Image Processing Toolbox for Julia

Release 1.2 Beta 1

Michael A. Wirth
Denis Nikitenko

School of Computer Science, University of Guelph

Abstract

This note describes a toolbox of image processing algorithms for Julia.

Julia is a new language for scientific computing, and due to its similarity to MATLAB, is an excellent conduit for image processing. This toolbox contains functions to perform elementary image processing operations. The toolbox is completely free of dependencies, meaning it does not require any other packages to work.

The toolbox is currently under (re)construction to make it compatible with the latest version of Julia. It was originally written in Julia 0.6, and subsequent versions of Julia introduced some source-breaking changes. The current version has been updated for Julia 0.7/1.0, and has the tools you need for A1. It will be expanded to include the rest of the algorithms as we go along.

The toolbox contains the following packages:
Image sharpening: various unsharp masking filters
Noise suppression: a series of varied filters to perform noise suppression
Histogram functions: generate and manipulate histograms, eg. histogram equalization
Noise generation: Functions to generate noise in images
Metrics: Algorithms to calculate image enhancement metrics

All functions in this library return new/processed images, and do not modify their arguments.

To work with common image formats (e.g. PNG or JPEG), you can use additional Julia packages:

- `FileIO` allows you to load those images. The resulting format is a matrix of 3 (RGB) or 4 (RGB and Alpha channel) values (https://github.com/JuliaIO/FileIO.jl). To convert these images to individual R, G, and B matrices, use the `channelview()` function.

  Also, keep in mind that the values in the matrix returned by `load()` are normalized, while the functions in our Julia library expect matrices with values between 0 and 255. You will need to scale them by 255, and explicitly convert the result to an integer type (Julia is strictly typed). The who process would look like this:

  ```
  img = load("image.png")
  mat = channelview(img)
  r=round.(Int16, 255*mat[1,:,:])
  g=round.(Int16, 255*mat[2,:,:])
  b=round.(Int16, 255*mat[3,:,:])
  ```

- You can use the `ImageView` library to display the loaded images, and their R, G, and B matrices (https://github.com/JuliaImages/ImageView.jl)

- You can use the `Plots` library for plotting histograms (https://github.com/JuliaPlots/Plots.jl)

Algorithms

Histogram Functions

Generate image histogram	**`getIMhist()`**	gray	imageHIST
Calculate cumulative histogram	**`cumhst()`**	gray	imageHIST
Histogram equalization	**`histEQ()`**	gray	imageHIST
Histogram hyperbolization	**`histHYPER()`**	gray	imageHIST
Bi-histogram equalization	**`bihistEQ()`**	gray	imageHIST
Generate a circular filter	**`roundFilter()`**	gray	imageENHADAPT
Adaptive histogram equalization	**`histeqADAPT()`**	gray	imageENHADAPT
Adaptive histogram equalization using circular filters	**`histeqADAPTcirc()`**	gray	imageENHADAPT

Basic Filters

Image convolution	**`filter_CONV`**		imageFILTER

Image Sharpening Filters

Traditional unsharp masking	**`filter_sharpUSM`**		imageFILTER
UM with Order Statistic Laplacian	**`filter_sharpUMOSLap`**		imageFILTER
UM with Laplacian of Gaussian	**`filter_sharpUSMLofG`**		imageFILTER
UM with Gaussian smoothing	**`filter_sharpUSMgauss`**		imageFILTER
Cubic UM	**`filter_sharpCUSM`**		imageFILTER

Noise Suppression (smoothing)

Gaussian smoothing	**`filter_GAUSSIAN`**		imageFILTER
Median filtering	**`filter_MEDIAN`**		imageFILTER
Truncated median filter	**`enh_truncMedian`**		imageENH
Mean (averaging) filter	**`filter_MEAN`**		imageENH
Hybrid median filter	**`enh_hybridMedian`**		imageENH
Alpha-Trimmed Means filter	**`enh_alphaTMean`**		imageENH
Weighted-median filter	**`filter_wMEDIAN`**		imageENH
Kuwahara filter	**`Kuwahara()`**		imageENH
Nagao Matsuyama filter	**`NagaoMatsuyama()`**		imageENH

Image Noise Generation Functions

Impulse noise	**`impulse()`**	gray	imageNoise
Gaussian noise	**`gaussian()`**	gray	imageNoise
Raleigh noise	**`raleigh()`**	gray	imageNoise
Negative exponential noise (speckle)	**`speckle()`**	gray	imageNoise
Gamma noise	**`gamma()`**	gray	imageNoise
Uniform noise	**`uniform()`**	gray	imageNoise

Image Enhancement Metric Algorithms

Rank’s Noise Estimation Index	**`RankNEI()`**	gray	imageMETRICS
Noise Amplification index	**`noiseAI()`**	gray	imageMETRICS
Perceptual blur metric	**`perblurMetric()`**	gray	imageMETRICS