The Seamstress Library

Hayden Walles*

August 26, 2015

Seamstress is an open source library for image seam carving. Seam carving is a technique described by Avidan and Shamir [1] for changing the size or shape of an image without necessarily changing the size or shape of the *things* in the image. The best way to explain is with some examples. The images[1] in Figures 1 and 2 show how seam carving can be used to resize an image. In a nutshell seam carving resizes images by taking out the "boring bits".

This document is a programming guide to the Seamstress library. You don't need to read it if you just want to use programs developed using the library. Sections 1–3 describe seam carving and the general capabilities of Seamstress. Section 4 is a combined guide and reference to the functionality of the the library.

To get Seamstress and also Arachne, a program that demonstrates the library at work, visit the project website at `http://seam-carver.sourceforge.net/`.

1 A brief overview of seam carving

There are two stages to seam carving. First, the *energy* of each pixel in the image is computed. A pixel with high energy is considered important, one with low energy unimportant. There are lots of ways to determine the energy of a pixel; at the moment Seamstress uses the rate of intensity change around a pixel as its energy (see Appendix A for details). So edges end up with high energy because there is a sharp change in intensity, and large uniform areas end up with low energy. This energy function seems to work well for a wide range of images, but future versions of Seamstress will probably allow alternative energy functions to be selected or provided by the user.

Once each pixel has an energy, we can work out how to shrink the image by removing pixels with as little energy as possible. Imagine we are making

*hgwalles@gmail.com

[1]The example images shown were made available by their owners at flickr (`http://www.flickr.com`) with creative commons rights (`http://creativecommons.org`). The source of each is identified in the figure captions.

Figure 1: The image on the right was created by removing seams from the image on the left. Blue sky remains at the top of the picture, but the bridge remains the same size. “Bridge” was taken in the Copland Valley on the West Coast of New Zealand by flickr user Tom@North (`http://flickr.com/photos/tomsen/466989106/`).

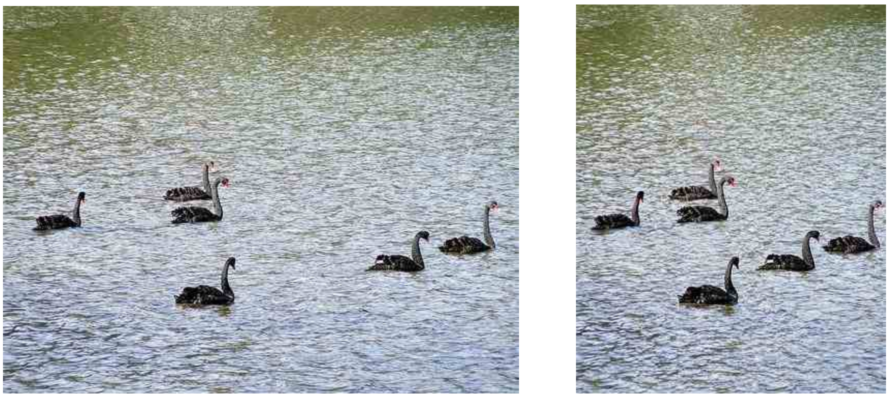

Figure 2: The image on the right was created by removing seams from the image on the left. The swans remain the same size and shape but appear closer together – only the water has been shrunk. “Birds @ Otago Peninsular, New Zealand” was taken on Otago Peninsula, New Zealand by flickr user timparkinson (`http://flickr.com/photos/timparkinson/263830393/`).

an image narrower, as in Figure 2. We could sum the energies of the pixels in each column, and remove the column with the lowest total. Unfortunately this distorts the image too much. Instead we work with seams. A vertical seem is a path through the image from top to bottom, with one pixel on each row, such that pixels on adjacent rows are no more than one column distant from each other.

In practice this means a vertical seam is a wiggly line from top to bottom. There are lots of vertical seams in an image, but some are *minimal* vertical seams. The total energy of all the pixels in a minimal vertical seam is the lowest of any vertical seam in the image. Exactly the same idea in the other direction leads to a minimal horizontal seam. Seam carving proceeds by identifying a minimal vertical (or horizontal) seam, and removing it, leaving an image one pixel narrower (or shorter). This can be repeated as often as desired to produce an output image of any size.

2 Seam maps

Imagine pulling *all* of the seams out of an image in a particular direction. You label all the pixels belonging to the first seam removed "1", all the pixels belonging to the second seam removed "2", and so on. Once you have pulled all the seams out, you have a *seam map*[1] in which every pixel is labelled with the number of the seam it belongs to.

Now an image with m seams removed can be reconstructed by gathering together only pixels from the original image whose label in the seam map is greater than m. This is how "real-time" seam carving is performed (the precomputation of the map only takes a few seconds).

Seamstress provides an easy interface to automatically compute seam maps for you (see Section 4.4).

3 Dynamic energy

Seamstress also provides a feature that is not discussed in Avidan and Shamir[1], dynamic energy. For seam carving as discussed above, the energy of each pixel is computed once, before any seams are removed. But since typical energy functions take in the context of each pixel, this approach can sometimes lead to poor results if many seams are removed.

Instead the energy can be recomputed between each seam removal, a technique I call dynamic energy. At first this may sound impractical, since the Seamstress energy function depends on convolution and is computationally expensive. However Seamstress takes advantage of the local nature of seam carving to only recompute the energy of pixels affected by each seam removal. Using dynamic energy is slower than using computing the energy only once, but only by about a factor of two.

In many cases dynamic energy provides better results.

4 The interfaces provided by Seamstress

Seamstress provides two interfaces for identifying seams in an image, as well as an way to manually protect and expose pixels of an image and a way to create a *seam map* from an image. The Unpicker seam interface carves seams only in one direction (only vertically, or horizontally) while the Unweaver interface allows arbitrary carving in both directions.

The reason two seam carvers are provided is that optimisation is possible when seam carving is in only one direction. Also, as Seamstress is currently implemented, true horizontal seam carving is *much* slower than vertical carving. Unpicker gets around this by performing horizontal seam carving as vertical seam carving on a transposed version of the input (that is, a version that is rotated 90 degrees and flipped).

The chief differences between the two interfaces are:

- The Unweaver interface allows arbitrary sequences of horizontal and vertical seams to be removed, the Unpicker interface allows only vertical or only horizontal seam removals.
- The Unpicker interface allows dynamic energy to be used, the Unweaver interface does not.
- Only Unpickers may be used to create seam maps.

The strategy required to use both is similar.

1. Create an unpicker or unweaver the same size as the image you want to carve.
2. Initialise the unpicker or unweaver with the image data.
3. Compute the energy.
4. Remove seams, either directly or via the Map interface.
5. Perhaps obtain further information about the seam carved image (determine the size or transform coordinates).
6. Delete the unpicker or unweaver.

4.1 The Unweaver interface

This is the most general interface. It allows you to remove horizontal and vertical seams from an image, but horizontal seam removal is *much* slower than vertical seam removal. It does currently support dynamic energy computation (see Section 3).

4.1.1 Creating an unweaver

`SEAM_UNWEAVER *seamstressNewUnweaver(int` *`actualwidth`*`, int` *`actualheight`*`,int` *`energyfunc`*`,int` *`dynamicenergy`*`,int` *`*error`*`);`
Creates a new Unweaver object.

actualwidth The width of the image you will be carving.

actualheight The height of the image you will be carving.

energyfunc Reserved for future development. This should always be zero at present.

dynamicenergy Reserved for future development. This should always be zero at present.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

The result, if successful, is a pointer to a new Unweaver object ready to be initialised with the image data. If unsuccessful, NULL will be returned, along with an appropriate error code if *error* is not NULL.

4.1.2 Initialising an unweaver

```
void unweaverSetRowRGB24(SEAM_UNWEAVER *pic, int row, unsigned char *pixels);
void unweaverSetRowRGB32(SEAM_UNWEAVER *pic, int row, unsigned char *pixels);
```

Initialises one row of an unweaver's image data. The unweaverSetRowRGB24 function accepts byte aligned RGB triplets. The unweaverSetRowRGB32 function accepts 32 bit words containing RGB bytes in the least significant bits, and ignores the high byte of each word. You should feed the entire input image into the unweaver immediately after creating it and before you compute the energy.

pic Pointer to the unweaver.

row the zero-based row number to set.

pixels pointer to the pixel data (aligned as described above) for the row.

4.1.3 Deleting an unweaver

```
int unweaverDelete(SEAM_UNWEAVER *weaver);
```

Frees all the memory associated with an unweaver. Use when you are finished.

weaver Pointer to the unweaver to delete.

4.1.4 Computing the energy

```
int unweaverComputeEnergy(SEAM_UNWEAVER *pic,SEAM_MARKS *marks,void (*updater)(int done,int of,void *user),void *user,int *error);
```

Computes the energy of an unweaver. Call this after the image data has been set, but before any seams are removed.

pic Pointer to the unweaver whose energy is to be computed.

marks Optional pointer to a Marks object whose annotations will be used to help compute the energy. This can be NULL if there are no manual marks to consider. See section 4.3 for details.

updater Optional pointer to a function that will be called from time to time as the computation proceeds and can be used to update a user interface with progress details. This can be NULL if no updater is desired. See section 4.5 for details.

user Pointer to user data that is passed to the updater function if one is provided.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

If successful nonzero is returned. If unsuccessful zero is returned, along with an appropriate error code if *error* is not NULL.

4.1.5 Seam carving

```
int unweaverNextVertical(SEAM_UNWEAVER *pic, int *seam,int *error);
int unweaverNextHorizontal(SEAM_UNWEAVER *pic, int *seam,int *error);
```

These remove a single vertical seam or a single horizontal seam, respectively. Once a seam is removed it cannot be inserted again.

pic Pointer to the unweaver to remove a seam from.

seam A pointer to a buffer long enough to hold the index of the pixel removed on each row or column of the seam. For example, if you are removing a vertical seam from an unweaver currently measuring 400×500 pixels, this buffer needs to have room for at least 500 integers. You can determine the exact size needed using the unweaverCurrentSize function, but in many cases this will be unnecessary. The length of the buffer needed will never exceed the original height (for a vertical seam) or width (for a horizontal seam), so it is often possible to allocate the buffer once and reuse for each seam removed.

The n^{th} value returned in this buffer indicate the column (for a vertical seam) or row (for a horizontal seam) of the pixel removed on the n^{th} row or column. These coordinates are relative to the image as it would be *if it had been carved to the state reached before the call*. They are *not* relative to the original image. This is a difference from the Unpicker interface.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

If successful, nonzero will be returned and the buffer pointed to by *seam* will be filled. If unsuccessful, zero will be returned, along with an appropriate error code if *error* is not NULL.

4.1.6 Obtaining the current size

```
void unweaverCurrentSize(SEAM_UNWEAVER *weaver, int *widthptr,int *heightptr);
```

Returns the current size of the specified unweaver. The current size is the size the input image would have reached after the sequence of horizontal and vertical seam removals already performed via unweaverNextHorizontal and unweaverNextVertical.

weaver Pointer to the unweaver whose size is desired.

widthptr A pointer to an integer that will receive the current width.

heightptr A pointer to an integer that will receive the current height.

4.1.7 Transforming coordinates

`int unweaverTransformToOriginal(SEAM_UNWEAVER *`*`weaver`*`, int *`*`xptr`*`, int *`*`yptr`*`);`

This function is useful when reconstructing the image that would result from the seam removals performed so far. It translates coordinates in the specified unweaver to coordinates in the original image. Put another way, it takes the coordinates of a pixel in the seam-carved image, and returns the coordinates of that pixel in the original image. So to reconstruct the seam-carved image, all a program need do is iterate over the width and height of the output, translating each position to the corresponding position in the input, where the colour data will be found for that pixel.

For many applications the actual seams removed (as returned in the buffers passed to unweaverNextVertical and unweaverNextHorizontal) can be ignored, and this function used to determine the result of the seam carving.

weaver Pointer to the unweaver to use.

xptr Pointer to the x coordinate to transform.

yptr Pointer to the y coordinate to transform.

Returns nonzero if the transformation was completed successfully. Returns zero if the coordinates were out of range of the unweaver's current size.

4.2 The Unpicker interface

The Unpicker interface lets you remove either horizontal or vertical seams from an image, but *not both.* If you want to remove seams in both directions arbitrarily, use the Unweaver interface (Section 4.1).

4.2.1 Creating an unpicker

`SEAM_UNPICKER *seamstressNewUnpicker(int `*`actualwidth`*`, int `*`actualheight`*`,int `*`direction`*`,int `*`energyfunc`*`,int `*`dynamicenergy`*`,int *`*`error`*`);`

Creates a new Unpicker object.

actualwidth The width of the image you will be carving.

actualheight The height of the image you will be carving.

direction The direction seams will be carved in. Zero indicates only vertical seams will be removed, one indicates only horizontal seams will be removed.

energyfunc Reserved for future development. This should always be zero at present.

dynamicenergy If nonzero then the energy will be recomputed after each seam is removed. See Section 3 for details.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

The result, if successful, is a pointer to a new Unpicker object ready to be initialised with the image data. If unsuccessful, NULL will be returned, along with an appropriate error code if *error* is not NULL.

4.2.2 Initialising an unpicker

```
void unpickerSetRowRGB24(SEAM_UNPICKER *pic, int row, unsigned
char *pixels);
void unpickerSetRowRGB32(SEAM_UNPICKER *pic, int row, unsigned
char *pixels);
```

Initialises one row of an unpicker's image data. The unpickerSetRowRGB24 function accepts byte aligned RGB triplets. The unpickerSetRowRGB32 function accepts 32 bit words containing RGB bytes in the least significant bits, and ignores the high byte of each word. You should feed the entire input image into the unpicker immediately after creating it and before you compute the energy.

pic Pointer to the unpicker.

row the zero-based row number to set.

pixels pointer to the pixel data (aligned as described above) for the row.

4.2.3 Deleting an unpicker

```
int unpickerDelete(SEAM_UNPICKER *picker);
```

Frees all the memory associated with an unpicker. Use when you are finished.

picker Pointer to the unpicker to delete.

4.2.4 Computing the energy

```
int unpickerComputeEnergy(SEAM_UNPICKER *pic,SEAM_MARKS
*marks,void (*updater)(int done,int of,void *user),void *user,int
*error);
```

Computes the energy of an unpicker. Call this after the image data has been set, but before any seams are removed.

pic Pointer to the unpicker whose energy is to be computed.

marks Optional pointer to a Marks object whose annotations will be used to help compute the energy. This can be NULL if there are no manual marks to consider. See section 4.3 for details. Note that if the unpicker is using dynamic energy that the Marks object must persist until all desired seams have been removed.

updater Optional pointer to a function that will be called from time to time as the computation proceeds and can be used to update a user interface with progress details. This can be NULL if no updater is desired. See section 4.5 for details.

user Pointer to user data that is passed to the updater function if one is provided.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

If successful nonzero is returned. If unsuccessful zero is returned, along with an appropriate error code if *error* is not NULL.

4.2.5 Seam carving

`int unpickerNextSeam(SEAM_UNPICKER *pic, int *seam,int translate,int *error);`
Removes a single seam from an unpicker.

pic Pointer to the unpicker to remove a seam from.

seam A pointer to a buffer long enough to hold the index of the pixel removed on each row or column of the seam. For example, if you are removing a vertical seam from an unpicker 500 pixels high, this buffer needs to have room for at least 500 integers. The length of the buffer needed will always be the original height (for a vertical seam) or width (for a horizontal seam), so it is often possible to allocate the buffer once and reuse for each seam removed.

The n^{th} value in the buffer indicates the column (for a vertical seam) or row (for a horizontal seam) of the pixel removed on the n^{th} row or column. The precise interpretation depends on the value of the *translate* parameter. If this is zero then these coordinates are relative to the image as it would be *if it had been carved to the state reached before the call.* If it is nonzero they are relative to the *original image.*

translate A flag that determines whether the returned coordinates will be relative to the current or original image. See the description of the *seam* parameter for details.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

If successful, nonzero will be returned and the buffer pointed to by *seam* will be filled. If unsuccessful, zero will be returned, along with an appropriate error code if *error* is not NULL.

4.2.6 Obtaining the current size

`void unpickerCurrentSize(SEAM_UNPICKER *picker, int *widthptr,int *heightptr);`

Returns the current size of the specified unpicker. The current size is the size the input image would have reached after the sequence of seam removals already performed via unpickerNextSeam.

picker Pointer to the unpicker whose size is desired.

widthptr A pointer to an integer that will receive the current width.

heightptr A pointer to an integer that will receive the current height.

4.2.7 Transforming coordinates

`int unpickerTransformToOriginal(SEAM_UNPICKER *picker, int *xptr, int *yptr);`

This function is useful when reconstructing the image that would result from the seam removals performed so far. It translates coordinates in the current unpicker to coordinates in the original image. Put another way, it takes the coordinates of a pixel in the seam-carved image, and returns the coordinates of that pixel in the original image. So to reconstruct the seam-carved image, all a program need do is iterate over the width and height of the output, translating each position to the corresponding position in the input, where the colour data will be found for that pixel.

For some applications the actual seams removed (as returned in the buffers passed to unpickerNextSeam) can be ignored, and this function used to determine the result of the seam carving.

picker Pointer to the unpicker to use.

xptr Pointer to the x coordinate to transform.

yptr Pointer to the y coordinate to transform.

Returns nonzero if the transformation was completed successfully. Returns zero if the coordinates were out of range of the unpicker's current size.

4.3 The Marks interface

The Marks interface allows the user to annotate locations in an image so that some parts are protected from or exposed to seam removal, overriding the automatic energy computation. This is done by setting the energy of pixels in protected regions to the maximum energy and that of pixels in exposed regions to zero energy. Seam removal is thus encouraged to remove exposed pixels and preserve protected ones. Marks objects can be passed to the Unweaver and Unpicker energy computing functions (See Sections 4.1.4 and 4.2.4 respectively)..

4.3.1 Creating a Marks object

`SEAM_MARKS *seamstressNewMarks(int` *`width`*`, int` *`height`*`,int *`*`error`*`);`
Creates a new Marks object.

width The width of the image the new Marks object will be used with. This must be the same as the original image's width.

height The height of the image the new Marks object will be used with. This must be the same as the original image's height.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

If successful, a pointer to a new blank Marks object will be returned. If unsuccessful, zero will be returned, along with an appropriate error code if *error* is not NULL.

4.3.2 Interpreting annotations

Annotations at each pixel should be interpreted as specified in the following table.

Value	Interpretation
0	The automatic energy should stand.
1	Protect the pixel by setting its energy to the maximum value.
2	Expose the pixel by setting its energy to zero.

4.3.3 Adding annotations

`void setMark(SEAM_MARKS *`*`marks`*`,int` *`row`*`,int` *`col`*`,int` *`type`*`);`
`int marksAnnotate(SEAM_MARKS *`*`marks`*`, int` *`l`*`, int` *`t`*`, int` *`r`*`, int` *`b`*`,int` *`type`*`);`
Annotates a pixel or a rectangular region of the Marks object.
The arguments for the setMark macro are as follows.

marks Pointer to the Marks object to annotate.

row Row of the pixel to annotate.

col Column of the pixel to annotate.

type Indicates the type of annotation to make. See Section 4.3.2 for the encoding.

The arguments for the marksAnnotate function are as follows.

marks Pointer to the Marks object to annotate.

l x coordinate of the left side of the rectangle.

t y coordinate of the top side of the rectangle.

r x coordinate of the right side of the rectangle.

b y coordinate of the bottom side of the rectangle.

type Indicates the type of annotation to make. See Section 4.3.2 for the encoding.

The function always returns one.

4.3.4 Reading annotations

`int getMark(SEAM_MARKS *marks,int row,int col);`
Reads back any annotation for the specified pixel. See Section 4.3.2 for the encoding.

4.3.5 Deleting a Marks object

`void marksDelete(SEAM_MARKS *marks);`
Deletes a Marks object, freeing all associated memory.

marks Pointer to the Marks object to delete.

4.4 The Map interface

The Map automates the construction of seam maps (see Section 2). During the construction of a seam map for an image, each pixel in the image is labelled with the number of the minimal seam to which it belongs. Seam maps can only be created when seams are removed in a single direction (horizontally or vertically).

4.4.1 Creating a seam map

`SEAM_MAP *unpickerMap(SEAM_UNPICKER *pic,void (*updater)(int done,int of,void *user),void *user, int *error);`
Creates a new Map object from an Unpicker object.

pic Pointer to the unpicker from which the map should be generated. The unpicker must be initialised and its energy must have been computed. No seams can have been removed from it. If the function returns successfully the unpicker will be exhausted (no seams will be left for removal). It is not required once the function returns.

updater Optional pointer to a function that will be called from time to time as the computation proceeds and can be used to update a user interface with progress details. This can be NULL if no updater is desired. See section 4.5 for details.

user Pointer to user data that is passed to the updater function if one is provided.

error Pointer to an integer in which an error code will be deposited, or NULL if no error code is desired.

The result, if successful, is a pointer to a new Map object. If unsuccessful, NULL will be returned, along with an appropriate error code if *error* is not NULL.

4.4.2 Deleting a seam map

`void mapDelete(SEAM_MAP *map);`
Deletes the specified Map object, freeing all memory associated with it.

map Pointer to the Map object to delete.

4.4.3 Reading a seam map

`int getMap(SEAM_MAP *map,int row,int col);`
A macro that returns the seam number of a pixel in the original image.

map Pointer to the Map object to read.

row Row of the pixel in the original image.

col Column of the pixel in the original image.

Returns the 1-based number of the seam that the pixel belongs to.

4.5 Updater functions

Some functions that do a lot of computation take updater call-back functions that will be called from time to time during the computation to allow the user to, for example, update an on-screen progress indicator. The arguments of this function are described below.
`void (*updater)(int done,int of,void *user);`

done A number indicating how much of the job is complete. The proportion of the job completed is given by $\frac{done}{of}$.

of A number to which the *done* parameter should be compared when judging completion.

user The pointer to user data passed in to the function that to which the updater function was also passed.

A Energy function

The energy function currently used by Seamstress is an approximate measure of the gradient parallel to the x and y axes of a slightly blurred version of the input image.
If $\mathbf{I}$ is the original image, then the energy of that image can be defined as

$$E(\mathbf{I}) = |\frac{\partial \mathbf{I}'}{\partial x}| + |\frac{\partial \mathbf{I}'}{\partial y}|$$

where $\mathbf{I}$ is the result of the convolution

$$\mathbf{I}' = ce^{\frac{-(x^2+y^2)}{\sigma^2}} * \mathbf{I}$$

with $\sigma = 1$ and c is a normalizing constant.

B GNU Free Documentation License

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 http://www.gnu.org/copyleft/.

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.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. 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".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

References

[1] Avidan, Shai and Shamir, Ariel. “Seam Carving for Content-Aware Image Resizing”. *ACM Transactions on Graphics* 26 (2007).