

WIFSTOOL, DIGITAL SUPPORT FOR IMAGE ANALYSIS RESEARCH

WESLEY SNYDER

1. Input File Formats	3
1.1. Wifs data types	3
2. Organization of Data	4
3. File Operations	5
3.1. Read	5
3.2. ReRead	5
3.3. SaveAs	5
3.4. Copy	5
3.5. Paste	6
4. Viewing an Image	6
4.1. Viewing Complex Images	8
5. The Mode Button	9
6. Overlays	9
6.1. Drawing an overlay	10
6.2. Viewing an Overlay	10
6.2.1. Using the overlay check box	10
6.2.2. Viewing Single Frames	10
6.3. Saving an Overlay	10
6.3.1. Saving Overlay and Image Together	10
6.3.2. Saving just the Overlay	11
6.4. Deleting an Overlay	11
7. Unary (one input) Operations	11
7.1. Functions available in mode 0	12
7.2. Functions available in mode 1	15
8. Binary Operations	17
9. Frequency Domain Operations	19
10. Comments	20
11. References	20
References	20
Index	21

Contents

Wifstool is a GUI which supports the Image Analysis researcher. It supports images of any size¹, any number of dimensions², and a wide variety of formats.

The program runs on Linux, Mac OSX, and Windows 7, 8 and 10. Files are interchangeable between platforms.

The program can do Fourier analysis and filtering.

Finally, the program can run an external script, writing an image out, running the script, and reading the script back in, thus providing general image analysis capability.

1. INPUT FILE FORMATS

Internally, Wifstool uses the *Wifs* (wonderful image file system). All images are internally converted into Wifs. Wifs supports any computer data type, including complex and structure.

1.1. Wifs data types. Most Wifs functions can operate on any of the following image data types:

u8bit: Unsigned 8 bit. (unsigned char)

char: Signed 8 bit

short: 16 bit

Ushort: unsigned short

int: 32 bit integer

float: 32 bit floating point

double: 64 bit floating point

32cmp: Complex number with 32 bit float real and 32 bit float imaginary

64cmp: Complex number with 64 bit float real and 64 bit float imaginary

S3uchar: Structure consisting of three unsigned characters per pixel

S4uchar: Structure consisting of four unsigned characters per pixel

S3ushort: Structure consisting of three 16 bit unsigned integers per pixel

S4ushort: Structure consisting of four 16 bit unsigned integers per pixel

S3float: Structure consisting of three 32 bit floating point numbers per pixel

S4float: Structure consisting of four 32 bit floating point numbers per pixel.

The structure data (with name starting with S), are only available in Wifstool 10.2 or later.

Wifstool can read and write images in any of the following formats:

¹Until the process runs out of memory

²Also limited by memory capacity

FIGURE 1. Any of eight different image buffers may be used as input or output. This example illustrates how to set up for an operation which will read one input from buffer zero, a second input also from buffer zero, and the result of the operation will be placed in buffer one. Reading a file is a special operation in that it does not require an input buffer, however the **OUTPUT** buffer still needs to be specified.



- Portable Network Graphics (PNG, png)
- Joint Photographics Experts Group (JPG, jpg, jpeg)
- Tagged Image File Format (TIFF, tif, tiff)
- Wifs File Format) (Wifs, ifs)
- Graphics Interchange Format (GIF)
- Bitmap File Format (bmp).

2. ORGANIZATION OF DATA

If the user has bound the Wifstool to a class of files (e.g. png), then one may simply double click a file and the application will load the file. Or, the user may choose to press the *file* button. A file select window will open, and the user may select the desired file.

Wifstool can simultaneously hold eight images in memory, referred to here as *buffers*, numbered 0-7, as illustrated in Figure 1. Any of the 8 image buffers may be used as **INPUT1**, **INPUT2**, or **OUTPUT** of operations. The Read operation in the example of Figure 1 will load the target file into the image buffered selected as **OUTPUT**.

The internal buffers are usually of type “float” or “complex float”, and have the same number of dimensions as the selected input image. If the image is color (as is default for png), it will be three dimensional and have three frames. An image may also have more than two dimensions and not be color. For example “movie loops” allow the user to play a sequence of images as if they were a movie.



FIGURE 2. File operations, left-to-right, are file Read, file ReRead, SaveAs, Copy, Paste, ReRun, View.



FIGURE 3. The ReRead button is emphasized

By using four dimensions, one for time and three for color, one may have color movies, however, this program performs all operations in memory, and therefore is not really intended for to be used for manipulating color movies.

Nor is it designed for operations typically performed by Photoshop®.

3. FILE OPERATIONS

Most file operations may be accessed by either the file pulldown, or across the top of the main window as illustrated in Figure 2.

3.1. Read. This operation will open a file selection window, allowing the user to choose any type of image file listed above. It will be read into whichever buffer is selected as **OUTPUT**.

3.2. ReRead. Once a file has been read, Wifstool saves the name of that file in two places, in the program and on the disk. Clicking ReRead (as illustrated in Figure 3) will read the file as stored in the program, and if that string is empty, it open the file as specified in the file list on disk. Thus, even the first time the program is loaded, using ReRead will load an image (unless, of course, the program has never previously been run).

3.3. SaveAs. This function will save the buffer selected as **OUTPUT** to a Wifs file of any data type or any “standard” format, including png, jpg, ..., as well as Wifs. If the output data type is Wifs, a popup will ask the user to specify the data type, (e.g. float, u8bit, int). Note that this operation saves one of the 8 image buffers to disk. It is different from the SaveWorkingImage function which is only available at the top of the working image³ The operation described here saves the buffer, unchanged by the operations on the working image, whereas the SaveWorkingImage saves those changes as well.

3.4. Copy. Copies a selected output buffer to the clipboard.

³SaveWorkingImage will be discussed in section 4.

3.5. **Paste.** Pastes a copied buffer from the clipboard to currently selected output buffer.

4. VIEWING AN IMAGE

The *view* button is the blue button at the center-top of the main window, between the ReRun and mode buttons. When the *view* button is pushed, the image selected as **OUTPUT** will be displayed into a new window, called *the viewing window* here. If the image just read in and selected as **OUTPUT** is 2-D or color, the *View* button will display that image. For purposes of display, the image is converted to eight bit unsigned byte. Color images with complex pixels are not supported.

When viewing, a new window, called the *viewing image* is opened displaying the image. There are additional functions available on the viewing window. These include:

Modify Brightness: A color image may be made perceptually brighter by adding white. These modifications are only made in the viewing image, and modifying them does not affect the master images, (those stored in buffers 0-7).

Viewing a Particular Frame: A color image may be considered as composed of three images, red, green, and blue. Usually, the user seeks to view the three together as a single color image, and if the color check box is checked, this is the model. However, the user may choose to view each frame independently as a single gray scale image. That view mode is used if the color check box is unchecked. If this is a color image, frame 0 is red, frame 1 is green, and frame 2 is blue. If an overlay exists, it will be in frame 3. Overlays are discussed in section 6.

Images consisting of more than three frames can also be viewed by unsetting the color check box and clicking the frame select button. This is the usual mode for viewing grayscale movies.

Color Model: The appearance of an image may be modified by adding or subtracting yellow or blue to increase the “warmth” or “coolness” of the image. These modifications are only made in the viewing image, and modifying them does not affect the master images, (those stored in buffers 0-7).

Color Enhancement: The appearance of an image may also be modified by adjusting the degree of red, green, or blue in the image. These modifications are only made in the viewing image, and modifying them does not affect the master images, (those stored in buffers 0-7). More sophisticated image enhancement is available by using the histogram equalization operation as described in section 7.1.

Zoom: Using the zoom slider makes the viewed image larger or smaller, but does not affect the master images, (those stored in buffers 0-7).

Bwindow and Blevel: Bwindow (Brightness window) and Blevel (Brightness Level) are parameters used to enhance or reduce contrast. They allow the image to have a brightness range of more or less than 0-256. This is particularly useful with medical

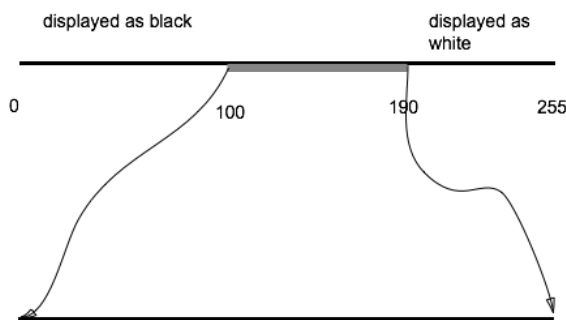


FIGURE 4. The Blevel and Bwindow controls choose a brightness range with a Bwindow of 90 and a Blevel of 100 and map it to 0-255. A brightness range of Bwindow value =90 and Blevel value of 100 will take the range 100-190 and display it as 0-255. All pixels with brightness below 100 will be black and all above 190 will be white.

images which are created with ten bit analog-digital converters and therefore have brightnesses ranging from 0 to 1023.

The terms *Bwindow* and *Blevel* are most meaningful if one considers display hardware in which the brightness values range between, say, 0 and 1023, but has internal hardware that can only display brightnesses between 0 and 255. The *brightness window* defines the brightness range which is mapped (linearly) to lie between 0 and 255. The *brightness level* defines where in the total range the brightness window lies. For example, one might have an image which has brightnesses between 0 and 1023, but which only has information between brightnesses of 150 and 180. One could display this with a level of 150 and a window of 30. Thus a brightness of 150 would appear as totally black and 180 as totally white. This is the most controllable way to modify contrast.

As an example, consider a total allowable brightness range of 0-255; however, suppose in this image, almost all the pixels have brightness between 100 and 190. Then, we might want to display any image with a brightness less than or equal to 100, and display them as 0 (black), and take any pixel with brightness greater than or equal to 190 and display it as 255 (bright), as illustrated in Figure 4. Then, brighter pixels are set to white and darker ones to black as illustrated in Figure 5.

Brightness Profile: A graph of brightness along an arbitrary path in the image can be acquired as follows:

- (1) The words "Profiler OFF" appear at the upper left of the viewing window.
- (2) Clicking on this will change the words to read "Profiler ON."
- (3) When profiling is on, hold the mouse button down and move the pointer over the image. The path is shown in red.

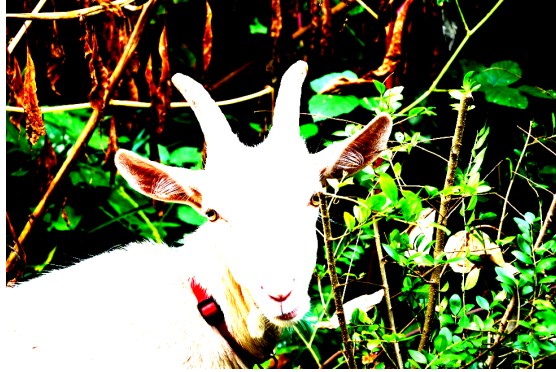


FIGURE 5. An image which has a small brightness window, as a result of manipulating Bwindow and Blevel.

- (4) Upon release of the mouse, a graph will be plotted showing the brightness along that path.

This function works on grayscale and color images.

Histogram: A histogram (see page 153 of [1]) of a selected set of pixels can be obtained as follows:

- (1) The words “Histogram OFF” appear at the upper left of the viewing window.
- (2) Clicking on this will change the words to read “Histogram ON.”
- (3) When Histogram is ON, hold the mouse button down and move the pointer over the image. The path will be shown in red.
- (4) Upon release of the mouse, a graph will be plotted showing the histogram of brightness values along that path. NOTE: If the path is closed, this does NOT histogram the pixels interior to the path, only the pixels along the path.

SaveWorkingImage: The user may modify the image begin viewed using the functions previously described in this section. The viewing image uses 8bits per color, rather than the 32 bit floating point representation in the buffers. SaveWorkingImage will save the working image as well as the overlay, if one exists. Since the working image is 8 bits per color (24 bits per pixel), it is not capable of the dynamic range of the buffers. The purpose of Wifstool is not to perform picture processing, but to allow the user to see the result of Image Analysis operations.

4.1. Viewing Complex Images. Here, the term *complex* means each pixel of the image is a complex number (it has a real and an imaginary component).

If the image just read in and selected as **OUTPUT** is complex, the user should select it as **INPUT1**, and use the COMPLEX pulldown. Operations are listed below:



FIGURE 6. (LEFT) Image operations available in mode 0. (RIGHT) Operations available in mode 1. These operations are performed on the eight floating point image buffers.

Magnitude: will compute the magnitude of the **INPUT1** buffer and return it in the selected output buffer. This operation requires complex input.

Phase: will compute the phase of the **INPUT1** buffer and return it in the selected output buffer. This operation requires complex input.

Real: will extract the real part of the **INPUT1** buffer and return it in the selected output buffer. This operation requires complex input.

Imaginary: will extract the imaginary part of the **INPUT1** buffer and return it in the selected output buffer. This operation requires complex input.

5. THE MODE BUTTON

At the top of the main window, to the right of the blue viewing button, are the words “mode 0”. Clicking on these words will select “mode 1”. Changing modes changes the functions available to the user to a different set. The buttons to select operations are illustrated in Figure 6

6. OVERLAYS

Wifstool supports a single overlay frame. This is implemented by adding an additional frame to a color image. Thus, a color image with an overlay actually has four frames, r, g,

b, and o. The overlay frame is intended to be binary, either zero or 255. If a pixel in the overlay frame is nonzero, the corresponding pixel in the red frame will be complemented.

6.1. Drawing an overlay. To construct an overlay, display the input image, which may have 3 frames or 4 frames (if it already has an overlay). Then click the words “DrawOverlay OFF”, which may be found over the viewed image. The word “OFF” will switch to “ON.” Changing this to ON will create an overlay, if one does not already exist. Then, while holding the left mouse button down, draw a series of curves over the image. The curves you draw will appear in red.

Note: The current version of Wifstool requires that zoom be off (scale == 1) when drawing an overlay.

When you are finished, remember to click the words “DrawOverlay ON” to switch it to “OFF.”

6.2. Viewing an Overlay.

6.2.1. Using the overlay check box. When you draw the overlay, it will appear on the screen. To make it disappear, note the overlay check box on the top right of the viewing panel. When this is checked, the overlay will appear superimposed over the image. Depending on the state of the drawing process, you may need to check it on, then off, then back on.

Two dimensional images may not have an overlay, however, if you have a 2D image, it can be trivially changed into a 3D image (in which all three frames are the same) by using the 2D 2 3D function.

6.2.2. Viewing Single Frames. If you wish to view the overlay as a single frame, click the COLOR check box to turn off color. Then, the individual frames, red, green, blue, and overlay may be viewed separately.

6.3. Saving an Overlay. You may save an overlay as a file together with the image, or you may save just the overlay. Remember that SaveWorkingImage saves the image on the screen, so if you save the image when it is visible, it will be part of the RGB image, as well as being a fourth frame. You probably want to uncheck the overlay check box before doing a save.

6.3.1. Saving Overlay and Image Together. If you wish to save the image and its overlay together as a single file, you must use the Wifs file format. Click the SaveWorkingImage button, specify the file name with the extension “Wifs”, and the target directory. You will be prompted for the data type (float is recommended). The resulting Wifs file will be readable by Wifstool.

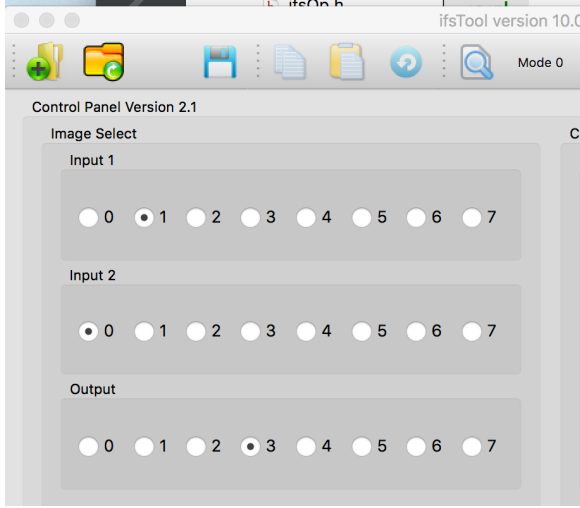


FIGURE 7. A possible organization of input and output buffer selection for the DX command: **INPUT1** is the image in buffer 1, **OUTPUT** is buffer 3, and **INPUT2** will be ignored. Many of the functions will prompt the user for the scale of the operation (σ of the Gaussian whose derivative is used as the kernel).

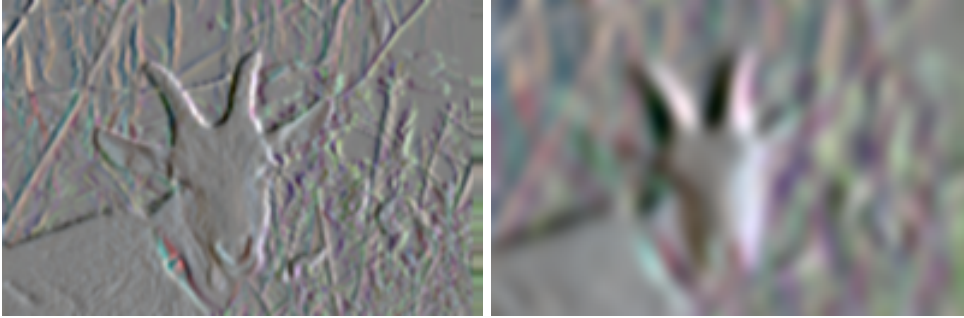


FIGURE 8. The output of the derivative with respect to x with a scale of 1 (LEFT) and 4 (RIGHT).

6.3.2. *Saving just the Overlay.* To save the Overlay independent of the original image, use the single frame viewing mode, as mentioned above, and view the overlay. Then, use the SaveWorkingImage button to save the overlay. You may use any of the standard data types such as jpg or png, or you may use Wifs.

6.4. **Deleting an Overlay.** Pressing the mouse left-click while holding down the Shift key while the drawOverlay is ON will remove the overlay entirely.

7. UNARY (ONE INPUT) OPERATIONS

The input image to all the unary operations is the one selected as **INPUT1**. The output will be the one selected as **OUTPUT**. For example, Figure 7 illustrates one choice of input and output for the DX operation(derivative with respect to x).

With large images (e.g. 4000×6000), some functions can require noticeable time to run. When executing a function, no other functions are available. The buttons are grayed out. In addition, the View button is grayed out until the function is complete.

7.1. Functions available in mode 0.

$\frac{\partial f}{\partial x}$: Computes an estimate of the derivative of brightness with respect to x (the column direction) See page 61 of [1].

$\frac{\partial f}{\partial y}$: Computes an estimate of the derivative of brightness with respect to y (the row direction).

$\frac{\partial^2 f}{\partial x^2}$: Computes an estimate of the second derivative of brightness with respect to x (the column direction).

$\frac{\partial^2 f}{\partial y^2}$: Computes an estimate of the second derivative of brightness with respect to y (the row direction).

$\frac{\partial^2 f}{\partial x \partial y}$: Computes an estimate of the second derivative of brightness with respect to x and y (the cross term).

Connected Components - CCL: Finds all the connected components in a 2D image. See page 166 of [1]. The output image identifies all the pixels in a particular component by the number of the component. The user is prompted for the parameters as shown below⁴ To choose the defaults just press the OK button.

- **Thresh** The threshold for connectivity. That is, if two adjacent pixels have a brightness difference greater than this amount, they are not in the same component (region).
- **Minval** A pixel must be at least this bright to be considered not background.
- **Maxval** A pixel must be less than or equal to this brightness to be considered not background.
- **EraseSmall** If this number is not zero, then any regions with area less than this specified value will be “erased,” that is, set to zero.
- **EraseLarge** If this number is not zero, then any regions with area greater than this specified value will be “erased,” that is, set to zero.
- **filenameofareas** If the user specified something in this field, that name will be used as the name (and path) of a file in which the area of each region will be specified.

Figure 9 illustrates filling in the connected components parameter box.

⁴Using the default values is generally a good starting point.

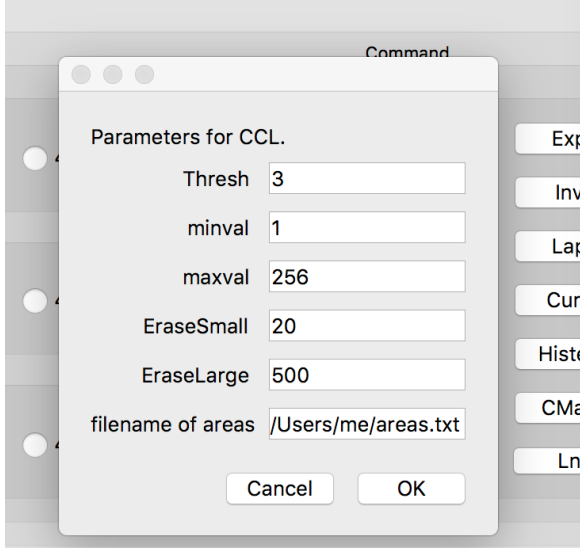


FIGURE 9. The parameters popup for the ccl (connected components labeling) command. Prior to a user entering values, the default values are shown. If no file name is given, the list of areas file will not be saved.

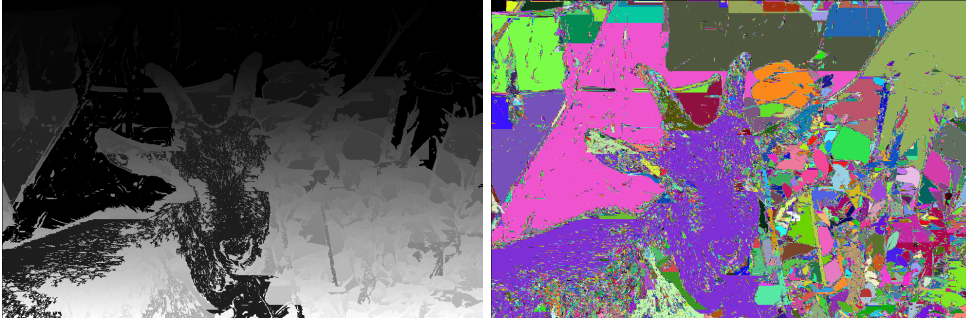


FIGURE 10. Illustrating the result of connected component labeling. The image on the left is the output of ccl. The pixel numbers are represented as brightnesses. The image on the right is the result of using random color assignment.

(This function is available only in release 10 of *Wifstool*.) Figure 10 illustrates the result of running connected components on the Vincent⁵ image.

Blur: The blur is implemented by a convolution with a Gaussian with a user-selectable kernel diameter.

Exponential: Computes $\exp\left(\frac{f(x,y)}{\tau}\right)$ at each point in the image. The scale value, τ , for which the user is prompted, allows brightness scaling changing a typical brightness of 200 to a brightness of 2.0, which, when exponentiated, is something reasonable. If the result is over 255.0, it is set to 255.0.

⁵Vincent Van Goat

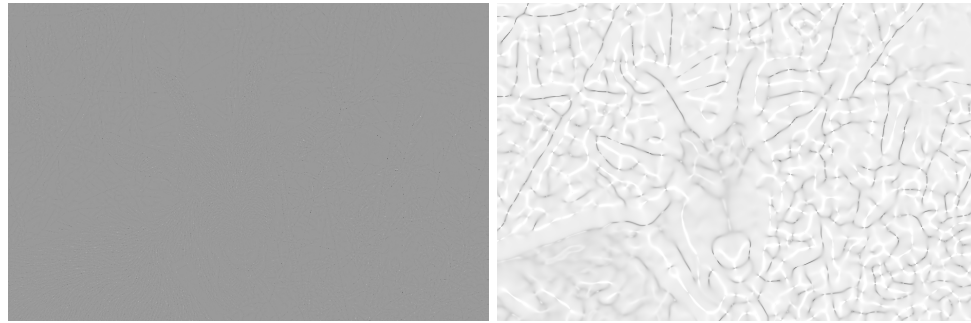


FIGURE 11. LEFT: curvature computed with a scale of 1.0, which matches only the noise in the image. RIGHT: curvature computed with a scale of 5.0, which is much closer to the scale of the features in the image.

Inverse: Dark areas become bright and bright areas become dark.

Laplacian: See page 69 of [1]. Computes $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ at each point in the image.

Curvature: See pages 176 and 244 of [1]. Computes the curvature at each point.

The curvature of a function f is defined by $\kappa = \frac{f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2}{(f_x^2 + f_y^2)^{3/2}}$. Curvature is heavily scale-dependent, Figure 11 illustrates two images illustrating the curvature of the same image, but this different scales (see page 245 in ??).

HistEq: Histogram equalization finds the image whose histogram is as close to flat as possible. Figure 12 illustrates a low contrast (overexposed) image on the left, and the same image after histogram equalization on the right. The modification of the brightness histogram makes the image much more realistic in this case. However, this version of the algorithm does nothing to constrain that the modification of pixel brightness preserve colors accurately.

Figure 13 illustrates another application of histogram equalization. On the left is a very high contrast image. Use of Window and Level will not produce detail in both the dark area and bright area simultaneously, but histogram equalization can.

PseudoColor: Given a 2D grayscale image in **INPUT1**, **OUTPUT** will be a color image where the brightness of each pixel has been remapped to a color. The user must specify how the mapping occurs using a “colormap.” Possible maps are:

- 0: gray scale
- 1: inverse gray scale
- 2: Bronson
- 3: Hot metal
- 4: Heated Spectrum



FIGURE 12. An overexposed image before and after histogram equalization.

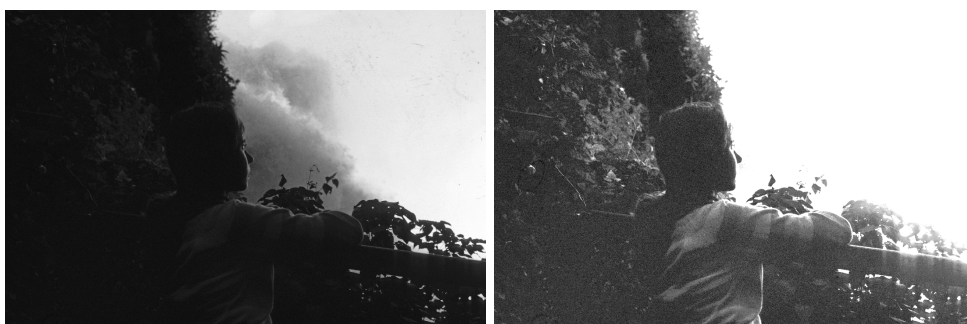


FIGURE 13. Histogram equalization can be used to show content in a very high contrast image.

- 5: Log
- 6: Random.

The Random map is surprisingly useful when evaluating noise removal algorithms (see page 96 in [1]). Because the colors are assigned randomly, if two adjacent points are identical in brightness, they will be displayed as the same color. Therefore large areas of constant color suggest a very smooth image.

NatLog: Computes $\ln f(x, y)$ at each point.

7.2. Functions available in mode 1.

Histogram of Gradient Orientation: This operation, abbreviated HoG, accepts a gray scale image and computes the histogram of the gradient orientation (see page 284 of [1]) for each 8×8 block of pixels. The output is a 3D image with 9 frames. Each frame represents a gradient direction. The brightness of point x, y in frame z denotes how often the gradient points in that direction.

ShowHoG: Since 9-dimensional images are difficult for humans to view, this function accepts as input the HoG image, and produces a better display image. That image displays 9 lines in each 8×8 block (See page 285 of [1]). The orientation of each line is the corresponding orientation, and the brightness of the line corresponds to the histogram entry, so that more common directions are displayed brighter. In this way, the user can easily determine dominant orientations at all locations over the image.

gMag: Displays the magnitude of the gradient at each point. (See page 17 of [1])

gDir: Displays the direction of the gradient at each point.

Color to Gray: Converts a color image to a gray scale image. This image may be saved to disk using the saveBuffer button at the top of the display window. The Clr2gray function converts a 3D, 3 frame color image to a 2D, 1 frame Grayscale image.

Absolute Value: It is possible for images to have negative values (for example, after using the SUB function). Then the ABS function can produce the absolute value, pixel-by-pixel.

Crop: Crop an image. Accomplished by displaying the image, clicking on the upper left and lower right corners of the cropped image, then selecting the output buffer and hitting the Crop button. The cropped image will be saved in the output buffer. It may be saved to disk using file→SaveAs, which will save the original, or by using the saveBuffer button on the display window, which will save the cropped image as an unsigned char Wifs image, or png, jpg, or other "standard" images.

Resize: This feature allows more precise control over zooming of an image than the zoom feature in the viewing window. The size of the destination image is specified, and the input image is spatially scaled to match that. Scaling may be up or down, and the horizontal direction and vertical directions may be scaled differently. This "zooming" process requires interpolation, and the user is asked to choose of of three modes as shown below:

Enter this number	To use this interpolation mode
0	Bilinear interpolation
1	Pixel replication
2	Four pixel averaging

We have found bilinear interpolation superior for most applications. If the line is left empty, bilinear will be used.

Qvar: Computes the *Quadratic Variation* (See page 64 of [1]) over the image. This is a more sensitive and better normalized version of the second derivative.

Harris Operator: The HarrisOp function computes the value of the Harris operator (See page 272 of [1]) at each point in the image, with user-selectable scale. Maxima of the Harris Operator generally indicate corners.

MarkMax: Sets to red all points in an image which are local maxima. (useful in conjunction with HarrisOp).

2D 2 3D: Sometimes, one needs to convert an image which is stored as a 2D image, to one which is 3D, for example, to create an overlay over the gray scale version of an image. This function provides that capability

8. BINARY OPERATIONS

arithmetic: The operations of add, subtract, multiply, and divide work as expected on float and complex data types (so they can be used for Fourier analysis). If the two input images are non-complex, the user has the option of adding, subtracting them, etc. with one translated in the image plane. In this case, the user is prompted for a displacement in the row direction and a displacement in the column direction. The displacement is the position of **INPUT2** relative to the (0,0) point of **INPUT1**. Figure 14 illustrates the result of adding two images with the displacement sufficiently large to place the second image outside the first.

When performing these four operations, both inputs must be grayscale or both must be color.

correlate: The form of correlation used here is the two-dimensional normalized correlation⁶. It works with both color and grayscale images. Since it performs in the space domain, it is more appropriate to applications in template matching, where one image is fairly small (under 64×64). It can take a **long** time to run.

You can always use Fourier methods to accomplish correlation of large images.

Edge-preserving noise removal: uses Piecewise-Constant Mean Field Annealing to remove the noise while keeping sharp edges, see [1], chapter 6.

This function finds the image, f , which minimizes the function

$$(1) \quad \sum_i \left(\frac{f_i - g_i}{\sigma^2} \right)^2 - \Gamma(f_i, \tau) ,$$

where Γ is a function which is maximized when f has the desired properties (in this case, is piecewise-constant). The function uses *annealing*, the process of slowly reducing some parameter as the algorithm runs to find an improved solution.

The function implemented in Wifstool has several parameters:

σ : The standard deviation of the noise in the image. The noise is assumed to be Gaussian and additive. We have found that for most images with brightnesses in the range 0-255, $\sigma = 3$ works well.

β : The relative strength of the *prior* term. (the β of Eq. 1), which weights the importance of smoothness vs. fidelity.

⁶See [1], page 268



FIGURE 14. The smaller image was moved up until it was outside the larger image. Then the two were added.

t1: The starting temperature. This is normally always set to 1.0 for 8 bit images. because it is normalized by the program.

t2: Normally 2 or three orders of magnitude lower than t1.

dlt: The rate of decrease of temperature. A geometric annealing schedule is used: $t = t \times dlt$, so 0.999 is much slower than .99.

All the parameters and defaults are shown on the popup which pops up when the pcon button is pushed.

This function only works on gray scale images.

Unlike ever other function in Wifstool, this function runs *in-place*. INPUT1 is f INPUT2 is g , and *No output needs to be specified*. The output can be viewed by setting OUTPUT to whichever buffer is chosen as INPUT1. Since it runs in place, (as illustrated in figure 15 it may be run in phases by setting t2 to something like 0.1, viewing the output, then changing t1 to 0.1 and t2 to 0.01 and continuing the program. This allows the user to run the program in segments and view the output

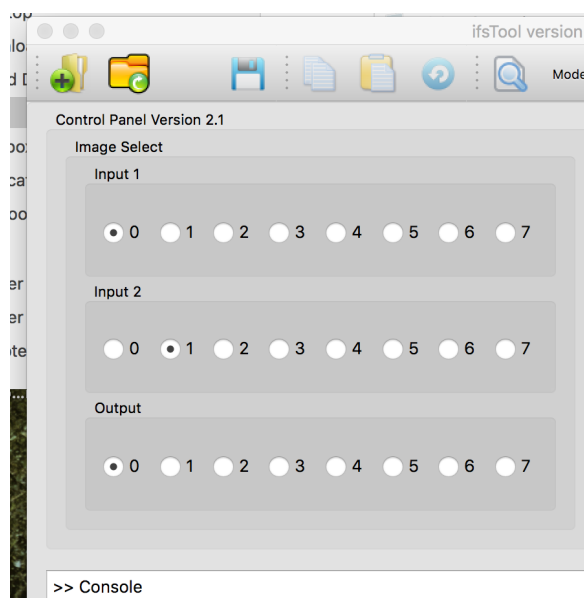


FIGURE 15. Unlike all the other functions in Wifstool, this function has no output! Just two inputs. It runs in place, so the first input is also the output.



FIGURE 16. Result of applying pcon. (You will probably need to zoom this image to see the differences clearly.)

during the process. Figure 16 illustrates the ability of pcon to sharpen an image while removing noise.

9. FREQUENCY DOMAIN OPERATIONS

All frequency domain operations are accessed via the COMPLEX pulldown.

fft: will compute the Fourier transform of **INPUT1**. Internally, this uses the complex two-dimensional fast Fourier transform. Images having dimension which are not a power of 2 will be padded. If the input is real, it will be copied into a complex image first. If the input is three-dimensional (a movie) or color, the fft is only applied to the first frame.

Note that the origin of the FFT will be at 0,0, the upper left of the image. Fourier transforms are often displayed with the origin in the center of the image, showing negative frequencies as positive. If the user is expecting this, the results may be surprising. In particular, instead of the DC term being in the center of the image, it will be at the upper left.

ifft: will compute the inverse Fourier transform of **INPUT1**. Internally, this uses the complex two-dimensional fast Fourier transform. Images having dimension which are not a power of 2 will be padded. The input image must be complex.

Complex Magnitude: Constructs a float image from a complex image by calculating the magnitude of the complex number.

Complex Phase: Constructs a float image from a complex image by calculating the phase of the complex number.

Real: Constructs a float image from a complex image by extracting the real part of the complex number.

Imaginary: Constructs a float image from a complex image by extracting the imaginary of the complex number.

10. COMMENTS

I am maintaining this by myself, so I cannot afford the time or effort to maintain it as if it were a commercial software produce. However, if you catch an error or a crash, please let me know. Please email me at wesley.e.snyder@me.com with as much information as you can provide. I'll try to fix it.

11. REFERENCES

REFERENCES

- [1] W.E. Snyder and H. Qi. *Fundamentals of Computer Vision*. Cambridge University Press, 2018.

INDEX

2D 2 3D, 10

Blevel, 6

bmp, 4

brightness-modify, 6

brightness-profile, 7

button-view, 6

Bwindow, 6

Bwindow&Blevel, 6

CCL, 12

char, 3

check box, 10

check box, color, 6

complex, 3

complex float, 4

complex images, 8

connected components, 12

contrast, 6

copy, 5

Data types, 3

default values, 12

derivative of brightness, 12

dimensions, 4

double, 3

drawOverlay, 11

DX operation, 11

enhancement, 6

float, 3, 4

floating point, 8

Formats,file, 3

frame number, 6

gif, 4

histogram, 8

Image buffers, 4

imaginary, 9

INPUT1, 4

INPUT2, 4

int, 3

jpg, 4

master images, 6

medical images, 7

mode, 9

movies, 6

Organization, 4

OUTPUT, 4

overlay, 6, 9

overlay-saving, 10

paste, 6

phase, 9

png, 4

profile, 7

Read, 5

ReRead, 5

RGB image, 10

SaveAs, 5

SaveWorkingImage, 5, 8

script, 3

short, 3

threshold for connectivity, 12

tif, 4

u8bit, 3

unary operations, 11

unsigned short, 3

view button, 6

viewing, 6

viewing image, 6

viewing window, 6

warmth, 6

Wifs, 4

zoom, 6