Combining Multiple Images with Enblend 4.2

Andrew Mihal*) Christoph Spiel

2026-09-13†)

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

Abstract

This manual is for Enblend version ⟨4.2⟩, a tool for compositing images in such a way that the seam between the images is invisible, or at least very difficult to see.

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] 5

4 Invocation 17

5 Seam Generators 47

6 Visualization Image 49

7 Color Spaces[c] 52

8 Understanding Masks[c] 59

A Helpful Programs[c] 63

B Bug Reports[c] 66

C Authors[c] 70

D GNU FDL[c] 71

Indices 79

Contents

List of Tables v
List of Figures vi
List of Examples vii

1 Overview **1**

2 Known Limitations **3**

3 Workflow[c] **5**
3.1 Standard Workflow 5
3.2 External Masks 7
3.3 Interacting with Enblend[c] 8
3.3.1 Finding Out Details 8
3.3.2 Console Messages 14
3.3.3 Environment Variables 16

4 Invocation **17**
4.1 Image Requirements 17
4.2 Command-Line Options 18
4.2.1 Common Options[c] 18
4.2.2 Advanced Options[c] 20
4.2.3 Mask Generation Options 25
4.2.4 Expert Options 28
4.2.5 Expert Mask Generation Options 30
4.2.6 Information Options[c] 34
4.2.7 Program Flow Charts 35
4.3 Option Delimiters[c] 37
4.3.1 Numeric Arguments 37
4.3.2 Filename Arguments 37
4.4 Response Files[c] 39
4.4.1 Response File Format 40
4.4.2 Syntactic Comments 41
4.4.3 Globbing Algorithms 42
4.4.4 Default Layer Selection 42
4.5 Layer Selection[c] 44

4.5.1 Layer Selection Syntax . . . 44
4.5.2 Tools for Multi-Page Files . . . 46

5 Seam Generators **47**

6 Visualization Image **49**

7 Color Spaces[c] **52**
7.1 Mathematical Preliminaries . . . 53
7.2 Floating-Point Images . . . 54
7.3 Color Profiles . . . 56
7.4 Blending Color Spaces . . . 56
7.5 Practical Considerations . . . 58

8 Understanding Masks[c] **59**
8.1 Masks In Input Files . . . 59
8.2 Weight Mask Files . . . 60

A Helpful Programs[c] **63**
A.1 Raw Image Conversion . . . 63
A.2 Image Alignment and Rendering . . . 63
A.3 Image Manipulation . . . 64
A.4 High Dynamic Range . . . 64
A.5 Libraries . . . 64
A.6 Meta-Data Handling . . . 65
A.7 Camera Firmware Extension . . . 65

B Bug Reports[c] **66**
B.1 Found a Bug? . . . 66
B.2 How to Report Bugs . . . 67
B.3 Sending Patches . . . 68

C Authors[c] **70**

D GNU FDL[c] **71**

Indices **79**
Syntactic Comment Index . . . 79
Program/Application Index . . . 80
Option Index . . . 81
General Index . . . 82

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

List of Tables

3.1 Image formats and bit-depths 13

4.1 Verbosity levels 21
4.2 Mask generation options 26
4.3 Optimizer strategies 27
4.4 Mask template characters 29
4.5 Grammar of response files 40
4.6 Grammar of syntactic comments 42
4.7 Globbing algorithms 43
4.8 Grammar of layer specifications 44

6.1 Visualization colors an symbols 50

List of Figures

3.1 Photographic workflow 6
3.2 External mask workflow 9

4.1 Internal work flow 36
4.2 Optimizer chain 38

6.1 Seam-line visualization 51

7.1 Log-transform 55

List of Examples

3.1 Output of '`enblend -version`' 10
3.2 Output of '`enblend -version -verbose`' 12
3.3 Output of '`enblend -show-software-components`' 14
4.1 Complete response file 41
4.2 Filename-globbing syntactic comment 42
8.1 Using **identify**. 60
8.2 Using **tiffinfo**. 61

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, Enblend
Command	Name of a binary in the running text	**convert**, **enblend**
Common part	Chapter, section, or any other part that appears in both manuals	Response Files[c]
Default	Value as compiled into the **enblend** binary that belongs to this documentation	⟨1⟩, ⟨*a.tif*⟩
Environment variable	Variable passed to enblend 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 enblend	`-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	Enblend.
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

Enblend overlays multiple images using the Burt-Adelson multi-resolution spline algorithm.[1]) This technique tries to make the seams between the input images invisible. The basic idea is that image features should be blended across a transition zone proportional in size to the spatial frequency of the features. For example, objects like trees and windowpanes have rapid changes in color. By blending these features in a narrow zone, you will not be able to see the seam because the eye already expects to see color changes at the edge of these features. Clouds and sky are the opposite. These features have to be blended across a wide transition zone because any sudden change in color will be immediately noticeable.

Enblend expects each input file to have an alpha channel. The alpha channel should indicate the region of the file that has valid image data. Enblend compares the alpha regions in the input files to find the areas where images overlap. Alpha channels can be used to indicate to Enblend that certain portions of an input image should not contribute to the final image.

Enblend does *not* align images. Use a tool such as Hugin or PanoTools to do this. The TIFF files produced by these programs are exactly what Enblend is designed to work with. Sometimes these GUIs allow to select feathering for the edges the images. This treatment is detrimental to Enblend. Turn off feathering by deselecting it or setting the "feather width" to zero.

Enblend blends the images in the order they are specified on the command line. You should order your images according to the way that they overlap, for example from left-to-right across the panorama. If you are making a multi-row panorama, we recommend blending each horizontal row individually, and then running Enblend a last time to blend all of the rows together vertically. 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 Enblend considers the next image on the command line. Consult Section 4.5 on how to change the

[1]) 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.

images' order within multi-layer image files.

Find out more about Enblend on its SourceForge[2]) web page[3]).

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

Chapter 2

Known Limitations

Enblend 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.
- Each "next" image must overlap with the result of the blending of all previous images. In special occasions option '`-pre-assemble`' can circumvent this sequential-blending restriction.
- No pair of images must overlap too much. In particular, no two images must be identical.

  The overlap is exclusively defined by the masks of the overlapping images. This is exactly what the input masks are built for. Let A be the number of pixels that overlap in both masks. We use A as a measure of the overlap area – something 2-dimensional; technically it is a pixel count.
  Construct the smallest circumscribed, par-axial rectangle of the overlap area. The rectangle has a circumference

  $$U = 2(a + b),$$

  which is of course 1-dimensional. Internally U again is a number of pixels just as A.
  The threshold when we consider a pair of images sufficiently different is when A is larger than $\langle 2 \rangle$ times the number of pixels on the circumference U

  $$A > \langle 2 \rangle \times U.$$

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

> *Avoiding the term "fractal dimension", we have constructed a simple measure of how 2-dimensional the overlap area is. This way we steer clear of feeding later processing stages with nearly 1-dimensional overlap regions, something that wreaks havoc on them.*

- Option '`-wrap=both`' performs blending in $\mathsf{E}(1) \times \mathsf{E}(1)$, which is only *locally* isomorphic to $\mathsf{S}(2)$. This will cause artifacts that do not appear in $\mathsf{S}(2)$. Enblend cannot blend within $\mathsf{S}(2)$.

Chapter 3

Photographic Workflow[C]

Enblend and Enfuse 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 Enblend and Enfuse 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.

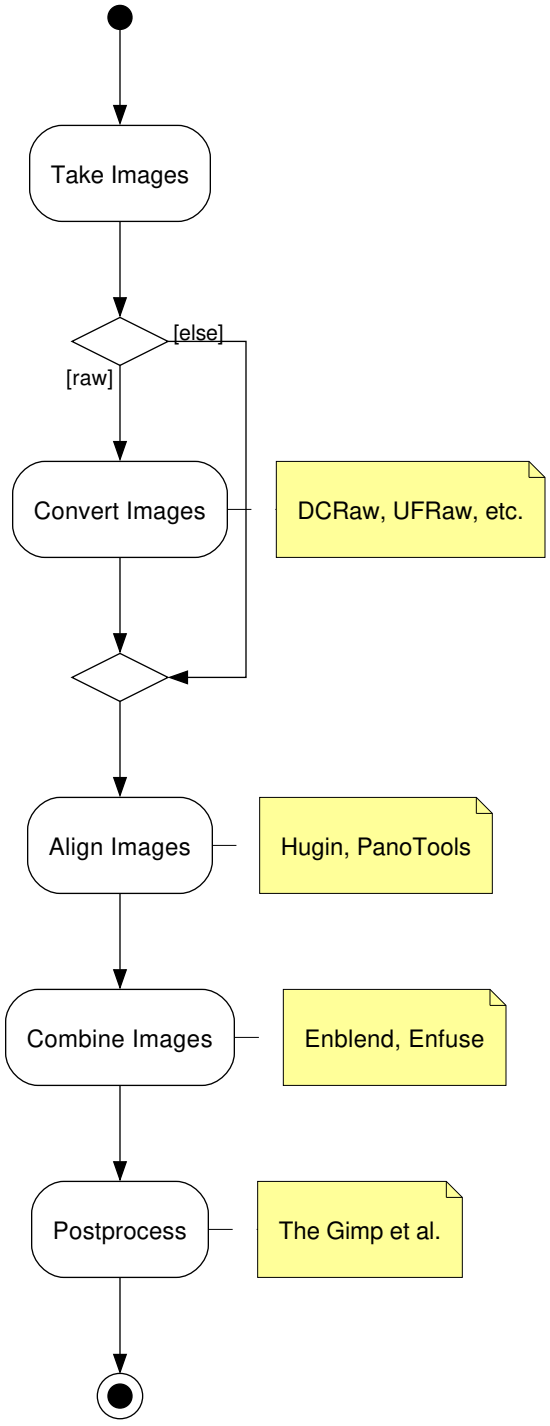


Figure 3.1: Photographic workflow with Enblend and Enfuse.

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
: Enblend and Enfuse 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 8 on page 59. 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 Enblend and Enfuse 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 17, 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 Enblend and Enfuse 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 Enblend^c

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

3.3.1 Finding Out Details About enblend

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

– what its exact version number is (see Sec. 3.3.1 on page 10 and option `-version`, page 35),
– what features it does support (see Sec. 3.3.1 on page 11 and options `-version`, page 35 and `-verbose`, page 20),

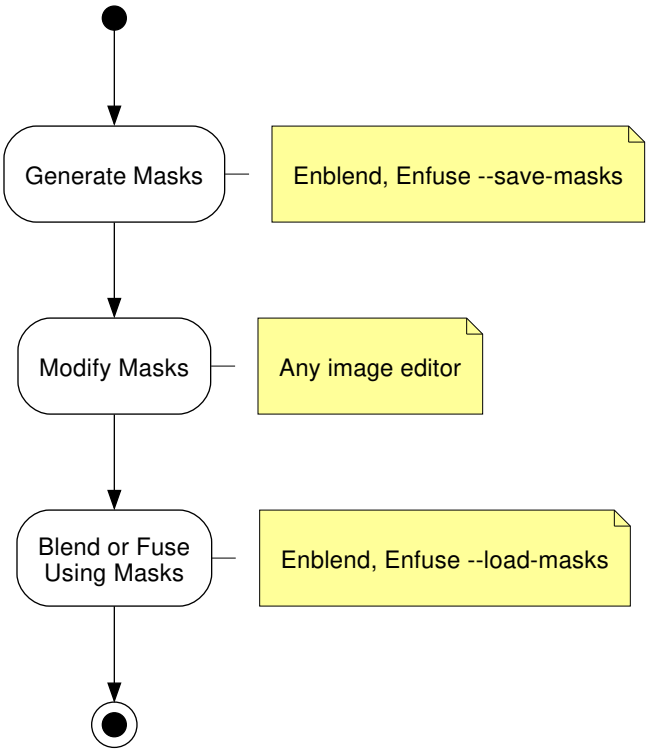


Figure 3.2: Workflow for externally modified masks. The "Blend or Fuse Using Masks" step utilizes the multi-resolution algorithm just as for internal workflow without mask files.

```
$ enblend -version
enblend 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 **enblend** 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 12 and option `-show-image-formats`, page 34),
- who built it, (see Sec. 3.3.1 on page 13 and option `-show-signature`, page 35), and finally
- what compiler and libraries were used to do so (see Sec. 3.3.1 on page 13 and option `-show-software-components`, page 35).

The information are explained in detail in the following sections.

Exact Version Number

Example 3.1 shows a possible output of '`enblend -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 viii) 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 12 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 **enblend** "suddenly" behaves strangely.

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

Check for extra feature OpenMP.

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

Check for extra feature OpenCL.

(3.) *"**enblend** 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

```
$ enblend -version -verbose
enblend 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 **enblend** when called with options `-version` and `-verbose` together.

(4.) *"**enblend** 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 **enblend** 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

Enblend 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 **enblend** binary.

The only three image formats always supported are

- JPEG,
- PNG, and

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 Enblend's. The "Mask"-column indicates whether the format supports an image mask (alpha-channel), see also Chapter 8. Column "Profile" shows whether the image format allows for ICC-profiles to be included; see also Chapter 7.

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

```
$ enblend -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 **enblend** when asked to reveal the compiler that was used to build it along with the libraries it was linked against.

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

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

3.3.2 Console Messages

Enblend 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 **enblend** is instructed to only show information about its configuration (see Section 4.2.6 on page 34) the text goes to Standard Output. **enblend** sends error and warning messages to Standard Error. The messages follow a fixed format.

`enblend:` [*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 **enblend** 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 **enblend**, it returns code 1 to the operating system. On successful termination the return code is 0.

`warning:` A problem that forces **enblend** 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 **enblend**. 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: **enblend** 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 **enblend** is correct, this type of message may warrant a bug report as explained in Appendix B on page 66.

In very unfortunate circumstances Enblend 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 **enblend**. All of them depend on **enblend** having been compiled with certain features. The hint "(direct)" indicates genuine variables in **enblend**, whereas "(implicit)" denotes variables that control libraries that are linked with **enblend**.

`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 11).

> *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 **enblend** uses as many OPENMP-threads as there are CPUs. Use this variable for example to free some CPUs for other processes than **enblend**.

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

`TMPDIR` determines the directory and thus the drive where **enblend** 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

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

`enblend` [*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 Enblend must comply with the following requirements.

- Parts of the images overlap.
- Each image has an alpha channel also called "mask".
- The images agree on their number of channels:
  - one plus alpha or
  - three plus alpha.

  This is, either all images are black-and-white (one channel and alpha channel) or all are RGB-color images (three channels and alpha channel).
- The images agree on their number of bits-per-channel, i.e., their "depth":
  - `uint8`,
  - `uint16`,
  - `float`,
  - etc.

  See option `-depth`, page 22 for an explanation of different output depths.

- Enblend understands the images' filename extensions as well as their file formats.

  You can check the supported extensions and formats by calling Enblend with option `-show-image-formats` (page 34).

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

```
$ enblend -help
```

does and sort them alphabetically within their groups. For an alphabetic list of *all* options consult the Option Index, page 81.

enblend 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 **enblend** 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 Enblend. They are called "common" because they are used most often. However, in fact, Enblend and Enfuse 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, Enblend 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 Enblend 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.

[1]) https://ukoethe.github.io/vigra/
[2]) As DR. DANIEL JACKSON correctly noted, actually, it is not a pyramid: "Ziggaurat, it's a Ziggaurat."

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 × height pixels cannot be deconstructed into a pyramid of more than

$$\lfloor \log_2(\min(\mathit{width}, \mathit{height})) \rfloor \quad \mathit{levels}.$$

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. Enblend 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 blended output image in *FILE*. If '`-output`' is omitted, Enblend 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 Enblend more verbose; see Section 3.3.1 on page 8 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 Enblend 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.

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 enblend; each level includes all messages of the lower levels.

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

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

Usually, Enblend 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, Enblend 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. 7.1 on page 53) for grayscale images and
- for color images utilize 3-dimensional RGB-cube (see Eqn. 7.6 on page 54) spanned by the input ICC profile or sRGB if no profiles are present. In the latter case, consider passing option `-fallback-profile`, page 28 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.

Enblend only.

Please keep in mind that by using different blend colorspaces, blending may not only change the colors of the output image, but Enblend may choose different seam line routes as some seam-line optimizers are guided by image differences, which are different when viewed in different colorspaces.

□

`-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".

Enblend 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. Enblend 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, Enblend 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.

  The Gimp before version 2.0 and CinePaint (see Appendix A on page 63) exhibit unusual behavior when loading images with unassociated alpha channels. Use option `-g` to work around this problem. With this flag Enblend 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 Enblend 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:

[3]) http://www.openexr.com/about.html#features
[4]) http://www.awaresystems.be/imaging/tiff/tifftags/extrasamples.html
[5]) https://en.wikipedia.org/wiki/Born-von_Karman_boundary_condition

`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.

`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 Mask Generation Options

These options control the generation and the usage of masks.

`-coarse-mask`[`=`*FACTOR*]
: Use a scaled-down or "coarse" version of the input images to create the seam line. This option reduces the number of computations necessary to compute the seam line and the amount of memory necessary to do so. It is the default.

If omitted *FACTOR* defaults to $\langle 8 \rangle$, this means, option `-coarse-mask` shrinks the overlapping *areas* by a factor of $\langle 8 \rangle \times \langle 8 \rangle$. With $FACTOR = 8$ the total memory allocated during a run of Enblend shrinks approximately by 80% and the maximum amount of memory in use at a time is decreased to some 40% in comparison to a full-size ("fine") mask.

Valid range: $FACTOR = 1, 2, 3, \ldots$, where 1 reproduces `-fine-mask`.

Also see the negated option, '`-fine-mask`', page 25 and Table 4.2.

Active Options	`-no-optimize`	`-optimize`
`-fine-mask`	Use NFT mask.	Vectorize NFT mask, optimize vertices with simulated annealing and DIJKSTRA'S shortest path algorithm, fill vector contours to recover mask.
`-coarse-mask`	Scale down overlap region, compute NFT mask and vectorize it, fill vector contours.	Scale down overlap region, vectorize NFT mask, optimize vertices with simulated annealing and DIJKSTRA'S shortest path algorithm, fill vector contours to recover mask.

Table 4.2: Various options that control the generation of masks. All mask computations are based on the Nearest-Feature Transformation (NFT) of the overlap region.

`-fine-mask`
: Instruct Enblend to employ the full-size images to create the seam line, which can be slow. Use this option, for example, if you have very narrow overlap regions.

  Also see option `-coarse-mask`, page 25 and Table 4.2.

`-load-masks`[`=`*`IMAGE-TEMPLATE`]
: Instead of generating masks, load those in *IMAGE-TEMPLATE*. The default is '⟨`mask-%n.tif`⟩'. The mask images must be 8 bit grayscale images.

  See option `-save-masks`, page 26 below for details.

`-optimize`
: Use a multi-strategy approach to route the seam line around mismatches in the overlap region. This is the default. Table 4.3 explains these strategies; also see Table 4.2.

  Option `-no-optimize` negates `-optimize` and thus turns off seam line optimization. Combined with option `-fine-mask`, page 25 this will produce the same type of mask as Enblend version 2.5, namely the result of a Nearest-Feature Transform (NFT).

`-save-masks`[`=`*`IMAGE-TEMPLATE`]
: Save the generated masks to *IMAGE-TEMPLATE*, which defaults to '⟨`mask-%n.tif`⟩'. Enblend saves masks as 8 bit grayscale, i.e. single channel images. For accuracy we recommend to choose a lossless format.

Algorithm	Tuning Parameters
Simulated Annealing	Tune with option `-anneal` = *TAU* : *DELTA-E-MAX* : *DELTA-E-MIN* : *K-MAX*. Simulated-Annealing[6]). FIX Explain Simulated-Annealing! ME
Dijkstra Shortest Path	Tune with option `-dijkstra` = *RADIUS*. Dijkstra algorithm[7]). FIX Explain Dijkstra algorithm! ME

Table 4.3: Enblend's strategies to optimize the seam lines between overlapping images.

Use this option if you wish to edit the location of the seam line by hand. This will give you images of the right sizes that you can edit to make your changes. Later, use option `-load-masks`, page 26 to blend the project with your custom seam lines.

Enblend will stop after saving all masks unless option `-output`, page 20 is given, too. With both options given, this is, '`-save-masks`' and '`-output`', Enblend saves all masks and then proceeds to blend the output image.

> *IMAGE-TEMPLATE defines a template that is expanded for each input 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 (see Section 4.2.1 on page 18). Table 4.4 lists all variables.*
> *A fancy mask filename template could look like this:*
>
> `%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.*

`-visualize`[=*VISUALIZE-TEMPLATE*]

Create an image according to *VISUALIZE-TEMPLATE* that visualizes the un-optimized mask and the applied optimizations (if any). The default is '⟨`vis-%n.tif`⟩'.

This image will show Enblend's view of the overlap region and how it decided to route the seam line. If you are experiencing artifacts or unexpected output, it may be useful to include this visualization image in your bug report. For a detailed description of the image, consult Chapter 6 on page 49.

> *VISUALIZE-TEMPLATE defines a template that is expanded for each input 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 (see option `-output`*, page 20). Table 4.4 on page 29 lists all variables.*

4.2.4 Expert Options

Control inner workings of Enblend and in particular the interpretation of images.

`-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 7.3 on page 56 on color profiles.

`-layer-selector=`*ALGORITHM*

Override the standard layer selector algorithm '⟨`all-layers`⟩'.

Enblend 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.

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

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

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

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 Enblend 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.*

`-a`
`-pre-assemble`

Pre-assemble non-overlapping images before each blending iteration.

This overrides the default behavior which is to blend the images sequentially in the order given on the command line. Enblend will use fewer blending iterations, but it will do more work in each iteration.

This option has the negated form '`-no-pre-assemble`', which restores the default.

`-x` Checkpoint partial results to the output file after each blending step.

4.2.5 Expert Mask Generation Options

These options allow for a detailed control of the seam-line optimizers which govern the mask generation.

`-anneal=`*`TAU`*`[:`*`DELTA-E-MAX`*`[:`*`DELTA-E-MIN`*`[:`*`K-MAX`*`]]]`

Set the parameters of the Simulated Annealing optimizer. See Table 4.3 on page 27.

`TAU` *TAU* is the temperature reduction factor in the Simulated Annealing; it also can be thought of as "cooling factor". The closer *TAU* is to one, the more accurate the annealing run will be, and the longer it will take.

Append a percent sign ('`%`') to specify *TAU* as a percentage.

Valid range: $\langle 0 \rangle < TAU < \langle 1 \rangle$.

The default is $\langle 0.75 \rangle$; values around 0.95 are reasonable. Usually, slower cooling results in more converged points.

`DELTA-E-MAX, DELTA-E-MIN`

DELTA-E-MAX and *DELTA-E-MIN* are the maximum and minimum cost change possible by any single annealing move.

Valid range: $\langle 0 \rangle <$ *DELTA-E-MIN* $<$ *DELTA-E-MAX*.

In particular they determine the initial and final annealing temperatures according to:

$$T_{\text{initial}} = \frac{\textit{DELTA-E-MAX}}{\log(\textit{K-MAX}/(\textit{K-MAX}-2))}$$
$$T_{\text{final}} = \frac{\textit{DELTA-E-MIN}}{\log(\textit{K-MAX}^2 - \textit{K-MAX} - 1)}$$

The defaults are: *DELTA-E-MAX*: $\langle 7000.0 \rangle$ and *DELTA-E-MIN*: $\langle 5.0 \rangle$.

K-MAX

K-MAX is the maximum number of "moves" the optimizer will make for each line segment. Higher values more accurately sample the state space, at the expense of a higher computation cost.

Valid range: *K-MAX* $\geq \langle 3 \rangle$.

The default is $\langle 32 \rangle$. Values around 100 seem reasonable.

`-dijkstra=`*RADIUS*

Set the search *RADIUS* of the DIJKSTRA Shortest Path algorithm used in DIJKSTRA Optimization (see Table 4.3 on page 27).

A small value prefers straight line segments and thus shorter seam lines. Larger values instruct the optimizer to let the seam line take more detours when searching for the best seam line.

Valid range: *RADIUS* $\geq \langle 1 \rangle$.

Default: $\langle 25 \rangle$ pixels.

`-image-difference=`*ALGORITHM*`[:`*LUMINANCE-WEIGHT*`[:`*CHROMINANCE-WEIGHT*`]]`

Enblend calculates the difference of a pair of overlapping color images when it generates the primary seam with a Graph-Cut and also before it optimizes the seams. It employs a user-selectable *ALGORITHM* that is controlled by the

- weights for luminance differences, *LUMINANCE-WEIGHT*, w_{luma} and
- color differences, *CHROMINANCE-WEIGHT*, w_{chroma}.

For black-and-white images the difference is simple the absolute difference of each pair of pixels.

`maximum-hue-luminance`
`maximum-hue-lum`
`max-hue-luminance`
`max-hue-lum`
`max`

Calculate the difference d as the maximum of the differences of the luminances l and hues h of each pair of pixels P_1 and P_2:

$$d = \max\left(w_{\text{luma}} \times |l(P_1) - l(P_2)|, w_{\text{chroma}} \times |h(P_1) - h(P_2)|\right).$$

This algorithm was the default for Enblend up to version 4.0.

`delta-e`
`de`

Calculate the difference d as the EUCLIDEAN distance of the pixels in L*A*B* space:

$$d = \Big[w_{\text{luma}} \times \left(L(P_1) - L(P_2)\right)^2 + w_{\text{chroma}} \times \left(a(P_1) - a(P_2)\right)^2 + w_{\text{chroma}} \times \left(b(P_1) - b(P_2)\right)^2\Big]^{1/2}$$

This is the default in Enblend version 4.1 and later.

Note that the "delta-E" mentioned here has nothing to do with *DELTA-E-MAX* and *DELTA-E-MIN* of option `-anneal`, page 30.

Both *LUMINANCE-WEIGHT* and *CHROMINANCE-WEIGHT* are non-negative. Enblend automatically normalizes the sum of *LUMINANCE-WEIGHT* and *CHROMINANCE-WEIGHT* to one. Thus,

```
-image-difference=delta-e:2:1
```

and

```
-image-difference=delta-e:0.6667:0.3333
```

define the same weighting function.

The default *LUMINANCE-WEIGHT* is ⟨1.0⟩ and the default *CHROMINANCE-WEIGHT* is ⟨1.0⟩.

At higher verbosity levels Enblend computes the true size of the overlap area in pixels and it calculates the average and standard deviation of the difference per pixel in the normalized luminance interval $(0 \ldots 1)$. These statistical measures are based on *ALGORITHM*, therefore they should only be compared for identical *ALGORITHM*s. The average difference is a rough measure of quality with lower values meaning better matches.

`-mask-vectorize=`*DISTANCE*

Set the mask vectorization *DISTANCE* that Enblend uses to partition each seam. Thus, break down the seam to segments of length *DISTANCE* each.

If Enblend uses a coarse mask (`-coarse-mask`, page 25) or Enblend optimizes (`-optimize`, page 26) a mask it vectorizes the initial seam line before performing further operations. See Table 4.2 for the precise conditions. *DISTANCE* tells Enblend how long to make each of the line segments called vectors here.

The unit of *DISTANCE* is pixels unless it is a percentage as explained in the next paragraph. In fine masks one mask pixel corresponds to one pixel in the input image, whereas in coarse masks one pixel represents for example $\langle 8 \rangle$ pixels in the input image.

Append a percentage sign ('`%`') to *DISTANCE* to specify the segment length as a fraction of the diagonal of the rectangle including the overlap region. Relative measures do not depend on coarse or fine masks, they are recomputed for each mask. Values around 5%–10% are good starting points.

This option strongly influences the mask generation process! Large *DISTANCE* values lead to shorter, straighter, less wiggly, less baroque seams that are on the other hand less optimal, because they run through regions of larger image mismatch instead of avoiding them. Small *DISTANCE* values give the optimizers more possibilities to run the seam around high mismatch areas.

What should *never* happen though, are loops or cusps in the seam line. Counter loops and cusps with higher weights of *DISTANCE-WEIGHT* (option `-optimizer-weights`, page 33), larger vectorization *DISTANCE*s, and *TAU*s (option `-anneal`, page 30) that are closer to one. Use option `-visualize`, page 27 to check the results.

Valid range: *DISTANCE* $\geq \langle 4 \rangle$.

Enblend limits *DISTANCE* so that it never gets below $\langle 4 \rangle$ even if it has been given as a percentage. The user will be warned in such cases.

Defaults: $\langle 4 \rangle$ pixels for coarse masks and $\langle 20 \rangle$ pixels for fine masks.

`-optimizer-weights=`*DISTANCE-WEIGHT*[`:`*MISMATCH-WEIGHT*]

Set the weights of the seam-line optimizer. If omitted, *MISMATCH-WEIGHT* defaults to 1.

The seam-line optimizer considers two qualities of the seam line:

- The distance of the seam line from its initial position, which has been determined by NFT (see option `-no-optimize`, page 26).
- The total "mismatch" accumulated along it.

DISTANCE-WEIGHT and *MISMATCH-WEIGHT* define how to weight these two criteria. Enblend up to version 3.2 used 1:1. This version of Enblend uses ⟨8.0⟩:⟨1.0⟩.

A large *DISTANCE-WEIGHT* pulls the optimized seam line closer to the initial position. A large *MISMATCH-WEIGHT* makes the seam line go on detours to find a path along which the mismatch between the images is small. If the optimized seam line shows cusps or loops (see option `-visualize`, page 27), reduce *MISMATCH-WEIGHT* or increase *DISTANCE-WEIGHT*.

Both weights must be non-negative. They cannot be both zero at the same time. Otherwise, their absolute values are not important as Enblend normalizes their sum.

`-primary-seam-generator=`*ALGORITHM*
Select the algorithm responsible for generating the general seam route.

This is the *ALGORITHM* that produces an initial seam line, which serves as the basis for later, optional optimizations. Nearest Feature Transform (NFT) is the only algorithm up to and including Enblend version 4.0. Version 4.1 added a Graph-Cut (GC) algorithm, which is the default for version 4.2 and later.

Valid *ALGORITHM* names are:

`nearest-feature-transform`
`nft`
Nearest Feature Transform

`graph-cut`
`gc`
Graph-Cut

See Chapter 5 on page 47 for details on Enblend's primary seam generators.

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 42.

`-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 **enblend** 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 Enblend followed by the versions of all important libraries against which Enblend 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 20 to show configuration details, like the extra features that may have been compiled in. For details consult Section 3.3.1 on page 10.

4.2.7 Program Flow And Option Settings

Figure 4.1 depicts Enblend's internal work flow and shows what option influences which part. Enblend works incrementally. Thus, no matter whether starting with the first image or the result of blending any previous images, it loads the next image. After blending it the result again is a single image, which serves as base for the next input image, this is the next iteration.

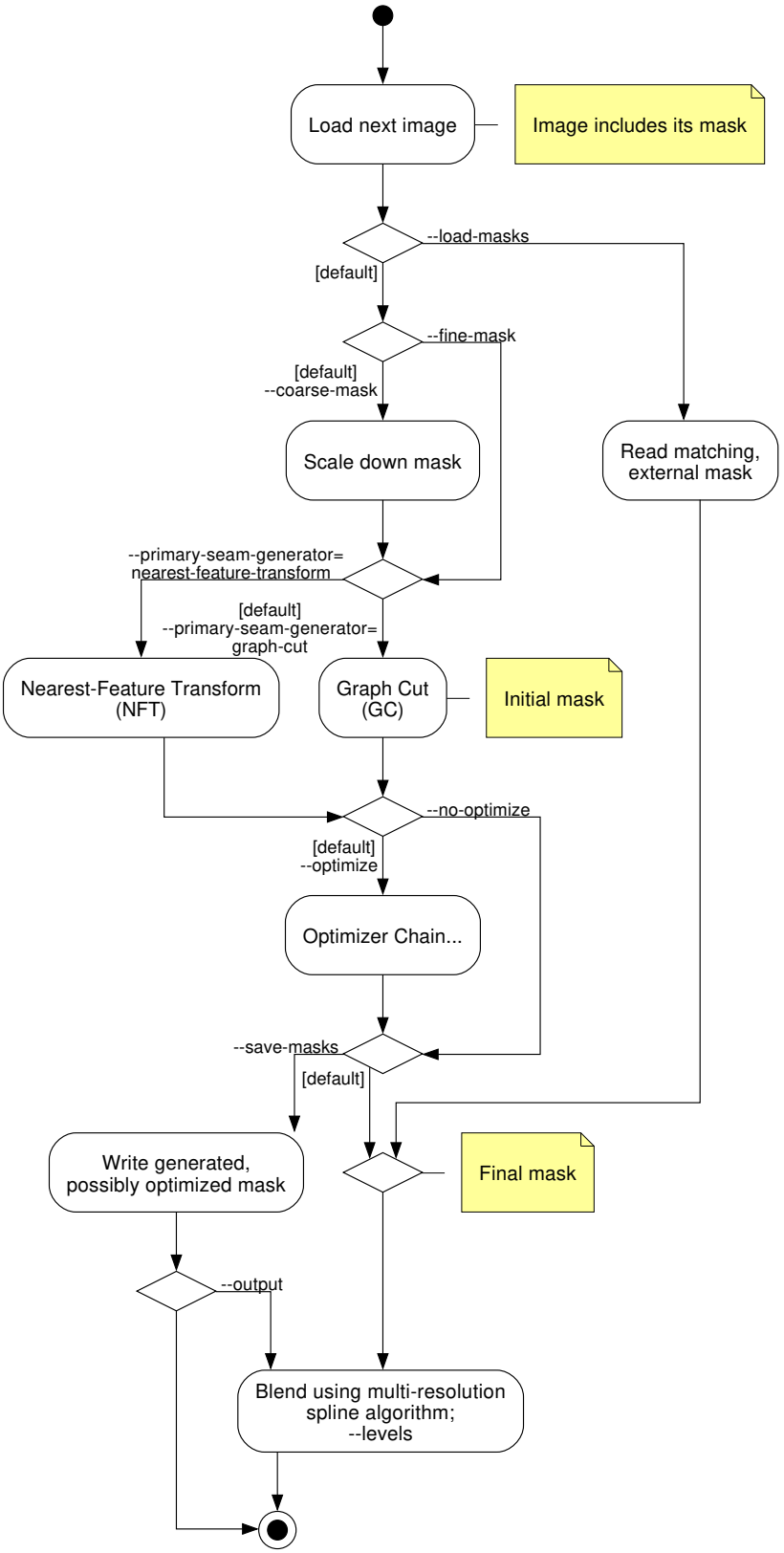


Figure 4.1: Enblend's internal work flow for the "next" image. The defaults are indicated like '`[default]`'. The Optimizer Chain is complicated enough to warrant its own chart in Figure 4.2.

Figure 4.1 somewhat simplifies the program flow.

- *The Graph Cut algorithm needs an initial "guess" of the seam line. It gets it by running a Nearest-Feature Transform.*
- *If the overlap between the previous image and the next image is too small, the "Scale down mask" step is skipped and Enblend works with the mask in its original size ("fine mask") no matter what the command-line options specify.*

4.3 Option Delimiters[c]

Enblend and Enfuse 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.

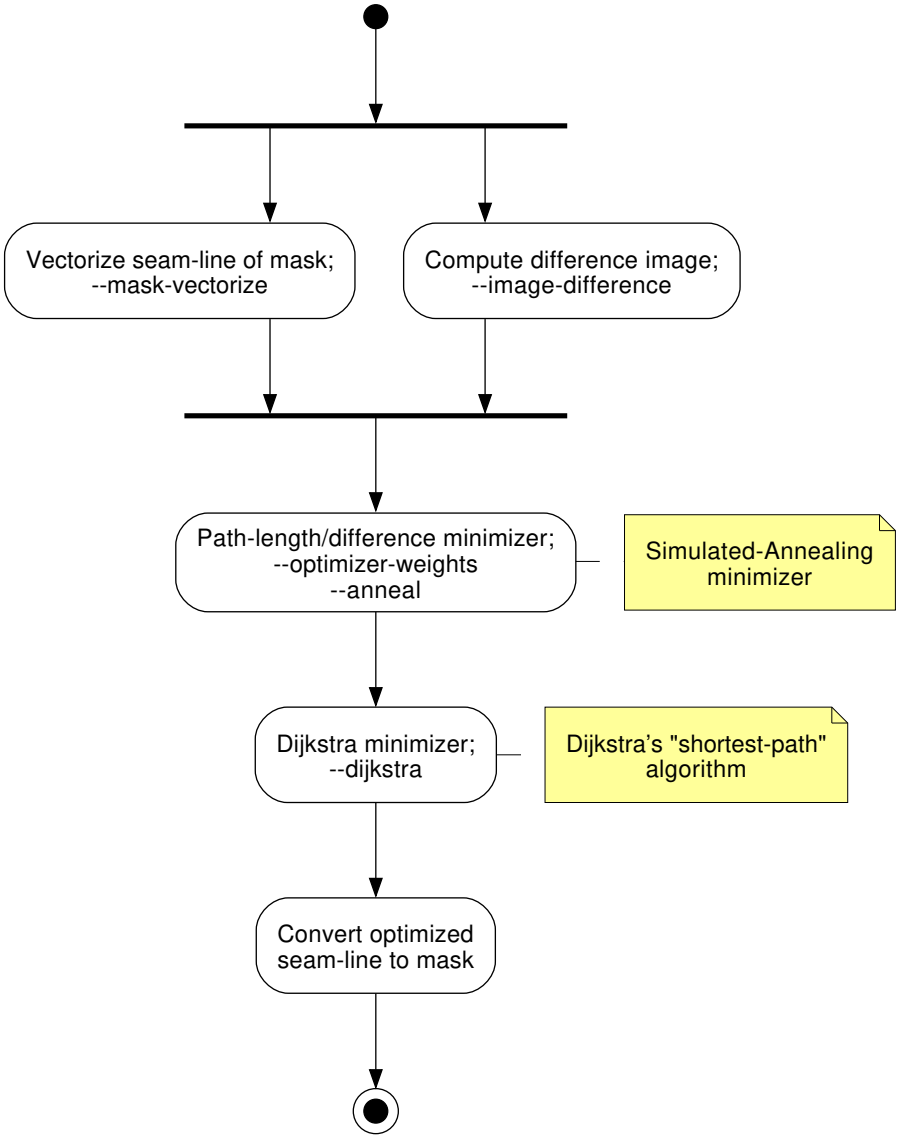


Figure 4.2: A closer look the Optimizer Chain of Figure 4.1.

`-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.

Enblend and Enfuse 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
```

where file *master.list* comprises of

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

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[8]).

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.

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

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

```
# 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 4.1: Example of a complete response file.

```
#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 Enblend or Enfuse 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 4.1 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

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.

```
# Horizontal panorama
# 15 images

# filename-globbing: wildcard

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

Example 4.2: Control filename-globbing in a response file with a syntactic comment.

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 79 lists them in alphabetic order.

4.4.3 Globbing Algorithms

The three equivalent syntactic keys

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

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

All versions of Enblend and Enfuse 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 Enblend or Enfuse use option `-show-globbing-algorithms`.

Example 4.2 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.

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 Enblend and Enfuse 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 28 and Section 4.4 on page 39. 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[9] 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

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

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

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

Quoting will also be required if Enblend’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.

[9] 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,

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

or

```
$ enblend $(ls -1 image-??.tif | sed -e 's/$/[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 63.

- 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[10]) 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[11])-suite, like, for example, **convert** and **display** use a similar syntax[12]) as Enblend 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.
- Enblend and Enfuse by default apply the '⟨`all-layers`⟩' selector (see option `-layer-selector`, page 28) 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.

[10]) http://www.remotesensing.org/libtiff/
[11]) https://www.imagemagick.org/script/index.php
[12]) https://www.imagemagick.org/script/command-line-processing.php#input

Chapter 5

Seam Generators

This version of Enblend supports two main algorithms to generate seam lines. Use option `-primary-seam-generator`=*ALGORITHM* to select one of the generators.

Nearest Feature Transform (NFT)
ALGORITHM=`nearest-feature-transform`

The Nearest Feature Transform[1]), also known as Distance Transform[2]) (DT), is a fast and efficient technique to produce a seam line route given the geometries of multiple overlapping images.

NFT as implemented in this version of Enblend only takes into account the *shape of the overlap area.* It completely ignore the images' contents.

Graph-Cut (GC)
ALGORITHM=`graph-cut`

Graph-Cut[3]) is a region-oriented way of segmenting images.

The generator is based on the idea of finding a minimum cost "cut" of a graph created from a given image pair. A "cut" is where the seam line appears. GC determines the cost from the overlapping images' contents.

The most significant difference between the two algorithms is the output mask gradation. NFT produces a coarse approximation of the seam, running as far away from the overlap-region borders as possible. The resulting mask could then be blended as-is, however, Enblend by default runs image-content dependent optimizers to increase the mask gradation and for example omits the regions where the images differ. The result is a finer seam line, which only loosely follows the shape of NFT's primary seam.

[2]) https://en.wikipedia.org/wiki/Distance_transform

[2]) Muhammad H. Alsuwaiyel and Marina Gavrilova, "On the Distance Transform of Binary Images", Proceedings of the International Conference on Imaging Science, Systems, and Technology, June 2000, Vols. I and II, pages 83–86.

[3]) https://en.wikipedia.org/wiki/Graph_cuts_in_computer_vision

Graph-Cut, on the other hand, is capable of producing the final mask in one pass without the need of further optimizers. It looks for a seam line that is *globally* optimal, taking into account feature frequency, as well as image dissimilarity. This means, the seam is less likely to cross lines like for example fences, lampposts, or road markings, where they would be visible.

The optimizers which run after NFT can also be run after GC. Nevertheless, GC works best just with a fine mask (option `-fine-mask`); optimizers are then automatically *turned off* to take full advantage of the detailed seam GC produces.

GC requires more memory and computation time to complete than NFT. Thus, it is best to prefer NFT where the images used are large and execution time is crucial. If quality is the priority, using GC and fine mask usually produces visually more pleasing results.

GC is currently limited to seams that begin and end on the images' borders. This means that the algorithm cannot run in cases where, for example, one image is contained in another, resulting in a loop-like seam. In such cases, though, Enblend automatically falls back to a NFT-generated seam, making its application transparent to the user.

Chapter 6

Visualization Image

The visualization image shows the symmetric difference of the pixels in the rectangular region where two images overlap. The larger the difference the lighter shade of gray it appears in the visualization image. Enblend paints the non-overlapping parts of the image pair – these are the regions where *no* blending occurs – in ⟨dark red⟩. Table 6.1 shows the meanings of all the colors that are used in seam-line visualization images.

Figure 6.1 shows an example of a seam-line visualization. It was produced with an Enblend run at all defaults plus passing options `-fine-mask` and `-visualize`.

The large ⟨dark red⟩ border is "off-limits" for Enblend, for the images do not overlap there. The dark wedge inside the ⟨dark red⟩ frame is where the images share a common region.

The initial seam-line (⟨dark yellow⟩) is almost straight with the exception of a single bend on the left side of the image and the final seam-line (⟨bright yellow⟩) meanders around it.

⟨dark red⟩ areas
: Non-overlapping parts of image pair.

various shades of gray
: Difference of the pixel values in the overlap region.

⟨dark blue⟩ dot
: Location of an optimizer sample.

⟨medium green⟩ dot
: First sample of a line segment.

⟨light green⟩ dot
: Any other but first sample of a line segment.

⟨bright cyan⟩ dot
: State space sample inside the DIJKSTRA radius.

⟨bright magenta⟩ dot
: Non-converged point.

⟨dark yellow⟩ line
: Initial seam line as generated by the primary seam generator.

⟨bright yellow⟩ line
: Final seam line.

⟨bright white⟩ ⟨cross⟩
: Non-movable, or "frozen" endpoint of a seam-line segment that no optimizer is allowed to move around.

⟨light orange⟩ ⟨diamond⟩
: Movable endpoint of a seam-line segment, which seam-line optimizers can move.

Table 6.1: Colors and symbols used in seam-line visualization images.

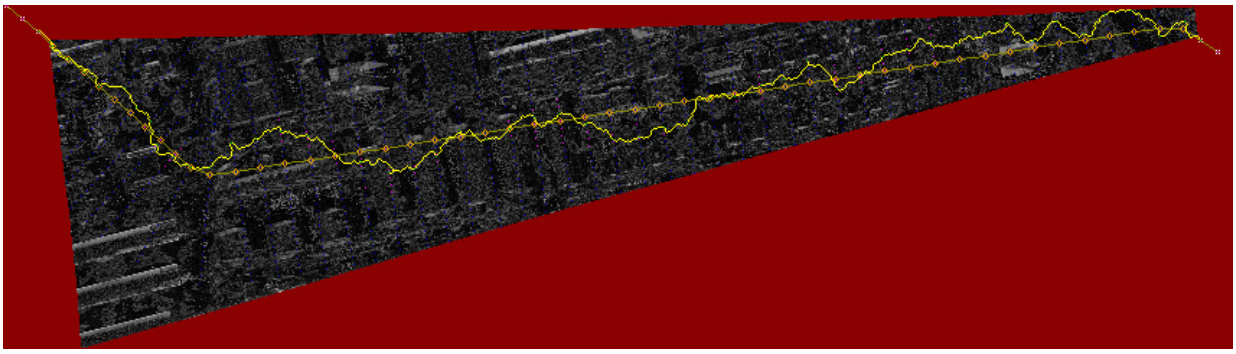


Figure 6.1: Seam-line visualization of a simple overlap. The 853×238 pixel image has been rescaled to fit the width of the current page.

Chapter 7

Color Spaces And Color Profiles[C]

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

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 Enblend's and Enfuse'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 Enblend and Enfuse 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.

7.1 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

$$L = tL_1 + (1-t)L_2 \quad \text{with} \quad 0 \le t \le 1, \tag{7.1}$$

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"*:

$$(0, L_{\max}) \to (0, 1) \tag{7.2}$$

Obviously,

$$0 \le \bar{L} \le 1 \tag{7.3}$$

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

$$L' = \left(tL_1^{1/\gamma} + (1-t)L_2^{1/\gamma}\right)^{\gamma}, \tag{7.4}$$

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 Enblend 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 (7.2) 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 (7.3) holds

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \le \begin{pmatrix} r \\ g \\ b \end{pmatrix} \le \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \tag{7.5}$$

[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, (7.1), 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

$$P := \begin{pmatrix} r \\ g \\ b \end{pmatrix} = t \begin{pmatrix} r_1 \\ g_1 \\ b_1 \end{pmatrix} + (1-t) \begin{pmatrix} r_2 \\ g_2 \\ b_2 \end{pmatrix} \quad \text{with} \quad 0 \le t \le 1, \tag{7.6}$$

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.

7.2 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.

$$\mathrm{Log}(L) := \begin{cases} 1 + \log(1+L) & \text{for } L \ge 0 \text{ and} \\ 1/(1-L) & \text{otherwise,} \end{cases} \tag{7.7}$$

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 7.1 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 Enblend and Enfuse 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.

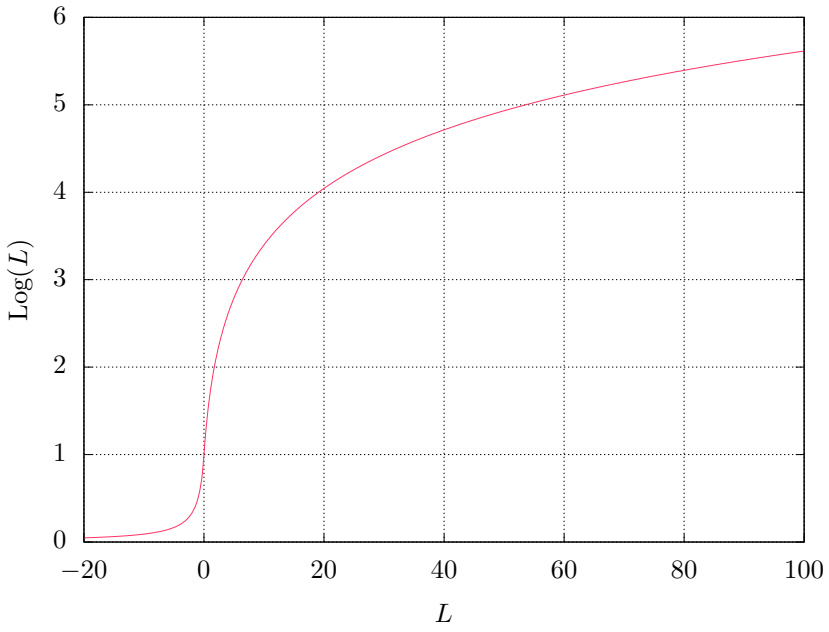


Figure 7.1: Forward Log-transform shown for the range of $-20 \leq L \leq 100$. The domain of $\mathrm{Log}(L)$ is the whole real axis.

7.3 Color Profiles

ICC-color profiles completely absorb gamma encodings (7.4) and ICC profile aware software like Enblend and Enfuse 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, Enblend and Enfuse 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 28.

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.

7.4 Blending Color Spaces

Enblend and Enfuse offer to work inside the RGB-cube (7.5) 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 (7.1) for grayscale images and inside the RGB-color cube as given in (7.6).

[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.

  Enblend 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.

7.5 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 22. No blend color space can avoid banding if parts of the input images are almost "monochrome".
- Enblend only. No color space can fix a seam-line gone haywire! First re-run Enblend with '`-visualize`', page 27 to inspect the seam-lines, then try one or more of the following
  - Reorder the input images.
  - Force blending with the full-resolution masks, using option `-fine-mask`, page 25.
  - Generate the initial seam with the simpler and more robust nearest-feature transform (NFT) by employing option `-primary-seam-generator=nearest-feature-transform`, page 34. See also Chapter 5 on page 47.
  - Disable the seam-line optimizer with '`-no-optimize`', page 26.

Chapter 8

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.

8.1 Masks In Input Files

Each of the input files for Enblend and Enfuse 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 8.1) or, for TIFF files only, **tiffinfo** (see Example 8.2) to inquire quickly whether a file contains a mask. Appendix A on page 63 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])

Alpha Channel
: The alpha channel stores the transparency value for each pixel, typically in

[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 8.1: Using **identify** to find out about the mask in *image-0000.tif*. '`Matte`' indicates the existence of a mask.

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 − alpha. See ImageMagick[2]) for further explanations.

Enblend and Enfuse 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.

8.2 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 8.2: 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; Enblend and Enfuse 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.

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 Enblend or Enfuse.

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, Enblend, and Enfuse. 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 Enblend and Enfuse 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 Enblend or Enfuse get a fatal signal, for any options or input images, that is a bug.
- If Enblend or Enfuse produce incorrect results, for any input whatever, that is a bug.
- If Enblend or Enfuse produce an error message for valid input, that is a bug.
- If Enblend or Enfuse 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 Enblend. You can get the data by running enblend with the options `-version` and `-verbose` together. See also Section 3.3.1 on page 8 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 Enblend and Enfuse.
- The *exact command line* you use to call Enblend or Enfuse, 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 Enblend or Enfuse 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 Enblend or Enfuse.

- 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 Enblend or Enfuse

If you would like to write bug fixes or improvements for Enblend or Enfuse, 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, 41
enfuse-response-file, 41
filename-globbing, 42
globbing, 42
glob, 42
layer-selector, 42
response-file, 41

Program/Application Index

If a program belongs to a larger package, it has its association mentioned in parenthesis.

CinePaint, 64
Cinepaint, 24
Darktable, 63
Gimp, 6, 24, 64
GraphicsMagick, 64
Hugin, 1, 6, 46, 63, 68
ImageFuser, 68
ImageMagick, 64
LittleCMS, 65
Magic Lantern, 65
Mercurial, 11
PanoTools, 1, 6, 63
RawTherapee, 63
UFRaw, 63
`PFSTools`, 64
`PTOptimizer` (PanoTools), 63
`PTmender` (PanoTools), 63
`align_image_stack` (Hugin), 63
`convert` (ImageMagick), 46, 64
`dcraw`, 6, 63
`display` (ImageMagick), 46, 64
`exiftool`, 65
`exrdisplay` (OpenEXR), 64
`fulla` (Hugin), 63
`gm` (GraphicsMagick), 64
`gmic`, 64
`identify` (ImageMagick), 59, 64
`montage` (ImageMagick), 64
`nona` (Hugin), 24, 46, 60, 63
`tiffcp` (LibTIFF), 46
`tiffinfo` (LibTIFF), 46
`tiffinfo` (libtiff), 59, 65
`tiffsplit` (LibTIFF), 46
`tificc` (LittleCMS), 65
`ufraw-batch`, 63
`ufraw`, 6, 63

Option Index

Locations marked like 12 indicate the place of the option's genuine description.

-anneal, 30
-blend-colorspace, 20
-ciecam, 22
-coarse-mask, 25
-compression, 18
-depth, 22
-dijkstra, 31
-fallback-profile, 28
-fine-mask, 25
-help, 34
-image-difference, 31
-layer-selector, 28
-levels, 19
-load-masks, 8, 26
-mask-vectorize, 32
-no-ciecam, 22
-no-optimize, 26
-no-parameter, 30
-no-pre-assemble, 30
-optimizer-weights, 33
-optimize, 26
-output, 8, 20
-parameter, 28
-pre-assemble, 30
-primary-seam-generator, 34
-save-masks, 8, 26
-show-globbing-algorithms, 34
-show-image-formats, 12, 34
-show-signature, 13, 35
-show-software-components, 13, 35
-verbose, 11, 20
-version, 10, 11, 35
-visualize, 27
-wrap, 4, 24
-V (long: -version), 35
-a (long: -pre-assemble), 30
-c (long: -ciecam), 22
-d (long: -depth), 22
-f, 24
-g, 24
-h (long: -help), 34
-l (long: -levels), 19
-o (long: -output), 20
-v (long: -verbose), 20
-w (long: -wrap), 24
-x, 30

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), 40
@ (response file prefix), 39

Numbers

360°
    horizontal panorama, *see* panorama, 360°
    vertical panorama, *see* panorama, 360°

A

Adelson, Edward H., 1
advanced options, 20–25
affine transformation, *see* transformation, affine
algorithm, *see* globbing algorithm
    globbing, 34
algorithms
    globbing, 43
alignment
    photometric, 7
alpha, 59
alpha channel
    associated, 24
Alsuwaiyel, Muhammad H., 47
anneal parameters, *see* optimize, anneal parameters
arithmetic JPEG compression, *see* compression
assemble, *see* preassemble
associated alpha channel, *see* alpha channel, associated
authors, 70

B

BigTIFF, 3
binary mask, *see* mask, binary
binary version, *see* software, version
bit depth, *see* bits per channel
bits per channel, 13, 22
black-and-white image, *see* image, black-and-white
black-point, 56
blend colorspace, *see* colorspace, blend
blending
    sequential, 3, 30
blending color space, *see* colorspace, for blending and fusing
blending pixels, *see* pixels, blending
Born, Max, 24
bracketing
    HDR, 65
branches of Enblend/Enfuse
    development, 10
Brasseur, Eric, 53
bug
    database at LaunchPad, 66
    reports, 66–69
        how to, 67
        identification of bugs, 66
        sending patches, 68
builder, 13
Burt, Peter J., 1
Burt-Adelson, 1

C

CARTER, SAMANTHA, 45
channel
  alpha, 1, 7, 13, 59
  depth, 22
  width, *see* channel, depth
checkpoint results, 30
chrominance weight, *see* weight, chrominance
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`
coarse mask, *see* mask, coarse
code,
see also return code, 14
code repository, *see* public repository
color appearance model, 20
color cube
  RGB, 21, 56
  sRGB, 56
  RGB, 21
color profile, 52, 56
color profiles
  math, 53
colors
  visualization image, *see* seamline, visualization image colors
colorspace, 52–58
  CIECAM02, 22, 57
  CIEL*A*B*, 21
  CIEL*U*V*, 21, 57
  L*A*B*, 32, 57
  blend, 20
  CIELUV, 21
  for blending and fusing, 56
  practical considerations, 58
command-line, 17
common options, 18–20
compiled-in features, *see* features
compiler, 13
compression, 18
  JPEG, 18
  JPEG of TIFF, 19
  LZW, 19
  arithmetic JPEG, 19
  deflate, 19
  packbits, 19
console messages, 14
conventions
  typographic, *see* notation
conversion
  raw, 5

D

D50 white-point, *see* white-point, D50
default layer selection, 42
default output filename, *see* filename, output, default
deflate compression, *see* compression
delimiters,
see also options, delimiters, 37
delta-E, 32
detailed configuration, 8
development branch, *see* branches, development
difference image, 31
DIJKSTRA radius, *see* radius, DIJKSTRA, *see* radius, DIJKSTRA
distance transform, 47
double precision float (IEEE754), *see* IEEE754, double precision float
DT, *see* distance transform
dual-ISO, 65
dynamic-library environment, 35

E

environment,
see also environment variable, 16
environment variable, 16, *16*

`CILK_NWORKERS`, 16
`OMP_DYNAMIC`, 16
`OMP_NUM_THREADS`, 16
`TMPDIR`, 16
EXIF, 65
expert mask generation options, 30–34
expert options, 28–30
EXR, 64
EXR image, *see* image, EXR
external mask, 7
extra features, *see* features
extra samples, 59

F

fallback profile, *see* profile, fallback
FDL, *see* GNU Free Documentation License
feathering, 1
features, 11
file
  multi-page, 44
  response, 17, 39
filename
  literal, 17
  output, 20
    default, 20
fine mask, *see* mask, fine
floating-point TIFF image, *see* image, floating-point TIFF
floating-point VIFF image, *see* image, floating-point VIFF
floating-point image, *see* image, floating-point
flow, *see* program flow
format
  image, 34
format of response file, *see* response file format
frozen seam-line endpoint, *see* seam-line, endpoint, frozen
fusing pixels, *see* pixels, fusing

G

Ganos Lal, *see* Morgan le Fay
Gavrilova, Marina, 47
GC, *see* graph-cut algorithm
generator
  seam, 34
glob, 43
globbing algorithm, 42
  `literal`, 42, 43
  `none`, 43
  `shell`, 43
  `sh`, 43
  `wildcard`, 42, 43
globbing algorithms, *see* algorithm, globbing, 43
GNU FDL, *see* GNU Free Documentation License
GNU Free Documentation License, 71
grammar
  response file, *see* response file, grammar
  syntactic comment, 42
graph-cut (GC), 34
graph-cut algorithm, 47
graphcut
  details, 47
  limitations, 48

H

half precision float (OpenEXR), *see* OpenEXR, half precision float
HDR, 64
HDR-bracketing, *see* bracketing, HDR
header files, 35
help, 34
helpful libraries, 64
helpful programs, 63–65
  HDR, 64
  camera firmware, 65
  High Dynamic Range, 64
  image alignment, 63
  image manipulation, 64
  image rendering, 63
  libraries, 64
  meta-data handling, 65
  raw image conversion, 63

hue-luminance maximum, 31

I

ICC, 65
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, 23
  single precision float, 23
image
  EXR, 54
  black-and-white, 56
  directory, 44
  floating-point, 54
  floating-point TIFF, 54
  floating-point VIFF, 54
  frame, 46
  seam-line, *see* seam-line, visualization
image difference, 31
image formats, 12, 13, *see* format, image
  JPEG, 12
  OpenEXR, 13
  PNG, 12
  TIFF, 12
image match quality, *see* quality, match
image processing order, *see* order of image processing
image requirements, 17
information
  on software components, 35
information options, 34–35
input mask, *see* mask, input files
interaction with Enblend, 8–16
invocation, 17–46

J

Jackson, Daniel, 19, 30, 62
JFIF, *see* JPEG
JPEG, 64
JPEG compression, *see* compression
JPEG quality level, *see* compression

K

von Kármán, Theodore, 24
known limitations, 3–4

L

L*a*b* colorspace, *see* colorspace, L*a*b*
LaunchPad
  bug database, *see* bug, database at LaunchPad
layer
  image, 44
  selection, 44–46
layer selection, 28
  all layers, 28
  default, 42
  first layer, 28
  largest-layer, 28
  last layer, 28
  no layer, 28
layer selection syntax, 44
lens distortion
  correction of, 7
level
  verbosity, 20
levels
  pyramid, 19
LibJPEG, 64
LibPNG, 64
libraries, 13
LibTiff, 65
Lindbloom, Bruce, 53
literal filename, *see* filename, literal
load mask, *see* mask, load
log-transform, *see* transform, log
loops in seam line, *see* seam line, loops
luminance, 52
luminance interval, 53

normalized, 53
trivial, 21, 28
luminance weight, *see* weight, luminance
luminance-hue maximum, 31
LZW compression, *see* compression

M

mask, 59
binary, 59
coarse, 25
external, 7
fine, 25
generation, *26*
input files, 59
load, 26
optimization visualization, 27
save, 26
template character, *29*
'%', 29
'B', 29
'b', 29
'D', 29
'd', 29
'E', 29
'e', 29
'F', 29
'f', 29
'i', 29
'n', 29
'P', 29
'p', 29
vectorization distance, 32
weight, 59, 60
mask generation options, 25–28
match quality, *see* quality, match
matte, 59, 60
maximum hue-luminance, 31
message
category, 14
error, 14
info, 15
note, 15
timing, 15
warning, 14
console, 14
debug, 15
foreign sources, 15
"should never happen", 15
Morgan le Fay, 62
movable seam-line endpoint, *see* seam-line, endpoint, movable
multi-page file,
see also layer, 44, *see* file, multi-page
tools, 46
multi-resolution spline, 1

N

name of builder, 13
nearest feature transform, 47
nearest feature transform (NFT), 34
nearest-feature transform (NFT), 26
NFT, *see* nearest feature transform
notation, vii

O

O'Neill, Jack, 30
`OMP_DYNAMIC`, *see* environment variable, `OMP_DYNAMIC`
`OMP_NUM_THREADS`, *see* environment variable, `OMP_NUM_THREADS`
only save mask, *see* save mask, only
OpenEXR, 64
data format, 23
half precision float, 23
OpenMP, 14, 16
optimize
anneal parameters, 30
seam, 26
strategy, 27
optimizer
seam-line, 22
simulated annealing, 30

optimizer weights, *see* weight, optimizer
option delimiters, *see* options, delimiters
options, 18–37
  advanced, 20
  common, 18
  delimiters, 37
    filename arguments, 37
    numeric arguments, 37
  expert, 28
  information, 34
  mask generation, 25
  mask generation for experts, 30
  program flow, 35
order
  of image processing, 39
output
  file compression, 18
output filename, *see* filename, output
  default, *see* filename, output, default
output image
  set size, 24
overview, 1–2

P

packbits compression, *see* compression
panorama
  360°
    horizontal, 25
    vertical, 25
parallax error, 7
perceptual rendering intent, *see* rendering intent
photographic workflow, 5–16
photometric alignment, *see* alignment, photometric
pixels
  blending, 52
  fusing, 52
PNG, 64
preassemble, 30
primary seam generator, *see* seam, primary generator
problem reports, *see* bug reports
profile
  fallback, 28
  ICC, 13, 21, 52
program flow, 35–37
  internal, 35
public repository, 11
pyramid levels, *see* levels, pyramid

Q

quality
  match, 32
query
  compiler, 13
  features, 11
  image formats, 12
  libraries, 13
  name of builder, 13
query version, *see* version, query

R

radius
  Dijkstra, 31
  Dijkstra, 50
raw conversion, *see* conversion, raw
rendering intent
  perceptual, 57
requantization, 22
requirements
  image, 17
response file, *see* file, response
  comment (‘#’), *see* ‘#’
  force recognition of, 41
  format, 40
  grammar, 40
  syntactic comment, 41
response file prefix
  ‘@’, *see* ‘@’
response files, 39–44
result
  checkpoint, 30
return code, 14
RGB color cube, *see* color cube, RGB, *see* color cube,

RGB
RGB-cube, 28

S

save mask, *see* mask, save
  only, 27
seam
  primary generator, 34
seam generation, 47, 47–48
  details, 47
seam line
  loops, 33
seam optimization, *see* optimize, seam, 33
seam-line
  endpoint
    frozen, 50
    movable, 50
  visualization, 49
  visualization example, 49
  visualization image colors, 50
sequential blending, *see* blending, sequential, *see* blending, sequential
signature, 13, 35
signed binary, *see* signature
simulated annealing optimizer, *see* optimizer, simulated annealing
single precision float (IEEE754), *see* IEEE754, single precision float
size
  canvas, 24
software
  components, *see* information, on software components
  version, 35
software version, *see* version, software
source code repository, *see* public repository
SourceForge, 2
spline, *see* multi-resolution spline
sRGB, 21
sRGB color cube, *see* color cube, sRGB
standard workflow, *see* workflow, standard
syntactic comment
  grammar, 42
  response file, *see* response file, syntactic comment
syntax
  layer selection, 44
    grammar, 44

T

TIFF, 65
`TMPDIR`, *see* environment variable, `TMPDIR`
transform
  floating-point images, 54
  log, 54, 55
transformation
  affine, 7
typographic conventions, *see* notation

U

unassociated alpha channel, *see* alpha channel, associated
understanding masks, 59–60

V

variable, *see* environment variable
vectorization distance, *see* mask, vectorization distance
verbosity level, *see* level, verbosity
version, *see* software, version
  query, 10
  software, 10
VIFF floating-point image, *see* image, floating-point VIFF
virtual reality, 24
visualization image, *see* seam-line, visualization
visualization image colors, *see* seam-line, visualization image colors

visualization image example, *see* seam-line, visualization example
visualization of mask, *see* mask, optimization visualization
VR, *see* virtual reality

W

weight
  chrominance, 31
  luminance, 31
  mask, *see* mask, weight
  optimizer, 33
weighting factor, 52
white-point, 56
  D50, 57
workflow, *see* photographic workflow
  Enblend, 6
  Enfuse, 6
  external mask, 9
  standard, 5
wrap around, 24