Combining Multiple Images with Enfuse 4.2

Andrew Mihal*) Christoph Spiel

2020-11-21†)

*) Original author
†) Date determined via file-system – potentially inaccurate.

Abstract

This manual is for Enfuse version ⟨4.2⟩, a tool to merge different exposures of the same scene to produce an image that looks much like a tone-mapped image.

Copyright © 2004–2009 Andrew Mihal.
Copyright © 2009–2016 Christoph Spiel.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Short Contents

1 Overview 1

2 Known Limitations 3

3 Workflow[c] 4

4 Invocation 16

5 Weighting Functions 48

6 Color Spaces[c] 75

7 Understanding Masks[c] 82

8 Applications of Enfuse 85

A Helpful Programs[c] 99

B Bug Reports[c] 102

C Authors[c] 106

D GNU FDL[c] 107

Indices 115

Contents

List of Tables vi
List of Figures vii
List of Examples viii

1 Overview **1**

2 Known Limitations **3**

3 Workflow[c] **4**
3.1 Standard Workflow 4
3.2 External Masks 6
3.3 Interacting with Enfuse[c] 7
3.3.1 Finding Out Details 7
3.3.2 Console Messages 13
3.3.3 Environment Variables 15

4 Invocation **16**
4.1 Image Requirements 16
4.2 Command-Line Options 17
4.2.1 Common Options[c] 17
4.2.2 Advanced Options[c] 20
4.2.3 Fusion Options 24
4.2.4 Expert Options 25
4.2.5 Expert Fusion Options 27
4.2.6 Information Options[c] 38
4.3 Option Delimiters[c] 39
4.3.1 Numeric Arguments 39
4.3.2 Filename Arguments 40
4.4 Response Files[c] 40
4.4.1 Response File Format 41
4.4.2 Syntactic Comments 42
4.4.3 Globbing Algorithms 43
4.4.4 Default Layer Selection 43
4.5 Layer Selection[c] 45
4.5.1 Layer Selection Syntax 45

4.5.2 Tools for Multi-Page Files . . . 47

5 Weighting Functions **48**
5.1 Weighting Pixels . . . 48
5.1.1 Weighted Average . . . 49
5.1.2 Disabling Averaging . . . 49
5.1.3 Single Criterion Fusing . . . 49
5.2 Exposure Weighting . . . 50
5.2.1 Built-In Functions . . . 50
5.2.2 User-Defined Functions . . . 57
5.2.3 User-Defined Dynamic Functions . . . 57
5.3 Saturation Weighting . . . 67
5.4 Local Contrast Weighting . . . 68
5.4.1 Standard Deviation . . . 68
5.4.2 Laplacian of Gaussian . . . 70
5.4.3 Blend SDev and LoG . . . 72
5.4.4 Scaling and Choice of Mode . . . 73
5.5 Local Entropy Weighting . . . 73

6 Color Spaces[c] **75**
6.1 Mathematical Preliminaries . . . 76
6.2 Floating-Point Images . . . 77
6.3 Color Profiles . . . 79
6.4 Blending Color Spaces . . . 79
6.5 Practical Considerations . . . 81

7 Understanding Masks[c] **82**
7.1 Masks In Input Files . . . 82
7.2 Weight Mask Files . . . 83

8 Applications of Enfuse **85**
8.1 What Makes Images Fusable? . . . 85
8.2 Repetition – Noise Reduction . . . 86
8.3 Exposure Series – Dynamic Range Increase . . . 87
8.3.1 Tips For Beginners . . . 87
8.3.2 Common Misconceptions . . . 88
8.4 Flash Exposure Series – Directed Lighting . . . 89
8.5 Polarization Series – Saturation Enhancement . . . 89
8.6 Focus Stacks – Depth-of-Field Increase . . . 89
8.6.1 Why create focus stacks? . . . 89
8.6.2 Preparing Focus Stacks . . . 90
8.6.3 Local Contrast Based Fusing . . . 90
8.6.4 Basic Focus Stacking . . . 91
8.6.5 Advanced Focus Stacking . . . 91
8.6.6 Tips For Focus Stacking Experts . . . 97

A Helpful Programs[c] **99**
A.1 Raw Image Conversion . . . 99
A.2 Image Alignment and Rendering . . . 99
A.3 Image Manipulation . . . 100
A.4 High Dynamic Range . . . 100
A.5 Libraries . . . 100
A.6 Meta-Data Handling . . . 101
A.7 Camera Firmware Extension . . . 101

B Bug Reports[c] **102**
B.1 Found a Bug? . . . 102
B.2 How to Report Bugs . . . 103
B.3 Sending Patches . . . 104

C Authors[c] **106**

D GNU FDL[c] **107**

Indices **115**
Syntactic Comment Index . . . 115
Program/Application Index . . . 116
Option Index . . . 117
General Index . . . 118

Chapters or sections marked with a "[c]"-sign appear in both manuals, i.e. the Enfuse manual and the Enblend manual. The commonality extends to *all* sub-sections of the marked one.

List of Tables

3.1 Image formats and bit-depths 12

4.1 Verbosity levels 19
4.2 Mask template characters 28
4.3 Flexible exposure cutoff 34
4.4 Exposure weight functions 36
4.5 Grammar of response files 41
4.6 Grammar of syntactic comments 43
4.7 Globbing algorithms 44
4.8 Grammar of layer specifications 45

5.1 Default weights 50
5.2 Exposure weight functions 52

List of Figures

3.1 Photographic workflow 5
3.2 External mask workflow 8

4.1 Entropy cutoff function 31
4.2 Exposure cutoff function 33

5.1 Gaussian weight function 51
5.2 Lorentzian function 53
5.3 Half-Sine function 54
5.4 Full-Sine function 55
5.5 Bi-Square function 56
5.6 Comparison of exposure weight functions 57
5.7 Local analysis window 69
5.8 Laplacian-of-Gaussian 71
5.9 Entropy function 74

6.1 Log-transform 78

8.1 Sharp edge 92
8.2 Smooth edge 93
8.3 Focus stacking decision tree. 96

List of Examples

3.1 Output of '`enfuse --version`' 9
3.2 Output of '`enfuse --version --verbose`' 11
3.3 Output of '`enfuse --show-software-components`' 13
4.1 ImageMagick for exposure cutoff 32
4.2 Complete response file 43
4.3 Filename-globbing syntactic comment 44
5.1 Simple dynamic exposure weight function 62
5.2 Templated dynamic exposure weight function 63
5.3 Dynamic exposure weight function with extra arguments 64
7.1 Using **identify**. 83
7.2 Using **tiffinfo**. 84

Notation

This manual uses some typographic conventions to clarify the subject. The markup of the ready-to-print version differs from the web markup.

Category	Description	Examples
Acronym	Common acronym	sRGB, OpenMP
Application	GUI or CLI application	Hugin, Enfuse
Command	Name of a binary in the running text	**convert**, **enfuse**
Common part	Chapter, section, or any other part that appears in both manuals	Response Files[c]
Default	Value as compiled into the **enfuse** binary that belongs to this documentation	⟨1⟩, ⟨*a.tif*⟩
Environment variable	Variable passed to enfuse by the operating system	`PATH`, `TMPDIR`
Filename	Name of a file in the filesystem	*a.tif*
Filename extension	Name of a filename extension with or without dots	*.png*, *tiff*
Fix me!	Section that needs extending or at least some improvement	FIX Explain ME
Literal text	Text that (only) makes sense when typed in exactly as shown	`uint16`
Option	Command-line option given to enfuse	`--verbose`
Optional part	Optional part of a syntax description in square brackets	`--verbose` [=*LEVEL*]
Placehoder	Meta-syntactic variable that stands in for the actual text	*ICC-PROFILE*
Proper name	Name of a person or algorithm	Dijkstra
Restricted note	Annotation that applies only to a particular program, configuration, or operating system	Enfuse.
Sample	Literal text in quotes	'`%`' or '`--help`'
Side note	Non-essential or "geeky" material	*Gory details*
White space	Indispensable white space	`r␣g␣b`

If we must break an identifier like for example `--show-software-compo=nents` or `ENBLEND_OPENCL_PATH` at the end of a line, we indicate the additional character which does not occur when the identifier is written as one word with a '`=`'-character.

Chapter 1

Overview

Enfuse merges overlapping images using the Mertens-KautzVan Reeth exposure fusion algorithm.[1]) This is a quick way for example to blend differently exposed images into a nice output image, without producing intermediate high-dynamic range (HDR) images that are then tone-mapped to a viewable image. This simplified process often works much better than tone-mapping algorithms.

Enfuse can also be used to build extended depth-of-field (DoF) images by blending a focus stack.

The idea is that pixels in the input images are weighted according to qualities such as, for example, proper exposure, good local contrast, or high saturation. These weights determine how much a given pixel will contribute to the final image.

A Burt-Adelson multi-resolution spline blender[2]) is used to combine the images according to the weights. The multi-resolution blending ensures that transitions between regions where different images contribute are difficult to spot.

Enfuse uses up to four criteria to judge the quality of a pixel:

Exposure
: The exposure criteria favors pixels with luminance close to the middle of the range. These pixels are considered better exposed than those with high or low luminance levels.

Saturation
: The saturation criteria favors highly-saturated pixels. Note that saturation is only defined for color pixels.

[1]) Tom Mertens, Jan Kautz, and Frank van Reeth, "Exposure Fusion", Proceedings of the 15th Pacific Conference on Computer Graphics and Applications 2007, pages 382–390.

[2]) Peter J. Burt and Edward H. Adelson, "A Multiresolution Spline With Application to Image Mosaics", ACM Transactions on Graphics, Vol. 2, No. 4, October 1983, pages 217–236.

Local Contrast
: The contrast criteria favors pixels inside a high-contrast neighborhood. Enfuse can use standard deviation, Laplacian magnitude, or a blend of both as local contrast measure.

Local Entropy
: The entropy criteria prefers pixels inside a high-entropy neighborhood. In addition, Enfuse allows the user to mitigate the problem of noisy images when using entropy weighting by setting a black threshold.

See Table 5.1 for the default weights of these criteria.

For the concept of pixel weighting, and details on the different weighting functions, see Chapter 5 on page 48.

Adjust how much importance is given to each criterion by setting the weight parameters on the command line. For example, if you set

```
--exposure-weight=1.0 --saturation-weight=0.5
```

Enfuse will favor well-exposed pixels over highly-saturated pixels when blending the source images. The effect of these parameters on the final result will not always be clear in advance. The quality of the result is subject to your artistic interpretation. Playing with the weights may or may not give a more pleasing result. The authors encourage the users to experiment, perhaps using down-sized or cropped images for speed.

> *Down-sizing (also called "down-sampling") with a good interpolator reduces noise, which might not be desired to judge the image quality of the original-size image. Cropping can offer an alternative, though.*

Enfuse expects but does not require each input image to have an alpha channel. By setting the alpha values of pixels to zero, users can manually remove those pixels from consideration when blending. If an input image lacks an alpha channel, Enfuse will issue a warning and continue assuming all pixels should contribute to the final output. Any alpha value other than zero is interpreted as "this pixel should contribute to the final image".

The input images are processed in the order they appear on the command line. Multi-layer images are processed from the first layer to the last before Enfuse considers the next image on the command line. Consult Section 4.5 on how to change the images' order within multi-layer image files.

Find out more about Enfuse on its SourceForge[3]) web page[4]).

[3]) http://sourceforge.net/
[4]) http://enblend.sourceforge.net/

Chapter 2

Known Limitations

Enfuse has its limitations. Some of them are inherent to the programs proper others are "imported" by using libraries as for example VIGRA[1]). Here are some of the known ones.

- The BigTIFF image format is not supported.
- Total size of *any* – even intermediate – image is limited to 2^{31} pixels, this is two giga-pixels.

[1]) https://ukoethe.github.io/vigra/

Chapter 3

Photographic Workflow[c]

Enfuse and Enblend are parts of a chain of tools to assemble images.

- Enblend combines a series of pictures taken at the same location but in different directions.
- Enfuse merges photos of the same subject at the same location and same direction, but taken with varying exposure parameters.

3.1 Standard, All-In-One Workflow

Figure 3.1 shows where Enfuse and Enblend sit in the tool chain of the standard workflow.

Take Images
: Take *multiple* images to form a panorama, an exposure series, a focus stack, etc.. . .

  There is one exception with Enfuse when a single raw image is converted multiple times to get several – typically differently "exposed" – images.

  Exemplary Benefits:

  - Many pictures taken from the same vantage point but showing different viewing directions. – Panorama
  - Pictures of the same subject exposed with different shutter speeds. – Exposure series
  - Images of the same subject focused at differing distances. – Focus stack

  Remaining Problem: The "overlayed" images may not fit together, that is the overlay regions may not match exactly.

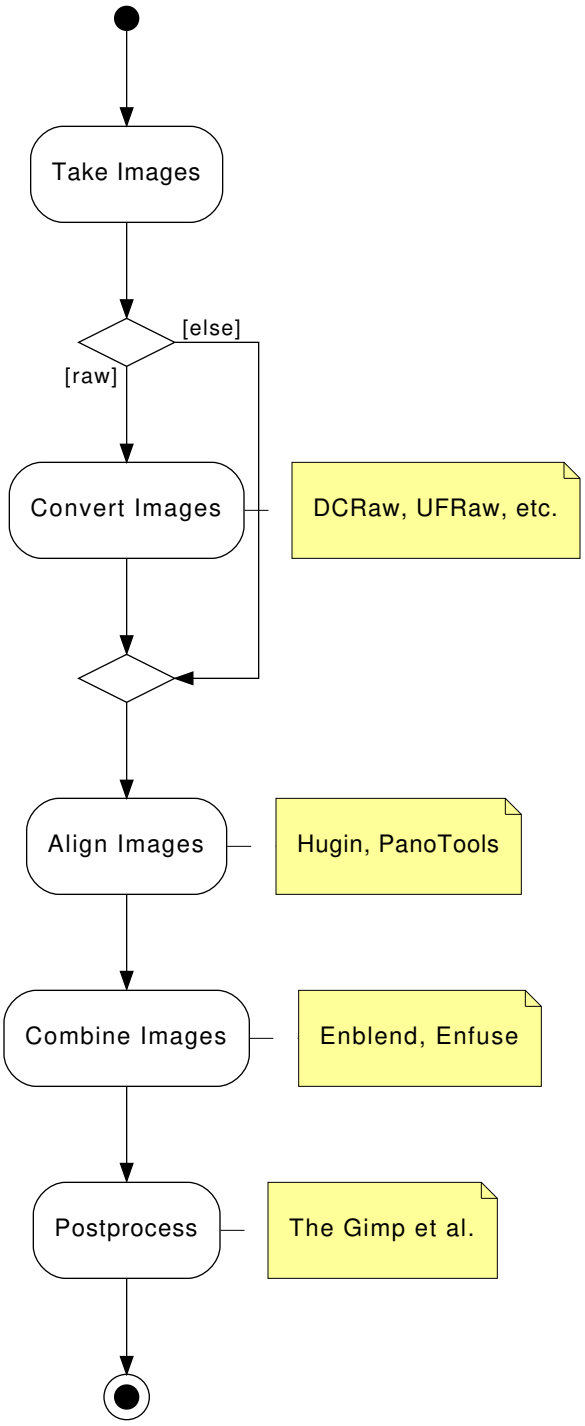


Figure 3.1: Photographic workflow with Enfuse and Enblend.

Convert Images
: Convert the raw data[1]) exploiting the full dynamic range of the camera and capitalize on a high-quality conversion.

Align Images
: Align the images so as to make them match as well as possible.

  Again there is one exception and this is when images naturally align. For example, a series of images taken from a rock solid tripod with a cable release without touching the camera, or images taken with a shift lens, can align without further user intervention.

  This step submits the images to affine transformations.

  If necessary, it rectifies the lens' distortions (e.g. barrel or pincushion), too.

  Sometimes even luminance or color differences between pairs of overlaying images are corrected ("photometric alignment").

  Benefit: The overlay areas of images match as closely as possible given the quality if the input images and the lens model used in the transformation.

  Remaining Problem: The images may still not align perfectly, for example, because of parallax[2]) errors, or blur produced by camera shake.

Combine Images
: Enfuse and Enblend combine the aligned images into one.

  Benefit: The overlay areas become imperceptible for all but the most misaligned images.

  Remaining Problem: Enblend and Enfuse write images with an alpha channel; for more information on alpha channels see Chapter 7 on page 82. Furthermore, the final image rarely is rectangular.

Post-process
: Post-process the combined image with your favorite tool. Often the user will want to crop the image and simultaneously throw away the alpha channel.

View

Print

Enjoy

3.2 External Masks

In the usual workflow Enfuse and Enblend generate the blending and fusing masks according to the command-line options and the input images including

[1]) https://luminous-landscape.com/understanding-raw-files-explained/
[2]) https://en.wikipedia.org/wiki/Parallax

their associated alpha-channels, and then they immediately use these masks for multi-resolution blending or multi-resolution fusing the output image.

Sometimes more control over the masks is wanted. To this end, both applications provide the option pair `--load-masks` and `--save-masks`. See Chapter 4 on page 16, for detailed explanations of both options. With the help of these options the processing can be broken up into two phases:

1. Save masks with `--save-masks`. Generate masks and save them into image files.

   Avoid option `--output` here unless the blended or fused image at this point is wanted.

2. Load possibly modified masks with `--load-masks` from files and then blend or fuse the final image with the help of the loaded masks only.

   Neither application (re-)generates any mask in this phase. The loaded masks completely control the multi-resolution blending or multi-resolution fusing the output image.

In between these two steps the user may apply whatever transformation to the mask files, as long as their geometries and offsets remain the same. Thus the "Combine Images" box of Figure 3.1 becomes three activities as is depicted in Figure 3.2.

To further optimize this kind of workflow, both Enfuse and Enblend stop after mask generation if option `--save-masks` is given, but *no output file* is specified with the `--output` option. This way the time for pyramid generation, blending, fusing, and writing the final image to disk is saved, as well as no output image gets generated.

Note that options `--save-masks` and `--load-masks` cannot be used simultaneously.

3.3 Interacting with Enfuse^C

This section explains how to find out about the inner workings of *your* version of Enfuse without looking at the source code. And it states how to interact with Enfuse besides passing command-line options and image filenames.

3.3.1 Finding Out Details About enfuse

An **enfuse** binary can come in several configurations. The exact name of the binary may vary and it may or may not reflect the "kind of enfuse". Therefore, **enfuse** offers several options that allow the user to query exactly...

- what its exact version number is (see Sec. 3.3.1 on page 9 and option `--version`, page 39),
- what features it does support (see Sec. 3.3.1 on page 10 and options `--version`, page 39 and `--verbose`, page 19),

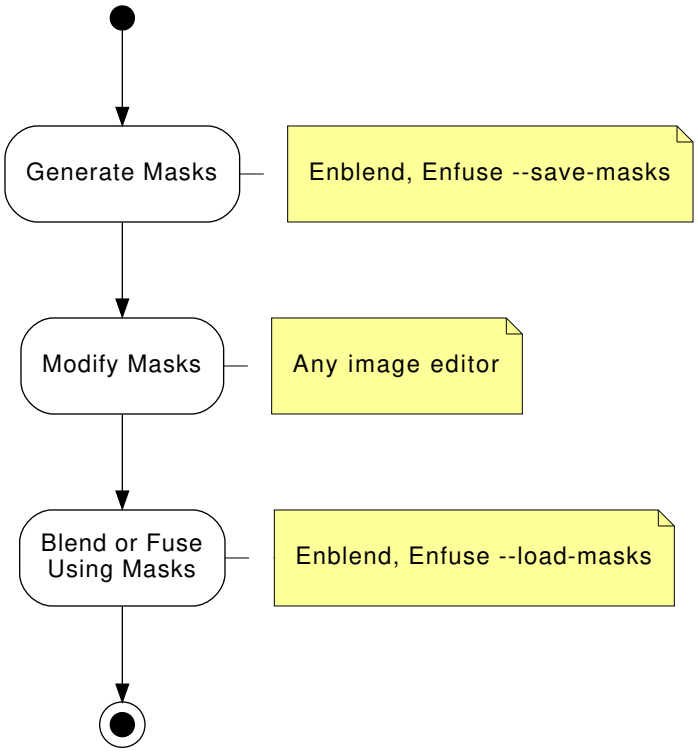


Figure 3.2: Workflow for externally modified masks. The "Blend or Fuse Using Masks" step utilizes the multi-resolution algorithm just as for internal workflow without mask files.

```
$ enfuse --version
enfuse 4.2-02c1f45857b4

Copyright (C) 2004-2009 Andrew Mihal.
Copyright (C) 2009-2015 Christoph Spiel.

License GPLv2+: GNU GPL version 2 or later
<http://www.gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute
it.
There is NO WARRANTY, to the extent permitted by law.

Written by Andrew Mihal, Christoph Spiel and others.
```

Example 3.1: Example output of **enfuse** when called with option `--version`.

- which image formats it can read and write without the need of conversion (see Sec. 3.3.1 on page 11 and option `--show-image-formats`, page 38),
- who built it, (see Sec. 3.3.1 on page 12 and option `--show-signature`, page 38), and finally
- what compiler and libraries were used to do so (see Sec. 3.3.1 on page 12 and option `--show-software-components`, page 38).

The information are explained in detail in the following sections.

Exact Version Number

Example 3.1 shows a possible output of '`enfuse --version`'. The version number at the beginning of the text tells about the *exact* version of the binary. It is the number that can be compared with the version number of this document, which by the way is ⟨4.2⟩. Our slightly cranky markup (see also Notation, page ix) dispels copy-paste errors.

The version indicator consist of a two (major and minor version) or three (major and minor version plus patch-level) numbers separated by dots and an optional hexadecimal identifier. Binaries from the "Development Branch" are assigned two-part version numbers, whereas a three-part version number is reserved for the "Stable Branch" of development. Officially released versions lack the hexadecimal identifier.
Examples:

`4.1.3-0a816672d475`
: Some unreleased version from the "Stable Branch", which finally will lead to version 4.1.3.

`4.1.3`
: Officially released version 4.1.3 in the "Stable Branch".

`4.2-1e4d237daabf`
: Some unreleased version from the "Development Branch", which finally will lead to version 4.2.

`4.2`
: Officially released version 4.2 in the "Development Branch".

Matching the version codes is the only reliably way to pair a given binary with its manual page ("manual page for enblend 4.2-1e4d237daabf") and its documentation. This document mentions the version code for example on its Title Page and the Abstract, page i.

The twelve-digit hexadecimal *ID-CODE* is automatically generated by our source-code versioning system, Mercurial[3]). Use the *ID-CODE* to look up the version on the web in our public source code repository[4]) or, if you have cloned the project to your own workspace, with the command

`hg log --verbose --rev` *ID-CODE*

Compiled-In Features

Adding option `--verbose` to `--version` will reproduce the information described in the previous section plus a list of "extra features". Any unavailable feature in the particular binary queried returns

```
Extra feature: ...: no
```

whereas available features answer "`yes`" followed by a detailed report on the feature and its connection to some library or specific hardware. Example 3.2 on page 11 shows such a report. Remember that your binary may include more or less of the features displayed there.

The '`--version --verbose`' combo is one of the first things test if **enfuse** "suddenly" behaves strangely.

(1.) *"I'm running my* ***enfuse*** *on a multi-core system, but it does not make use of it."*

Check for extra feature OpenMP.

(2.) *"My* ***enfuse*** *complains when I call it with '*`--gpu`*'!"*

Check for extra feature OpenCL.

(3.) *"****enfuse*** *is so slow!"*

Ensure that *neither* feature `mmap-view` *nor* `image-cache` has been compiled in.

[3]) https://www.mercurial-scm.org/
[4]) http://hg.code.sf.net/p/enblend/code

```
$ enfuse --version --verbose
enfuse 4.2-95f1fed2bf2d

Extra feature: dynamic linking support: yes
Extra feature: image cache: no
Extra feature: OpenMP: no

Copyright (C) 2004-2009 Andrew Mihal.
Copyright (C) 2009-2015 Christoph Spiel.

License GPLv2+: GNU GPL version 2 or later <http://www.gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Andrew Mihal, Christoph Spiel and others.
```

Example 3.2: Example output of **enfuse** when called with options `--version` and `--verbose` together.

(4.) *"**enfuse** eats away too much memory! Can I tell it to unload that onto the disk?"*

No, there is no command-line switch for that, but you can use a version with `mmap-view` feature.

(5.) *"My **enfuse** has* OpenMP *enabled. Does it support dynamic adjustment of the number of threads?"*

Under extra feature OpenMP look for "support for dynamic adjustment of the number of threads".

Supported Images Formats

Enfuse can read and write a fixed set of image formats if it was compiled to support them. For example the EXR-format requires special support libraries. Use option `--show-image-formats` to find out

- what image-data formats are supported,
- what filename extensions are recognized, and
- what per-channel depths have been compiled into the **enfuse** binary.

The only three image formats always supported are

- JPEG,

Format	Mask	Profile	Channel Bit-Depth				
			Integral			Floating-Point	
			`uint8`	`uint16`	`uint32`	`float`	`double`
JPEG	–	•	•	–	–	–	–
PNG	•	•	•	•	–	–	–
PNM	?	–	•	•	•	–	–
[V]TIFF	•	•	•	•	•	•	•

Table 3.1: Bit-depths of selected image formats. These are the maximum capabilities of the formats themselves, not Enfuse's. The "Mask"-column indicates whether the format supports an image mask (alpha-channel), see also Chapter 7. Column "Profile" shows whether the image format allows for ICC-profiles to be included; see also Chapter 6.

– PNG, and
– TIFF.

All others are optional. In particular the high-dynamic range (HDR) format OpenEXR only gets compiled if several non-standard libraries are available.

The provided per-channel depths range from just one, namely "8 bits unsigned integral" (`uint8`) up to seven:

– 8 bits unsigned integral, '`uint8`'
– 16 bits unsigned or signed integral, '`uint16`' or '`int16`'
– 32 bits unsigned or signed integral, '`uint32`' or '`int32`'
– 32 bits floating-point, '`float`'
– 64 bits floating-point, '`double`'

Table 3.1 summarizes the channel bit depths of some prominent image formats.

Name Of Builder

During building each **enfuse** is automatically signed to give the users an extra level of confidence that it was constructed by someone that they can trust to get it right. Access this signature with '`--show-signature`' and **enfuse** will print something like

```
Compiled on sgctrl03 by Walter Harriman on Wed, Dec 22 2004, 16:07:22
GMT-7.
```

where machine name, person, and date-time depend on the build.

Compiler And Libraries Used To Build

Sometimes **enfuse** refuses to start or runs into trouble because the libraries supplied to it do not match the ones it was compiled with. Option `--show-software-components` can be helpful to diagnose the problem in such cases,

```
$ enfuse --show-software-components
Compiler
  g++ 4.9.1
  implementing OpenMP standard of 2013-7
  implementing Cilk version 2.0
    without support of "_Cilk_for" keyword

Libraries
  GSL:        1.15
  Little CMS: 2.7.0
  Vigra:      1.10.0
```

Example 3.3: Output of **enfuse** when asked to reveal the compiler that was used to build it along with the libraries it was linked against.

because it shows the version information of Enfuse's most important libraries as they have identified themselves during compile-time.

Furthermore the report reveals the compiler used to build **enfuse** along with the most important compiler extensions like, for example, OpenMP. Example 3.3 shows such a report.

3.3.2 Console Messages

Enfuse is meant to read multiple images, "montage" them together, and finally write a single output image. So, any console messages either serve the entertainment desire of the user or indicate problems.

When **enfuse** is instructed to only show information about its configuration (see Section 4.2.6 on page 38) the text goes to Standard Output. **enfuse** sends error and warning messages to Standard Error. The messages follow a fixed format.

`enfuse:` [*CATEGORY*`:`] *MESSAGE*

where *CATEGORY* is

`error:` A serious problem that sooner or later will lead to a program stop. The result will definitely not be what the user wants – including no output image at all, as **enfuse** deletes corrupted or incomplete output images.

Most messages drop category name '`error`' and plainly write *MESSAGE*:

```
enblend: input image "1.tif" does not have an alpha channel
```

If an '`error`' actually leads to a premature termination of **enfuse**, it returns code 1 to the operating system. On successful termination the return code is 0.

`warning:` A problem that forces **enfuse** to take an alternate execution path or drop some assumptions about the input.

`info:` No problem, just a "nice-to-know" information for the user.

`note:` Not a standalone *CATEGORY*, but an additional explanation that sometimes trails messages in one of the above categories. Errors, warnings and infos tell the causes, notes inform about the actions taken by **enfuse**. Here is an example, where a `warning` gets augmented by a `note`:

```
enblend: warning: input images too small for coarse mask
enblend: note: switching to fine mask
```

`timing:` A measurement of the execution duration and thus a sub-category of `info`.

Sadly, not all messages can be sorted in the category scheme.

Debug Messages: Though debug messages generally come devoid of a specific form the more civilized of them start each line with a plus sign '`+`'. Example:

```
+ checkpoint: leaving channel width alone
```

Foreign Sources: **enfuse** depends on various foreign software components that issue their own messages. We try to catch them and press them in our category scheme, but some of them invariably slip through. The most prominent members of this rogue fraction are the notices of VIGRA[5]) as e.g.

```
enfuse: an exception occurred
enfuse: Precondition violation!
...
```

and LibTIFF[6]):

```
TIFFReadDirectory: Warning, img0001.tif: wrong data type 1
for "RichTIFFIPTC"; tag ignored.
```

"Should-Never-Happen": An internal consistency check fails or a cautious routine detects a problem with its parameters and racks up the digital equivalent of a nervous breakdown. Sometimes these messages end in the word '`Aborted`'.

[5]) https://ukoethe.github.io/vigra/
[6]) http://www.remotesensing.org/libtiff/

```
terminate called after throwing an instance of '...'
what(): ...
Aborted
```

If the installation of **enfuse** is correct, this type of message may warrant a bug report as explained in Appendix B on page 102.

In very unfortunate circumstances Enfuse quits because of a problem, but does not show any message. The output file then either does not exist or it is broken. One known reason are out-of-memory situations, where the process needs additional memory, but cannot allocate it and while terminating needs even more memory so that the operating system wipes it out completely wherever it then happens to be in its execution path.

3.3.3 Environment Variables

A small set of environment variables influences the execution of **enfuse**. All of them depend on **enfuse** having been compiled with certain features. The hint "(direct)" indicates genuine variables in **enfuse**, whereas "(implicit)" denotes variables that control libraries that are linked with **enfuse**.

`CILK_NWORKERS` (implicit) CILK-enabled versions only.

This environment variable works for CilkPlus[7]) as `OMP_NUM_THREADS` (see below) does for OPENMP. It can be helpful for load balancing.

`OMP_DYNAMIC` (implicit) OPENMP-enabled versions only.

Control whether the OPENMP[8]) sub-system should parallelize nested parallel regions. This environment variable will only have an effect is the OPENMP sub-system is capable of dynamic adjustment of the number of threads (see explanations in Section 3.3.1 on page 10).

> *The important hot spots in the source code override the value of `OMP_DYNAMIC`.*

`OMP_NUM_THREADS` (implicit) OPENMP-enabled versions only.

Control – which typically means: reduce – the number of threads under supervision of the OPENMP[9]) sub-system. By default **enfuse** uses as many OPENMP-threads as there are CPUs. Use this variable for example to free some CPUs for other processes than **enfuse**.

`TMPDIR` (direct) '`mmap_view`'-branch only.

`TMPDIR` determines the directory and thus the drive where **enfuse** stores all intermediate images. The best choice follows the same rules as for a swap-drive: prefer the fastest disk with the least load.

---

[7]) https://www.cilkplus.org/
[8]) http://openmp.org/wp/
[9]) http://openmp.org/wp/

Chapter 4

Invocation

Fuse the sequence of images *INPUT*... into a single *IMAGE*.

`enfuse` [*OPTIONS*] [`--output=`*IMAGE*] *INPUT*...

INPUT images are either specified literally or via so-called response files (see Section 4.4). The latter are an alternative to specifying image filenames on the command line. If omitted, the name of the output *IMAGE* defaults to ⟨*a.tif*⟩.

4.1 Input Image Requirements

All input images for Enfuse must comply with the following requirements.

- The images overlap.
- The images agree on their number of bits-per-channel, i.e., their "depth":
  - `uint8`,
  - `uint16`,
  - `float`,
  - etc.

  See option `--depth` below for an explanation of different (output) depths.
- Enfuse understands the images' filename extensions as well as their file formats.

  You can check the supported extensions and formats by calling Enfuse with option `--show-image-formats`.

Moreover, there are some good practices, which are not enforced by the application, but almost certainly deliver superior results.

- Either all files lack an ICC profile, or all images are supplied with the *same* ICC profile.

- If the images' meta-data contains resolution information ("DPI"), it is the same for all pictures.

4.2 Command-Line Options

In this section we group the options as the command-line help

```
$ enfuse --help
```

does and sort them alphabetically within their groups. For an alphabetic list of *all* options consult the Option Index, page 117.

enfuse accepts arguments to any option in uppercase as well as in lowercase letters. For example, '`deflate`', '`Deflate`' and '`DEFLATE`' as arguments to the `--compression` option described below all instruct **enfuse** to use the DEFLATE compression scheme. This manual denotes all arguments in lowercase for consistency.

4.2.1 Common Options[c]

Common options control some overall features of Enfuse. They are called "common" because they are used most often. However, in fact, Enfuse and Enblend do have these options in common.

`--compression=`*COMPRESSION*

Write a compressed output file. The default is not to compress the output image.

Depending on the output file format, Enfuse accepts different values for *COMPRESSION*.

JPEG format.

The compression either is a literal integer or a keyword-option combination.

LEVEL

Set JPEG quality *LEVEL*, where *LEVEL* is an integer that ranges from 0–100.

`jpeg`[`:`*LEVEL*]

Same as above; without the optional argument just switch on standard JPEG compression.

`jpeg-arith`[`:`*LEVEL*]

Switch on arithmetic JPEG compression. With optional argument set the arithmetic compression *LEVEL*, where *LEVEL* is an integer that ranges from 0–100.

TIF format.

Here, *COMPRESSION* is one of the keywords:

`none`
: Do not compress. This is the default.

`deflate`
: Use the DEFLATE compression scheme also called ZIP-in-TIFF. DEFLATE is a lossless data compression algorithm that uses a combination of the LZ77 algorithm and HUFFMAN coding.

`jpeg`[`:`*LEVEL*]
: Use JPEG compression. With optional argument set the compression *LEVEL*, where *LEVEL* is an integer that ranges from 0–100.

`lzw`
: Use LEMPEL-ZIV-WELCH (LZW) adaptive compression scheme. LZW compression is lossless.

`packbits`
: Use PACKBITS compression scheme. PACKBITS is a particular variant of run-length compression; it is lossless.

Any other format.
: Other formats do not accept a *COMPRESSION* setting. However, the underlying VIGRA[1]) library automatically compresses *png*-files with the DEFLATE method. (VIGRA is the image manipulation library upon which Enfuse is based.)

`-l` *LEVELS*
`--levels=`*LEVELS*
: Use at most this many *LEVELS* for pyramid[2]) blending if *LEVELS* is positive, or reduce the maximum number of levels used by $-$*LEVELS* if *LEVELS* is negative; '`auto`' or '`automatic`' restore the default, which is to use the maximum possible number of levels for each overlapping region.

  The number of levels used in a pyramid controls the balance between local and global image features (contrast, saturation, ...) in the blended region. Fewer levels emphasize local features and suppress global ones. The more levels a pyramid has, the more global features will be taken into account.

  > *As a guideline, remember that each new level works on a linear scale twice as large as the previous one. So, the zeroth layer, the original image, obviously defines the image at single-pixel scale, the first level works at two-pixel scale, and generally, the n^{th} level contains image data at 2^n-pixel scale. This is the reason why an image of width $\times$ height pixels cannot be deconstructed into a pyramid of more than*
  >
  > $$\lfloor \log_2(\min(\mathit{width}, \mathit{height})) \rfloor \quad \mathit{levels}.$$

[1]) https://ukoethe.github.io/vigra/

[2]) As DR. DANIEL JACKSON correctly noted, actually, it is not a pyramid: "Ziggaurat, it's a Ziggaurat."

Level	Messages
0	only warnings and errors
1	reading and writing of images
2	mask generation, pyramid, and blending
3	reading of response files, color conversions
4	image sizes, bounding boxes and intersection sizes
5	Enblend only. detailed information on the optimizer runs
6	estimations of required memory in selected processing steps

Table 4.1: Verbosity levels of enfuse; each level includes all messages of the lower levels.

> *If too few levels are used, "halos" around regions of strong local feature variation can show up. On the other hand, if too many levels are used, the image might contain too much global features. Usually, the latter is not a problem, but is highly desired. This is the reason, why the default is to use as many levels as is possible given the size of the overlap regions. Enfuse may still use a smaller number of levels if the geometry of the overlap region demands.*

Positive values of *LEVELS* limit the maximum number of pyramid levels. Depending on the size and geometry of the overlap regions this may or may not influence any pyramid. Negative values of *LEVELS* reduce the number of pyramid levels below the maximum no matter what the actual maximum is and thus always influence all pyramids. Use '`auto`' or '`automatic`' as *LEVELS* to restore the automatic calculation of the maximum number of levels.

The valid range of the absolute value of *LEVELS* is ⟨1⟩ to ⟨29⟩.

`-o` *FILE*
`--output=`*FILE*

Place fused output image in *FILE*. If '`--output`' is omitted, Enfuse writes the resulting image to ⟨*a.tif*⟩.

`-v` [*LEVEL*]
`--verbose`[`=`*LEVEL*]

Without an argument, increase the verbosity of progress reporting. Giving more `--verbose` options will make Enfuse more verbose; see Section 3.3.1 on page 7 for an exemplary output. Directly set a verbosity level with a non-negative integral *LEVEL*. Table 4.1 shows the messages available at a particular *LEVEL*.

The default verbosity level of Enfuse is ⟨1⟩.

4.2.2 Advanced Options[c]

Advanced options control e.g. the channel depth, color model, and the cropping of the output image.

`--blend-colorspace=`*`COLORSPACE`*

Force blending in selected *COLORSPACE*. Given well matched images this option should not change the output image much. However, if Enfuse must blend vastly different colors (as e.g. anti-colors) the resulting image heavily depends on the *COLORSPACE*.

Usually, Enfuse chooses defaults depending on the input images:

- For grayscale or color input images *with* ICC profiles the default is to use CIELUV colorspace.
- Images *without* color profiles and floating-point images are blended in the trivial luminance interval (grayscale) or RGB-color cube by default.

On the order of fast to slow computation, Enfuse supports the following blend colorspaces.

`identity`
`id`
`unit`

Compute blended colors in a naïve way sidestepping any dedicated colorspace.

- Use trivial, 1-dimensional luminance interval (see Eqn. 6.1 on page 76) for grayscale images and
- for color images utilize 3-dimensional RGB-cube (see Eqn. 6.6 on page 77) spanned by the input ICC profile or sRGB if no profiles are present. In the latter case, consider passing option `--fallback-profile`, page 25 to force a different profile than sRGB upon all input images.

`lab`
`cielab`
`lstar`
`l-star`

Blend pixels in the CIEL*A*B* colorspace.

`luv`
`cieluv`

Blend pixels in the CIEL*U*V* colorspace.

`ciecam`
`ciecam02`
`jch`

Blend pixels in the CIECAM02 colorspace.

`-c`
`--ciecam`
: Deprecated. Use '`--blend-colorspace=ciecam`' instead. To emulate the negated option `--no-ciecam` use `--blend-colorspace=identity`.

`-d` *DEPTH*
`--depth=`*DEPTH*
: Force the number of bits per channel and the numeric format of the output image, this is, the *DEPTH*. The number of bits per channel is also known as "channel width" or "channel depth".

  Enfuse always uses a smart way to change the channel depth to assure highest image quality at the expense of memory, whether requantization is implicit because of the output format or explicit through option `--depth`.

  - If the output-channel depth is larger than the input-channel depth of the input images, the input images' channels are widened to the output channel depth immediately after loading, that is, as soon as possible. Enfuse then performs all blending operations at the output-channel depth, thereby preserving minute color details which can appear in the blending areas.
  - If the output-channel depth is smaller than the input-channel depth of the input images, the output image's channels are narrowed only right before it is written to the output *FILE*, that is, as late as possible. Thus the data benefits from the wider input channels for the longest time.

  All *DEPTH* specifications are valid in lowercase as well as uppercase letters. For integer format, use

  `8`
  `uint8`
  : Unsigned 8 bit; range: $0 \ldots 255$

  `int16`
  : Signed 16 bit; range: $-32768 \ldots 32767$

  `16`
  `uint16`
  : Unsigned 16 bit; range: $0 \ldots 65535$

  `int32`
  : Signed 32 bit; range: $-2147483648 \ldots 2147483647$

  `32`
  `uint32`
  : Unsigned 32 bit; range: $0 \ldots 4294967295$

  For floating-point format, use

  `r32`

`real32`
`float`
IEEE754 single precision floating-point, 32 bit wide, 24 bit significant;

- Minimum normalized value: $1.2 \cdot 10^{-38}$
- Epsilon: $1.2 \cdot 10^{-7}$
- Maximum finite value: $3.4 \cdot 10^{38}$

`r64`
`real64`
`double`
IEEE754 double precision floating-point, 64 bit wide, 53 bit significant;

- Minimum normalized value: $2.2 \cdot 10^{-308}$
- Epsilon: $2.2 \cdot 10^{-16}$
- Maximum finite value: $1.8 \cdot 10^{308}$

If the requested *DEPTH* is not supported by the output file format, Enfuse warns and chooses the *DEPTH* that matches best.

Versions with OpenEXR read/write support only.

The OpenEXR data format is treated as IEEE754 float internally. Externally, on disk, OpenEXR data is represented by "half" precision floating-point numbers.

OpenEXR[3] half precision floating-point, 16 bit wide, 10 bit significant;

- Minimum normalized value: $9.3 \cdot 10^{-10}$
- Epsilon: $2.0 \cdot 10^{-3}$
- Maximum finite value: $4.3 \cdot 10^{9}$

□

`-f` *WIDTH*`x`*HEIGHT*[`+x`*XOFFSET*`+y`*YOFFSET*]
Ensure that the minimum "canvas" size of the output image is at least *WIDTH*×*HEIGHT*. Optionally specify the *XOFFSET* and *YOFFSET* of the canvas, too.

This option only is useful when the input images are cropped TIFF files, such as those produced by **nona**.

Note that option `-f` neither rescales the output image, nor shrinks the canvas size below the minimum size occupied by the union of all input images.

`-g` Save alpha channel as "associated". See the TIFF documentation[4] for an explanation.

[3] http://www.openexr.com/about.html#features
[4] http://www.awaresystems.be/imaging/tiff/tifftags/extrasamples.html

The Gimp before version 2.0 and CinePaint (see Appendix A on page 99) exhibit unusual behavior when loading images with unassociated alpha channels. Use option `-g` to work around this problem. With this flag Enfuse will create the output image with the "associated alpha tag" set, even though the image is really unassociated alpha.

`-w` [*MODE*]
`--wrap`[=*MODE*]

Blend around the boundaries of the panorama, or "wrap around".

As this option significantly increases memory usage and computation time only use it, if the panorama will be

- consulted for any kind measurement, this is, all boundaries must match as accurately as possible, or
- printed out and the boundaries glued together, or
- fed into a virtual reality (VR) generator, which creates a seamless environment.

Otherwise, always avoid this option!

With this option Enfuse treats the set of input images (panorama) of width w and height h as an infinite data structure, where each pixel $P(x, y)$ of the input images represents the set of pixels $S_P(x, y)$.

> *Solid-state physicists will be reminded of the* Born-von Kármán *boundary condition*[5]*.*

MODE takes the following values:

`none`
`open`

This is a "no-op"; it has the same effect as not giving '`--wrap`' at all. The set of input images is considered open at its boundaries.

`horizontal`

Wrap around horizontally:

$$S_P(x, y) = \{P(x + mw, y) : m \in Z\}.$$

This is useful for 360° horizontal panoramas as it eliminates the left and right borders.

`vertical`

Wrap around vertically:

$$S_P(x, y) = \{P(x, y + nh) : n \in Z\}.$$

This is useful for 360° vertical panoramas as it eliminates the top and bottom borders.

[5] https://en.wikipedia.org/wiki/Born-von_Karman_boundary_condition

`both`
`horizontal+vertical`
`vertical+horizontal`
Wrap around both horizontally and vertically:

$$S_P(x, y) = \{P(x + mw, y + nh) : m, n \in Z\}.$$

In this mode, both left and right borders, as well as top and bottom borders, are eliminated.

Specifying '`--wrap`' without *MODE* selects horizontal wrapping.

4.2.3 Fusion Options

Fusion options define the proportion to which each input image's pixel contributes to the output image.

`--contrast-weight=`*`WEIGHT`*
Sets the relative *WEIGHT* of high local-contrast pixels.

Valid range: $\langle 0 \rangle \leq WEIGHT \leq \langle 1 \rangle$, default: $\langle 0.0 \rangle$.

See Section 5.4 and 4.2.4, option `--contrast-window-size`, page 29.

`--entropy-weight=`*`WEIGHT`*
Sets the relative *WEIGHT* of high local entropy pixels.

Valid range: $\langle 0 \rangle \leq WEIGHT \leq \langle 1 \rangle$, default: $\langle 0.0 \rangle$.

See Section 4.2.4 and 5.5, options `--entropy-window-size`, page 30 and `--entropy-cutoff`, page 29.

`--exposure-optimum=`*`OPTIMUM`*
Determine at what normalized exposure value the *OPTIMUM* exposure of the input images is. This is, set the position of the maximum of the exposure weight curve. Use this option to fine-tune exposure weighting.

Valid range: $\langle 0 \rangle \leq OPTIMUM \leq \langle 1 \rangle$, default: $\langle 0.5 \rangle$.

`--exposure-weight=`*`WEIGHT`*
Set the relative *WEIGHT* of the "well-exposedness" criterion as defined by the chosen exposure weight function (see option `--exposure-weight-function`, page 34 below). Increasing this weight relative to the others will make well-exposed pixels contribute more to the final output.

Valid range: $\langle 0 \rangle \leq WEIGHT \leq \langle 1 \rangle$, default: $\langle 1.0 \rangle$.

See Section 5.2 on page 50.

`--exposure-width=`*`WIDTH`*
Set the characteristic *WIDTH* (FWHM) of the exposure weight function. Low numbers give less weight to pixels that are far from the user-defined

optimum (option `--exposure-optimum`, page 24) and vice versa. Use this option to fine-tune exposure weighting (See Section 5.2 on page 50).

Valid range: $WIDTH > \langle 0 \rangle$, default: $\langle 0.2 \rangle$.

`--hard-mask`
: Force hard blend masks on the finest scale. This is the opposite flag of option `--soft-mask`, page 25.

  This blending mode avoids averaging of fine details (only) at the expense of increasing the noise. However it considerably improves the sharpness of focus stacks. Blending with hard masks has only proven useful with focus stacks.

  See also Section 4.2.3 and options `--contrast-weight`, page 24 as well as `--contrast-window-size`, page 29 above.

`--saturation-weight=`*`WEIGHT`*
: Set the relative *WEIGHT* of high-saturation pixels. Increasing this weight makes pixels with high saturation contribute more to the final output.

  Valid range: $\langle 0 \rangle \leq WEIGHT \leq \langle 1 \rangle$, default: $\langle 0.2 \rangle$.

  Saturation weighting is only defined for color images; see Section 5.3.

`--soft-mask`
: Consider all masks when fusing. This is the default.

4.2.4 Expert Options

Control inner workings of Enfuse and the reading/writing of weight masks.

`--fallback-profile=`*`PROFILE-FILENAME`*
: Use the ICC profile in *PROFILE-FILENAME* instead of the default sRGB. This option only is effective if the input images come *without* color profiles *and* blending is not performed in the trivial luminance interval or RGB-cube.

  Compare option `--blend-colorspace`, page 20 and Chapter 6.3 on page 79 on color profiles.

`--layer-selector=`*`ALGORITHM`*
: Override the standard layer selector algorithm '⟨`all-layers`⟩'.

  Enfuse offers the following algorithms:

  `all-layers`
  : Select all layers in all images.

  `first-layer`
  : Select only first layer in each multi-layer image. For single-layer images this is the same as '`all-layers`'.

`last-layer`
: Select only last layer in each multi-layer image. For single-layer images this is the same as '`all-layers`'.

`largest-layer`
: Select largest layer in each multi-layer image, where the "largeness", this is the size is defined by the product of the layer width and its height. The channel width of the layer is ignored. For single-layer images this is the same as '`all-layers`'.

`no-layer`
: Do not select any layer in any image.

  This algorithm is useful to temporarily exclude some images in response files.

`--load-masks` (1st form)
`--load-masks=`*`SOFT-MASK-TEMPLATE`* (2nd form)
`--load-masks=`*`SOFT-MASK-TEMPLATE`*`:`*`HARD-MASK-TEMPLATE`* (3rd form)

Load masks from images instead of computing them.

The masks must be grayscale images. All image formats understood by Enfuse (see option `--show-image-formats`, page 38) are viable mask file formats, though those with floating-point pixels for example TIFF or VIFF are suited best.

1st form: Load all soft-weight masks from files that were previously saved with option `--save-masks`, page 27. *HARD-MASK-TEMPLATE* is effective only when loading hard masks (see option `--hard-mask`, page 25). The respective defaults are ⟨`softmask-%n.tif`⟩ and ⟨`hardmask-%n.tif`⟩.

In the 2nd form *SOFT-MASK-TEMPLATE* defines the names of the soft-mask files.

In the 3rd form *HARD-MASK-TEMPLATE* additionally defines the names of the hard-mask files. See option `--save-masks`, page 27 below for the description of mask templates.

Options `--load-masks` and `--save-masks` are mutually exclusive.

`--parameter=`*`KEY`*`[=`*`VALUE`*`][:...]`

Set a *KEY-VALUE* pair, where *VALUE* is optional. This option is cumulative. Separate multiple pairs with the usual numeric delimiters.

This option has the negated form '`--no-parameter`', which takes one or more *KEY*s and removes them from the list of defined parameters. The special key '`*`' deletes all parameters at once.

Parameters allow the developers to change the internal workings of Enfuse without the need to recompile or relink.

Daniel Jackson: *I just hope we won't regret giving them those gate addresses.*

Jack O'Neill: *I don't think we will, first one being a black hole and all. They get progressively darker after that.*

`--save-masks` (1st form)
`--save-masks=`*`SOFT-MASK-TEMPLATE`* (2nd form)
`--save-masks=`*`SOFT-MASK-TEMPLATE`*`:`*`HARD-MASK-TEMPLATE`* (3rd form)

Save the generated weight masks to image files.

1st form: Save all soft-weight masks in files. If option `--hard-mask`, page 25 is effective also save the hard masks. The defaults are ⟨`softmask-%n.tif`⟩ and ⟨`hardmask-%n.tif`⟩.

In the 2nd form *SOFT-MASK-TEMPLATE* defines the names of the soft-mask files.

In the 3rd form *HARD-MASK-TEMPLATE* additionally defines the names of the hard-mask files.

Enfuse will stop after saving all masks unless option `--output`, page 19 is given, too. With both options given, this is, '`--save-masks`' and '`--output`', Enfuse saves all masks and then proceeds to fuse the output image.

Both *SOFT-MASK-TEMPLATE* and *HARD-MASK-TEMPLATE* define templates that are expanded for each mask file. In a template a percent sign ('`%`') introduces a variable part. All other characters are copied literally. Lowercase letters refer to the name of the respective input file, whereas uppercase ones refer to the name of the output file. Table 4.2 on page 28 lists all variables.

A fancy mask filename template could look like

```
%D/mask-%02n-%f.tif
```

It puts the mask files into the same directory as the output file ('`%D`'), generates a two-digit index ('`%02n`') to keep the mask files nicely sorted, and decorates the mask filename with the name of the associated input file ('`%f`') for easy recognition.

The masks carry the totaled images' weights. They consist of single-channel, this is grayscale, floating-point data and thus preferably are saved in floating-point form. Enfuse defaults to floating-point TIFF.

Options `--load-masks` and `--save-masks` are mutually exclusive.

4.2.5 Expert Fusion Options

Expert fusion options control details of contrast-weight algorithms and they set ultimate cutoffs for entropy and exposure fusion.

Format	Interpretation
`%%`	Produces a literal '`%`'-sign.
`%i`	Expands to the index of the mask file starting at zero. '`%i`' allows for setting a pad character or a width specification: `% PAD WIDTH i` *PAD* is either '`0`' or any punctuation character; the default pad character is '`0`'. *WIDTH* is an integer specifying the minimum width of the number. The default is the smallest width given the number of input images, this is 1 for 2–9 images, 2 for 10–99 images, 3 for 100–999 images, and so on. Examples: '`%i`', '`%02i`', or '`%_4i`'.
`%n`	Expands to the number of the mask file starting at one. Otherwise it behaves identically to '`%i`', including pad character and width specification.
`%p`	This is the full name (path, filename, and extension) of the input file associated with the mask. Example: If the input file is called */home/luser/snap/img.jpg*, '`%p`' expands to */home/luser/snap/img.jpg*, or shorter: '`%p`' ↦ */home/luser/snap/img.jpg*.
`%P`	This is the full name of the output file.
`%d`	Is replaced with the directory part of the associated input file. Example (cont.): '`%d`' ↦ */home/luser/snap*.
`%D`	Is replaced with the directory part of the output file.
`%b`	Is replaced with the non-directory part (often called "basename") of the associated input file. Example (cont.): '`%b`' ↦ *img.jpg*.
`%B`	Is replaced with the non-directory part of the output file.
`%f`	Is replaced with the filename without path and extension of the associated input file. Example (cont.): '`%f`' ↦ *img*.
`%F`	Is replaced with the filename without path and extension of the output file.
`%e`	Is replaced with the extension (including the leading dot) of the associated input file. Example (cont.): '`%e`' ↦ *.jpg*.
`%E`	Is replaced with the extension of the output file.

Table 4.2: Special format characters to control the generation of mask filenames. Uppercase letters refer to the output filename and lowercase ones to the input files.

`--contrast-edge-scale=`*EDGE-SCALE*`[:`*LCE-SCALE*`:`*LCE-FACTOR*`]`

A non-zero value for *EDGE-SCALE* switches on the LAPLACIAN-of-GAUSSIAN (LoG) edge detection algorithm. *EDGE-SCALE* is the radius of the GAUSSIAN used in the search for edges. Default: ⟨0.0⟩ pixels.

A positive *LCE-SCALE* turns on local contrast enhancement (LCE) before the LoG edge detection. *LCE-SCALE* is the radius of the GAUSSIAN used in the enhancement step, *LCE-FACTOR* is the weight factor ("strength"). Enfuse calculates the *enhanced* values of the *original* ones with

```
enhanced :=
    (1 + LCE-FACTOR) * original -
    LCE-FACTOR * GaussianSmooth(original,
                                LCE-SCALE).
```

LCE-SCALE defaults to ⟨0.0⟩ pixels and *LCE-FACTOR* defaults to ⟨0.0⟩. Append '`%`' to *LCE-SCALE* to specify the radius as a percentage of *EDGE-SCALE*. Append '`%`' to *LCE-FACTOR* to specify the weight as a percentage.

`--contrast-min-curvature=`*CURVATURE*

Define the minimum *CURVATURE* for the LoG edge detection. Default: ⟨0⟩. Append a '`%`' to specify the minimum curvature relative to maximum pixel value in the source image (for example 255 or 65535).

A positive value makes Enfuse use the local contrast data (controlled with option `--contrast-window-size`, page 29) for curvatures less than *CURVATURE* and LoG data for values above it.

A negative value truncates all curvatures less than $-$*CURVATURE* to zero. Values above *CURVATURE* are left unchanged. This effectively suppresses weak edges.

`--contrast-window-size=`*SIZE*

Set the window *SIZE* for local contrast analysis. The window will be a square of *SIZE* × *SIZE* pixels. If given an even *SIZE*, Enfuse will automatically use the next odd number.

For contrast analysis *SIZE* values larger than 5 pixels might result in a blurry composite image. Values of 3 and 5 pixels have given good results on focus stacks.

Valid range: *SIZE* ≥ ⟨3⟩, default: ⟨5⟩ pixels.

See also Section 4.2.3 on page 24, options `--contrast-weight`, page 24 and `--hard-mask`, page 25.

`--entropy-cutoff=`*LOWER-CUTOFF*`[:`*UPPER-CUTOFF*`]`

Define a cutoff function Y' for the normalized luminance Y by *LOWER-CUTOFF* and *UPPER-CUTOFF*, which gets applied (only) before the local-entropy calculation. *LOWER-CUTOFF* is the value below which pixels are mapped to pure black when calculating the local entropy of the

pixel's surroundings. Optionally also define the *UPPER-CUTOFF* value above which pixels are mapped to pure white.

$$Y' = \begin{cases} 0 & \text{for } Y \leq \textit{LOWER-CUTOFF}, \\ 1 & \text{for } Y \geq \textit{UPPER-CUTOFF} \text{ and} \\ Y & \text{otherwise.} \end{cases} \tag{4.1}$$

Also see Section 5.5 on page 73 for an explanation of local entropy. Figure 4.1 on page 31 shows an example for the luminance mapping.

Note that the entropy cutoff does not apply directly to the local-entropy H of a pixel or its weight w_H, but the luminance image that get fed into the local-entropy weight calculation. However, assigning *constant* values to extreme shadows or highlights in general decreases their local entropy, thereby reducing the pixels' weights.

For color images *LOWER-CUTOFF* and *UPPER-CUTOFF* are applied separately and independently to each channel.

Append a '`%`'-sign to specify the cutoff relative to maximum pixel value in the source image (for example 255 or 65535). Negative *UPPER-CUTOFF* values indicate the maximum minus the absolute *UPPER-CUTOFF* value; the same holds for negative percentages.

Defaults: $\langle 0\% \rangle$ for *LOWER-CUTOFF* and $\langle 100\% \rangle$ for *UPPER-CUTOFF*, that is, all pixels' values are taken into account.

> *Note that a high LOWER-CUTOFF value lightens the resulting image, as dark and presumably noisy pixels are averaged with* equal *weights. With LOWER-CUTOFF = 0, the default, on the other hand, "noise" might be interpreted as high entropy and the noisy pixels get a high weight, which in turn renders the resulting image darker. Analogously, a low UPPER-CUTOFF darkens the output image.*

`--entropy-window-size=`*SIZE*

Window *SIZE* for local entropy analysis. The window will be a square of *SIZE* × *SIZE* pixels.

In the entropy calculation *SIZE* values of 3 to 7 yield an acceptable compromise of the locality of the information and the significance of the local entropy value itself.

Valid range: *SIZE* ≥ $\langle 3 \rangle$, default: $\langle 3 \rangle$ pixels.

If given an even *SIZE* Enfuse will automatically use the next-larger odd number.

`--exposure-cutoff=`*LOWER-CUTOFF*[`:`*UPPER-CUTOFF*[`:`*LOWER-PROJECTOR*`:`*UPPER-PROJECTOR*]]

Define an exposure-cutoff function by the luminances *LOWER-CUTOFF*

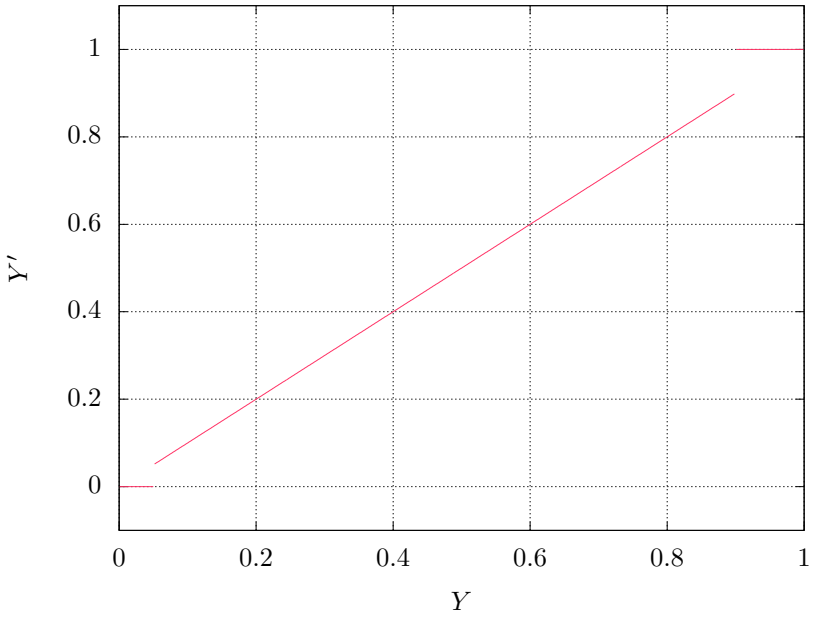


Figure 4.1: Modified lightness Y', Eqn. 4.1, for *LOWER-CUTOFF* = 5% and *UPPER-CUTOFF* = 90%, which are rather extreme values.

```
$ convert IMAGE \
          \( +clone -threshold LOWER-CUTOFF \) \
          -compose copy_opacity -composite \
          MASKED-IMAGE

$ convert IMAGE \
          \( \
              \( IMAGE -threshold LOWER-CUTOFF \) \
              \( IMAGE -threshold UPPER-CUTOFF -negate \) \
              -compose multiply -composite \
          \) \
          -compose copy_opacity -composite \
          MASKED-IMAGE
```

Example 4.1: Using ImageMagick for exposure cutoff operations. The first example only applies a lower cutoff, whereas the second one applies both a lower and an upper cutoff to the images.

and *UPPER-CUTOFF*. Pixels below the lower or above the upper cutoff get a weight of exactly zero irrespective of the active exposure-weight function.

For color images the values of *LOWER-CUTOFF* and *UPPER-CUTOFF* refer to the gray-scale projection as selected with *LOWER-PROJECTOR* and *UPPER-PROJECTOR*. This is similar to option `--gray-projector`.

Append a '`%`'-sign to specify the cutoff relative to maximum pixel value in the source image (for example 255 or 65535). Negative *UPPER-CUTOFF* values indicate the maximum minus the absolute *UPPER-CUTOFF* value; the same holds for negative percentages.

> *The impact of this option is similar, but not identical to transforming* all *input images with ImageMagick's*[6]*)* ***convert*** *(see Appendix A on page 99) prior to fusing with the commands demonstrated in Example 4.1.*
> *Transforming some or all input images as shown in the example gives the user more flexibility because the thresholds can be chosen for each image individually.*

The option allows to specify projection operators as in option `--gray-projector` for the *LOWER-CUTOFF* and *UPPER-CUTOFF* thresholds.

This option can be helpful if the user wants to exclude underexposed or overexposed pixels from the fusing process in *all* of the input images. The values of *LOWER-CUTOFF* and *UPPER-CUTOFF* as well as the

[6]) https://www.imagemagick.org/script/index.php

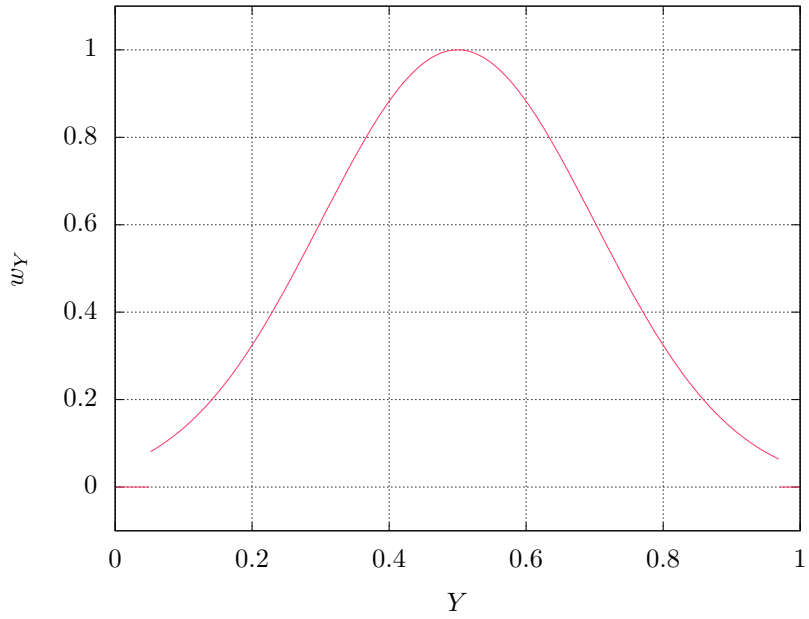


Figure 4.2: Exposure weight w_Y after an exposure-cutoff of *LOWER-CUTOFF* = 5% and *UPPER-CUTOFF* = 97% was applied to a GAUSSIAN with the *optimum* = 0.5 and *width* = 0.2.

gray-scale projector determine which pixels are considered "underexposed" or "overexposed". As any change of the exposure-weight curve this option changes the brightness of the resulting image: increasing *LOWER-CUTOFF* lightens the final image and lowering *UPPER-CUTOFF* darkens it.

Defaults: ⟨0%⟩ for *LOWER-CUTOFF* and ⟨100%⟩ for *UPPER-CUTOFF*, that is, all pixels' values are weighted according to the "uncut" exposure-weight curve.

Figure 4.2 shows an example.

The gray-scale projectors *LOWER-PROJECTOR* and *UPPER-PROJECTOR* default to '⟨`anti-value`⟩' and '⟨`value`⟩', which are usually the best choices for effective cutoff operations on the respective ends.

Note that the application of the respective cutoffs is completely independent of the actual shape of the exposure weight function.

If a set of images stubbornly refuses to "react" to this option, look at their histograms to verify the cutoff actually falls into populated ranges of the histograms. In the absence of an image manipulation program like

Task	Cutoff Setting	Effect
Suppress some noise.	`--exposure-cutoff=5%`	The percentage makes the cutoff specification channel-width agnostic.
Shave off pure white pixels.	`--exposure-cutoff=0:-1`	This cutoff specification only works for integral pixels, but it will set the weight of the very brightest pixels to zero.
Shave off bright white pixels.	`--exposure-cutoff=0:-1%`	Here we exclude the brightest 1% of pixels form the exposure fusion no matter whether the image is encoded with integers or floating-point numbers.
Suppress some noise and shave off pure white pixels.	`--exposure-cutoff=5%:-1`	Combine the effects of lower and upper cutoff, while mixing relative and absolute specifications.

Table 4.3: Some possible exposure-cutoff settings and their effects on the exposure weights.

The Gimp[7]), ImageMagick's[8]) can be used to generate histograms[9]), like, for example,

```
$ convert -define histogram:unique-colors=false \
            IMAGE histogram:- | \
        display
```

The syntax of this option is flexible enough to combine ease of use and precision, as Table 4.3 demonstrates.

`--exposure-weight-function=`*`WEIGHT-FUNCTION`* (1st form)
`--exposure-weight-function=`*`SHARED-OBJECT`*`:`*`SYMBOL`*`[:`*`ARGUMENT`*`[:...]]` (2nd form)

1st form: override the default exposure weight function (⟨`gaussian`⟩) and instead use one of the weight-functions in Table 4.4 on page 36.

Versions with dynamic-linking support only.

2nd form: dynamically load *SHARED-OBJECT* and use *SYMBOL* as user-defined exposure weight function. Optionally pass the user-defined function *ARGUMENT*s.

[7]) http://www.gimp.org/
[8]) https://www.imagemagick.org/script/index.php
[9]) https://www.imagemagick.org/Usage/files/#histogram

Depending on the operating system environment, a "shared object" is sometimes also called a "dynamic library".

□

In Table 4.4 the variable w_{exp} denotes the exposure weight and z represents the normalized luminance Y linearly transformed by the exposure optimum Y_{opt} (option `--exposure-optimum`) and *width* (option `--exposure-width`) according to the linear transform

$$z = \frac{Y - Y_{\text{opt}}}{width}. \tag{4.2}$$

Internally Enfuse uses a rescaled *width* that gives all weight functions the same full width at half of the maximum (FWHM), also see Figure 5.6. This means for the user that changing the exposure function neither changes the optimum exposure nor the width.

For a detailed explanation of all the weight functions Section 5.2 on page 50.

If this option is given more than once, the last instance wins.

`--gray-projector=`*`PROJECTOR`*

Use gray projector *PROJECTOR* for conversion of RGB images to grayscale:

$$(R, G, B) \to Y.$$

In this version of Enfuse, the option is effective for exposure weighting and local contrast weighting and *PROJECTOR* defaults to '⟨`average`⟩'.

Valid values for *PROJECTOR* are:

`anti-value`

Do the opposite of the '`value`' projector: take the minimum of all color channels.

$$Y = \min(R, G, B)$$

This projector can be useful when exposure weighing while employing a lower cutoff (see option `--exposure-cutoff`) to reduce the noise in the fused image.

`average`

Average red, green, and blue channel with equal weights. This is the default, and it often is a good projector for gamma = 1 data.

$$Y = \frac{R + G + B}{3}$$

`gauss`
`gaussian`
: The original exposure weight function of Enfuse and the only one up to version 4.1.

$$w_{\text{exp}} = \exp\left(-z^2/2\right) \tag{4.3}$$

`lorentz`
`lorentzian`
: Lorentz curve.

$$w_{\text{exp}} = \frac{1}{1 + z^2/2} \tag{4.4}$$

`halfsine`
`half-sine`
: Upper half-wave of a sine; exactly zero outside.

$$w_{\text{exp}} = \begin{cases} \cos(z) & \text{if } |z| \leq \pi/2 \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

`fullsine`
`full-sine`
: Full sine-wave shifted upwards by one to give all positive weights; exactly zero outside.

$$w_{\text{exp}} = \begin{cases} (1 + \cos(z))/2 & \text{if } |z| \leq \pi \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

`bisquare`
`bi-square`
: Quartic function.

$$w_{\text{exp}} = \begin{cases} 1 - z^4 & \text{if } |z| \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

Table 4.4: Predefined exposure weight functions. For a graphical comparison see Figure 5.6.

`channel-mixer:`*RED-WEIGHT*`:`*GREEN-WEIGHT*`:`*BLUE-WEIGHT*

Weight the channels as given.

$$\begin{aligned} Y = {} & \textit{RED-WEIGHT} \times R + \\ & \textit{GREEN-WEIGHT} \times G + \\ & \textit{BLUE-WEIGHT} \times B \end{aligned}$$

The weights are automatically normalized to one, so

```
--gray-projector=channel-mixer:0.25:0.5:0.25
--gray-projector=channel-mixer:1:2:1
--gray-projector=channel-mixer:25:50:25
```

all define the same mixer configuration.

The three weights *RED-WEIGHT*, *GREEN-WEIGHT*, and *BLUE-WEIGHT* define the relative weight of the respective color channel. The sum of all weights is normalized to one.

`l-star`

Use the L-channel of the L*a*b*-conversion of the image as its gray-scale representation. This is a useful projector for gamma = 1 data. It reveals minute contrast variations even in the shadows and the highlights. This projector is computationally expensive. Compare with '`pl-star`', which is intended for gamma-corrected images.

See Wikipedia[10]) for a detailed description of the LAB color space.

`lightness`

Compute the lightness of each RGB pixel as in an Hue-Saturation-Lightness (HSL) conversion of the image.

$$Y = \frac{\max(R, G, B) + \min(R, G, B)}{2}$$

`luminance`

Use the weighted average of the RGB pixel's channels as defined by CIE ("Commission Internationale de l'Éclairage") and the JPEG standard.

$$Y = 0.30 \times R + 0.59 \times G + 0.11 \times B$$

`pl-star`

Use the L-channel of the L*a*b*-conversion of the image as its gray-scale representation. This is a useful projector for gamma-corrected data. It reveals minute contrast variations even in the shadows and the highlights. This projector is computationally expensive. Compare with '`l-star`', which is intended for gamma = 1 images.

See Wikipedia[11]) for a detailed description of the LAB color space.

[10]) https://en.wikipedia.org/wiki/Lab_color_space
[11]) https://en.wikipedia.org/wiki/Lab_color_space

`value`
: Take the Value-channel of the Hue-Saturation-Value (HSV) conversion of the image.

$$Y = \max(R, G, B)$$

4.2.6 Information Options[c]

`-h`
`--help`
: Print information on the command-line syntax and all available options, giving a boiled-down version of this manual.

`--show-globbing-algorithms`
: Show all globbing algorithms.

  Depending on the build-time configuration and the operating system the binary may support different globbing algorithms. See Section 4.4.3 on page 43.

`--show-image-formats`
: Show all recognized image formats, their filename extensions and the supported per-channel depths.

  Depending on the build-time configuration and the operating system, the binary supports different image formats, typically: BMP, EXR, GIF, HDR, JPEG, PNG, PNM, SUN, TIFF, and VIFF and recognizes different image-filename extensions, again typically: *bmp*, *exr*, *gif*, *hdr*, *jpeg*, *jpg*, *pbm*, *pgm*, *png*, *pnm*, *ppm*, *ras*, *tif*, *tiff*, and *xv*.

  The maximum number of different per-channel depths any **enfuse** provides is seven:

  - 8 bits unsigned integral, '`uint8`'
  - 16 bits unsigned or signed integral, '`uint16`' or '`int16`'
  - 32 bits unsigned or signed integral, '`uint32`' or '`int32`'
  - 32 bits floating-point, '`float`'
  - 64 bits floating-point, '`double`'

`--show-signature`
: Show the user name of the person who compiled the binary, when the binary was compiled, and on which machine this was done.

  This information can be helpful to ensure the binary was created by a trustworthy builder.

`--show-software-components`
: Show the name and version of the compiler that built Enfuse followed by the versions of all important libraries against which Enfuse was compiled and linked.

Technically, the version information is taken from header files, thus it is independent of the dynamic-library environment the binary runs within. The library versions printed here can help to reveal version mismatches with respect to the actual dynamic libraries available to the binary.

`-V`
`--version`
Output information on the binary's version.

Team this option with `--verbose`, page 19 to show configuration details, like the extra features that may have been compiled in. For details consult Section 3.3.1 on page 9.

4.3 Option Delimiters[C]

Enfuse and Enblend allow the arguments supplied to the programs' options to be separated by different separators. The online documentation and this manual, however, exclusively use the colon ':' in every syntax definition and in all examples.

4.3.1 Numeric Arguments

Valid delimiters are the semicolon ';', the colon ':', and the slash '/'. All delimiters may be mixed within any option that takes numeric arguments.

Examples using some Enfuse options:

`--contrast-edge-scale=0.667:6.67:3.5`
Separate all arguments with colons.

`--contrast-edge-scale=0.667;6.67;3.5`
Use semi-colons.

`--contrast-edge-scale=0.667;6.67/3.5`
Mix semicolon and slash in weird ways.

`--entropy-cutoff=3%/99%`
All delimiters also work in conjunction with percentages.

`--gray-projector=channel-mixer:3/6/1`
Separate arguments with a colon and two slashes.

`--gray-projector=channel-mixer/30;60:10`
Go wild and Enfuse will understand.

4.3.2 Filename Arguments

Here, the accepted delimiters are comma ‘`,`’, semicolon ‘`;`’, and colon ‘`:`’. Again, all delimiters may be mixed within any option that has filename arguments.

Examples:

`--save-masks=soft-mask-%03i.tif:hard-mask-03%i.tif`
: Separate all arguments with colons.

`--save-masks=%d/soft-%n.tif,%d/hard-%n.tif`
: Use a comma.

4.4 Response Files[C]

A response file contains names of images or other response filenames. Introduce response file names at the command line or in a response file with an ⟨`@`⟩ character.

Enfuse and Enblend process the list *INPUT* strictly from left to right, expanding response files in depth-first order. Multi-layer files are processed from first layer to the last. The following examples only show Enblend, but Enfuse works exactly the same.

Solely image filenames.

Example:

```
enblend image-1.tif image-2.tif image-3.tif
```

The ultimate order in which the images are processed is: *image-1.tif*, *image-2.tif*, *image-3.tif*.

Single response file.

Example:

```
enblend @list
```

where file *list* contains

```
img1.exr
img2.exr
img3.exr
img4.exr
```

Ultimate order: *img1.exr*, *img2.exr*, *img3.exr*, *img4.exr*.

Mixed literal names and response files.

Example:

```
enblend @master.list image-09.png image-10.png
```

response-file	::=	*line**
line	::=	(*comment* | *file-spec*) [‘`\r`’] ‘`\n`’
comment	::=	*space** ‘`#`’ *text*
file-spec	::=	*space** ‘`@`’ *filename space**
space	::=	‘␣’ | ‘`\t`’

where *text* is an arbitrary string and *filename* is any filename.

Table 4.5: EBNF definition of the grammar of response files.

where file *master.list* comprises of

```
image-01.png
@first.list
image-04.png
@second.list
image-08.png
```

first.list is

```
image-02.png
image-03.png
```

and *second.list* contains

```
image-05.png
image-06.png
image-07.png
```

Ultimate order: *image-01.png*, *image-02.png*, *image-03.png*, *image-04.png*, *image-05.png*, *image-06.png*, *image-07.png*, *image-08.png*, *image-09.png*, *image-10.png*,

4.4.1 Response File Format

Response files contain one filename per line. Blank lines or lines beginning with a ⟨`#`⟩ sign are ignored; the latter can serve as comments. Filenames that begin with a ⟨`@`⟩ character denote other response files. Table 4.5 states a formal grammar of response files in EBNF[12]).

In a response file relative filenames are used relative the response file itself, not relative to the current-working directory of the application.

The above grammar might surprise the user in the some ways.

[12]) https://en.wikipedia.org/wiki/Ebnf

White-space trimmed at both line ends
: For convenience, white-space at the beginning and at the end of each line is ignored. However, this implies that response files cannot represent filenames that start or end with white-space, as there is no quoting syntax. Filenames with embedded white-space cause no problems, though.

Only whole-line comments
: Comments in response files always occupy a complete line. There are no "line-ending comments". Thus, in

```
#exposure series
img-0.33ev.tif #"middle" EV
img-1.33ev.tif
img+0.67ev.tif
```

only the first line contains a comment, whereas the second line includes none. Rather, it refers to a file called

```
img-0.33ev.tif #"middle" EV
```

Image filenames cannot start with ⟨@⟩
: A ⟨@⟩ sign invariably introduces a response file, even if the filename's extension hints towards an image.

If Enfuse or Enblend do not recognize a response file, they will skip the file and issue a warning. To force a file being recognized as a response file add one of the following syntactic comments to the *first* line of the file.

```
response-file: true
enblend-response-file: true
enfuse-response-file: true
```

Finally, Example 4.2 shows a complete response file.

4.4.2 Syntactic Comments

Comments that follow the format described in Table 4.6 are treated as instructions how to interpret the rest of the response file. A syntactic comment is effective immediately and its effect persists to the end of the response file, unless another syntactic comment undoes it.

Unknown syntactic comments are silently ignored.

A special index for syntactic comments, page 115 lists them in alphabetic order.

```
# 4\pi panorama!

# These pictures were taken with the panorama head.
@ round-shots.list

# Freehand sky shot.
zenith.tif

# "Legs, will you go away?" images.
nadir-2.tif
nadir-5.tif
nadir.tif
```

Example 4.2: Example of a complete response file.

syntactic-comment	::=	*space** '`#`' *space** *key* *space** '`:`' *space** *value*
key	::=	('`A`'...'`Z`' | '`a`'...'`z`' | '`-`')+

where *value* is an arbitrary string.

Table 4.6: EBNF definition of the grammar of syntactic comments in response files.

4.4.3 Globbing Algorithms

The three equivalent syntactic keys

- `glob`,
- `globbing`, or
- `filename-globbing`

control the algorithm that Enfuse or Enblend use to glob filenames in response files.

All versions of Enfuse and Enblend support at least two algorithms: `literal`, which is the default, and `wildcard`. See Table 4.7 for a list of all possible globbing algorithms. To find out about the algorithms in your version of Enfuse or Enblend use option `--show-globbing-algorithms`.

Example 4.3 gives an example of how to control filename-globbing in a response file.

4.4.4 Default Layer Selection

The key `layer-selector` provides the same functionality as does the command-line option `--layer-selector`, but on a per response-file basis. See Section 4.2.1.

`literal`
: Do not glob. Interpret all filenames in response files as literals. This is the default.

  Please remember that white-space at both ends of a line in a response file *always* gets discarded.

`wildcard`
: Glob using the wildcard characters '`?`', '`*`', '`[`', and '`]`'.

  The WIN32 implementation only globs the filename part of a path, whereas all other implementations perform wildcard expansion in *all* path components. Also see glob(7)[a].

`none`
: Alias for `literal`.

`shell`
: The `shell` globbing algorithm works as `literal` does. In addition, it interprets the wildcard characters '{', '`@`', and '`~`'. This makes the expansion process behave more like common UN*X shells.

`sh`
: Alias for `shell`.

[a] http://man7.org/linux/man-pages/man7/glob.7.html

Table 4.7: Globbing algorithms for the use in response files.

```
# Horizontal panorama
# 15 images

# filename-globbing: wildcard

image_000[0-9].tif
image_001[0-4].tif
```

Example 4.3: Control filename-globbing in a response file with a syntactic comment.

layer-specification	::=	'`[`' *selection-tuple* '`]`'
selection-tuple	::=	*selection* ['`:`' *selection*]
selection	::=	{ *singleton* | *range* }
range	::=	['`reverse`'] [*range-bound*] '`..`' [*range-bound*]
range-bound	::=	*singleton* | '`_`'
singleton	::=	*index* | '`-`' *index*

where *index* is an integral layer index starting at one.

Table 4.8: EBNF definition of the grammar of layer specifications. In addition to the *selection* separator '`:`' shown all usual numeric-option delimiters ('⟨`;:/`⟩') apply. The keyword for *range* reversal, '⟨`reverse`⟩', can be abbreviated to any length and is treated case-insensitively.

This syntactic comment affects the layer selection of all images listed after it including those in included response files until another `layer-selector` overrides it.

4.5 Layer Selection^c

Some image formats, like for example TIFF, allow for storing more than one image in a single file, where all the contained images can have different sizes, number of channels, resolutions, compression schemes, etc. The file there acts as a container for an *ordered* set of images.

In the TIFF-documentation these are known as "multi-page" files and because the image data in a TIFF-file is associated with a "directory", the files sometimes are also called "multi-directory" files. In this manual, multiple images in a file are called "layers".

The main advantage of multi-layer files over a set of single-layer ones is a cleaner work area with less image-files and thus an easier handling of the intermediate products which get created when generating a panorama or fused image, and in particularly with regard to panoramas of fused images.

The difficulty in working with layers is their lack of a possibly mnemonic naming scheme. They do not have telling names like *taoth-vaclarush* or *valos-cor*, but only numbers.

4.5.1 Layer Selection Syntax

To give the user the same flexibility in specifying and ordering images as with single-layer images, both Enfuse and Enblend offer a special syntax to select layers in multi-page files by appending a *layer-specification* to the image file name. Table 4.8 defines the grammar of *layer-specification*s.

Selecting a tuple of layers with a *layer-specification* overrides the active layer selection algorithm. See also option `--layer-selector`, page 25 and Section 4.4 on page 40. Layer selection works at the command-line as well as in Response Files; see Section 4.4.

The simplest *layer-specification* are the layer-*index*es. The first layer gets index 1, the second layer 2, and so on. Zero never is a valid index! For convenience indexing backwards[13]) is also possible. This means by prefixing an index with a minus-sign ('`-`') counting will start with the last layer of the *associated* multi-page image, such that the last layer always has index `-1`, the next to last index `-2` and so on. Out-of-range indexes are silently ignored whether forward or backward.

The single layer of a single-layer file always can be accessed either with index '`1`' or '`-1`'.

Select a contiguous *range* of indexes with the range operator '⟨`..`⟩', where the *range-bound*s are forward or backward indices. Leaving out a bound or substituting the open-range indicator '⟨`_`⟩' means a maximal range into the respective direction.

Layer specifications ignore white space, but usual shells do not. This means that at the command-line

```
$ enfuse --output=out.tif --verbose multi-layer.tif[2:]
```

works, whereas spaced-out out phrase '`multi-layer.tif [2 :  ]`' must be quoted

```
$ enfuse --output=out.tif --verbose 'multi-layer.tif[2 : ]'
```

Quoting will also be required if Enfuse's delimiters have special meanings to the shell.

Examples for an image with 8 layers.

`[]` The empty selection selects nothing and in that way works like the layer-selector '`no-layer`'.

`[2 : 4 : 5]` Select only layers 2, 4, and 5 in this order.

`[2 : -4 : -3]` Like before, but with some backwards-counting indices.

`[1 .. 4]` Layers 1 to 4, this is 1, 2, 3, and 4 in this order.

`[_ .. 4]` Same as above in open-range notation.

`[.. 4]` Same as above in abbreviated, open-range notation.

`[-2 .. _]` The last two layers, which are 7 and 8 in our running example.

`[_ .. _]` All layers in their natural order.

`[..]` All layers in their natural order selected with the abbreviated notation.

`[reverse _ .. _]` All layers in reverse order. This yields 8, 7, 6, 5, 4, 3, 2, and 1.

[13]) Samantha Carter: "There has to be a way to reverse the process. The answer has to be here."

`[rev ..]` All layers in reversed order as before selected with the abbreviated notation.

`[r -3 ..]` The last three layers in reverse order, this is 8, 7 and 6 in our running example.

Shell expansion will not work anymore with a file name terminated by a layer specification expression (or anything else), because to the shell it is not a file name anymore. Work around with, for example,

```
$ enfuse 'for x in image-??.tif; do echo $x[2]; done'
```

or

```
$ enfuse $(ls -1 image-??.tif | sed -e 's/$/[2]/')
```

The order of the indices determines the order of the layers, this is, the images. An index can occur multiple times, which causes layer to be considered again. *Consequently, this will lead to an error with Enblend, but may be desired with Enfuse in* `soft-mask` *mode to give the image more weight by mentioning it more than once.*

4.5.2 Tools for Multi-Page Files

Here are some tools that are particularly useful when working with multi-page files. For more helpful utilities check out Appendix A on page 99.

- Hugin's stitcher, **nona** produces multi-page TIFF file, when called with the '`-m TIFF_multilayer`'-option.
- The utility **tiffcp** of the TIFF-LibTIFF tool suite[14]) merges several TIFF-images into a single multi-page file.
- The sister program **tiffsplit** splits a multi-page file into a set of single-page TIFF-images.
- Another utility of the same origin, **tiffinfo**, is very helpful when inquiring the contents of single- or multi-page file TIFF-files.
- All tools of the ImageMagick[15])-suite, like, for example, **convert** and **display** use a similar syntax[16]) as Enfuse to select layers (which in ImageMagick parlance are called "frames") in multi-page files. Please note that ImageMagick tools start indexing at zero, whereas we start at one.
- Enfuse and Enblend by default apply the '⟨`all-layers`⟩' selector (see option `--layer-selector`, page 25) to each of the input images.

Please bear in mind that some image-processing tools – none of the above though – do *not* handle multi-page files correctly, where the most unfruitful ones only take care of the first layer and *silently* ignore any further layers.

[14]) http://www.remotesensing.org/libtiff/
[15]) https://www.imagemagick.org/script/index.php
[16]) https://www.imagemagick.org/script/command-line-processing.php#input

Chapter 5

Weighting Functions

As has been noted in the introductory Overview (page 1), Enfuse supports four different types of weighting. The following sections describe the concept of weighting and all weighting functions in detail.

5.1 Weighting Pixels

Image fusion maps each pixel $P(i, x, y)$ of every input image i to a single pixel $Q(x, y)$ in the output image:

$$P(i, x, y) \to Q(x, y),$$

where x runs from 1 to the width of the images, y from 1 to the height, and i from 1 to the number n of input images.

$$w(P(1, x, y))P(1, x, y) + \ldots + w(P(n, x, y))P(n, x, y) \to Q(x, y), \tag{5.1}$$

where

- each w is non-negative to yield a physical intensity and
- the sum of all w is 1 to leave the total intensity unchanged.

The pixel weights w themselves are weighted sums with the same constraints

$$\begin{aligned} w(P) = {} & w_{\text{exp}}\ f_{\text{exp}}(P) + \\ & w_{\text{sat}}\ f_{\text{sat}}(P) + \\ & w_{\text{cont}}\ f_{\text{cont}}(P, r_{\text{cont}}) + \\ & w_{\text{ent}}\ f_{\text{ent}}(P, r_{\text{ent}}). \end{aligned}$$

Here we have abbreviated $P(i, x, y)$ to P for simplicity. The user defines the constants w_{exp}, w_{sat}, w_{cont}, and w_{ent} with the options `--exposure-weight`, `--saturation-weight`, `--contrast-weight`, and `--entropy-weight` respectively. The functions f_{exp}, f_{sat}, f_{cont}, and f_{ent} along with the window sizes r_{cont} and r_{ent} are explained in the next sections.

5.1.1 Weighted Average

By default, Enfuse uses a weighted average, where *each* pixel contributes as much as its weight demands. Of course the weights can be extreme, favoring only a few pixels or even only one pixel in the input stack. Extremes are not typical, however.

Equal weights are another extreme that turns (5.1) into an arithmetic average. This is why we sometimes speak of the "averaging property" of this weighting algorithm, like smoothing out noise.

5.1.2 Disabling Averaging: Option '`--hard-mask`'

The weighted average computation as described above has proven to be widely successful with the exception of one special case: focus stacking, where the averaging noticeably softens the final image.

Use '`--hard-mask`' to switch Enfuse into a different weighting mode, where the pixel with the highest weight wins, this is, gets weight one, and all other pixels get the weight of zero. With option `--hard-mask` (5.1) becomes

$$P(i, x, y) \to Q(x, y),$$

where

$$w(P(i, x, y)) \geq w(P(j, x, y)) \quad \text{for all} \quad 1 \leq j \leq n.$$

Note that this "averaging" scheme lacks the nice noise-reduction property of the weighted average (5.1), because only a single input pixel contributes to the output.

5.1.3 Single Criterion Fusing

Enfuse allows the user to weight each pixel of an input image by up to four different criteria (see e.g. Chapter 1 on page 1). However, it does not force the user to do so. For some applications and more often simply to gain further insight into the weighting and fusing process, looking at only a single criterion is the preferred way to work.

The version of Enfuse for which this documentation was prepared, uses the default weights as stated in Table 5.1. Notice that by default *more than one* weight is larger than zero, which means they are *active*.

To disable a particular criterion set its weight to zero as for example

Criterion	Default Weight
Exposure	⟨1.0⟩
Saturation	⟨0.2⟩
Local Contrast	⟨0.0⟩
Local Entropy	⟨0.0⟩

Table 5.1: Enfuse's default weights as compiled into enfuse.

```
$ enfuse \
    --exposure-weight=1 --saturation-weight=0 \
    --contrast-weight=0 --entropy-weight=0 \
    image_[1-3].png
```

instructs Enfuse to consider only the exposure weight. Combine this with option `--save-masks` and it will become clearer how Enfuse computes the exposure weight for the set of images.

Another problem that can be inspected by fusing with just a single active criterion and saving the masks is if the weights of one criterion completely overpower all others.

5.2 Exposure Weighting

Exposure weighting prefers pixels with a luminance Y close to the user-chosen optimum value (option `--exposure-optimum`, default: ⟨0.5⟩) of the normalized, real-valued luminance interval $(0, 1)$.

RGB-pixels get converted to luminance before using the grayscale projector given by '`--gray-projector`', which defaults to `average`. Grayscale pixels simply are identified with luminance.

In the normalized luminance interval 0.0 represents pure black and 1.0 represents pure white independently of the data type of the input image. This is, for a JPEG image the luminance 255 maps to 1.0 in the normalized interval and for a 32 bit TIFF picture the highest luminance value 4294967295 also maps to 1.0. The middle of the luminance interval, 0.5, is where a neutral gray tone ends up with every camera that had no exposure correction dialed in, for example the image of any gray-card or white-card.

The exposure weighting algorithm only looks at a single pixel at a time; the pixel's neighborhood is not taken into account.

5.2.1 Built-In Functions

Up to Enfuse version 4.1 the only weighting function is the Gaussian

$$w_{\exp}(Y) = \exp\left(-\frac{1}{2}\left(\frac{Y - Y_{\text{opt}}}{\mathit{width}}\right)^2\right), \qquad \text{(4.3 rep.)}$$

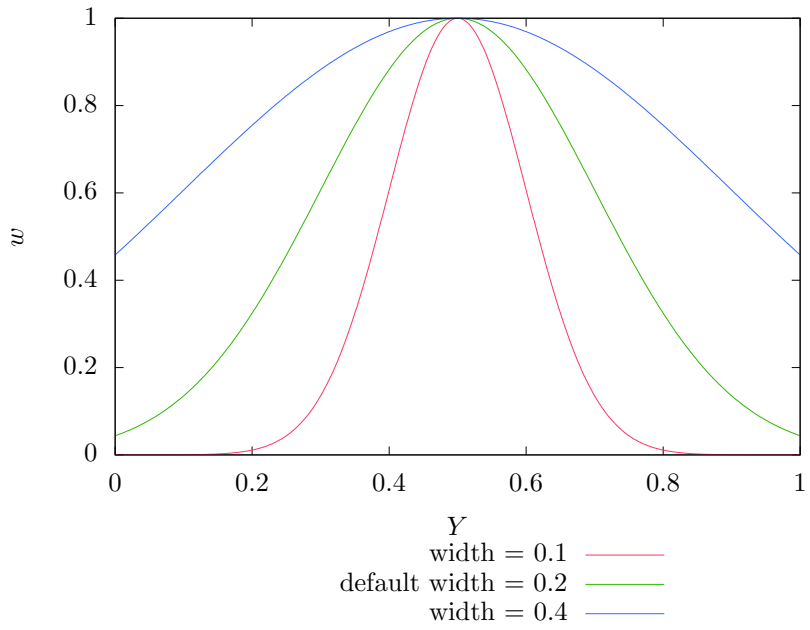


Figure 5.1: Enfuse's GAUSSIAN function with the parameters *optimum* = 0.5 and three different *width*s: 0.1, 0.2, and 0.4.

whose maximum position Y_{opt} and *width* are controlled by the command line options `--exposure-optimum` and `--exposure-width` respectively, where Y_{opt} defaults to $\langle 0.5 \rangle$ and *width* defaults to $\langle 0.2 \rangle$. Figure 5.1 shows some GAUSSIANS.

The options `--exposure-optimum` and `--exposure-width` serve to fine-tune the final result without changing the set of input images. Option `--exposure-optimum` sets the point of optimum exposure. Increasing the *optimum* makes Enfuse prefer lighter pixels, rendering the final image lighter, and vice versa. Option `--exposure-width` defines the *width* of acceptable exposures. Small values of *width* penalize exposures that deviate from *optimum* more, and vice versa.

In Enfuse version 4.2 several new exposure weight functions have been added. Select them with option `--exposure-weight-function`. For the following presentation we refer to the linear luminance transform

$$z = \frac{Y - Y_{\text{opt}}}{\mathit{width}}. \qquad \text{(4.2 rep.)}$$

as introduced in Eqn. 4.2 on page 35.

Functions GAUSSIAN

$$w_{\exp}(z) = \exp\left(-z^2/2\right) \qquad \text{(4.3 rep.)}$$

Exposure Weight	*WEIGHT-FUNC.*	Equ.	Chart
Gaussian curve (default)	`gauss`, `gaussian`	4.3, p. 36	5.1, p. 51
Lorentz curve	`lorentz`, `lorentzian`	4.4, p. 36	5.2, p. 53
Upper half-wave of a sine	`halfsine`, `half-sine`	4.5, p. 36	5.3, p. 54
Full sine-wave shifted upwards by one	`fullsine`, `full-sine`	4.6, p. 36	5.4, p. 55
Quartic, or bi-square function	`bisquare`, `bi-square`	4.7, p. 36	5.5, p. 56

Table 5.2: Available, compiled-in exposure weight functions.

and Lorentzian

$$w_{\text{exp}}(z) = \frac{1}{1 + z^2/2} \qquad \text{(4.4 rep.)}$$

behave like $1 - z^2$ around the optimum. However for large $|z|$ the Gaussian weight rolls off like $\exp(-z^2/2)$ and the Lorentzian only as z^{-2}.

Both, the Gaussian and the Lorentzian are easy to use, because they do not go exactly to zero. Thus, Enfuse can select "better" pixels even far away from the chosen optimum.

Again, Half-Sine

$$w_{\text{exp}}(z) = \begin{cases} \cos(z) & \text{if } |z| \le \pi/2 \\ 0 & \text{otherwise.} \end{cases} \qquad \text{(4.5 rep.)}$$

and Full-Sine

$$w_{\text{exp}}(z) = \begin{cases} (1 + \cos(z))/2 & \text{if } |z| \le \pi \\ 0 & \text{otherwise.} \end{cases} \qquad \text{(4.6 rep.)}$$

behave like $1-z^2$ around the optimum, like Gaussian and Lorentzian. However for large $|z|$ they both are exactly zero. The difference is how they decrease just before they reach zero. Half-Sine behaves like $z - z'$ and Full-Sine like $(z - z'')^2$, where z' and z'' are the respective zeros.

Bi-Square

$$w_{\text{exp}}(z) = \begin{cases} 1 - z^4 & \text{if } |z| \le 1 \\ 0 & \text{otherwise.} \end{cases} \qquad \text{(4.7 rep.)}$$

is the only predefined function that behaves like $1 - z^4$ around the optimum.

The weight functions Half-Sine, Full-Sine, and Bi-Square are more difficult to use, because they yield exactly zero if the normalized luminance of a pixel is far enough away from the optimum. This can lead to pathologies if the luminances of the same pixel position in all N input images are assigned a weight of zero. For all-zero weights Enfuse falls back to weighing equally. This is, each pixel gets a weight of $1/N$, which may or may not be the desired result. However, if the *width* is chosen so large that the weight never vanishes or the input images

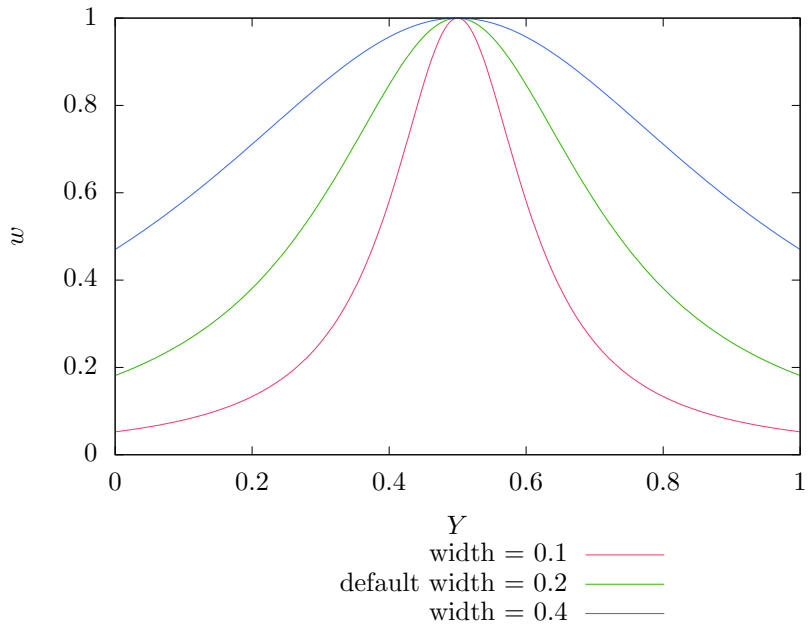


Figure 5.2: Enfuse's LORENTZIAN function with the parameters *optimum* = 0.5 and three different *width*s: 0.1, 0.2, and 0.4.

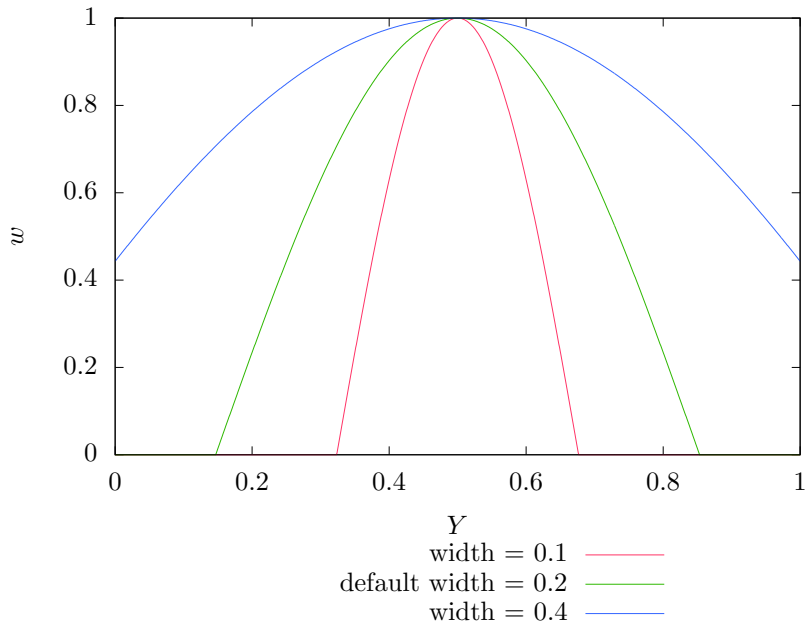


Figure 5.3: Enfuse's Half-Sine function with the parameters *optimum* = 0.5 and three different *width*s: 0.1, 0.2, and 0.4.

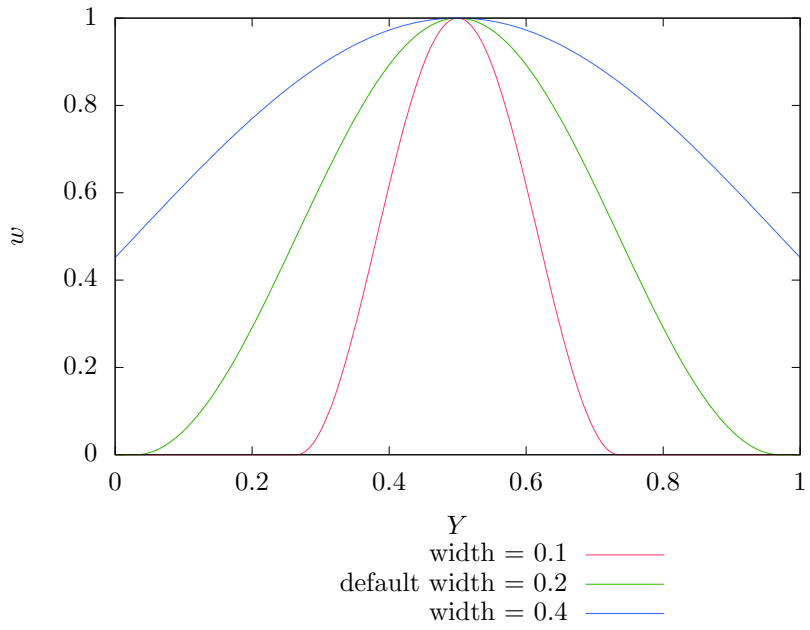


Figure 5.4: Enfuse's Full-Sine function with the parameters *optimum* = 0.5 and three different *width*s: 0.1, 0.2, and 0.4.

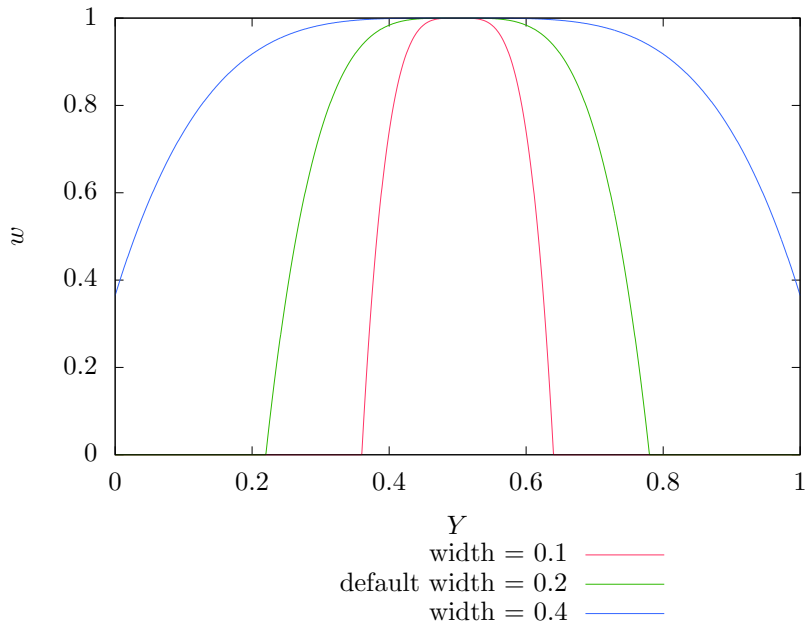


Figure 5.5: Enfuse's Bi-Square function with the parameters *optimum* = 0.5 and three different *width*s: 0.1, 0.2, and 0.4.

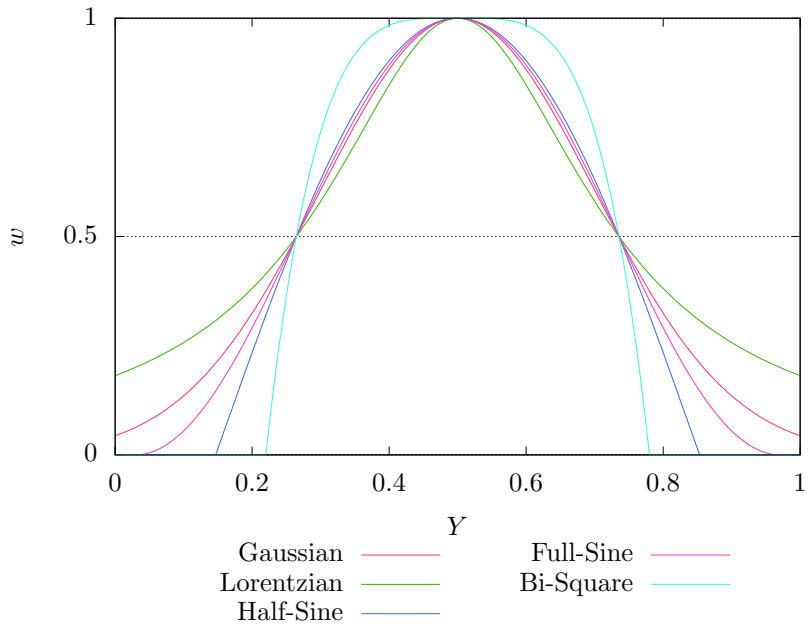


Figure 5.6: Comparison of all of Enfuse's built-in exposure weight functions $w(Y)$ for the default values of *optimum* $= 0.5$ and *width* $= 0.2$. Note that all functions intersect at $w = 1/2$, this is, they share the same FWHM.

span a large enough range of luminances for each and every pixel, the problem is circumnavigated.

Another way of cutting off underexposed or overexposed pixels is to use option `--exposure-cutoff`, which has the additional benefit of allowing to choose upper and lower cutoff separately.

Figure 5.6 compares all available exposure weight functions for the same parameters, namely their defaults. They all intersect at $w = 1/2$ independently of *optimum* or *width*, making it simple to just try them out without fundamentally changing brightness.

5.2.2 User-Defined Exposure Weighting Functions

5.2.3 User-Defined, Dynamically-Linked Exposure Weighting Functions

DYNAMIC LINKING-enabled versions only.

On operating systems, where dynamic linking (or synonymously: dynamic loading) of code is possible, for Enfuse executables compiled with dynamic-linking support (see Section 3.3.1 on page 7 on how to check this feature), Enfuse can

work with user-defined exposure weighting functions, passed with the long form of option `--exposure-weight-function`, load the exposure weight function identified by *SYMBOL* from *SHARED-OBJECT* and optionally pass some *ARGUMENT*s:

`--exposure-weight-function=`*SHARED-OBJECT*`:`*SYMBOL*[`:`*ARGUMENT*[`:...`]]

Some notes on the arguments of this option:

- *SHARED-OBJECT* is a filename (typically ending in *.so* or *.dll*). Depending on the operating system and the dynamic-loader implementation compiled into **enfuse**, *SHARED-OBJECT* may or may not require a path.

  Linux

  If *SHARED-OBJECT* does not contain a slash ('/'), the dynamic loader *only* searches along `LD_LIBRARY_PATH`; it even ignores the current working directory unless `LD_LIBRARY_PATH` contains a dot ('.'). So, to use a *SHARED-OBJECT* living in the current directory either say

  ```
  env LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. \
      enfuse --exposure-weight-function=SHARED-OBJECT:...
  ```

  or

  ```
  enfuse --exposure-weight-function=./SHARED-OBJECT:...
  ```

  For details of the search algorithm please consult the manual page of dlopen(3).

  □

  OS X

  If *SHARED-OBJECT* does not contain a slash ('/'), the dynamic loader searches the following paths or directories until it finds a compatible Mach-O file:

  1. `LD_LIBRARY_PATH`,
  2. `DYLD_LIBRARY_PATH`,
  3. current working directory, and finally
  4. `DYLD_FALLBACK_LIBRARY_PATH`.

  For details of the search algorithm please consult the manual page of dlopen(3).

  □

  Windows

  If *SHARED-OBJECT* specifies an absolute filename, exactly this file is used. Otherwise Enfuse searches in the following directories and in this order:

1. The directory from which **enfuse** is loaded.
2. The system directory.
3. The Windows directory.
4. The current directory.
5. The directories that are listed in the `PATH` environment variable.

For details consult the manual page of LoadLibrary.

□

- There is no way knowing which of the symbols inside of *SHARED-OBJECT* are suitable for *SYMBOL* without knowledge of source code of *SHARED-OBJECT*.
- A weight function has access to additional *ARGUMENT*s passed in by appending them after *SYMBOL* with the usual delimiters. How these *ARGUMENT*s are interpreted and how many of them are required is encoded in the weight-function. Enfuse supports *ARGUMENT*s; it neither restricts their number nor their type.

  Usually, neither the exposure optimum (`--exposure-optimum`=OPTIMUM) nor the width (`--exposure-width`=WIDTH) of the exposure function are ARGUMENTs, because they are always explicitly passed on to any exposure weight function.

  For example, assuming *variable_power.cc* of the supplied examples was compiled to *variable_power.so*, we can override the default exponent of 2 with

```
enfuse --exposure-weight-function=\
         variable_power.so:vpower:1.8 ...
```

Prerequisites

To use a home-grown exposure-weight function several prerequisites must be met. On the software side

1. The operating system allows loading additional code during the execution of an application.
2. Enfuse is compiled with the extra feature “dynamic linking support”.
3. Either
   (a) The same compiler that compiled Enfuse is available or at least
   (b) A compiler that produces compatible object code to the compiler that compiled Enfuse.

The latter is called "ABI-compatible". An example for a pair of ABI-compatible compilers is GNU's **g++** and INTEL's **icpc**.

To find out which compiler built *your* version of **enfuse** use option `--show-software-components`.

4. The base-class header file *exposure_weight_base.h* is available.

Between chair and keyboard:

- A firm understanding of weighting pixels in the fusion process and in particular in the cumulative ascription of different weights.
- A basic understanding of object-oriented programming paired with the ability to compile and link single-source C++-files.
- A realistic expectation of the limitations of tailoring weight functions.

Coding Guidelines

1. Derive the weight function from the supplied C++ base-class `ExposureWeight`, which is defined in header file *exposure_weight_base.h*. It resides in the *src* sub-directory of the source distribution and – for a correctly installed package – in directory */usr/share/doc/enfuse/examples*.
2. At least override method `weight`.
   - Domain: define `weight` for normalized luminance values y from zero to one including both interval ends: $0 \leq y \leq 1$.
   - Image: Let the `weight` w fall in the interval from zero to one: $0 \leq w \leq 1$. The `weight`s can be all the same, $w = \text{const}$. This is, they can encode a constant weight, as long as the constant is not zero.

     Enfuse checks this property and refuses to continue if any weight is outside the required range or all weights are zero.
   - (Optionally) Rescale the *WIDTH* of the function to match the FWHM of Enfuse's original Gauss curve. The macro `FWHM_GAUSSIAN` is defined exactly to this end.
3. If necessary, rewrite methods `initialize` and `normalize`, too.
4. OPENMP-enabled versions only.

   Enfuse never calls `initialize` in an OPENMP parallel execution environment. However, OPENMP-enabled versions of Enfuse call `normalize` and `weight` in parallel sections.

   Technically, the functors which the user-defined weight functions are part of are copy-constructed for each OPENMP worker thread. Thus, there is no contention within the `ExposureWeight` sub-classes. Although, if `normalize` or `weight` access a shared resource these accesses must be

protected by serialization instructions. One solution is to use OpenMP directives, like for example,

```
#pragma omp critical
{
    std::cout << "foobar!" << std::endl;
}
```

Experienced hackers will recognize occasions when to prefer other constructs, like, for example `#pragma omp atomic` or simply an atomic datatype (e.g. `sig_atomic_t` from *signal.h*).

Remember to compile all modules that use OpenMP directives with the (compiler-specific) flags that turn on OpenMP. For `g++` this is '`-fopenmp`' and for **icpc** it is '`-fopenmp`' or '`-openmp`'.

5. To raise an exception associated with the derived, user-defined exposure-weight class, throw

   ```
   ExposureWeight::error(const std::string& message)
   ```

   Enfuse catches these exceptions at an enclosing scope, displays *message*, and aborts.

6. Define an object of the derived class. This creates the *SYMBOL* to refer to at the Enfuse command line.

   The actual signature of the constructor (default, two-argument, ...) does not matter, because Enfuse *always* invokes `initialize` before calling any other method of a user-defined `ExposureWeight` sub-class. Method `initialize` sets (read: overwrites) *optimum* and *width* and ensures they are within the required parameter range.

Example 5.1 shows the C++-code of a suitable extension. If Enfuse has been compiled with support for user-defined weight functions, the examples presented here should have been duplicated in directory */usr/share/doc/enblend-enfuse/examples/enfuse* along with a GNU-Makefile called *Makefile.userweight*.

As the extension language is C++, we can write templated families of functions, like Example 5.2 demonstrates.

The last example, 5.3, shows a weight function that accesses an extra *ARGUMENT* passed in with `--exposure-weight-function`. A class like `VariablePower` allows full control over the exponent at the command line including fractional exponents thereby generalizing both of the previous examples.

Performance Considerations

Exposure weighting objects are created and destroyed only $O(1)$ times. Thus, method `initialize` could be used to perform all kinds of computationally expensive tasks. In contrast, methods `normalize` and `weight` are called for

```
#include <cmath>                        // std::fabs()

#include "exposure_weight_base.h"       // FWHM_GAUSSIAN, ExposureWeight

struct Linear : public ExposureWeight {
    void initialize(double y_optimum, double width_parameter,
                    ExposureWeight::argument_const_iterator arguments_begin,
                    ExposureWeight::argument_const_iterator arguments_end)
        override {
        ExposureWeight::initialize(y_optimum,
                                   width_parameter * FWHM_GAUSSIAN,
                                   arguments_begin, arguments_end);
    }

    double weight(double y) override {
        const double z = std::fabs(normalize(y));
        return z <= 1.0 ? 1.0 - z : 0.0;
    }
};

Linear linear;
```

Example 5.1: A dynamic exposure weight function that defines a "roof-top". The natural width is exactly one, so we override method `initialize` to rescale *WIDTH*, passed in as `width_parameter`, by multiplying with `FWHM_GAUSSIAN` to get the same width as the predefined Gaussian.

```
#include <algorithm>    // std::max()
#include <cmath>        // M_LN2, std::exp(), std::fabs()

#include "exposure_weight_base.h" // FWHM_GAUSSIAN, ExposureWeight

template <int n> double ipower(double x) {return x * ipower<n - 1>(x);}
template <> double ipower<0>(double) {return 1.0;}

template <int n> struct TemplatedPower : public ExposureWeight {
    void initialize(double y_optimum, double width,
                    ExposureWeight::argument_const_iterator arguments_begin,
                    ExposureWeight::argument_const_iterator arguments_end)
        override {
        const double fwhm = 2.0 / std::exp(M_LN2 / static_cast<double>(n));
        ExposureWeight::initialize(y_optimum,
                                   width * FWHM_GAUSSIAN / fwhm,
                                   arguments_begin, arguments_end);
    }

    double weight(double y) override {
        return std::max(1.0 - ipower<n>(std::fabs(normalize(y))), 0.0);
    }
};

TemplatedPower<2> tpower2;
TemplatedPower<3> tpower3;
TemplatedPower<4> tpower4;
```

Example 5.2: The templated class `TemplatedPower` allows to create a weight function for arbitrary positive exponents `n`. In particular, `TemplatedPower<4>` duplicates the built-in exposure-weight function `bisquare`.

```
#include <algorithm>    // std::max()
#include <cerrno>       // errno
#include <cmath>        // M_LN2, std::exp(), std::fabs(), std::pow()

#include "exposure_weight_base.h" // FWHM_GAUSSIAN, ExposureWeight


class VariablePower : public ExposureWeight {
    typedef ExposureWeight super;

public:
    void initialize(double y_optimum, double width,
                    ExposureWeight::argument_const_iterator arguments_begin,
                    ExposureWeight::argument_const_iterator arguments_end)
        override {
        if (arguments_begin == arguments_end) {
            exponent = 2.0;
        } else {
            char* tail;
            errno = 0;
            exponent = strtod(arguments_begin->c_str(), &tail);
            if (*tail != 0 || errno != 0) {
                throw super::error("non-numeric␣exponent");
            }
            if (exponent <= 0.0 || exponent > 4.0) {
                throw super::error("exponent␣x␣out␣of␣range␣0␣<␣x␣<=␣4");
            }
        }

        const double fwhm = 2.0 / std::exp(M_LN2 / exponent);
        super::initialize(y_optimum, width * FWHM_GAUSSIAN / fwhm,
                          arguments_begin, arguments_end);
    }

    double weight(double y) override {
      return std::max(1.0 - std::pow(std::fabs(normalize(y)), exponent), 0.0);
    }

private:
    double exponent;
};

VariablePower vpower;
```

Example 5.3: Dynamic exposure weight function that accesses the first extra argument from the tuple of arguments passed with option `--exposure-weight-function`.

every pixel in *each* of the input images. Therefore, if performance of the weight function is a problem, these two functions are the prime candidates for optimization.

Compiling, Linking, and Loading

Linux

Compile and link using the GNU-compiler[1]), `g++`, for example with

```
g++ -std=c++11 \
    -O2 -fpic -I<PATH-TO-BASE-CLASS-HEADER> \
    -shared -Wl,-soname,dynexp.so \
    -o dynexp.so \
    dynexp.cc
```

The important options are

`-fpic`
: Instruct the compiler's code-generator to produce position-independent code (PIC), which is suitable for a shared object. Some systems require '`-fPIC`' instead of '`-fpic`'.

`-shared`
: Tell the linker to create a shared object instead of the default executable. On some systems, the library must be "blessed", by passing the shared-object name (`soname`) directly to the linker (`-Wl`).

  Of course more than one object file can be linked into a single shared object.

Finally, the weight function can be selected by its *SYMBOL* name in the *SHARED-OBJECT*.

```
enfuse --exposure-weight-function=dynexp.so:linear...
```

□

OS X

On OS X the creation of shared objects – or loadable modules – has been tested with the C-language frontend of LLVM[2]), `clang++`[3]), and should work on OS X Mavericks (10.9) or higher.

```
clang++ -std=c++11 -stdlib=libc++ \
    -O2 -bundle -I<PATH-TO-BASE-CLASS-HEADER> \
    -o dynexp.so \
    dynexp.cc
```

[1]) https://gcc.gnu.org/

The important option here is '`-bundle`' which instructs the compiler's code-generator to produce a loadable module.

Finally, the weight function can be selected by its *SYMBOL* name in the *SHARED-OBJECT*.

```
enfuse --exposure-weight-function=dynexp.so:linear...
```

□

Windows

On Windows the creation of shared objects – or dynamic link libraries (DLL files) as they are called here – has been tested with the MinGW compiler chain and with MS-Visual C++ 2012.

- Compile and link using the MinGW compiler with

  ```
  g++ -g -O2 -I<PATH-TO-BASE-CLASS-HEADER> -c dyn=
  exp.cc
  g++ -g -shared -Wl,-soname,dynexp.dll -o dynexp.dll
  dynexp.o
  ```

  For details see the explanation for the GNU compiler above. Windows neither requires options `-fpic` nor `-fPIC`.

- When using the MS-Visual C++ compiler, you need to explicitly export *SYMBOL*. There are two possibilities to achieve this. Use only one variant, not both at the same time.

  1. Either use `"C"` linkage and define the object using the construction `__declspec(dllexport)`. For Example 5.1 the object definition has to be extended to

     ```
     extern "C"
     {
       __declspec(dllexport) Linear linear;
     }
     ```

  2. Or, alternatively, create a module-definition file (*.def*) and pass this file to the linker (in: Project Properties, Linker, Module Definition File). For Example 5.1, this file would look like

     ```
     LIBRARY dynexp
     EXPORTS
             linear @@1
     ```

Finally, the weight function can be selected by its *SYMBOL* in the dynamic link library.

[2]) http://llvm.org/
[3]) http://clang.llvm.org/

```
enfuse --exposure-weight-function=dynexp.dll:linear...
```

□

Summary of influential options

`--exposure-optimum`: Section 4.2.3 on page 24

`--exposure-weight-function`: Section 4.2.5 on page 34

`--exposure-weight`: Section 4.2.3 on page 24

`--exposure-width`: Section 4.2.3 on page 24

`--gray-projector`: Section 4.2.5 on page 35

5.3 Saturation Weighting

Saturation weighting prefers pixels with a high saturation, where the saturation S_{HSL} is computed in HSL color space. For an introduction to the HSL color space, please consult Wikipedia[4]).

Taking the largest and smallest components of the RGB-value (R, G, B) in the normalized RGB-cube

$$\begin{aligned} v_{\min} &= \min(R, G, B) \quad \text{and} \\ v_{\max} &= \max(R, G, B), \end{aligned}$$

we define chroma

$$C = v_{\max} - v_{\min}$$

and lightness

$$L = (v_{\max} + v_{\min})/2.$$

Enfuse computes the saturation of the pixel according to the following formula:

$$S_{\mathrm{HSL}} = \begin{cases} 0 & \text{if } C = 0 \\ \frac{C}{1-|2L-1|} & \text{otherwise.} \end{cases} \tag{5.2}$$

The saturation weighting algorithm only looks at a single pixel at a time. The neighborhood of the pixel is not taken into account.

Obviously, saturation weighting can only be defined for RGB-images, not for grayscale ones. If you need something similar, check out Section 5.5 on Entropy Weighting, which works for both RGB and grayscale pictures.

Summary of influential options

`--saturation-weight`: Section 4.2.3 on page 25

[4]) https://en.wikipedia.org/wiki/HSL_and_HSV

5.4 Local Contrast Weighting

Local contrast weighting favors pixels inside a high contrast neighborhood. The notion of "high contrast" is defined either by two different criteria or by a blend of both:

- The standard deviation (SDEV) of all the pixels in the local analysis window is large. See Section 5.4.1 on page 68.
- The Laplacian-of-Gaussian (LOG) has a large magnitude. See Section 5.4.2 on page 70.
- If the LOG magnitude is below a given threshold, use SDEV data, otherwise stick with LOG. See Section 5.4.3 on page 72.

Enfuse converts every RGB image to grayscale before it determines its contrast. Option `--gray-projector`, page 35 controls the projector function. Depending on the subject, one of several grayscale projectors may yield the best black-and-white contrast for image fusion.

In the following sections we describe each algorithm in detail.

5.4.1 Standard Deviation

The pixel under consideration C sits exactly in the center of a square, the so-called local analysis window. It always has an uneven edge length. The user sets the size with option `--contrast-window-size`. Figure 5.7 shows two windows with different sizes.

During the analysis, Enfuse scans the local analysis window across all rows and all columns[5]) of each of the input images to compute the contrast weight of every pixel.

Summary of influential options

`--contrast-weight`: Section 4.2.3 on page 24

`--contrast-window-size`: Section 4.2.5 on page 29

`--gray-projector`: Section 4.2.5 on page 35

`--hard-mask`: Section 4.2.3 on page 25

Statistical Moments

We start with the probability function w of the random variable X:

$$w : x \to p(\{\omega : X(\omega) = x\}).$$

It associates a probability p with each of the n different possible outcomes ω of the random variable X.

[5]) In the current implementation a `floor(contrast-window-size / 2)`-wide border around the images remains unprocessed and gets a weight of zero.

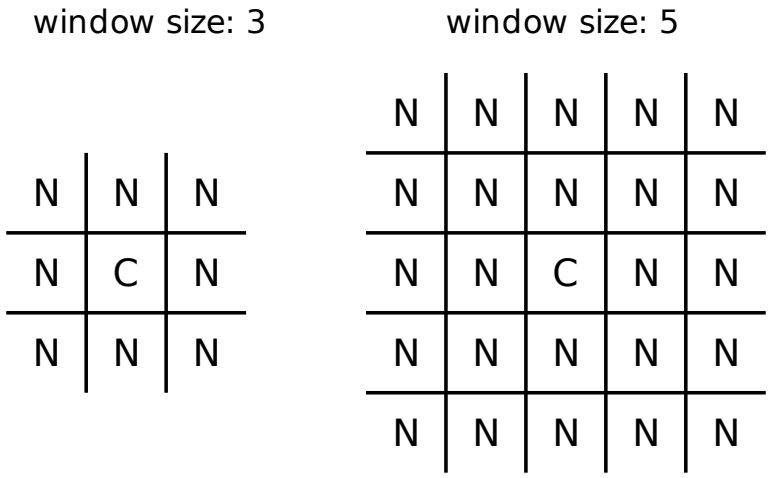


Figure 5.7: Examples of local analysis windows for the sizes 3 and 5. "C" marks the center where the pixel gets the weight. "N" denote neighboring pixels, which all contribute to the weight.

Based on w, we define the expectation value or "First Moment" of the random variable X:

$$\mathsf{Ex}\, X := \sum_{i=1}^{n} x_i w(x_i).$$

Using the definition of the expectation value, we define the variance, or "Second Moment" as

$$\mathsf{Var}\, X := \mathsf{Ex}\left((X - \mathsf{Ex}\, X)^2\right),$$

and the standard deviation as

$$\sigma\, X := \sqrt{\mathsf{Var}\, X}.$$

Obviously, the variance of X is the expectation value of the squared deviation from the expectation value of X itself. Note that the variance's dimension is X's dimension squared; the standard deviation rectifies the dimension to make it comparable with X itself again.

Estimators

In Enfuse, we assume that X follows a uniform probability function $w(x) = \text{const}$. That is, all pixel values in the local analysis window are considered to be equally probable. Thus, the expectation value and the variance can be estimated from the pixel values like this

$$\mathsf{Ex}\, X := \frac{1}{n} \sum_{i=1}^{n} x_i.$$

In other words: the expectation value is the arithmetic mean of the lightness of all pixels in the local analysis window. Analogously, the variance becomes

$$\mathsf{Var}\, X := \frac{1}{n-1}\, \mathsf{Ex}\, \left((X - \mathsf{Ex}\, X)^2\right).$$

5.4.2 Laplacian of Gaussian

The LAPLACIAN-of-GAUSSIAN (LoG) is an operator to detect edges in an image. Sometimes the LoG-operator is also called MARR-HILDRETH operator. A LAPLACIAN-of-GAUSSIAN operator, `vigra::laplacianOfGaussian`[6]) is part of the package VIGRA[7]) that Enfuse is built upon and is used for edge detection if option `--contrast-edge-scale` is non-zero and `--contrast-min-curvature` equal to or less than zero.

Let the GAUSSIAN function be

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

The parameter σ, the argument of option `--contrast-edge-scale`, is the length scale on which edges are detected by $g(x, y)$. We apply the LAPLACIAN operator in CARTESIAN coordinates

$$\triangle \equiv \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

to $g(x, y)$, to arrive at a continuous representation of the two-dimensional filter kernel

$$k(x, y) = \frac{\xi^2 - 1}{\pi\sigma^4} \exp(-\xi^2), \tag{5.3}$$

where we have used the dimensionless distance ξ from the origin

$$\xi^2 = \frac{x^2 + y^2}{2\sigma^2}.$$

Enfuse uses a discrete approximation of k in the convolution with the image. The operator is radially symmetric with respect to the origin, which is why we can easily plot it in Figure 5.8, setting $R = \sqrt{x^2 + y^2}$.
See also HIPR2: LAPLACIAN-of-GAUSSIAN[8]).

Sometimes the LoG is plagued by noise in the input images. After all, it is a numerical approximation of the second derivative and deriving always "roughens" a function. The (normalized) mask files relentlessly disclose such problems. Use option `--contrast-min-curvature` with a *negative* argument *CURVATURE* to suppress all edges with a curvature below $-$*CURVATURE* (which is a positive

[6]) https://ukoethe.github.io/vigra/doc-release/vigra/group__ConvolutionFilters.html
[7]) https://ukoethe.github.io/vigra/
[8]) http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

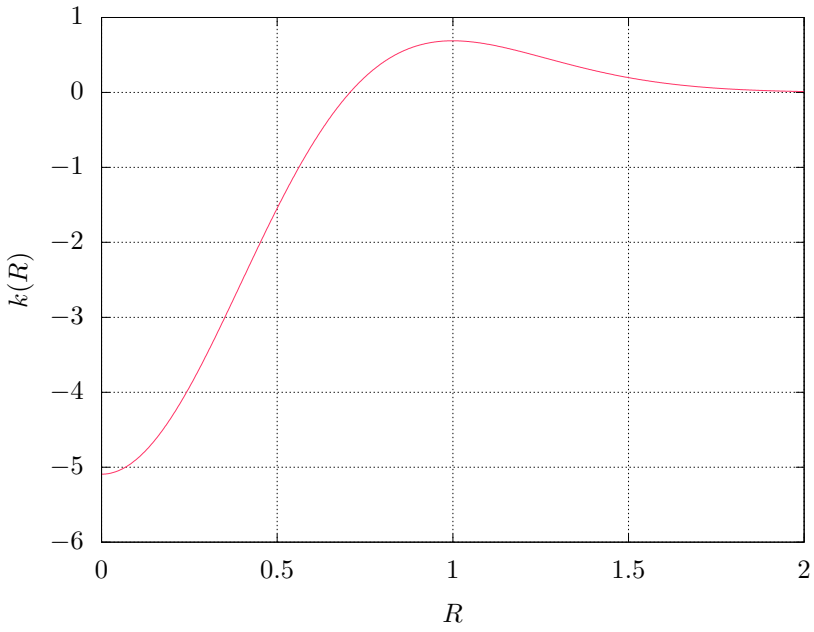


Figure 5.8: Plot of the LAPLACIAN-of-GAUSSIAN function $k(R)$, Eqn. 5.3, for $\sigma = 0.5$, using $R = \sqrt{x^2 + y^2}$.

value). Check the effects with the mask files and particularly the hard-mask files (⟨*hardmask-%n.tif*⟩) if using option `--hard-mask`.

To indicate the *CURVATURE* in relative terms, which is particularly comprehensible for humans, append a percent sign ('`%`'). Try minimum curvatures starting from −0.5% to −3%.

Summary of influential options

`--contrast-edge-scale`: Section 4.2.5 on page 27

`--contrast-min-curvature`: Section 4.2.5 on page 29

`--contrast-weight`: Section 4.2.3 on page 24

`--hard-mask`: Section 4.2.3 on page 25

5.4.3 Blend Standard Deviation and Laplacian-of-Gaussian

Enfuse can team the standard deviation computation and Laplacian of Gaussian to deliver the best of both methods. Use a *positive* argument *CURVATURE* with option `--contrast-min-curvature` to combine both algorithms. In this mode of operation Enfuse computes the SDev-weight and the LoG-weight, then uses the LoG to decide whether to go with that value or prefer the SDev data. If the LoG is greater than *CURVATURE* Enfuse uses the weight delivered by the LoG, otherwise the SDev-weight is rescaled such that its maximum is equal to *CURVATURE*, and the scaled SDev is used as weight.

This technique merges the two edge detection methods where they are best. The LoG excels with clear edges and cannot be fooled by strong but smooth gradients. However, it is bad at detecting faint edges and it is susceptible to noise. The SDev on the other hand shines with even the most marginal edges, and resists noise quite well. Its weakness is that is is easily deceived by strong and smooth gradients. Tuning *CURVATURE* the user can pick the best threshold for a given set of images.

Summary of influential options

`--contrast-edge-scale`: Section 4.2.5 on page 27

`--contrast-min-curvature`: Section 4.2.5 on page 29

`--contrast-weight`: Section 4.2.3 on page 24

`--contrast-window-size`: Section 4.2.5 on page 29

`--gray-projector`: Section 4.2.5 on page 35

`--hard-mask`: Section 4.2.3 on page 25

5.4.4 Scaling and Choice of Mode

Experience has shown that neither the parameters *EDGESCALE* and *CURVATURE* nor the mode of operation (SDev-only, LoG-only, or a blend of both) scales to different image sizes. In practice, this means that if you start with a set of reduced size images, say 2808×1872 pixels, carefully optimize *EDGESCALE*, *CURVATURE* and so on, and find LoG-only the best mode, and then switch to the original resolution of 5616×3744 pixels, multiplying (or dividing) the parameters by four and sticking to LoG-only might *not* result in the best fused image. For best quality, perform the parameter optimization and the search for the most appropriate mode at the final resolution.

5.5 Local Entropy Weighting

Entropy weighting prefers pixels inside a high entropy neighborhood.

Let S be an n-ary source. Watching the output of S an observer on average gains the information

$$H_a(n) := \sum_{x \in S} p(x) \log_a(1/p(x))$$

per emitted message, where we assume the knowledge of the probability function $p(S)$. The expectation value $H_a(n)$ is called entropy of the source S. Entropy measures our uncertainty if we are to guess which message gets chosen by the source in the future. The unit of the entropy depends on the choice of the constant $a > 1$. Obviously

$$H_b(n) = H_a(n)/\log_a(b)$$

holds for all $b > 1$. We use $a = 2$ for entropy weighting and set the entropy of the "impossible message" to zero according to

$$\lim_{p \to 0} p \log_a(1/p) = 0.$$

Figure 5.9 shows an entropy function.

For more on (information) entropy visit Wikipedia[9]).

Enfuse computes a pixel's entropy by considering the pixel itself and its surrounding pixels quite similar to Local-Contrast Weighting (5.4 on page 68). The size of the window is set by '`--entropy-window-size`'. Choosing the right size is difficult, because there is a serious tradeoff between the locality of the data and the size of the sample used to compute H. A large window results in a large sample size and therefore in a reliable entropy, but considering pixels far away from the center degrades H into a non-local measure. For small windows the opposite holds true.

Another difficulty arises from the use of entropy as a weighting function in dark parts of an image, that is, in areas where the signal-to-noise ratio is low.

[9]) https://en.wikipedia.org/wiki/Information_entropy

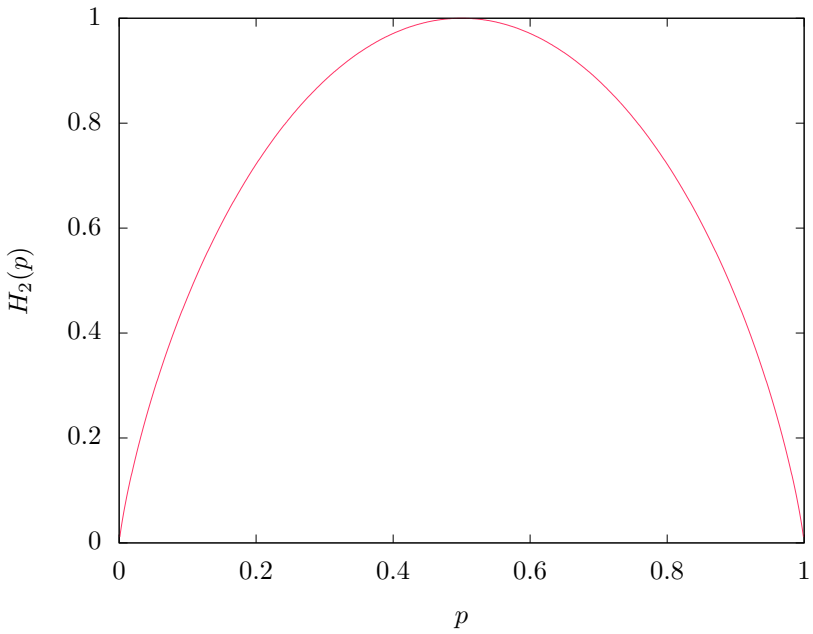


Figure 5.9: Entropy function $H_2(p)$ for an experiment with exactly two outcomes.

Without any precautions, high noise is taken to be high entropy, which might not be desired. Use option `--entropy-cutoff` to control the black level when computing the entropy.

On the other extreme side of lightness, very light parts of an image, the sensor might already have overflown without the signal reaching 1.0 in the normalized luminance interval. For these pixels the entropy is zero and Enfuse can be told of the threshold by properly setting the second argument of '`--entropy-cutoff`'.

Summary of influential options

`--entropy-cutoff`: Section 4.2.5 on page 29

`--entropy-weight`: Section 4.2.3 on page 24

`--entropy-window-size`: Section 4.2.5 on page 30

Chapter 6

Color Spaces And Color Profiles[c]

This chapter explains the connection of pixel data types, ICC[1])-color profiles, blend color spaces in Enfuse or Enblend.

Here, we collectively speak of blending and do not distinguish fusing, for the basic operations are the same. Furthermore, we assume the multi-resolution spline algorithm has calculated a set of weights w_i for $i = 1, 2, \ldots$ and $\sum w_i = 1$ for each pixel that must be blended from the participating input pixels $P_i, i = 1, 2, \ldots$.

In the simplest, non-trivial case we have to blend a pair of grayscale input pixels. Given their luminances L_1, L_2 and their weighting factor $0 \leq w \leq 1$, what luminance L is their "weighted average"? This is the heart of Enfuse's and Enblend's pyramidal blending operations! We are in particular interested in a weighted average that appears *visually* correct, this is, our eyes and brains consider L convincing or at the very least credible.

> *Note that Enfuse and Enblend face different obstacles in their respective domains of use.*
>
> *Enblend*
>
> > *The overlapping areas usually are* well matched *both geometrically and photometrically. The differences of the pixels that must be blended are small.*
>
> *Enfuse (using a Soft Mask[2]))*
>
> > *The input images* greatly differ *in exposure, saturation, or contrast. This is exactly why we want to fuse them. Thus, the luminance, saturation, and hue differences to be handled by Enfuse are generally quite high.*

The details of blending pixels and in particular color pixels is quite intricate, which is why we start this chapter with a mathematical introduction.

[1]) https://en.wikipedia.org/wiki/ICC_profile

[2]) Fusing with a Hard Mask is different, because exactly one weight factor is unity and all the others are zero. There is nothing to blend – just to copy.

6.1 Mathematical Preliminaries

Let us first address grayscale images because they only require us to talk about luminances. For a linear representation of luminances, we just blend for a given t with

$$L = tL_1 + (1-t)L_2 \quad \text{with} \quad 0 \le t \le 1, \tag{6.1}$$

where the luminances $L_i, i = 1, 2$, range from zero to their data-type dependent maximum value $L_{\max}$, thereby defining a "luminance interval". We can always map this interval to $(0, 1)$ by dividing the luminances by the maximum, which is why we call the latter *"normalized luminance interval"*:

$$(0, L_{\max}) \to (0, 1) \tag{6.2}$$

Obviously,

$$0 \le \bar{L} \le 1 \tag{6.3}$$

holds for all values $\bar{L} := L/L_{\max}$ in the normalized luminance interval.

Sometimes images are gamma-encoded with exponent γ and the blended luminance becomes

$$L' = \left(tL_1^{1/\gamma} + (1-t)L_2^{1/\gamma}\right)^{\gamma}, \tag{6.4}$$

which couples t and L' in a non-linear way. See also Eric Brasseur's explanation of the gamma error in picture scaling[3]).

> *Typical gamma values are $\gamma = 2.2$ for* sRGB *and* AdobeRGB*, 1.8 for* AppleRGB*, and* ProPhotoRGB*, 1.0 for Linear Rec709* RGB *and any others with "linear" in their names. For an extensive overview check out* Bruce Lindbloom's *Information on Working Color Spaces.*[4])

The usual color-input images fed into Enfuse are RGB-encoded, which means each pixel comes as a triple of values $(r, g, b)^{\mathrm{T}}$ that represent the red, green, and blue parts. We apply the normalization (6.2) to each of the three primary colors and arrive at an "RGB-*cube*" with unit edge length. The vectors of primary colors span the cube

$$\vec{r} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \vec{g} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \vec{b} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

For each point inside – familiarly called pixel – the generalization of (6.3) holds

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \begin{pmatrix} r \\ g \\ b \end{pmatrix} \le \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \tag{6.5}$$

[3]) http://www.4p8.com/eric.brasseur/gamma.html
[4]) http://www.brucelindbloom.com/index.html?WorkingSpaceInfo.html

Blending the pixels of *color* images is more complicated than blending plain luminances. Although we can write down the naïve blending equation, (6.1), again for RGB-coded pixels

$$P_1 := \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} \quad \text{and} \quad P_2 := \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix}$$

and trivially arrive at

$$P := \begin{pmatrix} r \\ g \\ b \end{pmatrix} = t \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + (1-t) \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix} \quad \text{with} \quad 0 \le t \le 1, \tag{6.6}$$

but this means

- we implicitly treat the color components r_i, g_i, b_i as *separate* luminances, which they are not and moreover
- we neglect the visual aspects, namely luminance, saturation, and hue of the blended color pixel P.

6.2 Floating-Point Images

Floating-point images (EXR, floating-point TIFF, or VIFF) get a special treatment. Their values L are first converted by the Log-transform.

$$\mathrm{Log}(L) := \begin{cases} 1 + \log(1+L) & \text{for } L \ge 0 \text{ and} \\ 1/(1-L) & \text{otherwise,} \end{cases} \tag{6.7}$$

which is undone by the inverse transform after blending. Here, $\log(x)$ with a lower-case initial denotes the natural logarithmic function (i.e. to base e). Figure 6.1 shows the forward transform in the range from -20 to 100. Around $L = 0$ function $\mathrm{Log}(L)$ has the series expansion

$$\mathrm{Log}(L) = 1 + L + \frac{L^2}{2} + O(L^3), \text{for } 0 \le L < 1.$$

This transform serves two purposes:

- During blending, even completely non-negative images can result in negative pixels. A Log-transform followed by the inverse guarantees all-positive output.
- For HDR data, the Log-transform puts the samples closer to a perceptual space making the blending a little more pleasing.

In the current version of Enfuse and Enblend it is *strongly recommended* to use blending inside the RGB-cube whenever the input data is in floating-point format; this is the default, too.

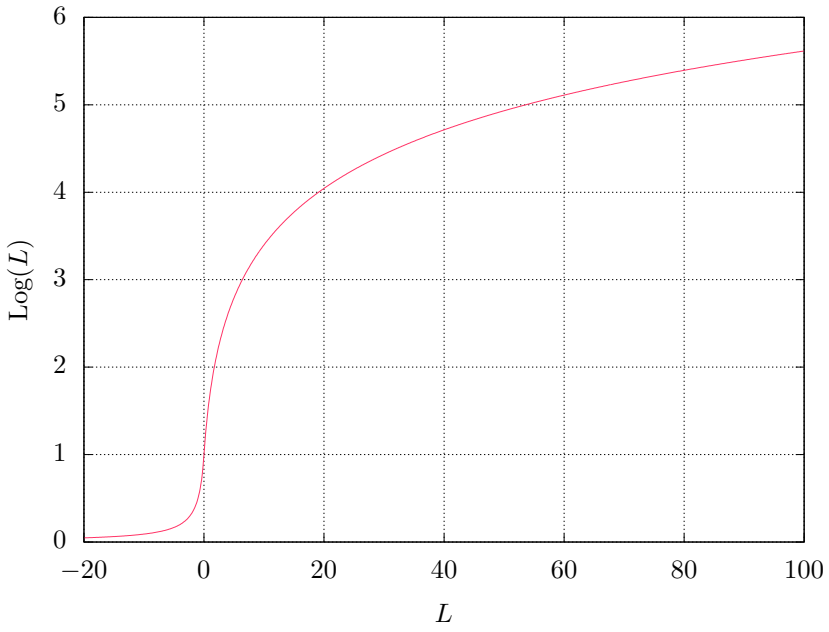


Figure 6.1: Forward Log-transform shown for the range of $-20 \leq L \leq 100$. The domain of $\mathrm{Log}(L)$ is the whole real axis.

6.3 Color Profiles

ICC-color profiles completely absorb gamma encodings (6.4) and ICC profile aware software like Enfuse and Enblend decode and encode images automatically respecting the gamma curves. Moreover color profiles define what is the darkest representable black, so called black-point

$$L = 0 \quad \text{and} \quad (r, g, b)^{\mathrm{T}} = (0, 0, 0)^{\mathrm{T}}$$

and analogously what is the purest and brightest white, the white-point

$$L = 1 \quad \text{and} \quad (r, g, b)^{\mathrm{T}} = (1, 1, 1)^{\mathrm{T}}.$$

By default, Enfuse and Enblend expect that either

1. no input image has a color profile or
2. all images come with the *same* ICC profile.

Even black-and-white images benefit from having attached appropriate profiles!

In Case 1. the applications blend grayscale images in the normalized luminance interval and color images inside the sRGB-cube. To override the default sRGB-profile select the desired profile with option `--fallback-profile`, page 25.

In Case 2. the images first are by default transformed to CIELUV[5]) color space – respecting the input color profile – then they are blended or fused, and finally the data get transformed back to RGB color space defined by the profile of the input images. Consequently, the input profile is assigned to the output image. Enforce a different blending color space than CIELUV with option `--blend-colorspace`, page 20.

Mixing different ICC profiles or alternating between images with profiles and without them generates warnings as it generally leads to unpredictable results.

Floating-Point images are an exception to the above rules. They are *always* blended in the RGB cube by default. The next section describes their treatment in detail.

6.4 Blending Color Spaces

Enfuse and Enblend offer to work inside the RGB-cube (6.5) or in several perceptually uniform color spaces. To override the default select a particular blending color space with option `--blend-colorspace`, page 20. Here are the four available color spaces.

Identity Space / RGB-Color Cube
: Calculate the blended pixel inside the luminance interval (6.1) for grayscale images and inside the RGB-color cube as given in (6.6).

[5]) https://en.wikipedia.org/wiki/CIECAM02

This is the fastest color space to do computations within, i.e. it consumes by far the least computing power, because no transform to or from any of the perceptually uniform color spaces is done.

L*A*B*[6])
: Represent each pixel as lightness L^*, red-green difference a^*, and yellow-blue difference b^*. The L*A*B* color space encompasses all perceivable colors. It is completely independent of any device characteristics, approximates human vision, and is perceptually uniform.

  Enfuse uses perceptual rendering intent and either the input profile's white-point or, if the ICC-profile lacks the `cmsSigMediaWhitePointTag`, fall back to the D50 white-point (see, e.g. Standard illuminant[7])).

  The conversions from and to L*A*B* are moderately fast to compute; L*A*B* mode is two to three times slower than working within the RGB-color cube.

CIEL*U*V*[8])
: Represent each pixel as lightness L^* and two color differences u^* and v^*. Formulas of each are too complicated to show them here.

  The L*U*V* tries to be perceptually uniform in lightness as well as in color.

  The applications use the same rendering intent and white-point as with L*A*B*.

  The conversions from and to L*U*V* are almost as fast to compute as L*A*B*.

CIECAM02[9])
: Represent each pixel as lightness J, chroma C ("colorfulness"), and hue angle h.

  Internally, the polar coordinates (C, h) are translated to Cartesian *coordinates for the pyramids.*

  The transformations to CIECAM02 color space and back use perceptual rendering intent, the D50 white point (see, e.g. Standard illuminant[10])), 500 lumen surrounding light ("average" in CIECAM02 parlance), and assume complete adaption.

  Both CIELUV and CIELAB only model the color information generated for small and isolated color samples. They cannot model the contextual effects of color perception. However, CIECAM02 can represent luminance adaptation, chromatic contrast and chromatic assimilation that arise in

[6]) https://en.wikipedia.org/wiki/Lab_color_space
[7]) https://en.wikipedia.org/wiki/Standard_illuminant
[8]) https://en.wikipedia.org/wiki/CIELUV_color_space
[9]) https://en.wikipedia.org/wiki/CIECAM02
[10]) https://en.wikipedia.org/wiki/Standard_illuminant

real world viewing conditions with heterogeneous, strongly contrasted, or three dimensional color sources.

Computationally, CIECAM02 is the most expensive blend color space. If an appreciable number of pixels need additional refinement steps the speed of the transformation further drops. Expect CIECAM02 mode to be 8–800 times slower than blending within the RGB-color cube.

Surprisingly often blending "inside the RGB-cube" works, although perceptually uniform color spaces, which represent luminance, saturation, and hue are preferable for blending and fusing operations.

6.5 Practical Considerations

- For small projects stick with the default blend colorspaces.
- For large projects switch on blending in the RGB color cube to speed up the assembly of the images. When satisfied with all other parameters use one of the computationally more expensive, but perceptually uniform color spaces.
- Banding is best fought by input images with a high bit depth ($\geq$ 16 bits per channel). A cheap and mostly vain trick is to force a large output bit depth with option `--depth`, page 21. No blend color space can avoid banding if parts of the input images are almost "monochrome".
- Enfuse only. No color space can fix blown highlights when fusing by exposure-weight; use '`--exposure-cutoff`', page 30 for this purpose and check whether saturation weight is zero.

Chapter 7

Understanding Masks[c]

A binary mask indicates for every pixel of an image if this pixel must be considered in further processing, or ignored. For a weight mask, the value of the mask determines how much the pixel contributes, zero again meaning "no contribution".

Masks arise in two places: as part of the input files and as separate files, showing the actual pixel weights prior to image blending or fusion. We shall explore both occurrences in the next sections.

7.1 Masks In Input Files

Each of the input files for Enfuse and Enblend can contain its own mask. Both applications interpret them as binary masks no matter how many bits per image pixel they contain.

Use ImageMagick's **identify** (see Example 7.1) or, for TIFF files only, **tiffinfo** (see Example 7.2) to inquire quickly whether a file contains a mask. Appendix A on page 99 shows where to find these programs on the web.

The "Matte" part of the image class and the "Extra Samples" line tell us that the file features a mask. Also, many interactive image manipulation programs show the mask as a separate channel, sometimes called "alpha". There, the white (high mask value) parts of the mask enable pixels and black (low mask value) parts suppress them.

The multitude of terms all describing the concept of a mask is confusing.

Mask
: A mask defines a selection of pixels. A value of zero represents an unselected pixel. The maximum value ("white") represents a selected pixel and the values between zero and the maximum are partially selected pixels. See Gimp-Savy.[1])

[1]) http://gimp-savvy.com/BOOK/index.html?node42.html

```
$ identify -version
Version: ImageMagick 6.7.7-10 2014-03-08 Q16
http://www.imagemagick.org
Copyright: Copyright (C) 1999-2012 ImageMagick Studio LLC
Features: OpenMP

$ identify -format "%f %m %wx%h %r %q-bit" image-0000.tif
image-0000.tif TIFF 917x1187 DirectClass sRGB Matte 16-bit
                                              ^^^^^
                                              mask
```

Example 7.1: Using **identify** to find out about the mask in *image-0000.tif*. 'Matte' indicates the existence of a mask.

Alpha Channel
: The alpha channel stores the transparency value for each pixel, typically in the range from zero to one. A value of zero means the pixel is completely transparent, thus does not contribute to the image. A value of one on the other hand means the pixel is completely opaque.

Matte
: The notion "matte" as used by ImageMagick refers to an inverted alpha channel, more precisely: $1 - \text{alpha}$. See ImageMagick[2]) for further explanations.

Enfuse and Enblend only consider pixels that have an associated mask value other than zero. If an input image does not have an alpha channel, Enblend warns and assumes a mask of all non-zero values, that is, it will use every pixel of the input image for fusion.

Stitchers like **nona** add a mask to their output images.

Sometimes it is helpful to manually modify a mask before fusion. For example to suppress unwanted objects (insects and cars come into mind) that moved across the scene during the exposures. If the masks of all input images are black at a certain position, the output image will have a hole in that position.

7.2 Weight Mask Files

FIX Show some weight masks and explain them. ME

[2]) https://www.imagemagick.org/Usage/transform/

```
$ tiffinfo
LIBTIFF, Version 4.0.2
Copyright (c) 1988-1996 Sam Leffler
Copyright (c) 1991-1996 Silicon Graphics, Inc.
...

$ tiffinfo image-0000.tif
TIFF Directory at off set 0x3a8182 (3834242)
  Subfile Type: (0 = 0x0)
  Image Width: 917 Image Length: 1187
  Resolution: 150, 150 pixels/inch
  Position: 0, 0
  Bits/Sample: 8
  Sample Format: unsigned integer
  Compression Scheme: PackBits
  Photometric Interpretation: RGB color
  Extra Samples: 1<unassoc-alpha>                mask
  Orientation: row 0 top, col 0 lhs
  Samples/Pixel: 4                               R, G, B, and mask
  Rows/Strip: 285
  Planar Configuration: single image plane
  ImageFullWidth: 3000
  ImageFullLength: 1187
```

Example 7.2: Using **tiffinfo** to find out about the mask in *image-0000.tif*. Here the line '`Extra Samples`' indicates one extra sample per pixel, which is interpreted as an unassociated alpha-channel; Enfuse and Enblend interpret this as mask. The second hint **tiffinfo** gives is in '`Samples/Pixel`', where – for a RGB-image – the $4 = 3 + 1$ tells about the extra channel.

Chapter 8

Applications of Enfuse

This section describes some of the novel possibilities that Enfuse offers the photographer. In contrast to the previous chapters, it centers around the effects on the final image.

8.1 What Makes Images Fusable?

Images should align well to be suitable for fusion. However, there is no hard mathematical rule what "well" means. The alignment requirements for 16 MPixel images to yield a sharp 4" × 6" print at 300 dpi or even for web presentation are relatively low, whereas the alignment of 8 MPixel images for a 12" × 18" print ought to be tight.

If the input images need to be aligned, Hugin (see also Appendix A.2 on page 99) is the tool of choice. It produces images exactly in the format that Enfuse expects.

Sometimes images naturally align extremely well so that no re-alignment is required. An image series with preprogrammed exposure steps taken in rapid succession where the camera is mounted on a heavy tripod and a humongous ball head, mirror lockup, and a cable release are used, comes to mind.

When in doubt about what will work, try it, and judge for yourself.
Useful ideas for a good alignment:

- Fix all camera parameters that are not explicitly varied.

  Aperture: Engage full manual or aperture-priority mode.

  Auto-focus: Disable "Auto Focus". Be aware that the auto-focus function could be linked to shutter-release button position "half pressed" or to the shutter release in insidious ways.

  Closed eyepiece: Close the eyepiece when using a cable release to suppress variations in stray light. (This tip applies only to single lens reflex cameras.)

Exposure time/Shutter speed: Use the shortest possible exposure time or, in other words, use the fastest shutter speed to avoid blur caused by camera shake or motion blur.

Flash power: Explicitly control the flash power of *all* flashes. This is sometimes called "flash exposure lock".

Sensitivity: Disable "Auto ISO".

White balance: Disable "Auto White Balance". Instead, use the most suitable fixed white balance or take the white balance off a white card. When in doubt, use the setting "Daylight" or equivalent.

- Steady the camera by any means.
  - Apply your best camera bracing technique combined with controlled breathing.
  - Prefer a monopod, or better, a rigid tripod with a heavy head.
  - Use a cable release if possible.
  - (This applies to cameras with a moving mirror only.) Engage "mirror lockup".
  - Consider automatic bracketing when applicable.
  - Activate camera- or lens-based image stabilization if you are sure that it improves the image quality in your particular case; otherwise disengage the feature.

    For some lens-based image stabilization systems, it is known that they "lock" into different positions every time they are activated. Moreover, some stabilization systems decrease the image quality if the lens is mounted on a tripod.
- Fire in rapid succession.

8.2 Repetition – Noise Reduction

Main Purpose: Reduce noise

With the default settings, Enfuse computes a weighted average of the input pixels. For a series of images, repeated with identical settings, this results in a reduction of (photon shot) noise. In other words, the dynamic range increases slightly, because the higher signal-to-noise ratio makes darker shades usable. Furthermore, smooth or glossy surfaces get a "cleaner" look, and edges become visually sharper. The nitty-gritty reportage look that sometimes stems from a high sensitivity setting disappears.

Averaged images, and therefore low-noise images, are the base for a multitude of techniques like, for example, differences. The most prominent method in this class is dark-frame subtraction.

Enfuse sets defaults for the exposure-weight to ⟨1.0⟩, for saturation-weight to ⟨0.2⟩ and for all other weights to zero, a good combination for noise reduction. Eliminating the saturation component with `--saturation-weight=0.0` sometimes can be worth the extra run.

8.3 Exposure Series – Dynamic Range Increase

Main Purpose: Increase manageable dynamic range

An exposure series is a set of images taken with identical parameters except for the exposure time. Some cameras even provide special functions to automate recording exposure series. See the instruction manual of your model for details. Also check out the features that Magic Lantern offers to shoot HDR-series and to extend the dynamic range right in the camera.

Enfuse's defaults for exposure weight, ⟨1.0⟩ and saturation weight, ⟨0.2⟩ are well suited for fusion of *color* images. Remember that saturation weighting only works for RGB data. Option `--saturation-weight` helps to control burnt-out highlights, as these are heavily desaturated. Alternatively, use option `--exposure-cutoff` to suppress noise or blown-out highlights without altering the overall brightness too much. If no image suffers from troublesome highlights, the relative saturation weight can be reduced and even be set to zero.

For black and white images '`--entropy-weight`' can be an alternative to '`--saturation-weight`' because it suppresses overexposed pixels, as these contain little information. However, entropy weighting is not limited to gray-scale data; it has been successfully applied to RGB images, too. Note that entropy weighting considers *each* color channel of an RGB image separately and chooses the channel with the minimum entropy as representative for the whole pixel.

Enfuse offers the photographer tremendous flexibility in fusing differently exposed images. Whether you combine only two pictures or a series of 21, Enfuse imposes no limits on you. Accordingly, the photographic effects achieved range from subtle to surreal, like the late 1980s "Max Headroom" TV-Series, to really unreal. Like some time ago in the chemical days of photography, when a new developer opened unseen possibilities for artists, exposure fusion extends a photographer's expressive space in the digital age. Whether the results look good or bad, whether the images are dull or exciting, is entirely up the artist.

In the next sections we give assistance to starters, and rectify several misconceptions about Enfuse.

8.3.1 Tips For Beginners

Here are some tips to get you in business quickly.

Include the best single exposure.

> Include the exposure you would have taken if you did not use Enfuse in your series. It gives you a solid starting point. Think of the other images as augmenting this best single exposure to bring out the light and dark features you would like to see.

Begin with as small a number of images as possible.
: Pre-visualizing the results of Enfuse is difficult. The more images put into the fusion process and the wider their EV-spacing is, the more challenging visualizing the output image becomes. Therefore, start off with as few images as possible.

  You can take a larger series of images and only use part of it.

Start with a moderate EV-spacing.
: As has been pointed out in the previous item, a wide EV-spacing makes pre-visualization harder. So start out with a spacing of $^2/_3$ EV to $^4/_3$ EV.

Use Magic Lantern's dual-ISO mode (and avoid Enfuse).
: Magic Lantern can reprogram a camera to take a *single* short with *two different* sensor speeds. They call it "dual-ISO" mode. The non-standard RAW image created that way must be processed with **cr2hdr**[1]), which produces a DNG file readable with the usual RAW converters.

8.3.2 Common Misconceptions

Here are some surprisingly common misconceptions about exposure series.

A single image cannot be the source of an exposure series.
: Raw-files in particular lend themselves to be converted multiple times and the results being fused together. The technique is simpler, faster, and usually even looks better than digital blending[2]) (as opposed to using a graduated neutral density filter) or blending exposures[3]) in an image manipulation program. Moreover, perfect alignment comes free of charge!

An exposure series must feature symmetrically-spaced exposures.
: Twice wrong! Neither do the exposures have to be "symmetric" like $\{0\,\text{EV}, -^2/_3\,\text{EV}, +^2/_3\,\text{EV}\}$, nor does the number of exposures have to be odd. Series like $\{-^4/_3\,\text{EV}, -^1/_3\,\text{EV}, +^1/_3\,\text{EV}\}$ or $\{-1\,\text{EV}, 1\,\text{EV}\}$ might be just right. By the way, the order in which the images were taken does not matter either.

An exposure series must cover the whole dynamic range of the scene.
: If you do not want to cover the whole range, you do not have to. Some HDR programs require the user to take a light probe, whereas Enfuse offers the user complete freedom of exposure.

> *Paul E. Debevec*[4]) *defines: "A light probe image is an omni-directional, high dynamic range image that records the incident illumination conditions at a particular point in space."*

[1]) **cr2hdr** is part of Magic Lantern's source distribution.
[2]) https://luminous-landscape.com/digital-blending/
[3]) https://www.gimp.org/tutorials/Blending_Exposures/
[4]) http://www.pauldebevec.com/

All exposure values must be different.

You can repeat any exposure as often as you like. That way you combine an exposure series with parts of Section 8.2, emphasizing the multiply occurring exposures and reducing noise.

8.4 Flash Exposure Series – Directed Lighting

Main Purpose: ???

FIX Text? ME

8.5 Polarization Series – Saturation Enhancement

Main Purpose: Reflection suppression, saturation enhancement

In the current implementation of Enfuse, it is not possible in general to fuse a polarization series. Naïvely (ab)using '`--saturation-weight`' will not work.

8.6 Focus Stacks – Depth-of-Field Increase

Main Purpose: Synthetic Depth-of-Field Increase

A focus stack is a series of images where the distance of the focal plane from the sensor varies. Sloppily speaking, the images were focused at different distances. Fusing such a stack increases the depth-of-field (DoF) beyond the physical limits of diffraction.

8.6.1 Why create focus stacks?

Given

- a fixed sensor or film size,
- a lens' particular focal length, and
- a notion about "sharpness", technically speaking the size of the circle-of-confusion (CoC)

the photographer controls the depth-of-field with the aperture. Smaller apertures – this is larger aperture numbers – increase the DoF and vice versa. However, smaller apertures increase diffraction which in turn renders the image unsharp. So, there is an optimum aperture where the photographer gets maximum DoF. Sadly, for some purposes like macro shots it is not enough. One way out is to combine the sharp parts of images focused at different distances, thereby artificially increasing the total DoF. This is exactly what Enfuse can do.

All lenses have a so called "sweet spot" aperture, where their resolution is best. Taking pictures at this aperture, the photographer squeezes the maximum quality out of the lens. But: the "sweet spot" aperture often is only one or

two stops away from wide open. Wouldn't it be great to be able combine these best-possible images to form one high-quality, sufficient-DoF image? Welcome to Enfuse's local-contrast selection abilities.

8.6.2 Preparing Focus Stacks

We are going to combine images with limited DoF to increase their in-focus parts. The whole process is about image sharpness. Therefore, the input images must align very well, not just well, but very well. For optimum results the maximum control point distance in Hugin should not exceed 0.3–0.5 pixels to ensure perfect blending.

As in all image fusion operations it is preferable to use 16 bit linear (i.e. gamma = 1) images throughout, but 8 bit gamma-encoded images will do. Naturally, high signal-to-noise (SNR) ratio input data always is welcome.

8.6.3 Local Contrast Based Fusing

A bare bones call to Enfuse for focus stacking could look like this.

```
$ enfuse \
    --exposure-weight=0 \
    --saturation-weight=0 \
    --contrast-weight=1 \
    --hard-mask \
    ... \
    --output=output.tif \
    input-<0000-9999>.tif
```

Here is what each option causes:

`--exposure-weight=0`
: Switch **off** exposure based pixel selection. The default weight is ⟨1.0⟩.

`--saturation-weight=0`
: Switch **off** saturation based pixel selection. The default weight is ⟨0.2⟩.

`--contrast-weight=1`
: Switch **on** pixel selection based on local contrast.

`--hard-mask`
: Select the best pixel from the image stack and ignore all others. Without this option, Enfuse uses all pixels in the stack and weights them according to their respective quality, which in our case is local contrast. Without '`--hard-mask`', the result will always look a bit soft. See also Section 5.4 on page 68.

If you want to see some entertaining progress messages – local-contrast weighting takes a while –, also pass the `--verbose` option for an entertaining progress report.

8.6.4 Basic Focus Stacking

For a large class of image stacks Enfuse's default algorithm, as selected in 8.6.3, to determine the sharpness produces nice results. The algorithm uses a moving square window, the so-called contrast window. It computes the standard deviation of the pixels inside of the window. The program then selects the window's center pixel of the image in the stack where the standard deviation is largest, that is, the local contrast reaches the maximum.

However, the algorithm fails to deliver good masks for images which exhibit high contrast edges on the scale of the contrast window size. The typical artifacts that show up are

- faint dark seams on the light side of the high contrast edges and
- extremely soft, slightly lighter seams on the dark side of the high contrast edges,

where the distance of the seams from the middle of the edge is comparable to the contrast window size.

If your results do not show any of these artifacts, stick with the basic algorithm. Advanced focus stacking, as described in the next sections, delivers superior results in case of artifacts, though requires manually tuning several parameters.

8.6.5 Advanced Focus Stacking

If your fused image shows any of the defects described in the previous section, you can try a more difficult-to-use algorithm that effectively works around the seam artifacts. It is described in the next section.

Detailed Look at the Problem

Let us use an example to illustrate the problem of relating the sharpness with the local contrast variations. Say we use a 5×5 contrast window. Moreover, let `sharp_edge` and `smooth_edge` be two specific configurations:

```
sharp_edge =  [   0,   0, 200,   0,   0;
                  0, 225,   0,   0,   0;
                  0, 255,   0,   0,   0;
                215,   0,   0,   0,   0;
                200,   0,   0,   0,   0]


smooth_edge = [   0,  62, 125, 187, 250;
                  1,  63, 126, 188, 251;
                  2,  65, 127, 190, 252;
                  3,  66, 128, 191, 253;
                  5,  67, 130, 192, 255]
```

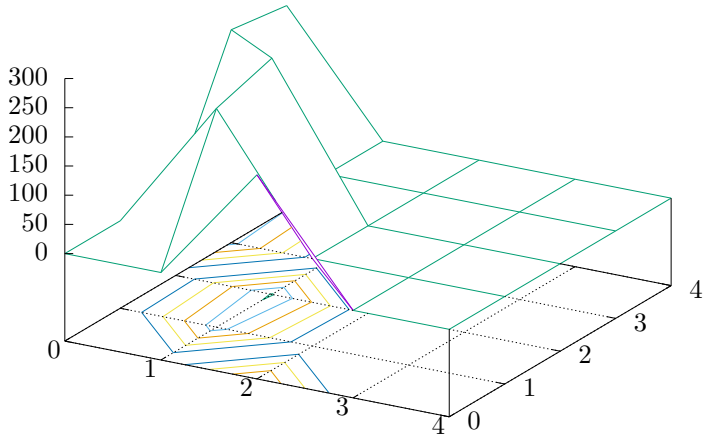


Figure 8.1: 3D-plot augmented by contour plot of the matrix `sharp_edge`.

where '`;`' separates the rows and '`,`' separates the columns. This is in fact Octave[5]) syntax. Figure 8.1 and 8.2 show plots of the matrices `sharp_edge` and `smooth_edge`.

Our intuition lets us "see" an extremely sharp edge in the first matrix, whereas the second one describes an extraordinarily smooth diagonal intensity ramp. Which one will be selected? Well, `sharp_edge` has a standard deviation of 88.07 and `smooth_edge` has 88.41. Thus, `smooth_edge` wins, contradicting our intuition, and even worse, our intention! Sadly, configurations like `smooth_edge` occur more often with high-quality, good bokeh[6]) lenses. In fact, they are the very manifestation of "good bokeh". Therefore, Laplacian edge detection plays an important role when working with high-quality lenses.

Laplacian Edge Detection

Enfuse provides a Laplacian-based algorithm that can help in situations where weighting based on the standard deviation fails. It is activated with a positive value for *SCALE* in `--contrast-edge-scale`=*SCALE*. The Laplacian will detect two-dimensional *curvature* on the scale of *SCALE*. Here and in the following

[5]) http://www.gnu.org/software/octave/
[6]) https://luminous-landscape.com/bokeh/

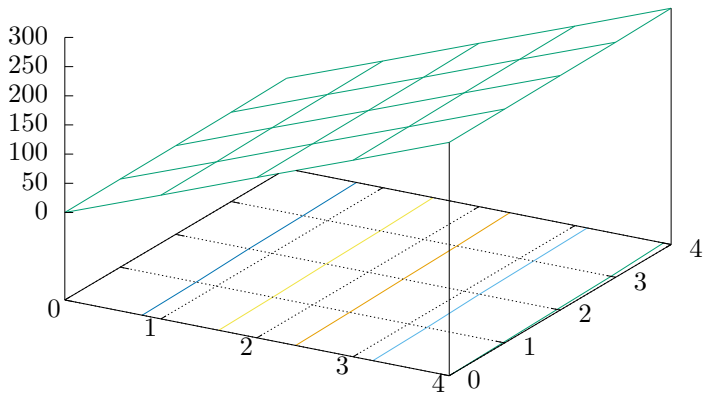


Figure 8.2: 3D-plot augmented by contour plot of the matrix `smooth_edge`.

we simply speak of "curvature" where we mean "magnitude of curvature". That is, we shall not distinguish between convex and concave edges. Enfuse always use the magnitude of curvature for weighting.

Typically, *SCALE* ranges between 0.1 pixels and 0.5 pixels, where 0.3 pixels are a reasonable starting point. To find the best value for *SCALE* though, usually some experimentation will be necessary. Use '`--save-masks`' to get all soft-mask (default: ⟨`softmask-%n.tif`⟩) and hard-mask files (default: ⟨`hardmask-%n.tif`⟩). Check how different scales affect the artifacts. Also see Chapter 7 on page 82.

Local Contrast Enhancement

Sometimes Enfuse misses smoother edges with '`--contrast-edge-scale`' and a little local contrast enhancement (LCE) helps. Set `--contrast-edge-scale`=*SCALE*:*LCE-SCALE*:*LCE-FACTOR*, where *LCE-SCALE* and *LCE-FACTOR* work like the unsharp mask[7]) filters in various image manipulation programs. Start with *LCE-SCALE* ten times the value of *SCALE* and a *LCE-FACTOR* of 2–5.

LCE-SCALE can be specified as a percentage of *SCALE*. *LCE-FACTOR* also can be specified as a percentage. Examples:

```
--contrast-edge-scale=0.3:3.0:3
--contrast-edge-scale=0.3:1000%:3.0
--contrast-edge-scale=0.3:3:300%
--contrast-edge-scale=0.3:1000%:300%
```

By default LCE is turned off.

Suppressing Noise or Recognizing Faint Edges

The Laplacian-based algorithm is much better at resisting the seam problem than the local-contrast algorithm, but it has two shortcomings:

1. The Laplacian is very susceptible to noise and
2. it fails to recognize faint edges.

The option `--contrast-min-curvature` option helps to mitigate both flaws.

The argument to `--contrast-min-curvature`=*CURVATURE* either is an absolute lightness value, e.g. 0…255 for 8 bit data and 0…65535 for 16 bit data, or, when given with a '`%`'-sign it is a relative lightness value ranging from 0% to 100%.

To suppress unreal edges or counter excessive noise, use the `--contrast-min-curvature` option with a *negative* curvature measure *CURVATURE*. This forces all curvatures less than $-$*CURVATURE* to zero.

A *positive* curvature measure *CURVATURE* makes Enfuse merge the LoG data with the local-contrast data. Every curvature larger than or equal to *CURVATURE* is left unchanged, and every curvature less than *CURVATURE*

[7]) http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm

gets replaced with the rescaled local-contrast data, such that the largest local contrast is just below *CURVATURE*. This combines the best parts of both techniques and ensures a precise edge detection over the whole range of edge curvatures.

Summary

`--contrast-edge-scale=0.3`
: Use LoG to detect edges on a scale of 0.3 pixels. Apply the default grayscale projector: `average`.

`--contrast-edge-scale=0.3 --gray-projector=l-star`
: Use LoG to detect edges on a scale of 0.3 pixels. Apply the L*-grayscale projector.

`--contrast-edge-scale=0.3:3:300%`
: Use LoG to detect edges on a scale of 0.3 pixels, pre-sharpen the input images by 300% on a scale of 3 pixels. Apply the default grayscale projector: `average`.

`--contrast-edge-scale=0.3 --contrast-min-curvature=-0.5%`
: Use LoG to detect edges on a scale of 0.3 pixels. Apply the default grayscale projector: `average` and throw away all edges with a curvature of less than 0.5%.

`--contrast-edge-scale=0.3 --contrast-min-curvature=0.5%`
`--contrast-window-size=7`
: Use LoG to detect edges on a scale of 0.3 pixels. Apply the default grayscale projector: `average` and throw away all edges with a curvature of less than 0.5% and replace the LoG data between 0% and 0.5% with SDev data. Use a window of 7×7 pixel window to compute the SDev.

Focus Stacking Decision Tree

Figure 8.3 helps the user to arrive at a well-fused focus stack with as few steps as possible.

Always start with the default, contrast weighting with a local contrast window. Only if seams appear as described in Section 8.6.5 switch to Laplacian-of-Gaussian contrast detection.

If some seams remain even in LoG-mode, decrease the sensitivity of the edge detection with a positive `--contrast-min-curvature`. A too high value of `--contrast-min-curvature` suppresses fine detail though. Part of the detail can be brought back with pre-sharpening, that is, 8.6.5 or combining LoG with local-contrast-window mode by using a negative `--contrast-min-curvature`.

Carefully examining the masks (option `--save-masks`) that Enfuse uses helps to judge the effects of the parameters.

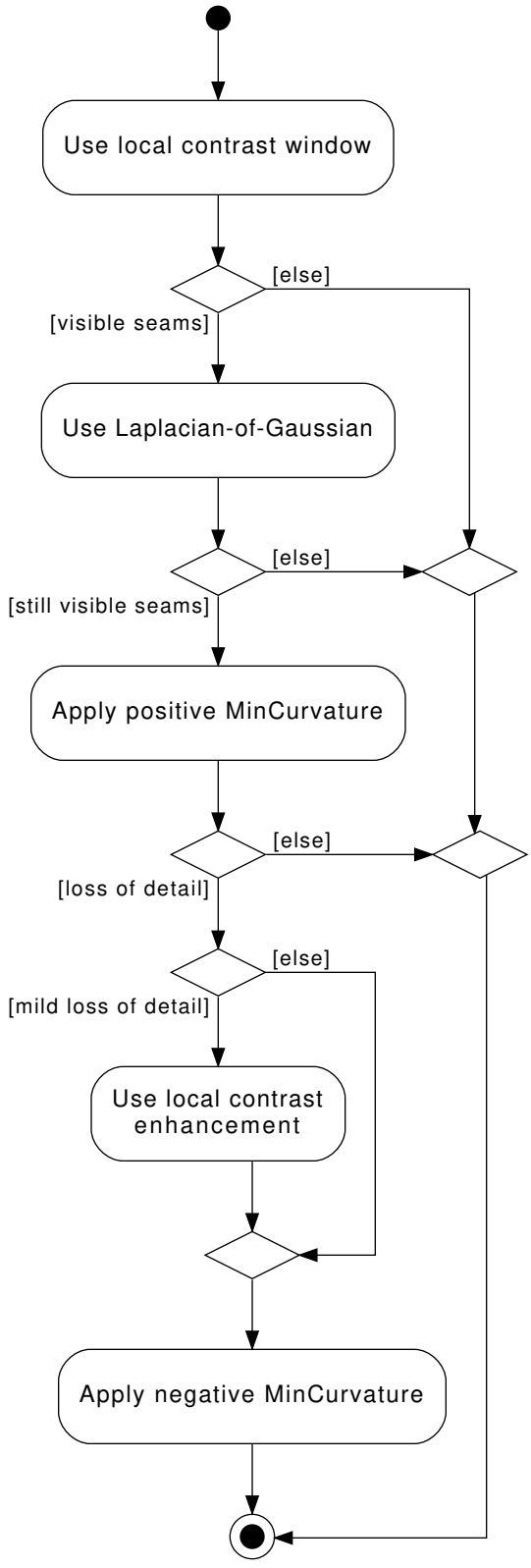


Figure 8.3: Focus stacking decision tree.

8.6.6 Tips For Focus Stacking Experts

We have collected some advice with which even focus-stacking adepts can benefit.

- Ensure that the sensor is clean.

  Aligning focus stacks requires varying the viewing angle, which corresponds to a changing focal length. Hence, the same pixel on the sensor gets mapped onto different positions in the final image. Dirt spots will occur not only once but as many times as there are images in the stack – something that is no fun to correct in post-processing.

  Along the same lines, the photographer may want to consider to prepare dark frames before, and possibly also after, the shoot of the focus stack, to subtract hot pixels before fusion.

- Prefer a low-sensitivity setting ("ISO") on the camera to get low-noise images.

  Fusing with option `--hard-mask` does not average, and thus does not suppress any noise in the input images.

- If the transition of in-focus to out-of-focus areas is too abrupt, record the images with closest and farthest focusing distances twice: first with the intended working aperture, and a second time with a small aperture (large aperture number).

  The small aperture will give the fused image a more natural in-focus to out-of-focus transition and the working-aperture shots supply the detail in the in-focus regions.

- Consider the use of Magic Lantern (Appendix A.7 on page 101) to automate the creation of focus stacks.

You have your answer, Daniel Jackson. I suggest you act on it. –
Morgan le Fay (Ganos Lal)

Appendix A

Helpful Programs And Libraries[c]

Several programs and libraries have proven helpful when working with Enfuse or Enblend.

A.1 Raw Image Conversion

- Darktable[1]) is an open-source photography workflow application and raw-image developer.
- DCRaw[2]) is a universal raw-converter written by David Coffin.
- RawTherapee[3]) is powerful open-source raw converter for Win∗, MacOS and Linux.
- UFRaw[4]) is a raw-converter written by Udi Fuchs and based on DCRaw (see above). It adds a GUI (**ufraw**), versatile batch processing (**ufraw-batch**), and some additional features such as cropping, noise reduction with wavelets, and automatic lens-error correction.

A.2 Image Alignment and Rendering

- Hugin[5]) is a GUI that aligns and stitches images.

  It comes with several command-line tools, like for example **nona** to stitch panorama images, **align_image_stack** to align overlapping images for HDR or create focus stacks, and **fulla** to correct lens errors.

[1]) http://www.darktable.org/
[2]) http://www.cybercom.net/~dcoffin/dcraw/
[3]) http://www.rawtherapee.com/
[4]) http://ufraw.sourceforge.net/
[5]) http://hugin.sourceforge.net/

- PanoTools[6]) the successor of Helmut Dersch's original PanoTools[7]) offers a set of command-line driven applications to create panoramas. Most notable are PTOptimizer for control point optimization and PTmender, an image stitcher.

A.3 Image Manipulation

- CinePaint[8]) is a branch of an early Gimp forked off at version 1.0.4. It sports much less features than the current Gimp, but offers 8 bit, 16 bit and 32 bit color channels, HDR (for example floating-point TIFF, and OpenEXR), and a tightly integrated color management system.
- The Gimp[9]) is a general purpose image manipulation program. At the time of this writing it is still limited to images with only 8 bits per channel.
- G'Mic[10]) is an open and full-featured framework for image processing, providing several different user interfaces to convert, manipulate, filter, and visualize generic image datasets.
- Both ImageMagick[11]) and GraphicsMagick[12]) are general-purpose command-line controlled image-manipulation programs, for example, **convert**, **display**, **identify**, and **montage**. GraphicsMagick bundles most ImageMagick invocations in the single dispatcher call to **gm**.

A.4 High Dynamic Range

- OpenEXR[13]) offers libraries and some programs to work with the EXR HDR-format, for example the EXR display utility **exrdisplay**.
- PFSTools[14]) read, write, modify, and tonemap high-dynamic range (HDR) images.

A.5 Major Libraries

- LibJPEG[15]) is a library for handling the JPEG (JFIF) image format.

[6]) http://panotools.sourceforge.net/
[7]) http://webuser.fh-furtwangen.de/~dersch/
[8]) http://www.cinepaint.org/
[9]) http://www.gimp.org/
[10]) http://gmic.sourceforge.net/
[11]) https://www.imagemagick.org/script/index.php
[12]) http://www.graphicsmagick.org/
[13]) http://www.openexr.com/
[14]) http://pfstools.sourceforge.net/
[15]) http://www.ijg.org/

- LibPNG[16]) is a library that handles the Portable Network Graphics (PNG) image format.
- LibTIFF[17]) offers a library and utility programs to manipulate the ubiquitous Tagged Image File Format, TIFF.

  The nifty **tiffinfo** command in the LibTIFF distribution quickly inquires the most important properties of TIFF files.

A.6 Meta-Data Handling

- EXIFTool[18]) reads and writes EXIF meta-data. In particular it copies meta-data from one image to another.
- LittleCMS[19]) is the color-management library used by Hugin, DCRaw, UFRaw, Enfuse, and Enblend. It supplies some binaries, too. **tificc**, an ICC color profile applier, is of particular interest.

A.7 Camera Firmware Extension

- Magic Lantern[20]) is a software add-on that runs from the SD (Secure Digital) or CF (Compact Flash) card and adds new features to cameras of a certain Japanese brand of cameras as for example
  - Dual-ISO (more precisely: simultaneous dual sensor speed); this operation mode may in fact obviate the need for Enfuse.
  - Focus stacking
  - HDR-bracketing

[16]) http://www.libpng.org/pub/png/libpng.html
[17]) http://www.remotesensing.org/libtiff/
[18]) http://www.sno.phy.queensu.ca/~phil/exiftool/
[19]) http://www.littlecms.com/
[20]) http://www.magiclantern.fm/index.html

Appendix B

Bug Reports[c]

Most of this appendix was taken from the Octave[1]) documentation.

Bug reports play an important role in making Enfuse and Enblend reliable and enjoyable.

When you encounter a problem, the first thing to do is to see if it is already known. To this end, visit the package's LaunchPad[2]) bug database[3]). Search it for your particular problem. If it is not known, please report it.

In order for a bug report to serve its purpose, you must include the information that makes it possible to fix the bug.

B.1 Have You Really Found a Bug?

If you are not sure whether you have found a bug, here are some guidelines:

- If Enfuse or Enblend get a fatal signal, for any options or input images, that is a bug.
- If Enfuse or Enblend produce incorrect results, for any input whatever, that is a bug.
- If Enfuse or Enblend produce an error message for valid input, that is a bug.
- If Enfuse or Enblend do not produce an error message for invalid input, that is a bug.

[1]) http://www.gnu.org/software/octave/
[2]) https://launchpad.net/
[3]) ⟨https://bugs.launchpad.net/enblend⟩

B.2 How to Report Bugs

The fundamental principle of reporting bugs usefully is this: report all the facts. If you are not sure whether to state a fact or leave it out, state it. Often people omit facts because they think they know what causes the problem and they conclude that some details do not matter. Play it safe and give a specific, complete example.

Keep in mind that the purpose of a bug report is to enable someone to fix the bug if it is not known. Always write your bug reports on the assumption that the bug is not known.

Try to make your bug report self-contained. If we have to ask you for more information, it is best if you include all the previous information in your response, as well as the information that was missing.

To enable someone to investigate the bug, you should include all these things:

- The exact version and configuration of Enfuse. You can get the data by running enfuse with the options `--version` and `--verbose` together. See also Section 3.3.1 on page 7 on how to find out the exact configuration of your binary.
- A complete set of input images that will reproduce the bug. Strive for a minimal set of *small* images, where images up to 1500×1000 pixels qualify as small.
- The type of machine you are using, and the operating system name and its version number.
- A complete list of any modifications you have made to the source. Be precise about these changes. Show a delta generated with **diff** for them.
- Details of any other deviations from the standard procedure for installing Enfuse and Enblend.
- The *exact command line* you use to call Enfuse or Enblend, which then triggers the bug.

  Examples:

```
$ ~/local/bin/enblend -v \
    --fine-mask \
    --optimizer-weights=3:2 \
    --mask-vectorize=12.5% \
    image-1.png image-2.png
```

  or:

```
$ /local/bin/enfuse \
    --verbose \
```

```
    --exposure-weight=0 --saturation-weight=0
--entropy-weight=1 \
    --gray-projector=l-star \
    --entropy-cutoff=1.667% \
    layer-01.ppm layer-02.ppm layer-03.ppm
```

If you call Enfuse or Enblend from within a GUI like, for example, Hugin[4]) or ImageFuser[5]) by Harry van der Wolf, copy&paste or write down the command line that launches Enfuse or Enblend.

- A description of what behavior you observe that you believe is incorrect. For example, "The application gets a fatal signal," or, "The output image contains black holes."

  Of course, if the bug is that the application gets a fatal signal, then one cannot miss it. But if the bug is incorrect output, we might not notice unless it is glaringly wrong.

B.3 Sending Patches for Enfuse or Enblend

If you would like to write bug fixes or improvements for Enfuse or Enblend, that is very helpful. When you send your changes, please follow these guidelines to avoid causing extra work for us in studying the patches. If you do not follow these guidelines, your information might still be useful, but using it will take extra work.

- Send an explanation with your changes of what problem they fix or what improvement they bring about. For a bug fix, just include a copy of the bug report, and explain why the change fixes the bug.
- Always include a proper bug report for the problem you think you have fixed. We need to convince ourselves that the change is right before installing it. Even if it is right, we might have trouble judging it if we do not have a way to reproduce the problem.
- Include all the comments that are appropriate to help people reading the source in the future understand why this change was needed.
- Do not mix together changes made for different reasons. Send them individually.

  If you make two changes for separate reasons, then we might not want to install them both. We might want to install just one.
- Use the version control system to make your diffs. Prefer the unified diff[6]) format: `hg diff --unified 4`.

[4]) http://hugin.sourceforge.net/
[5]) http://imagefuser.sourceforge.net/onlinemanual/
[6]) https://en.wikipedia.org/wiki/Diff#Unified_format

- You can increase the probability that your patch gets applied by basing it on a recent revision of the sources.

Appendix C

Authors[C]

Andrew Mihal (acmihal@users.sourceforge.net) has written Enblend and Enfuse.

Contributors (in alphabetical order)

- Pablo d'Angelo (dangelo@users.sourceforge.net) added the contrast criteria.
- Joe Beda: Win32 porting up to version 3.2.
- Kornel Benko, kornelbenko@users.sourceforge.net: CMake support for version 4.0.
- Roger Goodman: Proofreading of the manuals.
- Mikołaj Leszczyński, rosomack@users.sourceforge.net: GraphCut algorithm in Enblend.
- Max Lyons.
- Mark aka 'mjz': Win32 porting up to version 3.2.
- Thomas Modes, tmodes@users.sourceforge.net: continuous Win32 porting and permanent CMake support.
- Ryan Sleevi, ryansleevi@users.sourceforge.net: Win32 porting of version 4.0.
- Christoph Spiel (cspiel@users.sourceforge.net) added the gray projectors, the LoG-based edge detection, an $O(n)$-algorithm for the calculation of local contrast, entropy weighting, and various other features.
- Brent Townshend, btownshend@users.sourceforge.net: HDR support.

Thanks to Simon Andriot and Pablo Joubert for suggesting the Mertens-Kautz-Van Reeth technique and the name "Enfuse".

Appendix D

The GNU Free Documentation License[c]

Version 1.2, November 2002

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110–1301, USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that

contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple

HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING
You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY
If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover.

Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS
   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
   D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for

example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS
   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS
   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License

into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS
   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION
   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION
   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE
    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions

will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See GNU Copyleft[1]).

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

[1]) http://www.gnu.org/copyleft/

Syntactic Comment Index

enblend-response-file, 42
enfuse-response-file, 42
filename-globbing, 43
globbing, 43
glob, 43
layer-selector, 43
response-file, 42

Program/Application Index

If a program belongs to a larger package, it has its association mentioned in parenthesis.

CinePaint, 100
Cinepaint, 22
Darktable, 99
Gimp, 5, 22, 34, 100
GraphicsMagick, 100
Hugin, 5, 47, 85, 99, 104
ImageFuser, 104
ImageMagick, 32, 34, 100
LittleCMS, 101
Magic Lantern, 101
Mercurial, 10
PanoTools, 5, 99
RawTherapee, 99
UFRaw, 99
`PFSTools`, 100
`PTOptimizer` (PanoTools), 99
`PTmender` (PanoTools), 99
`align_image_stack` (Hugin), 99
`convert` (ImageMagick), 47, 100
`convert`, 32
`dcraw`, 5, 99
`display` (ImageMagick), 47, 100
`exiftool`, 101
`exrdisplay` (OpenEXR), 100
`fulla` (Hugin), 99
`gm` (GraphicsMagick), 100
`gmic`, 100
`identify` (ImageMagick), 82, 100
`montage` (ImageMagick), 100
`nona` (Hugin), 22, 47, 83, 99
`tiffcp` (LibTIFF), 47
`tiffinfo` (LibTIFF), 47
`tiffinfo` (libtiff), 82, 101
`tiffsplit` (LibTIFF), 47
`tificc` (LittleCMS), 101
`ufraw-batch`, 99
`ufraw`, 5, 99

Option Index

Locations marked like 12 indicate the place of the option's genuine description.

--blend-colorspace, 20
--ciecam, 21
--compression, 17
--contrast-edge-scale, 27
--contrast-min-curvature, 29
--contrast-weight, 24
--contrast-window-size, 29
--depth, 21
--entropy-cutoff, 29
--entropy-weight, 24
--entropy-window-size, 30
--exposure-cutoff, 30, 57
--exposure-optimum, 24, 50, 51, 59
--exposure-weight-function, 34, 51, 58
--exposure-weight, 24
--exposure-width, 24, 51, 59
--fallback-profile, 25
--gray-projector, 35, 50
--hard-mask, 25, 49, 97
--help, 38
--layer-selector, 25
--levels, 18
--load-masks, 7, 26
--no-ciecam, 21
--no-parameter, 26
--output, 7, 19
--parameter, 26
--saturation-weight, 25
--save-masks, 7, 27
--show-globbing-algorithms, 38
--show-image-formats, 11, 38
--show-signature, 12, 38
--show-software-components, 12, 38, 60
--soft-mask, 25
--verbose, 10, 19
--version, 9, 10, 39
--wrap, 23
-V (long: --version), 39
-c (long: --ciecam), 21
-d (long: --depth), 21
-f, 22
-g, 22
-h (long: --help), 38
-l (long: --levels), 18
-o (long: --output), 19
-v (long: --verbose), 19
-w (long: --wrap), 23

General Index

Locations marked like 12 indicate the location where a term gets introduced or defined. Tables, lists, etc. that summarize material are indicated like *34*.

Symbols

#(response file comment), 41
@ (response file prefix), 40

Numbers

360°
    horizontal panorama, *see* panorama, 360°
    vertical panorama, *see* panorama, 360°

A

active criteria, *see* criteria, active
Adelson, Edward H., 1
advanced focus stacking, *see* focus stacking, advanced
    recognizing faint edges, 94
    suppressing noise, 94
advanced options, 20–24
affine transformation, *see* transformation, affine
algorithm, *see* globbing algorithm
    globbing, 38
algorithms
    globbing, 44
alignment
    photometric, 6
alpha, 82
alpha channel
    associated, 22
`anti-value` gray projector, *see* gray projector, `anti-value`
aperture
    sweet spot, 89
applications of Enfuse, 85–97
arithmetic JPEG compression, *see* compression
associated alpha channel, *see* alpha channel, associated
authors, 106
average
    disabling, 49
    weighted, 49
`average` gray projector, *see* gray projector, `average`

B

basic focus stacking, *see* focus stacking, basic
BigTIFF, 3
binary mask, *see* mask, binary
binary version, *see* software, version
bit depth, *see* bits per channel
bits per channel, 12, 21
black-and-white image, *see* image, black-and-white
black-point, 79
blend colorspace, *see* colorspace, blend
blending color space, *see* colorspace, for blending and fusing
blending exposures, 88
blending pixels, *see* pixels, blending
Born, Max, 23
bracketing

HDR, 101
branches of Enfuse/Enblend
development, 9
Brasseur, Eric, 76
bug
database at LaunchPad, 102
reports, 102–105
how to, 103
identification of bugs, 102
sending patches, 104
builder, 12
Burt, Peter J., 1
Burt-Adelson, 1

C

Carter, Samantha, 46
channel
alpha, 2, 6, 12, 82
depth, 21
width, *see* channel, depth
`channel-mixer` gray projector, *see* gray projector, `channel-mixer`
CIECAM02 colorspace, *see* colorspace, CIECAM02, *see* colorspace, CIECAM02
CIEL*a*b* colorspace, *see* colorspace, CIEL*a*b*
CIEL*u*v* colorspace, *see* colorspace, CIEL*u*v*, *see* colorspace, CIEL*u*v*
`CILK_NWORKERS`, *see* environment variable, `CILK_NWORKERS`
circle-of-confusion, 89
CoC, *see* circle-of-confusion
code,
see also return code, 13
code repository, *see* public repository
coding guidelines, 60
color appearance model, 20
color cube
RGB, 20, 79
sRGB, 79
RGB, 20
color profile, 75, 79
color profiles
math, 76
colorspace, 75–81
CIECAM02, 20, 80
CIEL*a*b*, 20
CIEL*u*v*, 20, 80
HSL, 67
L*a*b*, 80
blend, 20
CIELUV, 20
for blending and fusing, 79
practical considerations, 81
command-line, 16
common options, 17–19
compiled-in features, *see* features
compiler, 12
compression, 17
JPEG, 17
JPEG of TIFF, 18
LZW, 18
arithmetic JPEG, 17
deflate, 18
packbits, 18
console messages, 13
contrast
local enhancement, 29
minimum curvature, 29
window size, 29
contrast enhancement
local, 94
contrast weight, *see* weight, local contrast
contrast weighting, *see* weighting, local contrast
contrast weighting using Laplacian-of-Gaussian, *see* weighting, contrast using Laplacian-of-Gaussian
contrast weighting using a blend of methods, *see* weighting, contrast using a blend of methods
contrast weighting using standard deviation, *see* weighting, contrast using standard deviation

conventions
typographic, *see* notation
conversion
raw, 4
RGB-L*A*B*, *see* RGB-L*A*B* conversion
criteria
active, 49
overpowering one another, 50
cutoff
exposure, *see* exposure, cutoff
cutoff entropy, *see* entropy, cutoff

D

D50 white-point, *see* white-point, D50
dark frame, 97
decision tree
focus stacking, 95
default layer selection, 43
default output filename, *see* filename, output, default
default weights, *see* weights, default
deflate compression, *see* compression
delimiters,
see also options, delimiters, 39
depth-of-field, 1, 89
depth-of-focus increase, 89
detailed configuration, 7
development branch, *see* branches, development
digital blending, 88
disabling average, *see* average, disabling
dlopen
Linux, 58
OS X, 58
DoF, *see* depth-of-field, *see* depth-of-field
double precision float (IEEE754), *see* IEEE754, double precision float
dual-ISO, 101
DYLD_FALLBACK_LIBRARY_PATH, *see* environment variable, DYLD_FALLBACK_LIBRARY_PATH
DYLD_LIBRARY_PATH, *see* environment variable, DYLD_LIBRARY_PATH
dynamic library, 34, 57
dynamic linking, *see* linking, dynamic
dynamic linking support, 59
dynamic loading, *see* loading, dynamic
dynamic range increase, 87, 89
dynamic-library environment, 38

E

edge detection
Laplacian, 92
entropy, *see* weighting, local entropy
cutoff, 29
definition, 73
window size, 30
entropy weight, *see* weight, entropy
entropy weighting, *see* weighting, local entropy
environment,
see also environment variable, 15
environment variable, *15*, 15
CILK_NWORKERS, 15
DYLD_FALLBACK_LIBRARY_PATH, 58
DYLD_LIBRARY_PATH, 58
LD_LIBRARY_PATH, 58
OMP_DYNAMIC, 15
OMP_NUM_THREADS, 15
TMPDIR, 15
estimators, 69
EXIF, 101
expectation value, 68
expert focus stacking tips, *see* tips, focus stacking experts
expert fusion options, 27–38
expert options, 25–27

exposure
cutoff, 30
optimum, 24
weight, 24
width, 24
exposure fusion, *see* fusion, exposure
exposure series, *see* series, exposure
common misconceptions, 88
tips for beginners, 87
exposure weight
function, *see* weight, function, exposure
linear transform, 35
exposure weight function, *36*
FWHM, 35
`bi-square`, 52
`bisquare`, 52
`full-sine`, 52
`fullsine`, 52
`gauss`, 50, 52
`half-sine`, 52
`halfsine`, 52
`lorentz`, 52
Gaussian, 50, 52
Lorentzian, 52
Lorentz curve, 52
bi-square, 36
bisquare, 36
full width half maximum, 35
full-sine, 36
fullsine, 36
gauss, 36
gaussian, 36
half-sine, 36
halfsine, 36
lorentz, 36
lorentzian, 36
exposure weight functions, 52
exposure weighting, *see* weighting, exposure, *see* weighting, exposure
functions, 48
EXR, 100
EXR image, *see* image, EXR
external mask, 6
extra features, *see* features
extra samples, 82

F

fallback profile, *see* profile, fallback
FDL, *see* GNU Free Documentation License
features, 10
file
Mach-O, 58
multi-page, 45
response, 16, 40
filename
literal, 16
output, 19
default, 19
filename template, *see* mask, filename template
first moment, *see* moment, first
flash exposure series, *see* series, flash exposure
floating-point TIFF image, *see* image, floating-point TIFF
floating-point TIFF mask, *see* mask, floating-point TIFF
floating-point VIFF image, *see* image, floating-point VIFF
floating-point image, *see* image, floating-point
focus stacking
advanced, 91
basic, 91
focus stacking decision tree, *see* decision tree, focus stacking
focus stacks, 89
fusing, 90
preparation, 90
why to create them?, 89
format
image, 38
format of response file, *see* response file format
full width half maximum, 35
fusing
local-contrast-based, 90

single criterion, 49
fusing pixels, *see* pixels, fusing
fusion
exposure, 1
fusion options, 24–25
FWHM, 24, *see* full width half maximum

G

Ganos Lal, *see* Morgan le Fay
glob, 44
globbing algorithm, 43
`literal`, 43, 44
`none`, 44
`shell`, 44
`sh`, 44
`wildcard`, 43, 44
globbing algorithms, *see* algorithm, globbing, 44
GNU FDL, *see* GNU Free Documentation License
GNU Free Documentation License, 107
grammar
response file, *see* response file, grammar
syntactic comment, 43
gray projector, 35
`anti-value`, 35
`average`, 35
`channel-mixer`, 35
`l-star`, 37
`lightness`, 37
`luminance`, 37
`pl-star`, 37
`value`, 37
grayscale projector, *see* projector, grayscale

H

half precision float (OpenEXR), *see* OpenEXR, half precision float
hard mask, *see* mask, hard
HDR, *see* high dynamic range, 100
HDR-bracketing, *see* bracketing, HDR
header files, 38
help, 38
helpful libraries, 100
helpful programs, 99–101
HDR, 100
camera firmware, 101
High Dynamic Range, 100
image alignment, 99
image manipulation, 100
image rendering, 99
libraries, 100
meta-data handling, 101
raw image conversion, 99
high dynamic range, 1
hot pixels, 97
HSL colorspace, *see* colorspace, HSL

I

ICC, 101
ICC profile, *see* profile, ICC, *see* profile, ICC
ICC profile black-point, *see* black-point
ICC profile white-point, *see* white-point
ICC profile, *see* profile, ICC
IEEE754
double precision float, 22
single precision float, 21
image
EXR, 77
black-and-white, 79
directory, 45
floating-point, 77
floating-point TIFF, 77
floating-point VIFF, 77
frame, 47
image formats, 11, 12, *see* format, image
JPEG, 11
OpenEXR, 12
PNG, 11
TIFF, 12

image processing order, *see* order of image processing
image requirements, *see* requirements, image
images
  fusable, 85
information
  on software components, 38
information options, 38–39
input mask, *see* mask, input files
interaction with Enfuse, 7–15
invocation, 16–47

J

Jackson, Daniel, 18, 26, 98
JFIF, *see* JPEG
JPEG, 100
JPEG compression, *see* compression
JPEG quality level, *see* compression

K

von Kármán, Theodore, 23
Kautz, Jan, 1
known limitations, 3

L

L*a*b* colorspace, *see* colorspace, L*a*b*
`l-star` gray projector, *see* gray projector, `l-star`
Laplacian edge detection, *see* edge detection, Laplacian
Laplacian-of-Gaussian, 27, 70
LaunchPad
  bug database, *see* bug, database at LaunchPad
layer
  image, 45
  selection, 45–47
layer selection, 25
  all layers, 25
  default, 43
  first layer, 25
  largest-layer, 26
  last layer, 25
  no layer, 26
layer selection syntax, 45
LCE, *see* local contrast enhancement
`LD_LIBRARY_PATH`, *see* environment variable, `LD_LIBRARY_PATH`
lens distortion
  correction of, 6
level
  verbosity, 19
levels
  pyramid, 18
LibJPEG, 100
LibPNG, 100
libraries, 12
library
  dynamic, *see* dynamic library, *see* dynamic library
LibTiff, 101
light probe, 88
`lightness` gray projector, *see* gray projector, `lightness`
Lindbloom, Bruce, 76
linking
  dynamic, 57
literal filename, *see* filename, literal
load mask, *see* mask, loading
loading
  dynamic, 57
LoadLibrary (Windows), 59
local analysis window, *see* window, local-analysis
local contrast, *see* weighting, local contrast
local contrast enhancement, *see* contrast, local enhancement, *see* contrast enhancement, local
local contrast problem, 91
local contrast weight, *see* weight, local contrast

local contrast weighting, *see* weighting, local contrast
local entropy, *see* weighting, local entropy
local entropy weighting, *see* weighting, local entropy, 73–74
local-contrast-based fusing, 90
LoG, *see* Laplacian-of-Gaussian
LoG, *see* Laplacian-of-Gaussian
log-transform, *see* transform, log
luminance, 75
`luminance` gray projector, *see* gray projector, `luminance`
luminance interval, 76
  normalized, 50, 76
  trivial, 20, 25
LZW compression, *see* compression

M

mask, 82
  binary, 82
  external, 6
  filename template, 26, 27
  floating-point TIFF, 27
  hard, 25
  input files, 82
  loading, 26
  save, 27
  soft, 25
  template character, *28*
    '`%`', 28
    '`B`', 28
    '`b`', 28
    '`D`', 28
    '`d`', 28
    '`E`', 28
    '`e`', 28
    '`F`', 28
    '`f`', 28
    '`i`', 28
    '`n`', 28
    '`P`', 28
    '`p`', 28
  weight, 82, 83
Math-O file, *see* file, Mach-O
matte, 82, 83
Mertens, Tom, 1
message
  category, 13
    error, 13
    info, 14
    note, 14
    timing, 14
    warning, 13
  console, 13
  debug, 14
  foreign sources, 14
  "should never happen", 14
mode of operation (SDev, LoG, ...), 73
moment
  first, 68
  second, 69
Morgan le Fay, 98
multi-page file,
see also layer, 45, *see* file, multi-page
  tools, 47
multi-resolution spline, 1

N

name of builder, 12
natural sharp-unsharp transition, 97
noise reduction, 86
normalized luminance interval, *see* luminance interval, normalized
notation, vii

O

O'Neill, Jack, 27
object
  shared, *see* shared object, *see* shared object
`OMP_DYNAMIC`, *see* environment variable, `OMP_DYNAMIC`
`OMP_NUM_THREADS`, *see* environment variable, `OMP_NUM_THREADS`
only save mask, *see* save mask only

OpenEXR, 100
    data format, 22
    half precision float, 22
OpenMP, 13, 15
optimum exposure, *see* exposure, optimum
option delimiters, *see* options, delimiters
options, 17–39
    advanced, 20
    common, 17
    delimiters, 39
        filename arguments, 40
        numeric arguments, 39
    expert, 25
    fusion, 24
    fusion for experts, 27
    information, 38
order
    of image processing, 40
output
    file compression, 17
output filename, *see* filename, output
    default, *see* filename, output, default
output image
    set size, 22
overpowering criteria, *see* criteria, overpowering
overview, 1–2

P

packbits compression, *see* compression
panorama
    360°
        horizontal, 23
        vertical, 23
parallax error, 6
perceptual rendering intent, *see* rendering intent
performance considerations, 61
photographic workflow, 4–15
photometric alignment, *see* alignment, photometric
pixels
    blending, 75
    fusing, 75
    hot, 97
`pl-star` gray projector, *see* gray projector, `pl-star`
PNG, 100
polarization series, *see* series, polarization
probability function, 68
problem reports, *see* bug reports
profile
    fallback, 25
    ICC, 12, 20, 75
projector
    grayscale, 50
projector to grayscale, *see* gray projector
public repository, 10
pyramid levels, *see* levels, pyramid

Q

query
    compiler, 12
    features, 10
    image formats, 11
    libraries, 12
    name of builder, 12
query version, *see* version, query

R

random variable, 68
raw conversion, *see* conversion, raw
Van Reeth, Frank, 1
rendering intent
    perceptual, 80
requantization, 21
requirements
    image, 16
response file, *see* file, response
    comment (‘`#`’), *see* ‘`#`’
    force recognition of, 42
    format, 41
    grammar, 41
    syntactic comment, 42
response file prefix

'@', *see* '@'
response files, 40–45
return code, 13
RGB color cube, *see* color cube, RGB, *see* color cube, RGB
RGB'-L*A*B* conversion, 37
RGB-cube, 25
RGB-L*A*B* conversion, 37

S

saturation enhancement, 89
saturation weight, 25
saturation weighting, *see* weighting, saturation, *see* weighting, saturation
save mask, *see* mask, save
  only, 27
scaling of parameters, 73
second moment, *see* moment, second
sensor
  use clean, 97
series
  exposure, 87
  flash exposure, 89
  polarization, 89
  simple, 86
shared object, 34, 57
signal-to-noise ratio, 90
signature, 12, 38
signed binary, *see* signature
simple series, *see* series, simple
single criterion fusing, *see* fusing, single criterion
single precision float (IEEE754), *see* IEEE754, single precision float
size
  canvas, 22
SNR, *see* signal-to-noise ratio
soft mask, *see* mask, soft
software
  components, *see* information, on software components
  version, 39
software version, *see* version, software
source code repository, *see* public repository
SourceForge, 2
spline, *see* multi-resolution spline
sRGB, 20
sRGB color cube, *see* color cube, sRGB
standard deviation, 69
standard workflow, *see* workflow, standard
statistical moments, 68
subtraction of dark frame, 97
sweet spot aperture, *see* aperture, sweet spot
syntactic comment
  grammar, 43
  response file, *see* response file, syntactic comment
syntax
  layer selection, 45
    grammar, 45

T

TIFF, 101
tips
  focus stacking experts, 97
TMPDIR, *see* environment variable, TMPDIR
transform
  floating-point images, 77
  log, 77, 78
transformation
  affine, 6
transition
  natural sharp-unsharp, 97
typographic conventions, *see* notation

U

unassociated alpha channel, *see* alpha channel, associated
understanding masks, 82–83

V

`value` gray projector, *see* gray projector, `value`
variable, *see* environment variable
variance, 69
verbosity level, *see* level, verbosity
version, *see* software, version
  query, 9
  software, 9
VIFF floating-point image, *see* image, floating-point VIFF
virtual reality, 23
VR, *see* virtual reality

W

weight
  entropy, 24
  exposure, *see* exposure, weight
  function
    exposure, 34
  local contrast, 24
  mask, *see* mask, weight
  saturation, *see* saturation weight
weighted average, *see* average, weighted
weighting
  contrast using Laplacian-of-Gaussian, 70
  contrast using a blend of methods, 72
  contrast using standard deviation, 68
  entropy, 67
  exposure, 1, 50
    built-in, 50
    prerequisites, 59
    user-defined, 57
  general concept of, 48
  local contrast, 1, 68
  local entropy, 2, 73
    window size, 73
  saturation, 1, 67
weighting factor, 75
weighting functions, 48–74
weights
  default, 50
white-point, 79
  D50, 80
width of exposure weight curve, *see* exposure, width
window
  local-analysis, 68
window size
  contrast, *see* contrast, window size
  entropy, *see* entropy, window size
workflow, *see* photographic workflow
  Enblend, 5
  Enfuse, 5
  external mask, 8
  standard, 4
wrap around, 23