ECE763 Project 01

Due 2/25/2020

How to submit your solutions: put your reports (word or pdf) and results images (.png, if had) in a folder named *[your_unityid]_project01* (e.g., twu19_project01), and then compress it as a **zip** file (e.g., twu19_project01.zip). Submit the zip file through **moodle**.

Important Note: We will **NOT** accept any replacement of submission after deadline ([+late days you use]), even if you can show the time stamp of the replacement is earlier than the deadline. So, **please double-check whether you submit correct files**.

HW and Project Schedule:

- HW1: out 01/17, due 01/31, [DONE]
- HW2: out 02/05, due 02/19
- HW3: out 02/21, due 03/06
- HW4: out 03/09, due 03/23
- HW5: out 03/25, due 04/08
- Project1: out 02/04, due 02/25
- Project2: out 02/27, due 03/12
- Project3: out 03/13, due 04/30 (no late days allowed)

Objectives: Face image classification using Gaussian model, Mixture of Gaussian model, t-distribution, Mixture of t-distribution, Factor Analysis and Mixture of Factor Analyzer

Importance of working on Project 01 individually: Getting hands-on experience is important, which benefits you the most if you play with it individually and independently.

Data Preparation. Prepare training and testing data. Go to https://github.com/betars/Face-Resources and select one of the provided 17 face datasets which has face bounding boxes annotated. Download the dataset (note that some of the 17 datasets might need registration to download and you can skip those to save time). Extract $n = 1000$ training images for face and non-face respectively, and $m = 100$ testing images for face and non-face respectively, both at 20×20 resolution (resizing accordingly, you can also try 10×10 and 60×60 to see how robust and efficient your codes are). Make sure training face images and testing face images are separate, that is no face testing images are from the same person in the training set of face images. And, non-face images should be cropped randomly from background in the provided images in the dataset you selected.

Organized the extracted image patches Write a self-contained data i/o module in python or matlab or c/c++.

- Why doing this step? To get familiar with loading datasets, extracting image patches based on provided annotations. The datasets usually provide some toolkit for

manipulating image and annotation which you can re-use and modify. This is the first step in almost of computer vision problems.

- You can try to use more training and testing images at your will.
- You can also try to use smaller or bigger patches so you need less or more computing power and memory footprint, depending on your laptop or workstation.
- You might need to extract more images initially, then manually prune those to make your dataset less messy. E.g., containing relatively clean faces, frontal and profile, without different types of occlusions (e.g., big glasses, mask, etc).

A useful and powerful imaging and computer vision toolbox provided by Prof. Wesley Synder (see the instruction attached, and feel free to ask questions to wes@ncsu.edu).

Tasks: With your own face dataset created, you can train your models and test the performance. For each model, report results as follows.

- *Visualize the estimated mean(s) and covariance matrix for face and non-face respectively*; Use RGB images, but you are welcome to try on other things such as gray images, gray images with histogram equalized, and HSI color space, etc.
- *Evaluate the learned model on the testing images using 0.5 as threshold for the posterior. Compute false positive rate (#negatives being classified as faces / #total negatives), and false negative rate (#positives being classified as non-face / #total positives), and the misclassification rate ((#false-positives + #false-negative)/#total testing images)*
- *Plot the ROC curve where x-axis represents false positive rate and y-axis true positive rate (i.e, 1-false negative rate). To plot the curve, you change the threshold for posterior from $+\infty$ (use maximum posterior across testing images in practice, then all being classified as non-faces) to $-\infty$ (use minimum posterior across testing images in practice, then all being classified as faces) with for example1000 steps. (ref: https://en.wikipedia.org/wiki/Receiver_operating_characteristic)*

Required:

Model 1 (20 points). Learn single Gaussian model using training images and report your results as stated above.

Model 2 (40 points). Learn Mixture of Gaussian model using training images and report your results as stated above. You can tune the number of components (e.g., based on cross validation strategy).

Model 3 (20 points). Learn t-distribution model using training images and report your results as stated above.

Model 4 (20 points). Learn factor analyzer using training images and report your results as stated above. You can tune the dimensionality of the subspace.

Optional [10 bonus points per model]:

Model 5. *Learn Mixture of t-distribution model using training images and report your results as stated above. You can tune the number of components (e.g., based on cross validation strategy).*

Model 6. *Learn Mixture of t factor analyzer model using training images and report your results as stated above. You can tune the dimensionality of the subspace and the number of components (e.g., based on cross validation strategy).*

Hint: Think about how to modularize your codes in a nice way. For example, you will only need one ROCplot functions, and you can write a common EM algorithm. It is important to learn how to connect understanding of mathematical derivation to actual efficient implementation.

WIFSTOOL, DIGITAL SUPPORT FOR IMAGE ANALYSIS RESEARCH

WESLEY SNYDER

Wifstool is a GUI which supports the Image Analysis researcher. It supports images of any size[1], any number of dimensions[2], and a wide variety of formats.

To get the tool

(1) Go to http://www.cambridge.org

(2) On the bar across the top, under Academic, choose Books Catalog

(3) In the search window, type Snyder Qi Computer Vision

(4) The picture of the book will pop up. Click on the picture.

(5) A larger picture will pop up, and under it, a description with a button that says "Resources". Click on that.

(6) An extension window will pop up with Wifstool Image Analyzer shown under General Resources, click on that

(7) Three options appear. Click on the one you prefer to download the app for your computer. (The Mac version is the best).

(8) You may also want to click on Resources and download the BookImages.zip file which contains some interesting images to play with.

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]Until the process runs out of memory

[2]Also limited by memory capacity

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 3 unsigned characters per pixel

S4uchar: Structure consisting of 4 unsigned characters per pixel

S3ushort: Structure consisting of 3 16 bit unsigned integers per pixel

S4ushort: Structure consisting of 4 16 bit unsigned integers per pixel

S3float: Structure consisting of 3 32 bit floating point numbers per pixel

S4float: Structure consisting of 4 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

- 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 simultanously 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 **INPUT**1,

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.

FIGURE 2. File operations, left-to-right, are file read, file reread, saveas, copy, paste, rerun, view.

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.

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:

FIGURE 3. The ReRead button is emphasized

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 image[3] 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.

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. Color images with complex pixels are not supported.

When viewing, a new window 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 display buffer, and modifying them does not affect the master images, (those stored in buffers 0-7).

[3]SaveWorkingImage will be discussed in section 4.

Choice of Frame Number: 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.

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 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 display buffer, 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 display buffer, 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 6.1.

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).

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

The terms *window* and *level* 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 *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.

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 will be shown in red.

(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 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. 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.** If the image just read in and selected as **OUTPUT** is complex, the user should select it as **INPUT**1, and use the COMPLEX pulldown. Operations are listed below:

Magnitude: will compute the magnitude of the **INPUT**1 buffer and return it in the selected output buffer. This operation requires complex input.

Phase: will compute the phase of the **INPUT**1 buffer and return it in the selected output buffer. This operation requires complex input.

Real: will extract the real part of the **INPUT**1 buffer and return it in the selected output buffer. This operation requires complex input.

Imaginary: will extract the imaginary part of the **INPUT**1 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 change them to "mode 1". Changing modes changes the twenty functions available to the user to a different twenty.

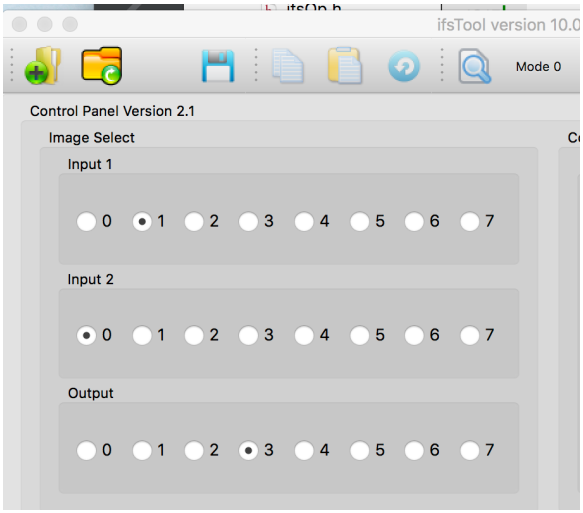

FIGURE 4. A possible organization of input and output buffer selection for the DX command: **INPUT**1 is the image in buffer 1, **OUTPUT** is buffer 3, and **INPUT**2 will be ignored. Many of the functions will prompt the user for the scale of the operation (σ of the Gaussian whose derivative is used as the kernel).

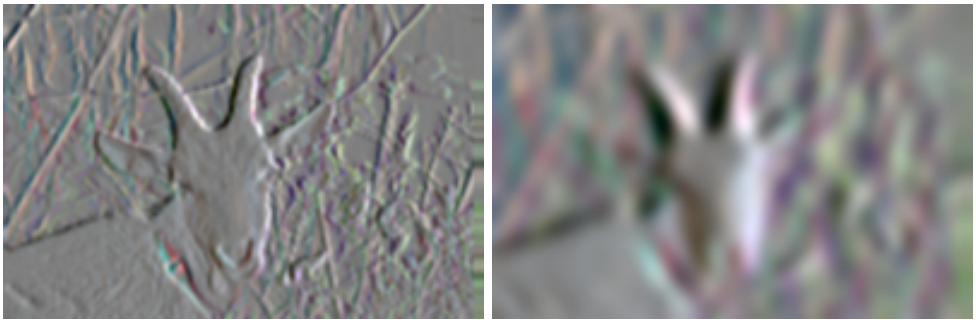

FIGURE 5. The output of the derivative with respect to x with a scale of 1 (LEFT) and 4 (RIGHT).

6. Unary (one input) Operations

The input image to all the unary operations is the one selected as **INPUT**1. The output will be the one selected as **OUTPUT**. For example, Figure 4 illustrates one choice of input and output for the DX operation (derivative with respect to x).

6.1. **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).

$\frac{\partial f}{\partial y}$: 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).

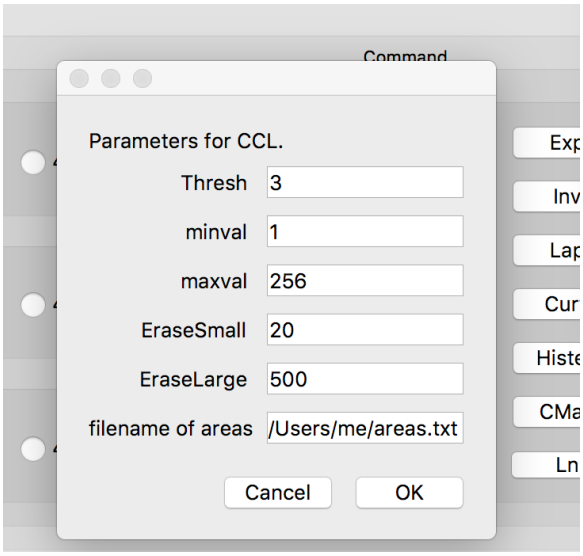

FIGURE 6. The parameters popup for the ccl (connected components labeling) command. Prior to a user entering values, the default values are shown. If no file name is given, the list of areas file will not be saved.

$\frac{\partial^2 f}{\partial y^2}$**:** 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. 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 follows:

- **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 6 illustrates filling in the connected components parameter box.

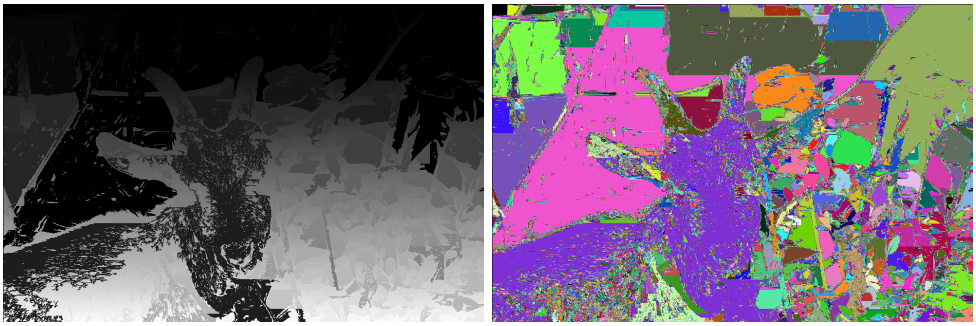

FIGURE 7. Illustrating the result of connected component labeling. The image on the left is the output of ccl. The pixel numbers are represented as brightnesses. The image on the right is the result of using random color assignment.

(This function is available only in release 10 of *Wifstool*.) Figure 7 illustrates the result of running connected components on the Vincent[4] 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.

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

Laplacian: Computes $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ at each point in the image.

Curvature: Computes the curvature at each point. The curvature of a function f is defined by $\kappa = \frac{f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2}{\left(f_x^2+f_y^2\right)^{3/2}}$. Curvature is heavily scale-dependent, Figure 8 illustrates two images illustrating the curvature of the same image, but this different scales (see page 245 in Snyder and Qi, 2018).

HistEq: Histogram equalization finds the image whose histogram is as close to flat as possible. Figure 9 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.

[4]Vincent Van Goat

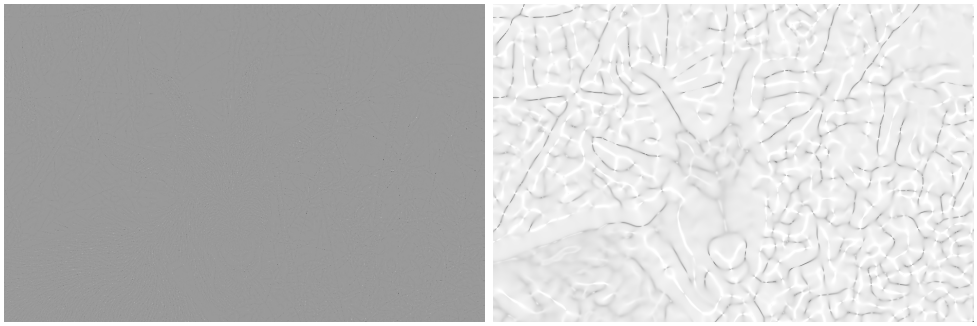

FIGURE 8. LEFT: curvature computed with a scale of 1.0, which matches only the noise in the image. RIGHT: curvature computed with a scale of 5.0, which is much closer to the scale of the featues in the image.

FIGURE 9. An overexposed image before and after histogram equalization.

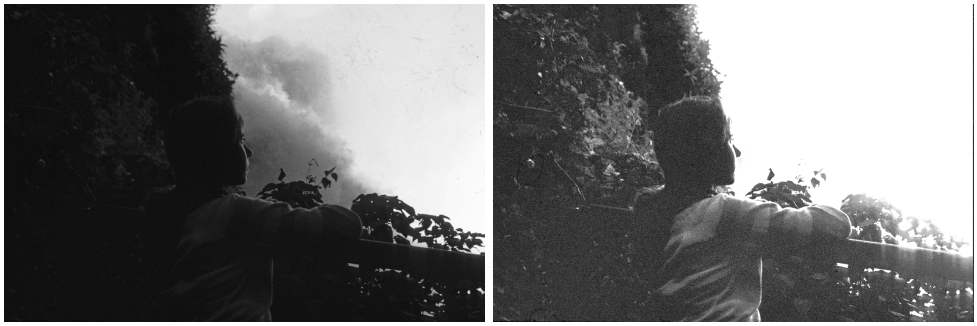

FIGURE 10. Histogram equalization can be used to show content in a very high contrast image.

Figure 10 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 **INPUT**1, **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
- 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.

6.2. 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.

Crop: Crop an image. Accomplished by displaying the image, and then 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).

7. 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 **INPUT**2 relative to the (0,0) point of **INPUT**1. Figure 11 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.

FIGURE 11. The smaller image was moved up until it was outside the larger image. Then the two were added.

correlate: The form of correlation used here is the two-dimensional normalized correlation[5]. 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

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

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.

[5]See Textbook, page 268

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 the for most images with brightnesses in the range 0-255, $\sigma = 3$ works well.

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

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, 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 during the process.

8. Frequency Domain Operations

All frequency domain operations are accessed via the COMPLEX pulldown.

fft: will compute the Fourier transform of **INPUT**1. 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 **INPUT**1. 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.

9. 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.

10. References

References

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