

RÜDIGER STOBBE

DIGITAL LITH PROCESSING 25.0



Contents

Introduction	5
Install and First Run	7
Preferences	9
Directories	9
Output	10
Previews	11
Compute	12
User Interface	14
Running Your First Development	18
Dealing With Color	24
Basics	24
Lookup Tables	28
Lookup Tables from description files	28
Lookup Table Editor	32
Defining a Color	33
Add a new point	34
Work with an existing point	34
Buttons	35
Other functions	35
Standard Development Module	36
Parameters	36
How to use the parameters	39
Presets	42
Image Adjustments	45
Session Files	48
The Digital Lith Exchange	49

Uneven Development	50
Pepper Fogging	50
Patterns	51
Streaking Modules	55
The streaking modules included in DigitalLith	57
Taming the effect	61
Development Parameters Revisited	62
What's next?	66
Thanks	67

Introduction

THIS TUTORIAL COVERS DIGITALLITH, version 25.0 a program which simulates lith prints for digital images. Lith printing is a special darkroom process. It is a technique which uses black&white negatives, black&white silver gelatin paper and a black&white developer to achieve quite moody and colorful images. If you are interested in the darkroom process, and I hope after having played around with DigitalLith you are interested in the real process, there are some books you might want to have a look at: *The Master Photographers Lith Printing Course* [1] and *The World of Lith Printing* [2]. There is also a very short section about lith printing in *Beyond Monochrome - A fine Art Printing Workshop* [3]. The real process produces beautiful unique images with a special mood and feel to it. DigitalLith is only a simulation and was never intended to replace the real process.

In the beginning it was anything but comfortable - a program that you used on the command line which went off to compute the image taking minutes to finish and then left you with a result that was not what you intended because you mixed up some of the twenty positional parameters.

Lith printing itself is a time consuming process where you sometimes have development times in the 10-15 minute range. And since it is a kind of exponential development, most of the time you stare down on a white sheet of paper in the development tray until towards the very end of the development time the image starts to show up and forms in a short amount of time. That means: having the digital process run within a few minutes is not too bad compared to the real process, but without looking at the image building up you are without control. That was when I started building a UI so that you can see the image develop and as a result you are able to set the parameters in a more comfortable way. Once that had been done things fell into place from version to version. Why do you

You might ask yourself why I did DigitalLith then anyway? That is a valid question and I will try to give you a valid answer. To do the real process you need a darkroom but I do not have a darkroom anymore. In the end I was using my darkroom just to do lith prints. To do lith prints and to get reproducible results you need to continuously do it and not just now and then. By the time I closed my darkroom I did not have enough time and also had started to use digital cameras. I thought I could live without lith prints. But I could not. Because of that I started to look into how to simulate the process (or the results) in Photoshop. Now, I am not a very experienced Photoshop user which means that I was not able to get even close the look of a real lith print. That was the moment I started thinking about how one would simulate it and that started DigitalLith.

have to remember the parameters, let's have presets. Why do I have to start from scratch for every new start, let's have a session and continue right where we left off. Simulate the process even more by having a snatch button. What is the easiest way to deliver some predefined presets, let's have the DigitalLith exchange. Why do I have to remember the settings used to process that image, let's write the development parameters into the image metadata. What about uneven development, let's have patterns and let's have streaking modules and let's load them dynamically from the DigitalLith Exchange. Why not let the users develop their own streaking modules, let's have an SDK. And so on.

Lots of functionality, lots of features, not enough lots of documentation. This is the reason I came up with this tutorial. It will walk you through the program's features one by one.

Note: Although DigitalLith runs on MacOS, Windows and Linux, all the images of the user interface are taken while running the program on MacOS. That is because MacOS is the development platform of DigitalLith and what I use for personal work. Also the keyboard shortcuts are from MacOS but can be translated easily to Windows.  on MacOS corresponds to  on Windows and Linux and  on MacOS matches to  on Windows and Linux. In case there is notion of a  +  then this refers to a click on the right mouse button.

OK, honestly there is no longer an SDK and there is no loading of process and streaking modules anymore because it was a maintenance and backward compatibility nightmare. Now what you get is the built-in modules and that's it. The good news, all the previous module data migrates smoothly into the new built-in ones.

Install and First Run

THIS CHAPTER WILL SHOW YOU how to install and run the program. The first step to install DigitalLith is to go to the DigitalLith home page and download the program: [Click!](#). Use the version most suitable for your operating system. You will be able to run the generic Unix version on any operating system which has a *Java Runtime Environment* installed, but if you are on Windows or Mac OS it is just more convenient to install the packages meant for these operating systems. And even on Linux it might be easier to install the Debian package. For Windows just run the executable and it will install the DigitalLith program. If you have a previous version installed, uninstall it first. On a Mac just double click on the zip file and move the application into the Applications folder. But also follow the special instructions on the DigitalLith homepage to work around MacOS gatekeeper. On Ubuntu Linux you change to the directory you downloaded the .deb file to and run

```
sudo dpkg -i DigitalLith_25.0.deb
```

Or you just double-click it in the file manager and follow the install instructions. On a generic Unix system things are a little bit more complicated. Here you have to unpack the archive into a folder where you want the program to reside. This will create the directory `digilith` which contains all the bits needed. To run the program, open a Terminal window, set the `JAVA_HOME` environment variable to point to your Java installation home (which should be a Java version 21 for version 25.0 of DigitalLith) and the `DIGITAL_LITH_HOME` environment variable to point to the `digilith` directory created when unpacking the archive. To start the program run

```
$DIGITAL_LITH_HOME/digilith.sh
```

Now that you have managed the stormy waters of installing the program let us start the program and do some basic setup

In previous versions you had to install a current Java version to run DigitalLith. This is no longer the case for Windows, Mac OS X and most Linux versions around, as the install packages are self-contained Java applications which are bundled with the Java runtime needed.

I am not a verified MacOS developer and on Catalina and later the program will not launch until you allow to run programs from unverified developers. One way of doing this is by unpacking the zip file, run DigitalLith once (which will fail with a message telling you the program is not working properly) and then open a Terminal window, `cd` to the folder where you unpacked the zip file and run the following: `sudo xattr -dr com.apple.quarantine DigitalLith.app` This is probably the easiest option.

of preferences. After clicking the program icon or running the program from a terminal window, what you are going to see is something like in Figure 1.

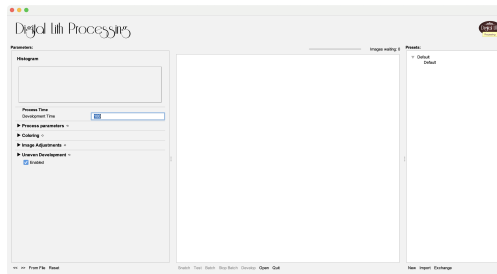


Figure 1: DigitalLith-UI after the first start

We do not care about all these elements you see in this window for now. But we will go ahead with some basic customization of the program. How that is done is described in the next chapter about Preferences.

During this tutorial we are using the Light UI mode as shown above. But you can also select a dark look of the program in the preferences which will look like the picture below.

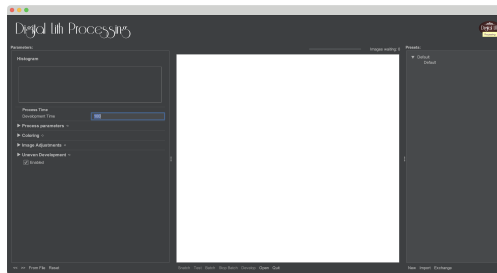


Figure 2: DigitalLith-UI in dark mode

Please keep in mind that to switch between dark and light mode you will have to restart the program.

Preferences

HERE WE ARE GOING TO DEAL with the preferences which control some parts of the program. Please have a look at them first.

When you start the program go to the Preferences menu. On MacOS this is done via  +  or via the menu  > . On Windows or Linux you will find them in the file menu  > . There you will be able to setup some needed settings to use all the program's features. Some of them may not make sense to you at this point, but here is a description of all of them and you may come back to this later to adjust the settings accordingly.

The preferences dialog has the following sections:

- Directories - where you set some places DigitalLith will use to store and look for presets, sessions, patterns etc
- Output - where you specify some output parameters
- Preview - where you specify preview sizes
- Compute - where you specify some parameters used to compute
- User Interface - where you can specify some basic user interface settings

Let us go through all of them.

Directories

In the Directories part of the preferences you can set some places in the file system where DigitalLith stores or looks up files. Figure 3 shows what the Directories tab looks like.

- **Session/Presets Directory** - DigitalLith uses four so-called dot files in the user's home directory. Coming from a Unix-like system you might be used to dot files. Here you can tell

 +  to go to Preferences

Please note, that the images do not show the default values, but the values I selected for this tutorial

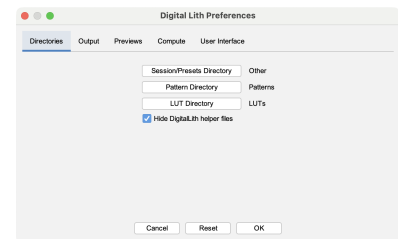


Figure 3: The Directories tab in the preferences dialog

DigitalLith to store at least the session, presets and preferences files in another location. I would recommend to store them in some separate directory (at least I do so) to keep the home directory clean. For DigitalLith the dot files are

- `.digital_lith_preferences` - to store the preferences you are about to edit in the above dialog window
- `.digital_lith_session` - to store the current session so that when you leave the program and start again, you are able to continue the work right where you left off
- `.digital_lith_presets` - to store the development presets that you create yourself or that you download from the preset exchange
- `.digital_lith_log` - to store some logging in case of errors so that we have a bit more info about what happened

The dot-files carry the program version in their names as a post-fix. The new version will initially read from the older dot-files. If you store the dot files in a specific directory and not in the home directory, then there is no need to hide them. In this case you can deselect *Hide DigitalLith helper files* which stores them without the leading dot.

- **Pattern Directory** - If you are going to use patterns to simulate uneven development (or for any other reason) then you need to specify where DigitalLith will search for patterns. How the pattern directory is set up is described in a later section of the tutorial.
- **LUT Directory** - to store the color lookup tables which can be used for image coloring and bypass the coloring dialog.

Output

The Output tab of the preferences dialog includes settings which customize the output of the developed images. Figure 4 shows the dialog.

- **Output to JPEG** - By default DigitalLith creates PNG output files. If you prefer to create JPEG files, you can define the output quality here.
- **Scale with high quality** - If checked, scaling of the image is done with higher quality. This also means that when the image is scaled, the scaling of the image takes more time

Having the version part of the file name will make sure that if you install a new version of the program which causes some issues, you will be able to go back to the version you ran before without having your settings destroyed by the new version.

It is up to you how to set up the directories where DigitalLith is going to store its data. But I would recommend to take control of it and not let the program do it.

Output to a JPEG file means smaller files at the cost of inferior quality due to the JPEG compression.

and slows down the user interface a bit (especially when up-scaling to a large image size) when loading or changing the image adjustments.

- **Add size, border and timestamp to file names** - This appends the used size and border to the file name as well as a current timestamp of image creation so that it gets a unique name all the time. Usually DigitalLith checks the names inside the output directory so that you will get a unique name, but if you moved some images outside the output directory that can no longer hold. With the timestamp it is not. You can give a pattern where `SIZE` is replaced by the size given in the process, `BORDER` is replaced by the border given in the process and `TIMESTAMP` is replaced by the timestamp. As you can see I am not interested in size or border, but have enabled the timestamp.
- **Embed thumbnail image in metadata** - This will embed a thumbnail preview into the generated image. When reading parameters from generated images this allows for a faster preview in the selection dialog.
- **Automatically choose a unique file name** - Often you will develop the same image multiple times. This will not bother you with asking for overriding an already existing image but just choose a unique file name if the file already exists. If you do have the timestamp in the filename this is very unlikely to happen.

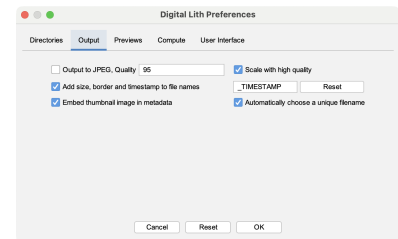


Figure 4: The Output tab in the preferences dialog

Previews

In this tab you can define the size of the preview. Running a Lith development takes some time and for that reason you can adjust the parameters and do test developments in a smaller size. See Figure 5

DigitalLith supports two sizes. You can select for each of those sizes if it shall use the Retina resolution which results in a finer preview, but also needs a bigger size which influences the time needed for a test development. The second preview can also be used in a mode where during the test development the user interface does not get updated which results in faster development. You can select the two previews with $\text{⌘} + 1$ for the first or $\text{⌘} + 2$ for the second.

- **First preview size** - The first preview size.

$\text{⌘} + 1$ or $\text{⌘} + 2$ to switch the preview

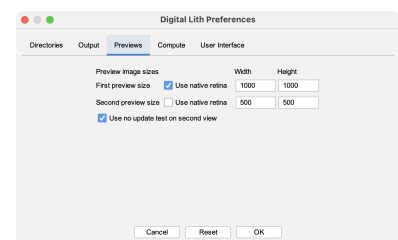


Figure 5: The Previews tab in the preferences dialog

- **Second preview size** - The second preview size
- **Use no update test on second view** - If selected the user interface does not get updated during a test development which results in faster development.

Compute

In the Compute tab you can set some parameters which customize the computation of the image. See Figure 6.

- **Recent images size** - DigitalLith maintains a list of your recently used images. If you want to open one of them just select them from the Recent Images sub-menu of the File menu. With this setting you can specify the maximum number of entries in that menu. The default is 15 and there is a hard maximum of 100.
- **Store test development in recent images** DigitalLith stores developed images in the list of recent images (including the development parameters). If this option is selected, then also test developments are stored in the list.
- **History size** - Whenever you run a computation, DigitalLith puts the used parameters into the history so that you can go back to the very same parameters later. Often you will end up playing around with some parameters and sometimes you realize that you want to go back in the history of computations. This parameter defines how many entries will be held in the history. Set it appropriately. Setting it to a high value in the hundreds will be no problem since the parameter set values stored do not use up that much memory. Remember that the history is saved from one session to the next. So if you quit the program and start it again later the history will still be there.
- **(Test)Parallel level** - DigitalLith is an iterative process which needs a lot of computing to produce the final image. Modern computers run with one or more processors each having multiple cores. This allows the image to be computed in parallel utilizing all processing cores. Here you can set the level of parallelism used by the program. If you run one quad core single Intel CPU machine which supports hyper-threading 8 might be a good choice here. If you run on Apple ARM then just use the number of CPU kernels of your processor. But if your machine is too fast to do a test development and stop at

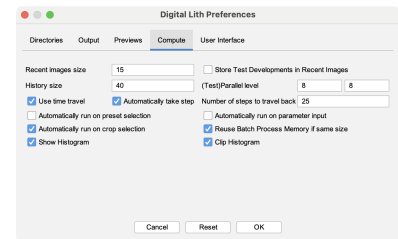


Figure 6: The Compute tab in the preferences dialog

If you do not care about the tech-speak here, just run a computation for an image and measure the time it needed. Then increase the parallel level and run the same image again. Increase the level until the computation does not get any faster.

the right time, then it might be good to not use all the CPU power for test developments. For that reason you can give a different parallel level for test developments or development/batch processing. The first number sets the parallel level for test development, the second sets the value for development or batch processing.

- **Use time travel** - If enabled the results of the last development steps are buffered and after a test development you can go back and forth in time by visiting the buffered steps. This allows you to better select the correct development time.
- **Automatically take step** - When selected this will automatically take over the time travel step into the development time parameter so that the user does not have to manually take over the step by pressing  + .
- **Number of steps to travel back** - This defines the number of development steps that are buffered for time traveling.
- **Automatically run on preset selection** - DigitalLith lets you define presets so that you can easily re-use settings that you prefer. There are two possibilities to select a preset. First you can click on one of the presets in the list on the right of the program window. The other method is to select one of the presets in the favorites list. Most of the times you select a preset you may want to immediately do a test development with it without changing any development parameters. This setting lets you select to automatically start a test development whenever you select a preset.
- **Automatically run on parameter input** - There are times where you just want to change one parameter and run a test development automatically. In that case select this option.
- **Automatically run on crop selection** - This starts a test development whenever a new crop is selected.
- **Reuse Batch Process Memory if same size** - DigitalLith uses a lot of memory and for that reason you might run out of memory for really big images. In that case it is better to release that memory after each batch run. However, if the memory is not an issue you can re-use the memory from one batch run to the next (for example when computing a series of images of the same size) which lets the program run a little bit more efficient.

- **Show Histogram** - Shows a histogram of the image above the parameter panel. This lets you better control the distribution of tonal values of the image.
- **Clip Histogram** - A histogram of a developed image might have quite some spikes in the shadows and highlights and thus might make it unusable for the rest of the tonal values. For that reason you can clip the spikes to get a better view on the rest of the histogram. If you select this option the clipped values are shown in red in the histogram. Click into the histogram to see the un-clipped version.

A warning here, if you are in a crop view, then only the histogram of the crop is shown

User Interface

In the User Interface tab you can set some parameters to customize the user interface. See Figure 7.

- **Language** - Here you can set the language. We do have two localizations, English and German. If you run in another locale DigitalLith will default to English.
- **The information to be logged in the log file** - This defines how detailed the info that is written into the log file is. This can be:
 - **Error** - only error messages are written into the log file.
 - **Error and Warning** - only error and warning messages are written into the log file.
 - **Error, Warning and Info** - error, warning and info messages are written into the log file.
 - **Everything** - everything is written into the log file. This might be overly detailed and only useful for debug purposes.
- **Action After Computation** - Since the compute process might take some time that you do not want to spend staring at the monitor you can configure what should happen once the computation is done. The following options exist:
 - **Do nothing** - there will be no action at all.
 - **Show a dialog and play a sound** - this opens a dialog window and beeps once the processing is done.
 - **Show a dialog only** - this will only show a dialog window but not play a sound.

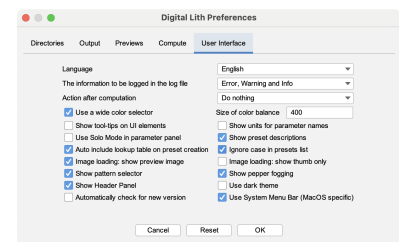


Figure 7: The User Interface tab in the preferences dialog

- **Play a sound** - this will not show a dialog, but only play a sound. On Mac OS once the computation finishes you will also see the program icon jumping.
- **Use wide color selector** - In the coloring dialog there are different gradient sliders which let you define the colors. The normal dialog has a resolution of 128 values out of 256 possible values. With this option you can enable to show wide sliders which let you select all 256 possible values but might no longer fit on very small displays.
- **Size of color balance** - In the color balance dialog there are some gradients which let you select the coloring as well as some curve controls to handle the mix of shadow, mid-tone and highlight colors. The bigger these controls are the more accurate your control of the values. But once a dialog does not fit onto the screen Java does weird things showing the dialog. Here you can select how big you want the controls to appear.
- **Show tool-tips on user interface elements** - For some user interface elements there are tool-tips. One example is the process parameters. Once you move the mouse over the parameter name you will get a small tool-tip window with a description of the parameter. If you are new to DigitalLith that might be an option to enable. Once you are more experienced you might choose to not show these tips because they are a distraction.
- **Show units for parameter names** - By default, DigitalLith shows just the parameter names. With this option you can enable the display of the parameter names with their corresponding units. This means that each parameter name contains info on whether this is a number or a percentage as well as info about the range. For the inexperienced user this might be helpful. Those who use DigitalLith regularly might choose to leave this turned off since the user interface looks cleaner without the unit info. There is still unit info in the tool tips.
- **Use solo mode in parameter panel** - This is similar to Lightroom's solo mode in the develop module. In solo mode, whenever you select a part of the parameter panel (Process Settings, Coloring, Image Adjustments or Uneven Development) the other parts collapse. This will keep the left side panel clear. When not in solo mode you can just expand or

collapse whatever part you like without changing the expansion of the other parts.

- **Show preset descriptions** - If you define a preset, you can also add a descriptive text in addition to the name. This option lets you define whether that description should be shown if you move the mouse over a preset.
- **Auto include lookup table on preset creation** - This automatically includes the lookup table when a new preset is created and the coloring is included. That way you do not have to remember to do so.
- **Ignore case in presets list** - This ignores upper and lower case when ordering the list of presets.
- **Image loading: show preview image** - If selected a preview of the selected image is shown in the image loading dialog. This makes it easier to select the image and to not rely on the name only.
- **Image loading: show thumb only** - If selected a preview is only shown if there is a thumbnail image embedded. This will speed up image selection when not the whole image is loaded.
- **Show pattern selector** - If not checked the uneven development part of the parameters will not show the pattern section - which does not mean that patterns are not applied if for example a preset which has a pattern is selected.
- **Show pepper fogging** - If not checked the uneven development part of the parameters will not show the pepper fogging section - which does not mean that pepper fogging is not applied if for example a preset which has pepper fogging is selected.
- **Show header panel** - The header panel uses some space and on small monitors this is a waste of real estate. Deselect this to no longer show the header panel.
- **Use dark theme** - This enables the dark theme. DigitalLith needs to be restarted to make the switch of the theme happening.
- **Automatically check for new versions** - There is a new version of the program every once in a while. If you do not

I have to admit, not showing patterns or pepper fogging is a very egoistic feature as I very rarely use patterns or pepper fogging. For me those features are more or less historic as today I have the streaking modules to make them obsolete. But hey, the whole program is kind of egoistic because I did it for my needs

want to go out and check the website yourself you can enable DigitalLith to check if there is a new version on start-up and if so lets you know that there is.

- **Use System Menu Bar** - Not seen here and only on MacOS, this allows to attach the menu bar to the window itself. It is sometimes uncomfortable when working on big monitors where DigitalLith might only occupy a small part of the area and maybe not at the top where you have to cross half the monitor with the mouse all the time to reach the menu bar. Although not very MacOS-ly it is more comfortable at times.

There is a bug in Java for MacOS M1/2 where you sometimes loose the menu bar being unusable after. This is not happening on Linux or Windows, nor on MacOS Intel. And it is not happening to the menu bar attached to the window. This is no longer happening on MacOS 14.4.1.

This completes the list of preferences. Have a look at them and maybe come back to them once you have more experience using the program.

Running Your First Development

NOW THAT YOU HAVE INSTALLED AND SET UP the program it is time to run your first image through DigitalLith. Therefore you first need to load an image into the program. It can be a black and white or a color jpg/png. Most of the time you might want to use a black and white image as DigitalLith only does a pretty simple black and white conversion.

DigitalLith is implemented so that you should be able to convert your images to black and white the usual way. There is no special preparation needed before you send it into the digital lith process.

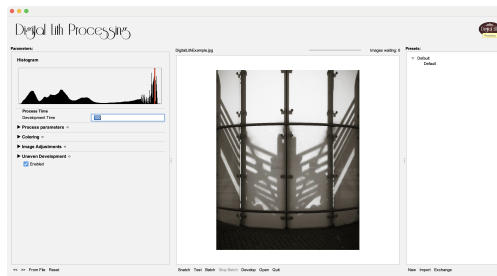


Figure 8: Example Image Loaded

This chapter will show you the possibilities you have to control the process. In Figure 8 you can see an image loaded into the program shown as a small preview image. On the left the process settings are expanded. When you run the program for the first time all the parts of the parameter panel are collapsed. Below the image you see a row of buttons used to drive the program. To load the image you can use either the `Open` button, the `File > Open` menu or the `⌘ + O` keyboard shortcut.

There are more ways to open an image. Via

`File > Recent Images ...`

you can select from a list of previous images you have loaded into DigitalLith. You can also load a recent image along with the development settings used the last time you dealt with that image. Do this via

`File > Recent Images With Settings ...`.

To follow this tutorial I recommend to download the example image: [Click!](#)

Here we are going to use the default settings as they are

`⌘ + O` to load/open an image

Some things to note, I am running on a retina display and have set my preview image area to be 1000×1000. If you are not running on a high resolution display (or maybe on an even bigger display) what you see might differ.

`⌘ + T` to do a test development
or `⌘ + ⌥ + T` to do an instant development

when you run the program the first time. Now let's go directly to your first development. First you will usually want to do a test development, so just press the **Test** button (or the $\text{⌘} + \text{T}$ keyboard shortcut) and watch the image develop or you do an instant development by pressing $\text{⌘} + \text{⌥} + \text{T}$. What you will end up with is something as shown in Figure 9 below.

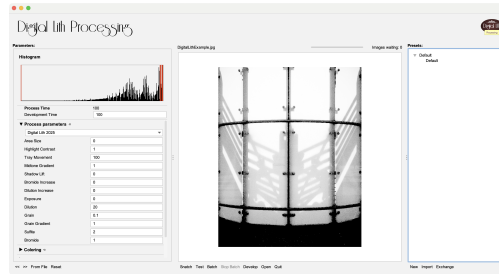


Figure 9: First development with default parameters

It already looks a bit lithy, but the color is still missing (which we will deal with in another chapter of this tutorial).

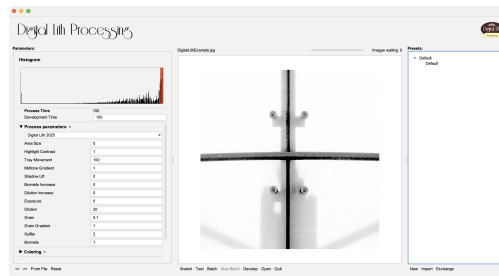


Figure 10: Development of a small crop of the original image

Also, since this is a process with some locality this will give you just a rough overview of how the image might turn out. You may want to have a look at a crop to see what is really going on. Just click into the image to get that part of the image enlarged. To get back to the normal view, click into the image area again and the full image will be shown. For now we are going to do a test development of an enlarged area. So enlarge one part and then press **Test** again and you may see something like the image in Figure 10.

This shows how the image is rendered at the pixel level. And that again might be too detailed. But there is at least one feature that may come in helpful here. We can switch from the detailed view to the full view again without losing the development of the detailed view or of the crop if you like. If you press $\text{⌘} + \text{I}$ you can see the developed crop well in the full view image - as you can see in Figure 11. Pressing $\text{⌘} + \text{I}$ again will bring you back to the cropped view. There is also $\text{⌘} + \text{J}$

DigitalLithis is a program which takes some getting used to and where you - as in the real process - need to gain some experience to interpret what you see on screen.

$\text{⌘} + \text{I}$ to switch between crop and crop in full view

$\text{⌘} + \text{J}$ to switch between original and lithed view

which allows you to switch between original and lithed preview. It works in cropped and full view.

Once you view the crop in the full view you can change to another crop without going to full view before. You do so by clicking into the image while holding the shift key. If you have checked the automatic development when selecting a crop, then the new crop will get a test development. If you are in a preview which does an instant development by default, then the new crop is also instant developed, otherwise the crop is developed normal. However pressing also the control key while clicking the behavior is switched. If you are in full view and want to select the crop but immediately show it in full view, then you can do this by holding the shift key when selecting the initial crop.

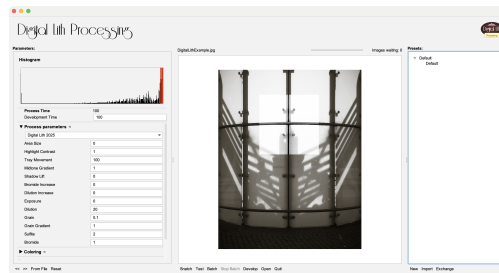


Figure 11: Viewing the crop in full view

Showing the crop embedded in the full view was an idea of my good friend Craig. I found it rather useless until I implemented it. Now I use it all the time.

There is another shortcut $\text{⌘} + \text{L}$ which lets you switch between the current and the last development. The last development is valid as long as there is no change in the crop or the image adjustments (size, border, etc). You can also switch back and fourth while the development is running. As soon as you load another image, change the image size/border or switch from crop mode to normal mode or vice versa the last development is invalidated.

Up to now you have only used the default development parameters. But how do you influence how the image is going to be rendered? First of all let's look at the processing time. That is the parameter right at the top of the parameter panel which you will find on the left. This parameter defines the number of iterations run by the process. Increase it and you will get a darker image, decrease it and it will turn out lighter. Every time you change a parameter you will have to run the test development again. You can also stay with the crop embedded in the full preview and do a test development. Instead of the full preview being developed you will now see the crop inside the preview being developed. You can also toggle back and forth between crop and crop embedded in full view while the develop-

$\text{⌘} + \text{L}$ to toggle between last and current development

I recommend in the settings to check auto-development on crop selection. Otherwise you will not recognize that you have selected a crop and only see this once you do a manual test development. I was already going on bug hunt when I tested the program in a clean environment and forgot to do that selection. I thought the cropping was broken.

ment is going on.

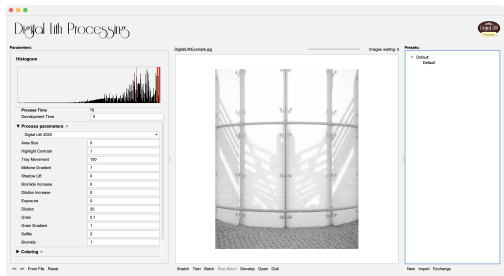


Figure 12: Development stopped right before the first pixel goes black

Setting the development time parameter to any value above zero is one way to stop the development process. There are two other ways to influence development time. One is by telling the program to stop development just before the first pixels turn into black. For that you will have to set the development time to 0. To go a step darker, which means that there are pixels which have *just* turned black: use -1. -2 means going one step further and so on. Figure 12 shows the test development run with a 0 as development time which means that the development stopped after 89 iterations. This leads to a much lighter image.

Sometimes, if you do not have any idea about what development time to choose, going with a 0 or -1 might turn out to be a good idea. At least until you learn about the third way to control the development time.

 +  to start and stop the process

Finally there is the method you may know from doing real lith prints. Here you start the test development by clicking on the  button (or the  +  keyboard shortcut). Now the development will run as long as you do not press the snatch button again. Once you stop the development the used process time will show up in the process time input field and you can refine it.

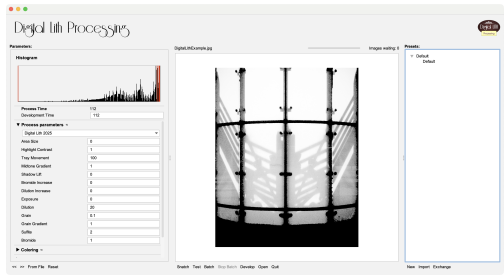


Figure 13: Development started and stopped with snatch button

Figure 13 shows the result of a development controlled with the snatch button. It is a development stopped a bit late, the darker parts turned into full black and you also see a bit more of the grain being built up. These blocked blacks are probably not what you want. So let's go back to the initial value for the process time (or some other values you found useful).

The time travel feature goes back to an idea coming from Craig, a good friend and the best beta tester ever. We talked about it for quite some time and I was always ending the discussion with a 'it is not going to happen without a major re-write of the program. So it is never going to happen.' But then out of curiosity I looked at it and found that it is very easy to implement and it resulted in one of the most useful additions to the program.

There is another way to find the process time that results in an image that you like. It is the time travel feature added in version 4.6.4.

With this feature you are able to go back in time - development step by development step. This means you overdevelop a bit and then go back step by step. You will see the image from the selected development step and so can decide on the development time. You can go back and forth in the development by using $\text{⌘} + \leftarrow$ and $\text{⌘} + \rightarrow$. Since DigitalLith 4.6.5 you can also go into the future. This means you can travel ahead in your development time.

You can even re-engage the development process by pressing $\text{⌘} + \text{⌘} + \text{S}$. Keep in mind that only the amount of development steps you defined in the preset are buffered. Once you go ahead too far you can not go back to where you started anymore. Well, except you re-develop. To go all the way back to the beginning of the buffered development steps buffer press $\text{⌘} + \downarrow$. To go all the way forward press $\text{⌘} + \uparrow$. To put the currently visited step into the development parameters press $\text{⌘} + \cdot$ (point) or set the preferences option to take over the time travel step automatically.

Then once you have the correct parameters you can decide to compute the image. There are two possibilities. First you can click the **Develop** button (or the $\text{⌘} + \text{D}$ keyboard shortcut) which opens a file save dialog where you can name to the processed image. You can then watch your image develop. During this stage you are still able to stop development using the snatch button. Once the image is developed you can continue with the same image or choose another one.

But processing an image might be a time consuming task and you may not want to wait for an image to be developed. In that case you can send it to the batch-queue via the **Batch** button and have it develop in the background while you work on the next image. If there are images in the batch queue it will be indicated at the top of the preview area. There will also be info about the progress of the current image in the queue. For each image in the queue an empty file is created once it is sent to the batch queue so that no other development may take the same file name. If you have many images in the queue and want to quit in order to not get stuck with empty image files, please use the **Stop Batch** button to clean up those files before you quit the program.

From the image that was snatched too dark I went back with time-travel to a development time of 93. The result is shown in

$\text{⌘} + \leftarrow$ and $\text{⌘} + \rightarrow$ to travel back and forth in time, $\text{⌘} + \downarrow$ to travel all the way back, $\text{⌘} + \uparrow$ to travel all the way *Back to the Future*.

$\text{⌘} + \cdot$ to set the process time to the current time travel step

The time travel feature is enabled in the preferences where you can also define how many development steps you want to buffer. The default is 25. Keep in mind that buffering the development steps takes up memory.

$\text{⌘} + \text{D}$ to develop on sight or $\text{⌘} + \text{B}$ to develop in batch

The only time I use a normal development over a batch development is right before I release a new version to check if that function is still working. All other times: Batch development is what I do

Figure 14.



Figure 14: Image developed with process time 93 found via time travel

Dealing With Color

Basics

NOW WE ARE GOING TO HAVE A CLOSER LOOK at the Coloring Dialog. The color is added later after the image is developed or if you have the development running in the preview window after each iteration.

There are different ways to deal with color. So let us have a look at the coloring parameter section first. What you see is two check boxes, a selector for lookup tables, a lookup table gradient input, a button `Set Colors ...` and a color gradient.

The color gradient shows how the tonal curve is mapped to color values from dark to light tones. The Keep Color check-box is used when you want to preserve the current coloring in a preset selection. We will talk about presets in a later chapter but if you select a preset and the preset includes the coloring, then the preset's coloring is not taken if this check box is selected. If the Black&White Only check box is selected, then no matter what the coloring is, a black and white image is created and stored. This is what you want to select if you add color later in another photo processor. Though in my opinion the more fun option is to have the coloring done while processing.

There is also the possibility to load color lookup tables or to store some color settings as a lookup table. How that works is described and shown later in this chapter along with the lookup gradient parameter.

Now let us jump right into the color dialog. To get there press the `Set Colors ...` button. In Figure 16 you can see how the dialog looks if you have never dealt with color before or if you selected a reset of all colors and curves. It might look a little bit confusing the first time you look at it, but once you know how it works it is not confusing anymore.

Right at the top you can select which coloring method to use. There is the coloring method of version 3.x of the program

The DigitalLith process itself is a pure grayscale process. It is actually just a grain machine.

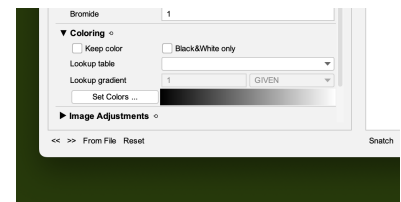


Figure 15: The color section

which is there for backward compatibility and the new 4.0 version. In version 3's coloring you could define tonal curves for the highlights and shadows while the mid-tone color is filled in. In version 4's coloring you can also set the tonal curve for the mid-tone color. Let us go with version 4 for now.

Below the coloring method selector you see the color gradient for the current selection. This is updated once you pull the sliders or change the curve so that you always see the effect of your changes. If you have an already computed image in the preview this image will also be adjusted as you work on the controls of the dialog. It would be pretty time consuming if you would have to run a test development every time you adjust the coloring.

Then you have the color selectors for the shadow, mid-tone and highlight color. You can set the hue and saturation.

Below there is a coordinate system showing five curves: the shadow, mid-tone and highlight curve, the global saturation and the gradient.

To adjust one of the curves you need to select the curve with the select box. The selected curve is shown in red. To add a new point to the curve just click where you want to place it and the point is added to the curve. To move a point, just click on it and drag the mouse. If you want to remove a point then you simply move it to another point and they merge into one. Alternatively you can also  + **click** on a point to remove it.

Besides the **Cancel** and **OK** buttons there are two more: **Reset** to reset the whole dialog and **Reset Current** to reset the currently selected curve.

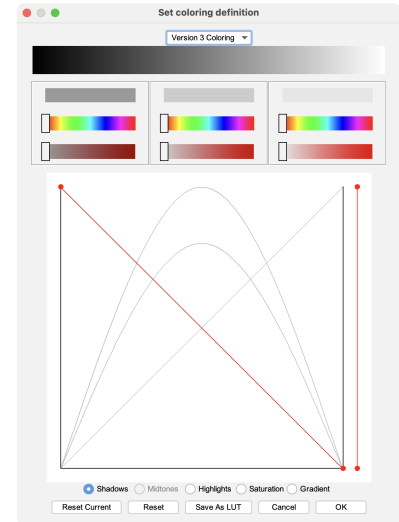


Figure 16: The coloring dialog

I for myself use the coloring dialog only to adjust some older presets of mine. But most of the time I am using lookup tables. For me that is the easiest way to get my coloring where I want it. Honestly, the old coloring dialog is not the brightest part of DigitalLith

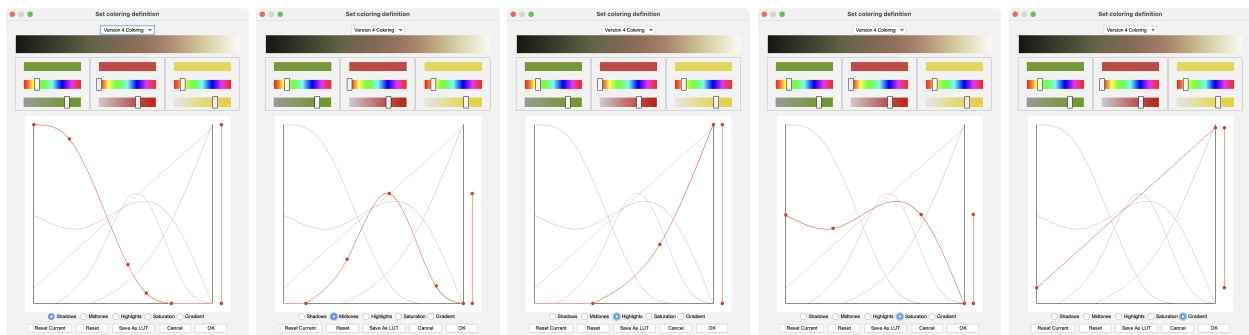


Figure 17: Example settings

For the next development let us adjust the values as shown in Figure 17 (at the end of the chapter you will find a download link containing a preset with the coloring as shown).

Play around with it a bit to get a feel for the different sliders. Maybe do a test development and then come back to the

dialog and place it on the screen so that you can see the developed image. As you change a slider or a curve you will not only see the gradient changing at the top of the dialog, but also the developed image will immediately show the change of the coloring.

With that coloring dialed in what we get in the UI is the following:

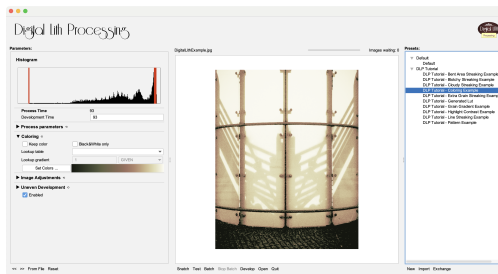


Figure 18: Development with color settings applied

Following this link you will get all the presets of the Tutorial, look at preset *DLP TUTORIAL - Coloring Example* with the above coloring: [Click!](#)

If you want to download the preset and add it to DigitalLith, here is a quick start in loading presets from files: Below the presets list on the right there is an **Import** button. Click it and select the preset file. Once loaded you can simply select the preset and the coloring will change.

Note: If you change the coloring after a test development, then the new coloring will be applied to the image without you having to run another development. However, if you use time travel to the past, then this does not work. That means that if you want to fool around with coloring after the development and time travel, please run another test development first.

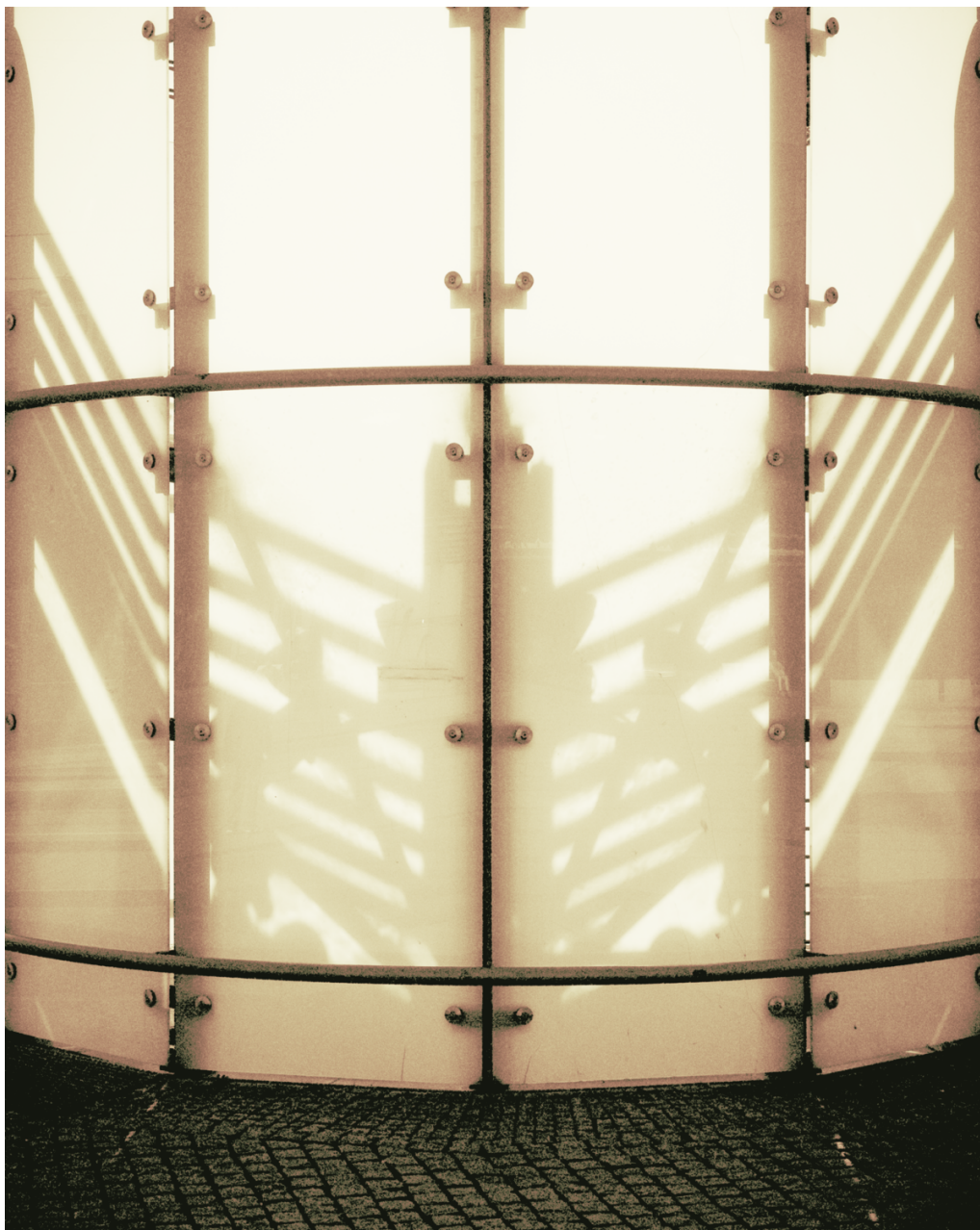


Figure 19: The image lithed with the color settings

Lookup Tables

If you are used to Photoshop or any other image processing program you might find the color handling of DigitalLith - rudimentary at best. Because of this there is an additional method to deal with coloring: creating color lookup tables. But do keep in mind that this has nothing to do with LUTs used in video processing. In DigitalLith we go a different route and in some sense a route where the generation of the lookup table does not need any specific knowledge nor any specific software. Every image processor which is able to create a PNG file can be used to create DigitalLith lookup tables. In fact, DigitalLith itself can produce such lookup tables. For example the coloring settings from this chapter converted into a DigitalLith LUT can be downloaded here: [Click!](#)

In DigitalLith we only map grayscale values so there is no need for a 3D LUT as we would only use a tiny part of it. What we use is a grayscale lookup table, which is stored as the first line in a PNG file. That means your LUT files will be PNG of a width corresponding to the size of the LUT and height 1. The dark values are stored on the left, the light values on the right.

To produce a DigitalLith LUT use your image editing program of choice and create an empty image which has the width equal to the size of the LUT. The height might be bigger, I would not start with just one line.

Now create a grayscale gradient ranging from black on the left to white on the right. Next you can tweak the color to your liking with all the available tools of your image editing program. Your image may now look similar to the color gradients shown in the DigitalLith color dialog. Finally reduce the image size so that it only has a height of one - you can also store the version without reducing the size, but DigitalLith will only look into the first line.

Store the image as 8bit RGB in png format. Give it a sound name and put it into the Lookup Table directory. Done. Now you have your first DigitalLith LUT.

Lookup Tables from description files

Besides the one line PNG files there is another way to create lookup tables. It often happens to me that I have one of my old lith prints in hand and then try to simulate that color in DigitalLith. It is difficult to deal with color in DigitalLith especially if you compare the result with the color of the real thing. For

Please note, if you download the example LUT, then you might just see an empty browser image. Just keep in mind, you are looking at a PNG with image height 1.

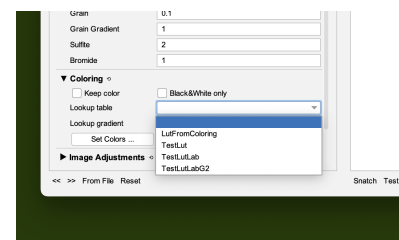


Figure 20: Selecting a lookup table

You already know about the place where DigitalLith looks for LUTs in the preferences section. So store the LUTs in some directory, then open the Preferences Dialog and point DigitalLith to that directory.

that reason there is a different way to define a lookup table. What you will need is a scan file of the print and a program which lets you read color values of specific areas of a file or of specific pixels. With a set of a few values there is a way to generate a lookup description file that you can store in the same directory as the lookup PNG files. This file extension should be `.lut`

Here is how you create such a file.

You need to either read the color values in RGB or in L/a/b. Then you create a file which contains the size of the table in the first line followed by lines with values of gray, R, G, B (or gray, L, a, b) separated by commas.

For this scan in Figure 21, I picked some values and created a file with the contents as shown in Figure 22.

The first line specifies that the size of the lookup table should be 2000. That also means that the first columns of the following lines need to be greater or equal to 0 and less than 2000 as these specify the gray tones that the colors map to. I picked the colors in L/a/b from the image. So the second column is the lightness, the third is the *a* and the fourth is the *b* part. The lightness should be in the range of [0..100] whereas the *a* and *b* values should be in the range of [-100..100]. If you specify the values as RGB then the values for red, green and blue should be in the range of [0..255].

There is a little tool that you can download from the digital lith homepage. It is a jar file so you need to have Java installed on your machine. That means this is not a self-contained application. You need to know how to run a program from the command line. Let's assume that you stored the above file in the same directory as the `lutgenerator.jar` file. In that case change into that directory and run the following command:

```
java -jar lutgenerator.jar -fromColorSpace=LAB testlut.data Te
```

This will generate a file `TestLut.lut` which you can copy into the directory where you store the lookup tables (see the preferences section on how to specify this directory). If you then select this newly created lookup table the coloring section of the parameter panel will look like what you see on the right. As you can see the gray values I selected lead to rather harsh changes which might look very lithy.

A former version of the generator was more complex and you could add gradient and distribution method. But since this is now supported in the UI this option has been removed.

But now you can specify the values in different color spaces



Figure 21: The scanned lith image

```
2000
0, 5, -3, 3
200, 10, -8, 5
400, 19, -7, 12
600, 29, -7, 19
650, 41, -6, 29
#800, 38, 18, 36
1100, 56, 24, 53
1300, 66, 20, 57
1600, 79, 10, 49
1800, 84, 5, 40
1900, 91, 0, 24
1999, 100, 0, 5
```

Figure 22: LUT definition

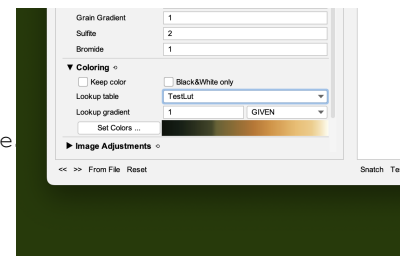


Figure 23: LUT gradient with given distribution

like LAB, HSL, HSV and RGB.

You can run

```
java -jar lutgenerator.jar -help
```

to get all the options shown and described.

The generator lost its importance since there is a lookup table editor in the UI. See the next chapter.

If you want things smoothed out a bit more there is the option to distribute the colors and overwrite the gray values you provided. The methods to distribute the colors are:

- **ADD** - determines the lightness of a color by the sum of R, G and B
- **MULT** - determines the lightness of a color by the product of R, G and B
- **LAB** - determines the lightness of a color by the lightness value of the color in L/a/b mode
- **GIVEN** - uses the lightness given in the file

Try different distribution methods and gradients. If you have a test development you will see the effect in the coloring of the developed image.

By default the gradient is 1, If the value is greater than one the light values are enhanced. For a value smaller than one the dark tones are enhanced. The gradient should be a value greater than zero. For a gradient of 2 the result is shown on the left.

And here is the tutorial image lithed with the colors distributed using the LAB method. But don't be surprised, this is not lithed with the default settings.

The data files, the generated lookup table used in the above development and the preset used can be found here: [*Click!*](#)

You do not need to install the lookup table as it is already included in the preset.

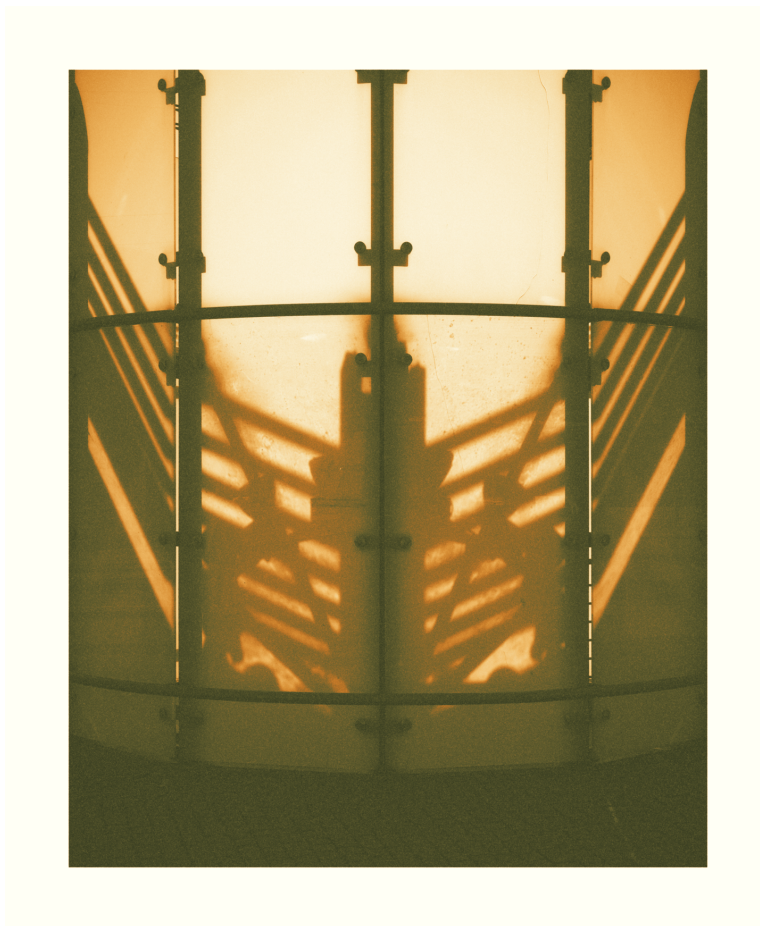


Figure 24: The image lithed with the generated LUT and LAB distribution

Lookup Table Editor

NOW THAT YOU WENT SO FAR WITH DEALING with lookup tables there is good news: There is an easier way to create them. Since version 25.0 there is a graphical lookup table editor built-in DigitalLith. You get to it by clicking into the color gradient in the coloring section of the parameter panel. This will open the lookup table editor. If you come from a selected lookup table then you can edit that data (you won't override the stored lookup table file). If you have defined the colors from the normal coloring dialog, then DigitalLith creates a lookup table which is almost the same than what you had.

Here is how the editor dialog looks like:

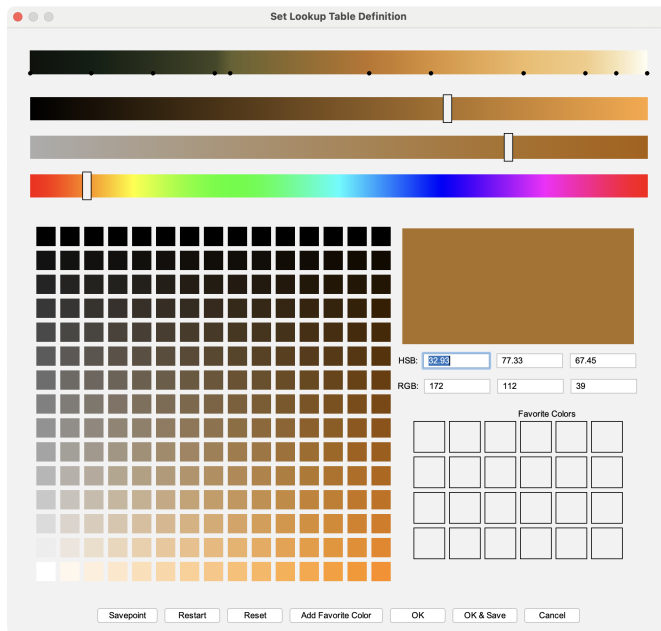


Figure 25: The Lookup Table Editor

The dialog of the Lookup Table Editor has many elements, let us have a quick look at each of them.

Defining a Color

The topmost element is the color gradient. You see small gray dots distributed on it. Each dot corresponds to a value as defined in the lookup table description. What you can do with those dots is described below.

Underneath the color gradient you find the blackness slider. With it you can define how dark or light a color is.

Next there is the saturation slider which lets you change the saturation of a selected color.

Then there is the hue slider which is used to select the base color.

Below that you find a field of color patches which show the base color as selected with the hue slider in different blackness (top-down) and saturation (left-right). If you change the hue slider you will see the patches change their color.

At the right side of the color patches you find some more elements. First there is a bigger color patch which corresponds to the color selected by the blackness, saturation and hue sliders. Below that there are text fields with the numeric values of hue, saturation and blackness as well as the corresponding RGB values. So you also can enter numbers to define the values.

Below that there is a field of 24 patches showing your favorite colors (at the beginning just all empty).

Once you start playing around with the sliders or start entering values for HSB or RGB you see that all those elements interact with each other. If you click on one of color patches showing the different shades of the base color you see that the numeric values are adjusted as well as the blackness and saturation sliders. Also the selected color patch is updated. Same if you select one of the favorite colors.

If you enter values into the text fields, the all other elements are updated accordingly. The value ranges are as follows:

- Hue: 0-360
- Saturation, Blackness: 0-100
- Red, Green, Blue: 0-255

Now that we know how to define a color we need to know how to add a color point to the lookup table.

If you start with an empty lookup table there are two colors defined, one at the very left (black) and one at the very right (white). You can start with an empty table by clicking on the button. You now have two options: Changing one of the

Do not laugh at me, but sometimes I find myself using the hue slider to just enjoy the many beautiful colors on those patches.

two points or adding a new point.

Add a new point

To add a new point you click into the color gradient where you want to place the selected color. Besides a simple click there are some modifiers you can use to change the action:

- : This adds the selected color at the point where you clicked to the lookup table and the added point is now the selected point.
-  + : This adds the selected color at the point where you clicked to the lookup table and the added point is not selected. Use this if you want to add more points and do not want to deselect the point to change the color controls for the next point
-  + : This adds a point with a color of the selected hue, but blackness and saturation correspond to the gray value of the lookup table. The more to right the point is the lighter it gets and less saturated. The added point is selected
-  +  + : Like -click but the point is not selected.
-  + : This adds a point with the color of the lookup table at that point. This is useful when you want to only slightly change a color of the lookup table. The point is selected.
-  +  + : Like -click but the point is not selected.

Work with an existing point

To work with an existing point of the lookup table you have to first select it. This is done by clicking on it. This also comes with some modifications:

- : This selects the point and updates the color controls with the color of the selected point.
-  + : Deselects the point.
-  + : This deletes the point from the lookup table. You can not delete the two last points.
-  +  + : This updates the color controls with the color of the point but does not select the point.

If a point is selected it can get updated by changing the color controls and by moving the point on the color gradient. You can even move it across another point.

Buttons

At the bottom you find several buttons that can be used:

- **Savepoint**: This memorizes the current lookup table as a new save point. Initially if you edit an existing lookup table the save point is the original lookup table. You can always go back to the last save point.
- **Restart**: This goes back to the last save point. If you have not yet defined a save point it gets back to the original lookup table.
- **Reset**: This resets the lookup table to a grayscale table with two points (black on the far left and white on the far right)
- **Add favorite Color**: Adds the current color as a favorite color (if there are still empty spaces)
- **OK**: Uses the edited lookup table.
- **OK & Save**: Uses the edited lookup table but also allows to save it into a file.
- **Cancel**: Cancels the edit and goes back to the original lookup table.

Other functions

The Lookup Table Editor has history. That means that if you by accident delete a point or move it or change it, you can press **⌘** + **Z** to undo changes.

To delete a favorite color you can **ctrl** + **click** on the color patch of the color.

Standard Development Module

IN THE LAST TWO SECTIONS WE DID DEVELOPMENTS without really knowing what all the parameters are meant for. Now we are going to have a closer look at all those development parameters. But keep in mind, this just covers the standard development parameters, there might be process modules installed with totally different parameters or parameters with the same name, but different behavior. By default, DigitalLith comes with three process modules installed. These are:

- **Digital Lith Classic** - the older standard DigitalLith process that has been available for the last few years.
- **Digital Lith Classic with Transparency** - like the classic process, but it also supports the development of transparent areas in the image.
- **Digital Lith 2025** - the new default process which has two parameters removed since they are no longer needed and which shows a smoother image generation (just for the fun of it). It also carries a new parameter to lift the shadows before the development.

Here we are looking at the default process (Digital Lith 2025) which is also the default setting if you start the program for the first time. Let us walk through the parameters from top to bottom:

Parameters

- **Development Time** - DigitalLith is an iterative process where at each step the image is computed out of the previous step. The Development Time defines the number of iterations. If you give a number greater than zero this defines the fixed number the process should run. But you can also have the process stop just before the first pixel turns

▼ Process parameters ◯	
Digital Lith 2025	
Area Size	0
Highlight Contrast	1
Tray Movement	100
Midtone Gradient	1
Shadow Lift	0
Bromide Increase	0
Dilution Increase	0
Exposure	0
Dilution	20
Grain	0.1
Grain Gradient	1
Sulfite	2
Bromide	1

Figure 26: The standard process parameters

into full black. In that case you set the development time to zero. If you want to go one iteration further then set the development time to -1, for two more iterations set it to -2 and so on. The Development Time is a parameter which is independent of the selected process module. Whatever process module you decide to use, it will always be an iterative process.

- **Area Size** - In the darkroom process the development runs faster where there is a silver grain seeded already. To simulate this behavior DigitalLith runs with some locality. That means to compute a pixel the area around it is also taken into consideration. This parameter gives the radius of this area. The area is going to be used kind of weighted so that those pixels with a greater distance will have less influence. If you give an area size of 1 then the area used will be the 3×3 matrix around the pixel. A size of 2 will result in a 5×5 matrix and so on. Having a bigger area will slow down the process since there is more work to do.

One exception is an area size of 0 where there is a 3×3 non-weighted area used. This and an area size of 1 are also the fastest options.

- **Highlight Contrast** - With this parameter you get control over the contrast level of the highlights. A value above one increases the highlight contrast whereas a value greater than zero, but lower than one decreases highlight contrast and thus also highlight definition. When you increase this parameter you may also want to decrease the exposure parameter, even into negative numbers.
- **Tray Movement** - With this parameter you get control over the deep shadows. A lower value means less tray movement which means that developer exhausts in the deep shadows. A higher value builds up the shadows faster. The value is a percentage with 100 being constant tray movement and 0 meaning no tray movement.
- **Midtone Gradient** - To have at least a little influence on the gradient you can define the mid-tone gradient which lets you adjust the contrast of the image. In cases where you use an image taken right from the camera this might be a good way to increase or decrease contrast. A value greater than 1 will increase contrast, a value smaller than 1 will decrease contrast.

- **Shadow Lift** - Lifts the shadows compared to the highlights. This is a percentage. A value of 100% would make all tonal values a highlight.
- **Bromide Increase** - As in the real process bromide will slow down development and increase shadow grain. In the real process bromide will build up in the developer during development. This parameter lets you simulate this. You specify the percentage bromide will be increased from one iteration to the next. Set it to 0 to run with constant bromide. A negative value will decrease bromide.
- **Dilution Increase** - The developer dilution might change slightly when developing an image as the developer gets weaker over time. With this parameter you can simulate that. What you specify is the percentage the dilution increases from one iteration to the next.
- **Exposure** - In the darkroom process you control image contrast by the amount of over-exposure you give to the image. The more light you give to the image the better the highlight tones catch up with the shadows. This parameter reflects this in DigitalLith and the behavior is the same, the highlight tones catch up if you increase it.
- **Dilution** - This is the developer dilution. Increase it to have the process running longer. This influences how grain builds up as that effect is more pronounced if there are more iterations.
- **Grain** - To have infectious development started DigitalLith lays down a grain pattern over the paper layer. This should be a value greater than zero. The higher the value the coarser the grain. Most of the time you will end up with a value below one. With this parameter you are kind of able to simulate the behavior of different papers. Use a low value to have DigitalLith produce effects that you may get with finer grained papers like Fomatone. Or increase the value to have DigitalLith simulate grainier lithing papers like Adox Vario Classic or Fomabrom.
- **Sulfite and Bromide** - These two parameters belong together and so will be described together. Bromide will slow down development and makes the grain coarser whereas sulfite will speed up development and make the grain smoother. These two parameters are tied together as they define a ratio

and not absolute values. That means that the effect of setting bromide/sulfite to 2/4 is the same as setting them to 1/2.

How to use the parameters

Now that we have described them all, let us try to go back to the image from last section and experiment a bit. In the previous example, the image that you see is computed with a development time of 93. And as you may have realized, there is a lot of highlight detail in the image that is not yet showing up in a satisfying way.

As in real lith we might want to increase the exposure to get the highlights catching up with the shadows. Let us start with a relatively high number of 5 in this case. You can see (first image) that this automatically brings in the mid-tones - and far too much, so we will have to decrease development time.

You can do this by using the snatch mechanism. And since exposure value of 5 seems a bit over the top, let us reduce it to 3. With snatching I ended up at a development time of 90. The second image (see next page) shows the result.

Next let us go with a little bit more lith grain. A higher grain value (0.3 in our case) results in faster development. So let's go up with the dilution too and set it to 30. Keep the development time for now (or snatch it if you like). The result is shown in the third image.

Now let us deal with another parameter pair. It is the sulfite/bromide ratio. A higher amount of sulfite smoothens the grain. As you see in the third image, it got rather grainy, so let us smooth it by setting the sulfite to 5 instead of 2. But keep in mind: Increasing sulfite also means a speed-up in development. So let's increase the dilution again, this time to 40 which might let us stay at 90 for the development time. See image four for the result of this development.

Now again, I do not think that the highlights look very promising and by just looking at the highlights and mid-tones we ended up with rather blocked shadows. So in our next run we are going to hold back development a bit by setting the tray movement down to a value of 50 and the development time to 80 (see the fifth image). This unblocks the shadows a bit but shadows are rather dull since the shadow contrast isn't very pronounced.

And this leads us to a parameter that I use all the time since it got introduced. It is the highlight contrast parameter. With

Exposure, Highlight Contrast and Tray Movement you are able to control contrast in your image to a degree earlier versions of DigitalLith were not able to. So let's give it a try.

Just remember, you need to play around a bit and get used to all these parameters to get the results you want. There is no free lunch. But to make a long story short: The highlight contrast value and the exposure value depend on each other a bit. So when increasing the highlight contrast you will very often go into the negative values with the exposure value, just to preserve shadow contrast as well. And since your shadows might block up too fast you will have to use the tray movement parameter to get differentiation in the shadows again.

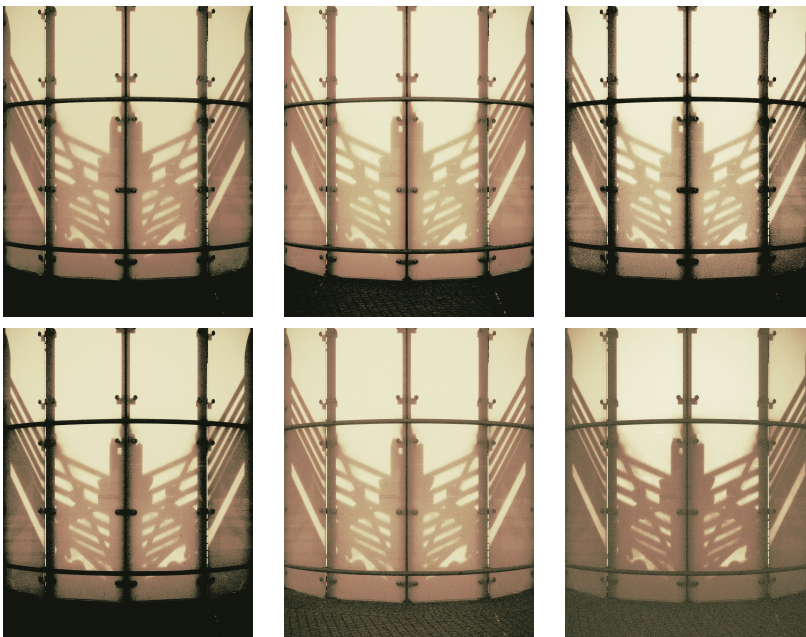


Figure 27: Example image lithed with different parameters

Once you've done this a few times it becomes second nature. Here we go with the following changes w.r.t. the last development: Highlight Contrast (5), Tray Movement (25) and Exposure (-2) and see the last image for the result. You can find these settings in preset *DLP Tutorial - Highlight Contrast Example* which is part of the Tutorial presets and can be downloaded here: [Click!](#)

So here is what you already learned:

- Increase the dilution to slow down development
- Increase grain for coarser grain (will speed up development too)

- Increase sulfite for finer grain (will speed up development too)
- Increase bromide for coarser grain (will slow down development too)
- Increase exposure to get the highlights to catch up with the shadows.
- Deal with highlight contrast, tray movement and exposure values to control highlight and shadow contrast.

Presets

ONCE YOU HAVE PLAYED AROUND WITH THE PROGRAM you may find parameters that you like more than others. And since there are many parameters, you do not want to remember them all. This is where presets come in. We already used them in the previous sections. If you went through the previous tutorial sections, your presets list might already show more than just the Default preset. In fact, it might look like what you can see on the right. I reduced the DigitalLith window size so that you can see the whole preset area elements which are the list of presets and the New, Import and Exchange buttons. And with the previous sections you might already be familiar with the import button.

And you may have already made friends with presets without deliberately defining one yourself anyway.

First there is the history. Whenever you do a development, the used parameters are stored in the history. You can browse through the history using the Previous and Next buttons at the bottom of the parameter area of the user interface. What you did with this was to just select an unnamed preset. The history is preserved from one session to the next, so once you quit the program you do not lose your history.

But having only the history would be very uncomfortable. There is a better way to remember parameters and that is through named presets. Once you have a parameter set that you like, click on the New button on the presets panel and a dialog will show up which lets you save these parameters to a preset with a name and a description of your choice. There are different types of presets. First, you can just take all the parameters which includes the development parameters, the color parameters, the uneven development parameters and the image adjustment parameters. Or you can just take a selection of the above. This means if you define a preset to only contain the color parameters, then all the other parameters

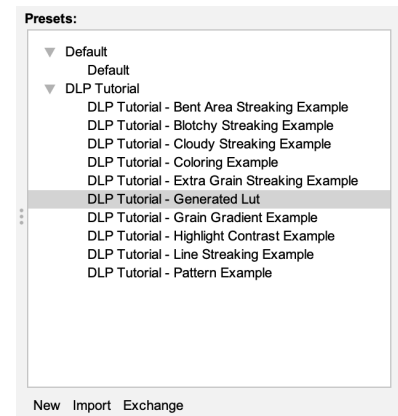


Figure 28: The presets pane you find on the right side of the program window

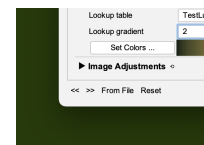


Figure 29: Development History

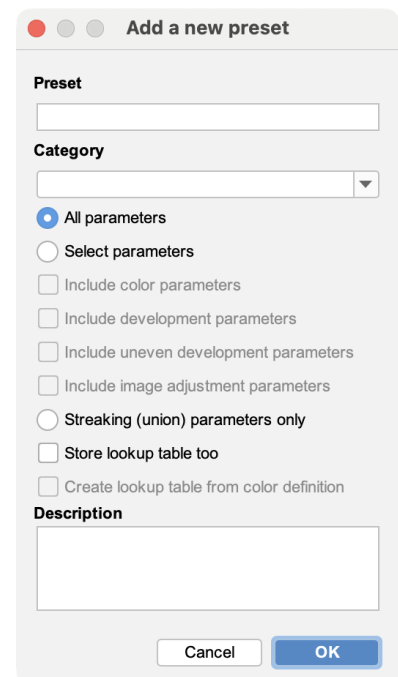


Figure 30: The add preset dialog

are unaffected once you select that preset. And third, you are also able to define a streaking union preset which will just deal with streaking parameters. Therefore when selected it does not overwrite the existing streaking parameters, but just adds to them. But at the moment you do not know about streaking at all - so let's deal with that later. And whenever storing the color parameters you also have the option to directly include the look-up table in the preset.

Once you hit OK the preset will show up in the presets list and you can select it from there.

If you would like to manage your presets then right or ctrl-click on a preset which pops up a context menu that lets you rename or redefine a preset from the current setting. There is also the possibility to delete the preset or export it into a file so that you can share the preset with others or move to another computer. And at the bottom there is the possibility to add the preset to the favorites menu.

To import presets from a file that you got from someone else or to re-import a preset that you have exported before you can use the Import button. This opens a file selection dialog so that you can select the preset file and import the presets stored in the file.

Another way to get your list of presets filled is to go to the DigitalLith Exchange. That is the button next to the Import button. The Exchange button will be handled in another chapter.

There is also another kind of preset. It is the processed image itself. Every time DigitalLith stores an image it puts the processing data into the metadata. You can read the processing data from an image file by clicking on the [Read From File](#) button. This lets you select the image file and populates the processing data from the data found in the metadata.

Over time you might store more and more presets which will fill up the list and you will end up scrolling up and down to find the preset you are looking for. To easily access the presets that you favor most there is the Favorites menu which lets you setup a list of the most used presets. On the right you can see the Favorites Menu with one preset already added via the popup we talked about above.

Since even that menu might fill up there is another way to define what will show up in the menu and what will not. If you select Manage Presets from the Presets menu you will see a window like the one below which lets you select the presets that you want to show up in the Favorites menu. As you can

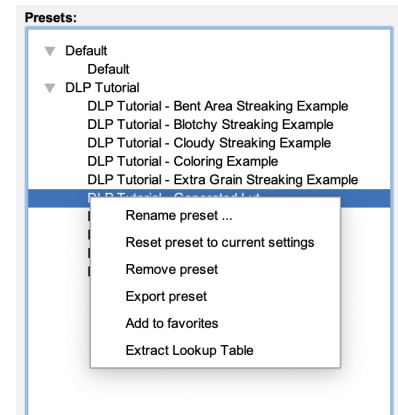


Figure 31: Popup description shown with mouse-over

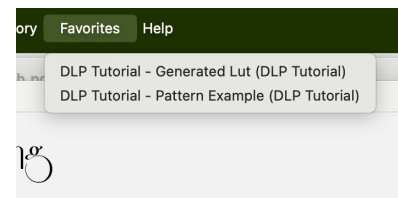


Figure 32: The favorites menu

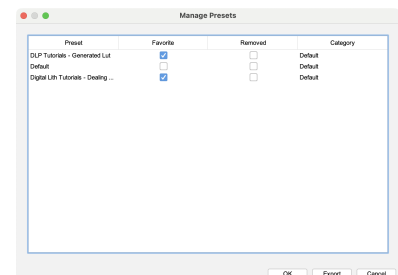


Figure 33: The dialog to manage presets

see the coloring example presets and the highlight contrast examples are already selected and as shown above that is correct, since they have been added to the favorite presets before. But you can also use this dialog to remove a whole set of presets. You may notice that over time you collect more and more presets which were state of the art the day you created them, but then after some time no longer worth the space they take up in the presets list. It would be time consuming and uncomfortable to remove them one after another. So go to the manage presets dialog and remove them all at once.

OK, we mentioned an image adjustments preset selection. In the next chapter we are going to deal with image adjustments.

Image Adjustments

IN THIS SECTION OF THE THE TUTORIAL WE ARE GOING TO cover the image adjustments part of the parameter panel. DigitalLith is an iterative process and also a process with quite a bit of locality. This means that the area around the pixel influences its next value. For this reason the image size is an important factor. There is a difference between lithing first and then scaling or vice versa. This also means that you should keep the pursued image size in mind. If you do want to lith the image for different sizes you do not have to provide the source in different sizes. DigitalLith can do that for you with the image adjustments.

Here you can scale the image before it goes through the process. What you provide is the desired length of the long edge. You are able to put a border around it which will then also be lithed along with the image. You either give the border as a percentage of the long or short edge of the image or as a specific number of pixels. If your brightest highlights in the coloring turn out to be too light so that the border does not pick up any color, you can adjust the lightness parameter. 100 means bright white where 0 means black. Keep in mind that even if you set the border to be white, it will take the color of the brightest highlights of the coloring. You can also select if the image border will be exposed like the rest of the image before running the development. There are settings where you will prefer to not expose the border because it would turn too dark otherwise.

The number for scale image denotes the number of pixels of the long edge used to scale. The border is an addition to that.

If you change the image adjustments the preview development is gone since the preview image is going to change as it is scaled from the adjustment settings.

In Figure 35 you can see an example of image scaling before and after processing with DigitalLith. The one on the left is be-

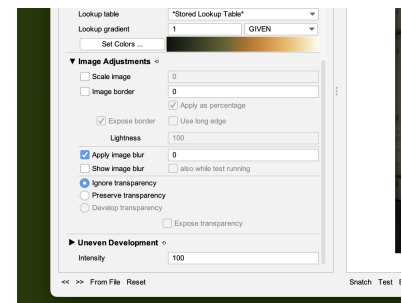


Figure 34: The image adjustments part of the parameters

ing processed in high resolution and then scaled down to 1500 pixels at the long edge. The one on the right is scaled down to 1500 and then processed. For the other parameters: Reset the parameters, apply the coloring preset, go with a development time of 100, dilution of 40 and grain of 0.3 and exposure of 1.

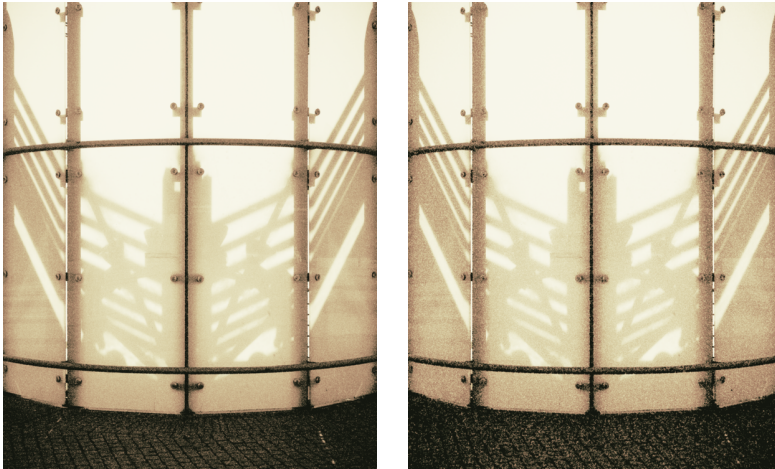


Figure 35: Example image scaled before and after lith processing

Then in Figure 36 you can see the effect of exposing the border versus not exposing it. The settings are the same as above without scaling. On the left you see the border exposed while on the right it is not exposed.

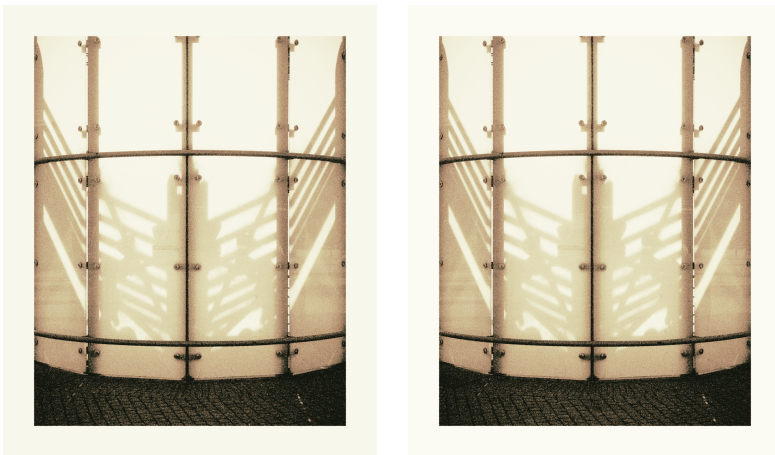


Figure 36: Example image with border exposed and not exposed

It looks as if the difference here is not much, but it is enough to have a better separation between image and border in the case where the border is not exposed.

The next part in the image adjustment deals with image blur. Applying a slight image blur results in more natural looking grain. But image blur comes at a cost of development time. For that reason you can decide for it to show up only after the test development ended or to not have it show up at all. You

can toggle to show it or not after the development is done. I usually apply a slight blur of 2 and show it but not during test development.

In Figure 37 you can see the difference of image blur applied versus not applied. We use the same adjustments as above but have the image scaled to 1500. The left image has no blur applied, the right has a blur of 2 applied.

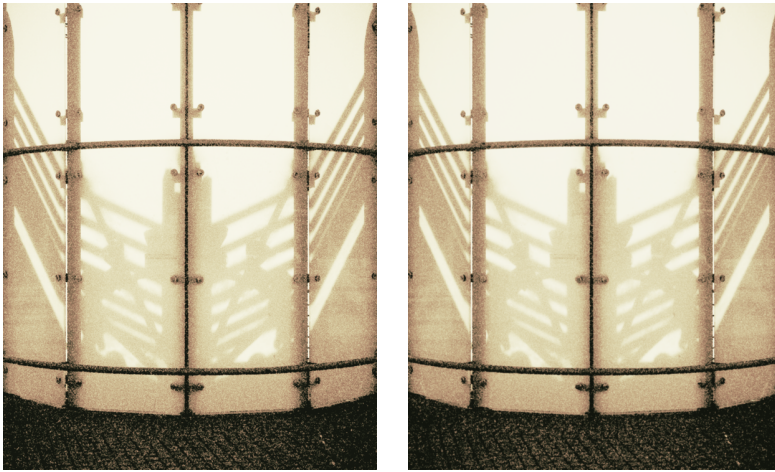


Figure 37: Example image with blur applied

The effect might be subtle but is visible. You might want to do your own testing.

And the last part of the adjustments deals with image transparency. For the standard process you can either select to preserve transparency or to convert it to white. There is another process which is able to also develop the transparency part which results in longer development since the transparency layer is developed separately.

This one was a very special requirement of my friend Craig who wanted to overlay some text layers with transparent background. To better match the images of his book he needed a way to deal with a transparent layer.

Session Files

DIGITALLITH SAVES THE STATE OF A SESSION so that you are able to continue your work after a re-start of the program. In this rather short section we look at where all this is stored in the file system. It deals with some helper files needed to save the session, presets, log information and so on.

These files are stored in the user's home directory as dot files, so on Linux/Mac they won't be visible.

- `.digital_lith` - the global dot file. This is used to specify the directory where the other dot-files are located. Default is the user home directory.
- `.digital_lith_session{\DigiLithVersion}` - contains the session. This includes the history and the current development settings.
- `.digital_lith_preferences{\DigiLithVersion}` - contains the preferences.
- `.digital_lith_presets{\DigiLithVersion}1` - contains all the presets you defined or imported.
- `.digital_lith_log` - contains the log data written by the file. The amount of info depends on the corresponding preferences setting.

If you come from an earlier version of DigitalLith, the content of the old files is read and stored in the new version files with the first call of the new version. That way the older files are not changed in case you still want to use the earlier version of the program for some reason.

Although the default location of these files is the user home directory, only the log file and the global dot file are always located in the home directory, the other file's location can be configured in the preferences.

Needless to say that the 25.0 in the file names above is going to be changed in the next version.

The Digital Lith Exchange

THERE IS A PLACE WHERE YOU can get predefined presets. You remember from a previous section that there is an [Exchange](#) button below the presets list. Clicking that button the program will lead you to the preset exchange. Instead of that button you can also use the Presets menu [Presets >> Presets Exchange ...](#).

Needless to say, you will have to be connected to the internet to find your way to the exchange.

If you do not want the program to connect to the internet - for whatever reason you can think of - then you can download presets using your browser. You see, somehow you have to go to the outside world. Here is the page that shows all presets with download links: [Click!](#)

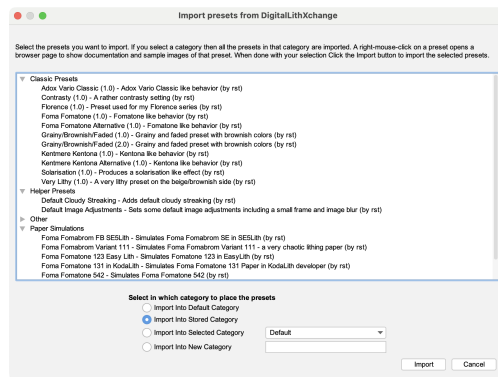


Figure 38: The preset exchange dialog

Having that in mind, let's go to the DigitalLith Exchange. You do this via one of the above possibilities. The program then downloads a list of available presets and show them in a dialog like the one in Figure 38.

Here you can select one or more presets and once you click on Import they will get imported and you will find them in the list of presets. You can select the section in which they appear.

If you like to get more info about a specific preset you can [ctrl + click](#) on the preset and this opens a browser window showing the help of the preset including sample images, see Figure 39.

The contents of the presets list in the dialog might look different because more presets are in the make and may show up here at some point in time.



Figure 39: Details for a selected preset

Uneven Development

IN THIS CHAPTER OF THE DigitalLith tutorial we are going to have a closer look at uneven development. This will be a three part topic and includes pepper fogging, patterns and streaking modules.

When doing lith prints, depending on the paper and developer setup you use, it may happen that you get an uneven development effect. Lith printing is vulnerable to inconsistent movement of the developer tray. So you may get streaking and underdeveloped parts. Lith printing also easily reveals emulsion faults where the normal process would run smoothly. You may also have to deal with chaotic infectious development which is called pepper fogging. Pepper fogging usually is controllable although there are papers where this is easier than with others.

Pepper Fogging

Let's start with pepper fogging first since this is just one single parameter. It is given as a percentage. That means you specify the percentage of the image area which will show pepper fogging. Usually this will be a very small number, way below one percent. The following images (Figure 41) show an increasing amounts of pepper fogging (1/100000%, 1/1000%, 1/100%). For the parameters, just reset the parameters, apply the coloring preset, use a sulfit parameter of 9, a dilution of 40 and a development time of 100.

Pepper fogging is applied to the image at the beginning of the process so the fogging gets developed and dots will grow in size - which explains why the 1/100% image does not really look like 1/100%.

Keep in mind that there are parameter settings which might produce very prominent pepper fogging which results in huge black splotches. For those reasons (and many other reasons)

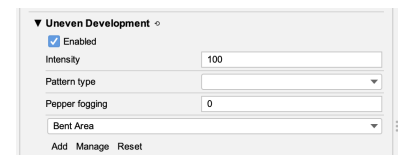


Figure 40: Uneven development parameters

Usually pepper fogging is something that you want to avoid and it is added to DigitalLith just for fun and completeness. I for example have it switched off in the preferences since I never use it - except for these demo images.

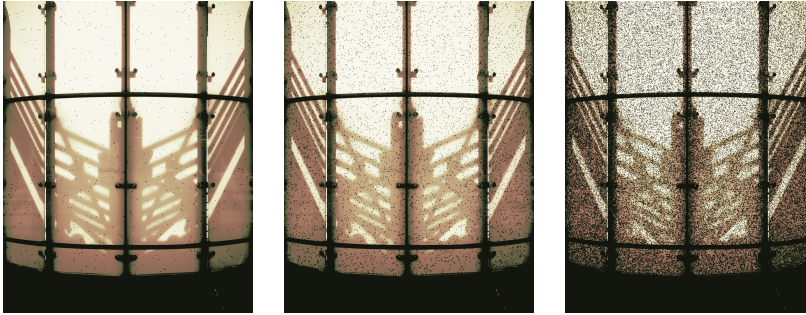


Figure 41: Different pepper fogging settings

you should rather use the extra grain streaking module.

Patterns

The next part of this chapter is the pattern overlay. Here you provide a set of patterns that you can choose from and then this is added to the image before processing. It is added to the paper layer only and so it is different from applying the pattern to the image before loading it into DigitalLith.

To work the tutorial you can create a directory on your file system and then in the preferences, point the pattern directory entry to this directory. At the end of this chapter you can find two zip files for download. Please unpack them into your pattern directory preserving the sub-directories - this means after unpacking the two files your pattern directory will contain two sub-directories called PaperCrushed and PaperCrushedSmooth. Each of the two sub-directories will contain two patterns.

Now, if you click on the pattern type drill-down menu in the uneven development part of the parameter panel you will be able to select between the two pattern types.

Patterns are organized into pattern types. As you already know from installing the demo patterns, there is a pattern directory on your disk that you specify in the preferences. Then you have the patterns organized in sub-directories which correspond to the pattern type. I have a whole set of images of crumpled paper which I have in original and smoothed versions in different directories. One example would be to have different grain patterns or you might photograph a gray cardboard with film and then scan the film and store that as a pattern to have the grain structure applied to the image. So the pattern type is the name of the sub-directory and the pattern name is the file name of the files found in that directory. There is no other

If you add patterns to the pattern directories while DigitalLith is running, just collapse the uneven development panel and expand it again and the new patterns will be shown.

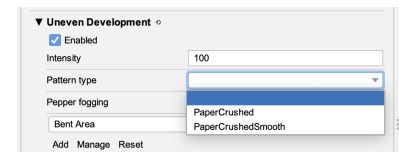


Figure 42: Selecting a pattern type

configuration file needed for this. The image on the left shows how this might look.

In Figure 43 you can see what happens if you select a pattern type: The first pattern of that type is selected. As you might notice I was pretty uninspired in naming them.

Once you select a pattern you also need to give a pattern strength (which corresponds a bit to the grain value of the standard process). In addition to that you might want to do some transformations like rotating it or flipping them horizontally or vertically. There might be more transformations to come.

Since the overall brightness of your patterns might be different you may check auto-contrast which will adjust the pattern so that its values will range from bright white to pure black and the average brightness will be a medium gray. That way you can change the pattern without adjusting the strength or any other development parameter to match the pattern brightness.

Last but not least you can also invert the pattern.

There is a button to randomly select another pattern of this type and another transformation in case you want to change this and be surprised. In Figure 44 you can see what happened to me after pressing the button.

Let's look at some examples. We start with the demo image, reset the development parameters and apply the coloring preset to have the same coloring. Patterns can be quite aggressive if the bromide is high. So let us adjust the development parameters a bit and set the sulfite to 5, the dilution to 30 and the development time to 85. Do not select any pattern yet - we do want to have something to compare to. See the first image below.

Then because we are adding a pattern, increase the dilution to 40 and select the pattern type, PaperCrushed with pattern name img002. This will set the pattern strength to 0.1 - keep it. The result can be seen in the second image below.

Next let us try the effect of auto contrast. Select that option and keep everything else the same. See the third image below.

We should also check invert - as above, keep everything else and you should get a result similar to image four below.

Crushed paper patterns might look a bit strange at first, but it is a matter of taste anyway. To get an impression of uneven development it is better to go with smoothed out patterns. Let us try this by selecting pattern type PaperCrushedSmooth and again with pattern name img002, this time without inverting.

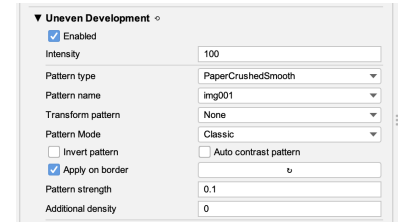


Figure 43: Pattern selection

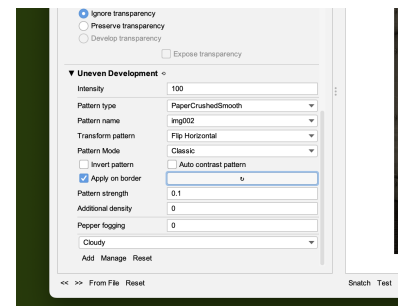


Figure 44: Selecting a random pattern

The result can be seen in the fifth image.

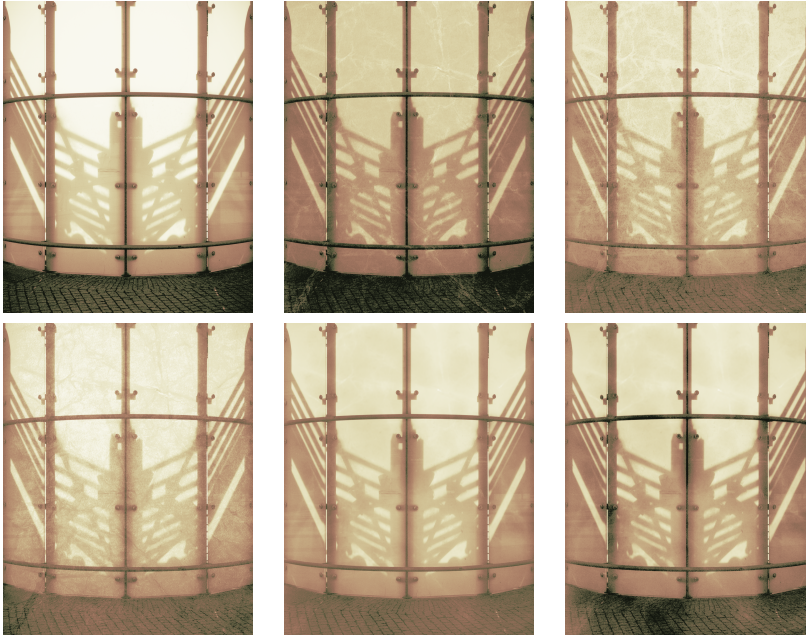


Figure 45: Pattern effects

OK and as a last step, since the images went a bit dull, let us bring some contrast back. For that we increase the highlight contrast to 2 and decrease the exposure to -1. Go back to 30 with the dilution and let us also get some slow-down and contrast by increasing the bromide by about 2% in each iteration. You can download the preset for this at the bottom of the page. The result is the sixth and last image in Figure 63.

If you want to reduce the pattern effect without having to adjust the image brightness once more you can go ahead and add extra density. The following example shows the effect. With a pattern strength of 0.3 the effect seems way too strong, reduce the strength to 0.1 and set the additional density parameter to 0.2 to balance out the strength reduction. Note that this is not using our coloring preset.

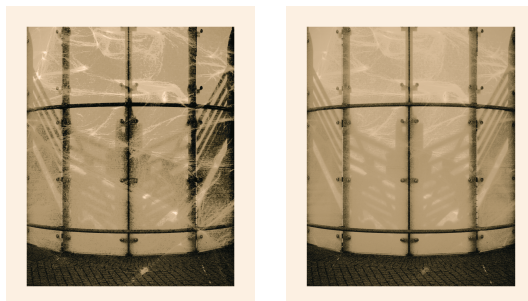


Figure 46: Adding density with lowered strength

There is another pattern adjustment that we have not talked about and which is only of use if you develop your image with a border. When having an image border you can decide if you want to add the pattern to the border too. In previous versions of DigitalLith the pattern has always been applied to the border and that means that for compatibility reasons the default is on. Here you can see the difference between the two settings, on the left the pattern is applied to the border, on the right it is not applied to the border.

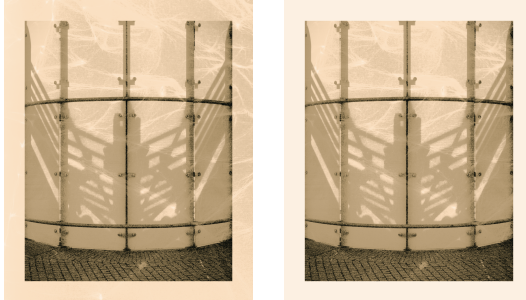


Figure 47: Applying the pattern to the border or not

Up to now we have not mentioned the mode to apply the pattern. We used the classic mode which is there since patterns got introduced and which applies the pattern independently of negative densities.

But there are more possibilities. You can use the *New* (as of version 24.0) mode which will take negative densities into account. There are two more variants of it, *New Lights* which works stronger on the lighter parts of the image and *New Shadows* which works stronger on the darker parts of the image, see Figure 49.

As you can see, there is a lot of room for creativity here. Play around with it and make sure that you do not end up walking around and looking for patterns only.

Here you can download some patterns:

- Crushed Paper Patterns: [Click!](#)
- Smoothed Crushed Paper Patterns: [Click!](#)

An example preset using patterns is part of the tutorial example presets and called *DLP Tutorial - Pattern Example*. The presets can be downloaded here: [Click!](#)

This completes the section about pattern overlays. The next section will shed some light on streaking modules which is another way of having uneven development effects simulated.

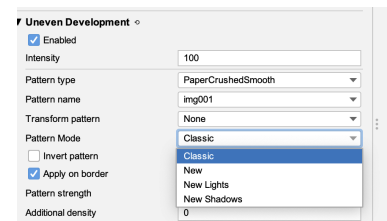


Figure 48: Pattern Mode selection

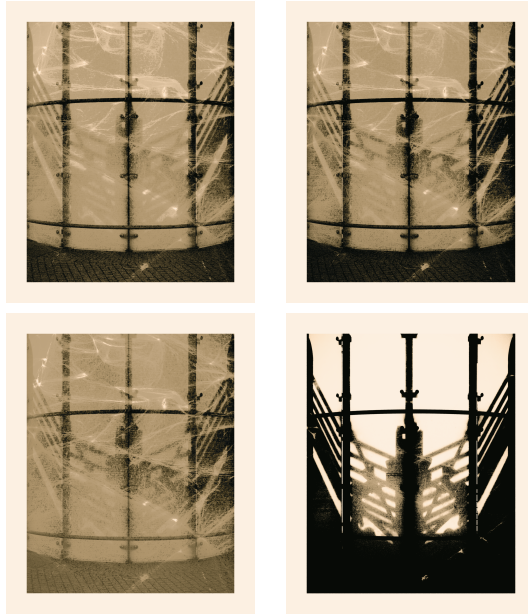


Figure 49: The different pattern modes from top left to bottom right: Classic, New, New Lights, New Shadows

Streaking Modules

Streaking modules are similar to pattern overlays. The difference however is that streaking modules compute a pattern overlay instead of applying a pattern stored in a file.

Since version 4.11 DigitalLith comes with a set of built-in streaking modules. Before that you had to first download the streaking modules from the DigitalLith-Exchange. Now since version 24.0 the streaking module exchange closed the doors. All relevant streaking modules are now built-in and it is by far fewer but more universal modules.

If you still have stored presets which use old streaking modules, then the data is migrated smoothly into the built-in modules.

There is a drill-down menu to select the kind of streaking you want to add. In previous versions that was a very long list because there were several different streaking type per streaking module like a paper only variant, a relative variant and so on. Now all that is handled by some parameters and there is only a few streaking modules left which are able to handle everything we had before. Do your selection and then click on **Add** to add that kind of streaking. Sometimes one streaking type is not enough, so you can add as many streaking entries as you want.

If you want to reset all streaking click on **Reset** which will remove all selected streaking. The **Manage** button will be covered

The reason for built-in modules only is that I removed the SDK. The SDK was a maintenance and backward compatibility nightmare which made every new feature of streaking modules difficult to implement since I had to make sure that also older modules are still functioning. There was never a feedback or bug when it comes to the SDK which in IT is a clear sign of a feature not being used. So I guess removing the SDK does not hurt anyone

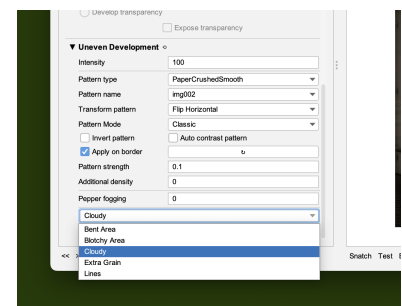


Figure 50: Selecting a streaking module

later in this section. Let us first select a streaking type from the pop-up menu and add it to the parameters.

In Figure 51 you will see what your parameters look like now. Here we added *Cloudy* streaking which adds a cloudy pattern to the image. Cloudy streaking is different from the other streaking modules as it does not add density through an amount of grain to be added. It just adds a cloudy pattern. It almost replaces using the pattern option which is added with its default parameters. We changed them a bit for the later example development. Let us first have a look at the parameters, just to get an idea of what the parameters of quite a few streaking modules look like. Usually there is an option to specify the amount of streaking to be applied. This is normally a percentage of the image area. Here we even have two amounts, first the one which specifies how much of the image area is covered with streaks and then the amount of streaking applied per streak. Then there is a variance, also given in percent so that not all streaks will have the same length and amount. Also the streak length is given, by specifying the distance from bottom and top edges, in this case 70% into the image area. Then we have the grain size which lets you specify - well - the size of the grains created. Next there is the strength. This is comparable with the grain parameter of the process. But you will see that you often have to give far higher values for the streaking. In this case there is a misalignment which means that the streaks might change direction a bit. And the last one is the flection which defines how much the streaks are bent. Play around with it and you will get used to the parameters.

Now let's have a look at the different UI elements. First there is a triangle for collapsing and expanding this specific streaking module's parameters. Then there are the two arrows which will let you move the streaking up and down if there is more than one streaking type added. A click on the minus sign will remove this streaking and a click on the plus sign will duplicate it with the same parameters.

If you ctrl-click the name of the streaking type a context menu like the one in Figure 52 will pop up.

You can disable this streaking or if it is disabled you can re-enable it. You can collapse it so that the parameters do not take up all the space. If it is collapsed then this lets you expand it again. There might be modules where you have vertical and horizontal parameters in one type. You can switch the vertical and horizontal ones.

With Set Defaults you can define the current values as de-

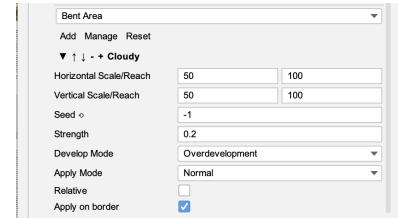


Figure 51: Module parameters

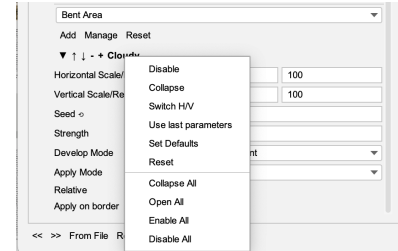


Figure 52: Streaking module context menu

fault values for this streaking type which means whenever you add streaking of that type the current values will be used as default values. With the reset you can reset the current values to the default values.

Now let us have a look at the streaking modules manage feature. For that let us first select the given parameters for our selected module as default values. After that click on the Manage button and you will get to the dialog to manage streaking modules.

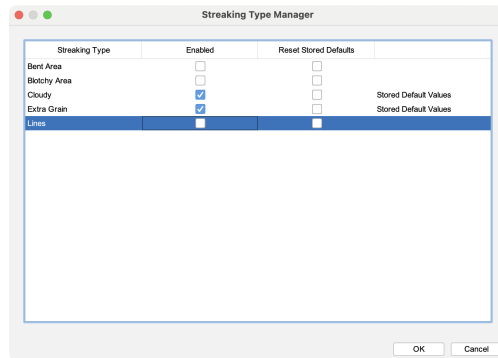


Figure 53: Manage streaking modules

Here you see a list of all installed streaking types. It may very well be the case, that you only like some of those types. So why bother with the types that you do not like. Here you can select to only see the ones you like see in the streaking selection. In addition to that you can also see for which streaking types you have overwritten the default values and select those which you want to reset to the original defaults.

Now it is time to describe the different streaking modules.

The streaking modules included in DigitalLith

Here is a list with a short description of the streaking modules built-in to DigitalLith.

All modules have the following parameters in common:

- **Develop Mode** - Indicates the kind of development. This can be over or under development. With over development density is added, with under development density is removed. Some streaking modules provide a third mode called leveled which applies streaking in some parts adding and in some parts lowering density.
- **Apply Mode** - Indicates how to use the negative densities when adding the streaking. It can be Normal where the

streaking is stronger in the shadow parts of the print, *Invert* where the streaking is stronger in the lighter parts of the image or *Paper Only* where the negative density is not used.

- **Relative** - Indicates if the strength is relative to the grain parameter of the process.
- **Apply on border** - Indicates that the effect is also applied to the image border.

Bent Area

As already shown in the previous section this adds bent streaks to the image. They creep into the image from the edges. For the invert flavors these are areas of underdevelopment, otherwise it is over development. The parameters are:

- **Amount Streaks** - the amount of streaking to be applied. This is a percentage of the image area.
- **Amount Per Streak** - the amount of streaking applied in each streak.
- **Streak Width** - the width of the streaks. This is a percentage of either the image height or the image width, depending on whether it is horizontal or vertical streaking.
- **Streak Length** - the length of the streaks. This is how far the streaks creep into the image and is a percentage of the image height or width, depending on whether this is vertical or horizontal streaking.
- **Variance** - the variance of the streak size. This is a percentage. If you give 10 and the length is given as 70 percent, then the length may vary between 63 and 77 percent.
- **Grain Size** - the grain size of the grains created. This is the radius of a grain. 0 means a one pixel grain, 1 means a 3x3 pixel grain. etc.
- **Strength** - the strength of the effect. This is comparable to the grain parameter of the process.
- **Misalignment** - some variance in the streaks so that they are not just straight. This is a percentage too.
- **Flection** - the flection of the single streaks.

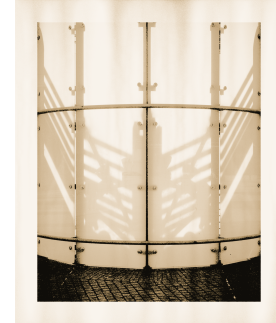


Figure 54: Streaking module Bent Area

Blotchy Area

Instead of streaks creeping in from the edges this module creates blotches of over or underdevelopment. The parameters are:

- **Amount** - the amount of streaking. This is a percentage of the image area.
- **Horizontal Spread** - the horizontal spread of a blotch. This is a percentage of the image width.
- **Vertical Spread** - the vertical spread of a blotch. This is a percentage of the image height.
- **Variance** - the variance of the blotches. This is a percentage.
- **Amount Per Blotch** - the amount of streaking per blotch.
- **Grain Size** - the grain size used. This is the radius of a grain. 0 means a one pixel grain, 1 means a 3x3 pixel grain. etc.
- **Strength** - the strength of the effect. This is comparable with the grain parameter of the process.
- **Misalignment** - the misalignment of a blotch. Set it to 0 and you get perfectly oval blotches.

Extra Grain

This module adds more grain to the image. The parameters are:

- **Amount Of Grain** - the amount of grain added. This is a percentage of the image area.
- **Grain Size** - the size of the grains. This is the radius of a grain. 0 means a one pixel grain, 1 means a 3x3 pixel grain. etc.
- **Strength** - the strength of the effect. This is comparable with the grain parameter of the process.
- **Uniformity** - the uniformity of a grain.



Figure 55: Streaking module Blotchy Area

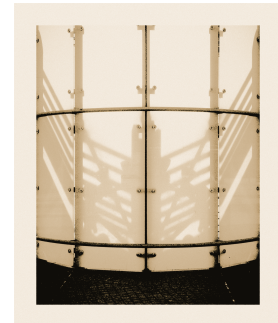


Figure 56: Streaking module Extra Grain

Line

This module adds lines of over or underdevelopment to the image. The parameters are:

- **Amount Streaks** - the amount of streaking applied. This is a percentage of the image area.
- **Spread** - the spread of the streaks. This is a percentage of the width or height (depending on the angle)
- **Amount Per Streak** - the amount of streaking per streak. This is a percentage.
- **Variance** - the variance of a streak. This is a percentage.
- **Strength** - the strength of the effect. This is comparable with the grain parameter of the process.
- **Grain Size** - the size of the grains. This is the radius of a grain. 0 means a one pixel grain, 1 means a 3x3 pixel grain. etc.
- **Angle** - the angle of the lines. 0 means horizontal, 90 means vertical.
- **Variance Angle** - the variance of the angle. This is a percentage.



Figure 57: Streaking module Line

Cloudy

This module applies a cloudy kind of streaking. The parameters are:

- **Horizontal Scale/Reach** - the horizontal spread of the effect and how far it reaches into the image.
- **Vertical Scale/Reach** - the vertical spread of the effect and how far it reaches into the image.
- **Seed** - A seed parameter used to randomize the effect. With a given number greater than zero you will see always the same effect. Set it to -1 and the program will choose a different seed for you all the time. You can also click on the little refresh icon (or the word seed) to get a number generated which you then can use. Having the number greater than zero makes sure that the effect in the processed image is the same as the effect in the test development. If you used -1 and found a nice setting, then you can re-use the seed parameter using the streaking popup and selection *Use last parameters*.

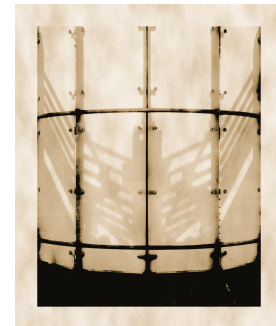


Figure 58: Streaking module Bent Area

- **Strength** - the strength of the effect. This is comparable to the grain parameter of the process.

Comparison of Streaking Modules

Here is an overview of all the streaking modules next to each other

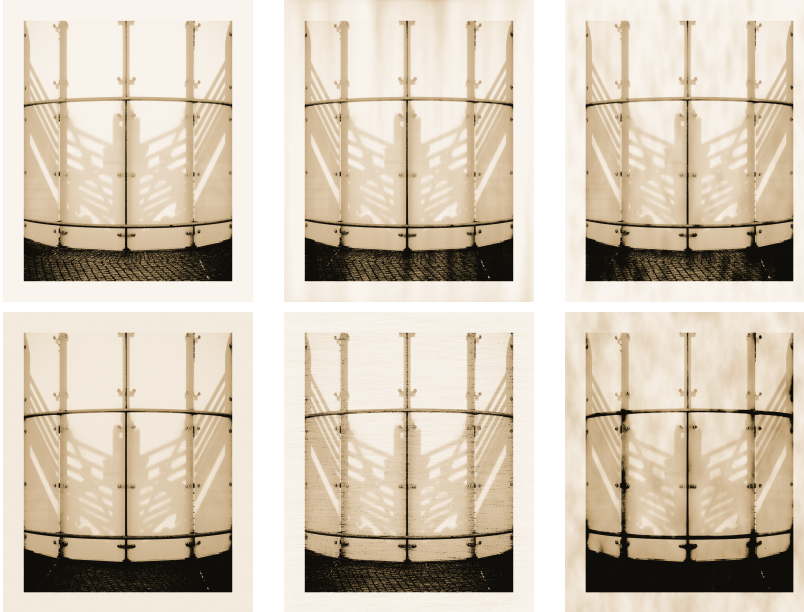


Figure 59: The different streaking modules applied from top left to bottom right: No streaking, Bent Area, Blotchy, Extra Grain, Line, Cloudy

I created a set of presets for the streaking module comparison as shown in Figure 59: Watch out for the Streaking examples in the tutorial presets which you can download here: [Click!](#)

Taming the effect

There is one parameter which might come in very handy. It is also shown when uneven development is collapsed. You might have set up a combination of streaking and patterns and now want to trim down the effect or maybe make it more prominent. Instead of going through all the streaking modules and adjusting the strength and also the strength of the pattern you can use the Intensity setting of the uneven development. The value is a percentage. To trim down use values lower than 100, to intensify use values greater than 100.



Figure 60: Reducing the effect of uneven development

Development Parameters Revisited

SO FAR WE HAVE NOT used all of the development parameters. So let us have a look at least at some of them. In the case studies series on my home page you will learn a lot more about all of them and how to use them to achieve a certain look. Here you can find them: [Click!](#)

DigitalLith is a kind of exponential process. This means it can happen that you see dramatic changes from one step to another and that it is not easy to land somewhere in the middle. For that reason you can slow down the process towards the end by slowly increasing the dilution. There is a parameter *Dilution increase* which defines a percentage that the dilution increases from step to step. You might want to use a very low number for this in the range between 0 and 2 percent. If you use too high a value you may never see black in your image.

Then there is another interesting parameter called *Bromide increase*. This specifies the percentage of bromide change for each iteration. You can set it to a positive number which means the bromide increases and grain gets coarser towards the end or you can set it to a negative number which means bromide decreases and grain gets smoother towards the end of the process.

Here are four examples where I processed them to a lower resolution so that the effect shows clearer. This is how you can do the same yourself. Reset the development parameters, apply our coloring preset and for each of the following examples set the grain to 0.3. Then select the scale image from image adjustments and set it to 1500. Now run developments with the following parameters shown in figure 62.

You see and already know that playing around with sulfite and bromide changes the development time. We wanted to keep the development time the same so I adjusted the dilution instead.

What we do is to show a first example with high sulfite which

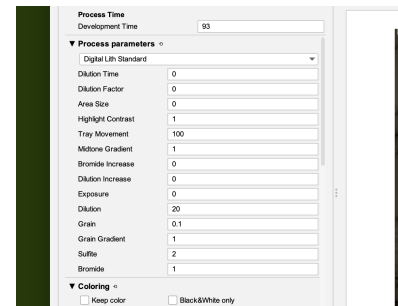


Figure 61: Parameters to tame the process

example	1	2	3	4
bromide increase	0	3	-3	0
dilution	60	45	40	30
sulfite	8	8	1	1
bromide	1	1	1	1

Figure 62: Example parameters

will smooth out the grain. Then there is the one which starts with sulfite/bromide ratio 8/1 and will end up with 1/1 (remember bromide and sulfite define a ratio which means 1/1 is the same as 8/8). The next one starts with 1/1 and will end up with 8/1. And the last example shows the effect of starting with 1/1 and staying there.

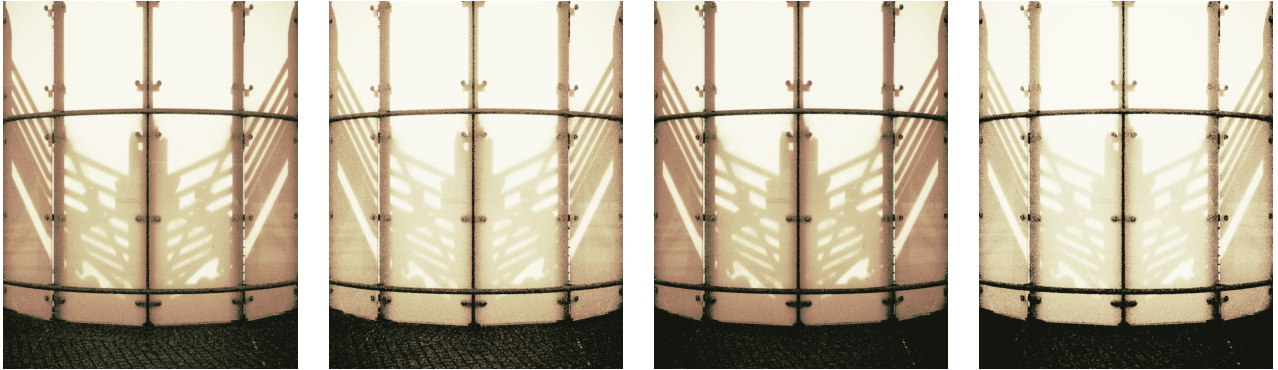


Figure 63: Usage of bromide increase parameter

Next let us visit the tray movement parameter. This also slows down development process but it works more towards the deep shadows. It simulates developer exhaustion which happens when the developer tray is not moved enough during development.

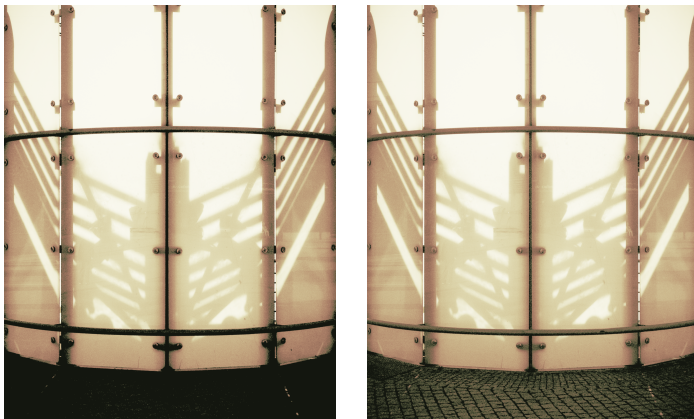


Figure 64: Usage of tray movement

Figure 64 shows the result of reduced tray movement. In both images we used default parameters with our coloring applied. But the left one has tray movement reduced to 50. See how the shadows are opened up but also how the highlights are nearly untouched.

The last parameter we look at in the tutorial part is the mid-tone gradient parameter which controls mid-tone contrast and therefore also influences highlight and shadow contrast a bit. A

higher value leads to increased mid-tone contrast and a lower value to flatter mid-tones. If you are curious, you can also use negative numbers.

Here is an example. As before reset the parameters and apply the coloring preset. Set the development time to 85, we will keep it there, but normally you would want to adjust it a bit.

The first image is with mid-tone contrast set to 0.25, the second is at 1, the third is at 1.5 and - out of curiosity - the fourth one has it set to -0.5. Here we go:

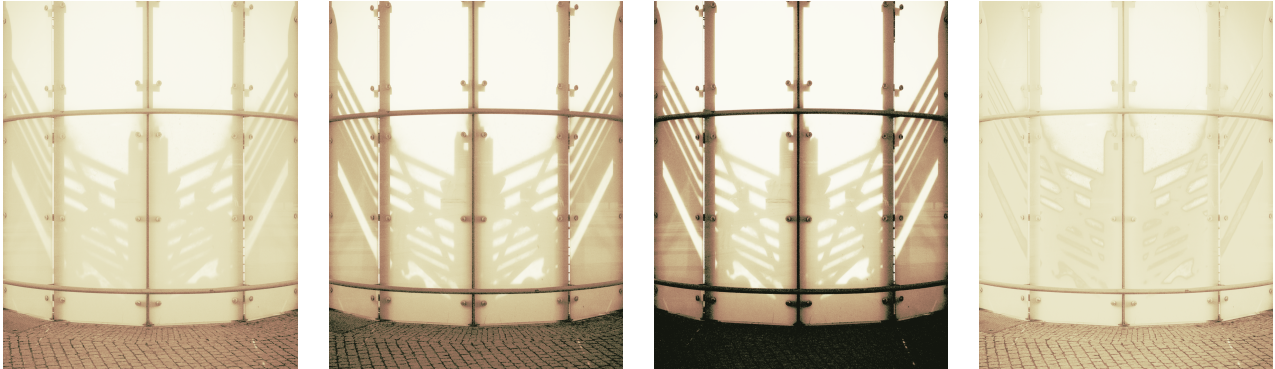


Figure 65: Usage of mid-tone gradient parameter

And now the very last topic of this chapter. It is a parameter that I use a lot: The grain gradient. The grain gradient lets you control when grain builds up. That means you can keep the highlights relatively grain free but then let it show up more prominently in the shadows. A value greater than 1 will make the highlights smoother and increase the shadow grain. A value greater than 0 but lower than 1 will make the grain appear earlier in the process. One thing that you may want to do when using this is to increase the grain to a high level and work with a high dilution too. But let us see an example of the grain gradient parameter in action - together with a few other parameters. There is a preset called *DLP Tutorial - Grain Gradient Example* which is part of the example tutorial presets which can be downloaded here: [Click!](#)

In Figure 67 you can see the smooth highlights and the grain starting for a certain tonal value. With the flat coloring you can have the grain be responsible for the tonal separation.

This image is also the last one in this chapter. Have fun with DigitalLith.

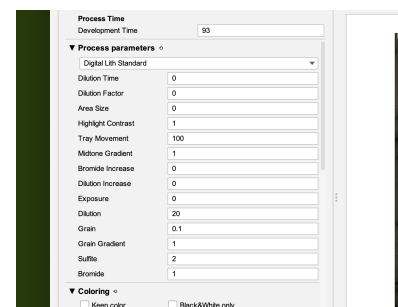


Figure 66: Parameters to tame the process



Figure 67: Grain gradient in action

What's next?

DIGITALLITH AT ITS CURRENT STATE is where I wanted it to be from the very beginning. So there is no big changes coming soon that I can think of in 2025. There will always be some minor bug fixes here and there and I am interested in all kinds of feedback. Some of the features of the program are there only because of user feedback. It might take a bit of time until some *enhancement request* gets realized in the program, but in the end many got it.

If you want to learn more about the program and on how to use the different parameters to meet a certain look, there is a whole series of case studies on my website. Please have a look at those too: [Click!](#)

I thought about including the case studies into this tutorial too. But there are more to come and so instead of having to update this tutorial with new case studies I decided to only have them on my website

Thanks

ALTHOUGH I WROTE ALL THE CODE on my own there are a few people involved into DigitalLith. Here I want to say thank you to:

- My daughter Judith for designing the icon. She could not stand the loveless icon I came up with in the beginning.
- My daughter Hannah for proofreading and making fun of my version of English.
- Sebastian for checking some install issues on Linux and beta-testing some versions.
- My dear friend Craig who is involved into the project more than anybody else. He is using every new version and has gone through highs and lows with those versions (including a loss of almost all his presets). But he has contributed idea after idea and often my initial thought was either nobody needs it or it is too complicated to implement - or both. But for the *too complicated to implement* I often proofed myself wrong and showcased the new feature to Craig a week later. The *nobody needs it* part I have to say: Many of those features I use myself all the time. One example is the crop view inside the whole image to have a better preview of the effect. That was very controversial and my initial reaction indeed was you do not need it, you will get a feel for the results over time. One week later I was already no longer willing to go without that feature. And there are many other examples. Craig, I hope you have as much fun as I have with this new version.
- Berend who took the seat of testing the many versions of the histogram and the LUT editor features. Thanks a lot for your patience.

Bibliography

- [1] T. Rudman. *The Master Photographer's Lith Printing Course*. Argentum, 1998. ISBN 1-902538-02-1.
- [2] T. Rudman. *The World of Lith Printing*. Argentum, 2006. ISBN 1-902538-45-5.
- [3] T. Worobiec and R. Spence. *Beyond Monochrome - A Fine Art Printing Workshop*,. Fountain Press Ltd, 1999. ISBN 0863433138.