Convolution

0.0.1

Generated by Doxygen 1.8.11

Contents

Chapter 1

Deprecated List

Global relativeEqualFloat (float a, float b, float maxRelDiff)
in favor fo almostEqualFloat due to limitations around 0

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 CONVOptions Struct Reference

Processing parameters for identifying mesh size and dataset.

`#include <conv.h>`

Data Fields

- int **gNI**
- int **gNJ**
- int pNI
- int **pNJ**
- int nIter
- int restartIter
- int verbose
- int doIO
- char dataset [1024]
- char outdir [1024]
- char prefix [64]

4.1.1 Detailed Description

Processing parameters for identifying mesh size and dataset.

4.1.2 Field Documentation

4.1.2.1 char CONVOptions::dataset[1024]

Whether I/O used

4.1.2.2 int CONVOptions::doIO

Whether verbose output used

4.1.2.3 int CONVOptions::nIter

Size of processor mesh

4.1.2.4 char CONVOptions::outdir[1024]

Source dataset path

4.1.2.5 int CONVOptions::pNI

Size of global mesh

4.1.2.6 char CONVOptions::prefix[64]

Output directory path

4.1.2.7 int CONVOptions::restartIter

Number of iterations

4.1.2.8 int CONVOptions::verbose

Used to determine when to create restart files

The documentation for this struct was generated from the following file:

- include/conv.h

4.2 CONVPatchDesc Struct Reference

Descriptor for managing processor mesh, data mesh and raster information.

```
#include <conv.h>
```

Data Fields

- MPI_Comm **comm**
- int patchI
- int **patchJ**
- int pNI
- int **pNJ**
- int left
- int **right**
- int **up**
- int **down**
- int ul
- int **ur**
- int **ll**
- int **lr**
- int gNI
- int **gNJ**
- int gI
- int **gJ**
- int lni
- int **lnj**
- double gAffine [6]
- float nodata
- char proj_str [1024]
- int nBands
- GDALDatasetH hDS

4.2.1 Detailed Description

Descriptor for managing processor mesh, data mesh and raster information.

4.2.2 Field Documentation

4.2.2.1 double CONVPatchDesc::gAffine[6]

Size of the local patch

4.2.2.2 int CONVPatchDesc::gI

Full size of the data mesh

4.2.2.3 int CONVPatchDesc::gNI

Upper left, upper right, lower left, lower right (needed for on option of the 9 point stencil

4.2.2.4 GDALDatasetH CONVPatchDesc::hDS

Number of bands within raster

4.2.2.5 int CONVPatchDesc::left

Size of processor mesh

4.2.2.6 int CONVPatchDesc::lni

Global I, J for the upper left of the local patch

4.2.2.7 int CONVPatchDesc::nBands

Proj.4 Projection string

4.2.2.8 float CONVPatchDesc::nodata

Affine transformation parameters for dataset

4.2.2.9 int CONVPatchDesc::patchI

Communicator for processor mesh

4.2.2.10 int CONVPatchDesc::pNI

(I,J) location of patch in processor mesh

4.2.2.11 char CONVPatchDesc::proj_str[1024]

No Data value to be used for null cells

4.2.2.12 int CONVPatchDesc::ul

Neighbors to this patch: left = (I, J-1), up=(I-1,J), etc. (standard matrix ordering)

The documentation for this struct was generated from the following file:

- include/conv.h

4.3 CONVTiming Struct Reference

Time tracker for profiling communication exchanges.

`#include <conv.h>`

Data Fields

- double **packtime**
- double **unpacktime**
- double exchtime
- double itertime

4.3.1 Detailed Description

Time tracker for profiling communication exchanges.

4.3.2 Field Documentation

4.3.2.1 double CONVTiming::exchtime

Time to pack/unpack data if separate

4.3.2.2 double CONVTiming::itertime

Time for exchange

The documentation for this struct was generated from the following file:

- include/conv.h

4.4 CRS Struct Reference

Container class for CRS data Useful for translating between proj and GeoTIFF, or LAS formats.

```
#include <CRS.h>
```

Data Fields

- ST_TIFF * **m_tiff**
- GTIF * m_gtiff
- char * proj

4.4.1 Detailed Description

Container class for CRS data Useful for translating between proj and GeoTIFF, or LAS formats.

4.4.2 Field Documentation

4.4.2.1 GTIF* CRS::m_gtiff

Tiff descriptor object

4.4.2.2 char* CRS::proj

GeoTiff descriptor object

The documentation for this struct was generated from the following file:

- include/CRS.h

4.5 Grid Struct Reference

Container struct for 2-dimensional array data.

`#include <Grid.h>`

Data Fields

- int **cols**
- int rows
- double nodata
- float * data

4.5.1 Detailed Description

Container struct for 2-dimensional array data.

4.5.2 Field Documentation

4.5.2.1 float* Grid::data

No data value

4.5.2.2 double Grid::nodata

Number of rows

4.5.2.3 int Grid::rows

Number of columns

The documentation for this struct was generated from the following file:

- include/Grid.h

4.6 Kernel Struct Reference

Data Fields

- int **cols**
- int **rows**
- float **coef**
- float * **data**

The documentation for this struct was generated from the following file:

- include/Kernel.h

4.7 mem_win Struct Reference

memory window for Asynchronous Remote Memory Access

Data Fields

- void * **mem**
- MPI_Win win
- int above_LRows
- int left_LCols
- int right_LCols

4.7.1 Detailed Description

memory window for Asynchronous Remote Memory Access

4.7.2 Field Documentation

4.7.2.1 int mem_win::above_LRows

MPI Window object used to open and close access

4.7.2.2 int mem_win::left_LCols

Number of ghost rows above(North) patch

4.7.2.3 int mem_win::right_LCols

Number of ghost columns left of patch

4.7.2.4 MPI_Win mem_win::win

Pointer to memory window

The documentation for this struct was generated from the following file:

- src/conv-fence.c

4.8 mGrid Struct Reference

Container struct for 2-dimensional multivariate array data.

`#include <Grid.h>`

Data Fields

- int **id**
- int ndims
- int * dims
- int * dimids
- int nvars
- int * varids

4.8.1 Detailed Description

Container struct for 2-dimensional multivariate array data.

Warning

Currently not functional, will be updated to allow for NetCDF IO.

4.8.2 Field Documentation

4.8.2.1 int* mGrid::dimids

Pointer to dimension definitions

4.8.2.2 int* mGrid::dims

Number of dimensions

4.8.2.3 int mGrid::ndims

Identifier

4.8.2.4 int mGrid::nvars

Pointer to dimension identifiers

4.8.2.5 int* mGrid::varids

Number of variables

The documentation for this struct was generated from the following file:

- include/Grid.h

4.9 Raster Struct Reference

This class contains basic metadata needed to georeference and project 2D arrays for raster datasets.

`#include <Raster.h>`

Data Fields

- Grid **grid**
- double affine [6]
- char proj4 [1024]

4.9.1 Detailed Description

This class contains basic metadata needed to georeference and project 2D arrays for raster datasets.

Raster class

4.9.2 Field Documentation

4.9.2.1 double Raster::affine[6]

Grid instance to hold pixel data

4.9.2.2 char Raster::proj4[1024]

Affine matrix to handle transformations

The documentation for this struct was generated from the following file:

- include/Raster.h

Chapter 5

File Documentation

5.1 include/CRS.h File Reference

```
#include <geotiff.h>
#include <geo_simpletags.h>
#include <geo_normalize.h>
#include <geovalues.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "GeoKey.h"
#include "VLR.h"
```

Data Structures

- struct CRS

  Container class for CRS data Useful for translating between proj and GeoTIFF, or LAS formats.

Typedefs

- typedef struct CRS CRS

  Container class for CRS data Useful for translating between proj and GeoTIFF, or LAS formats.

Functions

- void **CRS_init** (CRS *crs)
- int **CRS_read** (sGeoKeys *geo, CRS *crs)
- int **CRS_getProj4** (CRS *crs, char *proj4)
- int **CRS_getGTIF** (CRS *crs, VLR *vlr, unsigned long n_vlr)
- void **CRS_free** (CRS *crs)

5.1.1 Detailed Description

Coordinate Reference System Header File

5.2 include/Grid.h File Reference

```
#include <stdlib.h>
#include <netcdf.h>
#include "Kernel.h"
```

Data Structures

- struct Grid

  Container struct for 2-dimensional array data.
- struct mGrid

  Container struct for 2-dimensional multivariate array data.

Functions

- void Grid_init (Grid *grid)

  Initialize a Grid instance.
- int Grid_copy (Grid *g1, Grid *g2, int full)

  Copy constructor for grid instances, shallow or full.
- int Grid_alloc (Grid *grid)

  Allocate pixels for Grid.
- void Grid_free (Grid *grid)

  Free memory allocated for Grid instance.
- float Grid_get (int col, int row, const Grid *grid)

  Retrieve a pixel value from Grid.
- int Grid_set (int col, int row, Grid *grid, float val)

  Set the value of a Grid pixel.
- int Grid_convolve (const Grid *in_grid, const Kernel *kern, Grid *out_grid)

  Convolve a kernel across a grid.
- int Grid_subtract (const Grid *in_grid_1, const Grid *in_grid_2, Grid *out_grid)

  Perform map arithmetic to subtract one Grid from another.
- int Grid_square (const Grid *in_grid, Grid *out_grid)

  Calculate the square of a grid Raises all values to an exponent of 2.
- int Grid_zero_cross (const Grid *in_grid, Grid *out_grid)

  Create binary grid of zero-crossings.
- int Grid_IDW (const Grid *in_grid, int n_neighbors, float distance, int power, Grid *out_grid)

  Perform inverse distance weighting interpolation on grid.
- int Grid_read (const char *filePath, Grid *grid)

  Read a Grid from a GDAL-compliant file.
- int Grid_downsample (const Grid *in_grid, Grid *out_grid)

  Downsample a grid to 2x pixel size.
- float Grid_mean (const Grid *grid)

  Calculate the mean cell value for the Grid.
- float Grid_var (const Grid *grid, const float mean)

  Calculate the cell value variance for a Grid.
- float Grid_stdev (const float var)

  Calculate the standard deviation for a Grid.
- float Grid_median (const Grid *grid, int i, int j, int radius)

Calculate the neighborhood median value for a given pixel.

- float Grid_Lee (const Grid *grid, int i, int j, int radius, float g_var)

  Lee filter (Adaptive mean)
- float Grid_mean_win (const Grid *grid, int i, int j, int radius)

  Neighborhood mean value.
- float Grid_var_win (const Grid *grid, int i, int j, int radius, float mean)

  Window(neighborhood) variance.
- int Grid_laplace (const Grid *in_grid, Grid *out_grid)

  Calculate Laplacian Derivative from Grid.
- float Grid_erode (const Grid *in_grid, int i, int j, Kernel *k)

  Greyscale Morphologic erosion.
- float Grid_dilate (const Grid *in_grid, int i, int j, Kernel *k)

  Greyscale morphologic dilation.
- float calc_ridge (float x, float y, float xx, float xy, float yy)

  Calculate ridge strength factor.
- float ridge (float xx, float xy, float yy)

  Calculate Ridge Strength factor.
- float sdgd (float x, float y, float xx, float xy, float yy)

  Second derivative in the gradient direction.

5.2.1 Detailed Description

Author

Nathan Casler

Date

17 October 2017

5.2.2 Function Documentation

5.2.2.1 float calc_ridge (float *x*, float *y*, float *xx*, float *xy*, float *yy*)

Calculate ridge strength factor.

Warning

Deprecated Function is currently deprecated in favor of ridge() function

Parameters

x	First order derivative in x direction
y	First order derivative in y direction
xx	Second-order derivative in x direction
xy	Second-order diagonal derivative
yy	Second-order derivative in y direction

See also

https://dsp.stackexchange.com/questions/1714/best-way-of-segmenting-veins-in-leaves

$$\_\frac{y^2x'' + 2x'y'xy'' - x^2y''}{y^3}$$

Returns

Ridge strength (float32)

5.2.2.2 int Grid_alloc (Grid * *grid*)

Allocate pixels for Grid.

If the grid instance is a non-null pointer, the grid instance will be freed before it is allocated.

Parameters

grid	Grid defining pixel dimensions

Returns

1 on success, else -1

5.2.2.3 int Grid_convolve (const Grid * *in_grid*, const Kernel * *kern*, Grid * *out_grid*)

Convolve a kernel across a grid.

Parameters

in_grid	Source Grid instance
kern	Kernel instance
out_grid	Destination Grid instance

Returns

0

5.2.2.4 int Grid_copy (Grid * *g1*, Grid * *g2*, int *full*)

Copy constructor for grid instances, shallow or full.

Parameters

g1	Source Grid instance
g2	Destination Grid instance
full	If non-zero, do full copy of pixel data, else leave pixels uninitialized

Returns

0

5.2.2.5 float Grid_dilate (const Grid * *in_grid*, int *i*, int *j*, Kernel * *k*)

Greyscale morphologic dilation.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
k	Structuring Kernel

Returns

Maximum value masked by structuring kernel

5.2.2.6 int Grid_downsample (const Grid * *in_grid*, Grid * *out_grid*)

Downsample a grid to 2x pixel size.

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Warning

This methodology may suffer from edge effects

Returns

0

5.2.2.7 float Grid_erode (const Grid * *in_grid*, int *i*, int *j*, Kernel * *k*)

Greyscale Morphologic erosion.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
k	Structuring kernel

Returns

Minimum value within structure kernel

5.2.2.8 void Grid_free (Grid * *grid*)

Free memory allocated for Grid instance.

Parameters

Source	Grid instance

Returns

void

5.2.2.9 float Grid_get (int *col*, int *row*, const Grid * *grid*)

Retrieve a pixel value from Grid.

Note

This function uses mirroring on borders so out-of-bound indexes are reflected to their internal index values.

Parameters

col	Column index for pixel
row	Row index for pixel
grid	Source Grid instance

Returns

Pixel data value (float32)

5.2.2.10 int Grid_IDW (const Grid * *in_grid*, int *n_neighbors*, float *distance*, int *power*, Grid * *out_grid*)

Perform inverse distance weighting interpolation on grid.

Note

Currently the distance threshold is not utilized

Warning

Deprecated. Current implementation has errors and is replaced by null_fill

Parameters

in_grid	Source Grid instance
n_neighbors	Desired number of neighbors
distance	Maximum distance threshold
power	Weighting coefficient
out_grid	Destination Grid instance

Returns

0

5.2.2.11 void Grid_init (Grid * *grid*)

Initialize a Grid instance.

Parameters

grid	Destination Grid instance

Returns

void

5.2.2.12 int Grid_laplace (const Grid * *in_grid*, Grid * *out_grid*)

Calculate Laplacian Derivative from Grid.

See also

https://en.wikipedia.org/wiki/Laplace_operator

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Warning

May experience edge effects since Grid_get mirrors edges

Returns

1

5.2.2.13 float Grid_Lee (const Grid * *in_grid*, int *i*, int *j*, int *radius*, float *g_var*)

Lee filter (Adaptive mean)

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Neighborhood search radius
g_var	Global variance measure

Returns

Weighted mean value

5.2.2.14 float Grid_mean (const Grid * *in_grid*)

Calculate the mean cell value for the Grid.

Parameters

in_grid	Source Grid instance

Returns

Mean value

5.2.2.15 float Grid_mean_win (const Grid * *in_grid*, int *i*, int *j*, int *radius*)

Neighborhood mean value.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Search neighborhood radius

Returns

Neighborhood mean value

5.2.2.16 float Grid_median (const Grid * *in_grid*, int *i*, int *j*, int *radius*)

Calculate the neighborhood median value for a given pixel.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Search neighborhood radius

Returns

Neighborhood median value

5.2.2.17 int Grid_read (const char * *filePath*, Grid * *grid*)

Read a Grid from a GDAL-compliant file.

Parameters

filePath	File path to GDAL-compliant dataset
grid	Destination Grid instance

Returns

0 on success, -1 on IOError

5.2.2.18 int Grid_set (int *col*, int *row*, Grid * *grid*, float *val*)

Set the value of a Grid pixel.

Parameters

col	Column index of pixel
row	Row index of pixel
grid	Destination Grid instance
val	Specified pixel value

Note

This function will return -1 if column or row indexes are out of bounds.

Returns

1 on succes, else -1

5.2.2.19 int Grid_square (const Grid * *in_grid*, Grid * *out_grid*)

Calculate the square of a grid Raises all values to an exponent of 2.

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Returns

1

5.2.2.20 float Grid_stdev (const float *var*)

Calculate the standard deviation for a Grid.

Warning

Deprecated, literally just applies square root to input

Parameters

var	Variance for Grid

Returns

Standard deviation

5.2.2.21 int Grid_subtract (const Grid * *in_grid_1*, const Grid * *in_grid_2*, Grid * *out_grid*)

Perform map arithmetic to subtract one Grid from another.

Parameters

in_grid_1	Source Grid to be subtracted from
in_grid_2	Source Grid to subtract
out_grid	Destination Grid

Returns

0

5.2.2.22 float Grid_var (const Grid * *in_grid*, const float *mean*)

Calculate the cell value variance for a Grid.

Parameters

in_grid	Source Grid instance
mean	Mean pixel value

Returns

Variance

5.2.2.23 float Grid_var_win (const Grid * *in_grid*, int *i*, int *j*, int *radius*, float *mean*)

Window(neighborhood) variance.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Neighborhood search radius
mean	Neighborhood mean value

Returns

Neighborhood variance value

5.2.2.24 int Grid_zero_cross (const Grid * *in_grid*, Grid * *out_grid*)

Create binary grid of zero-crossings.

A zero-crossing occurs in a neighborhood where both positive and negative values are found. This can be used as an edge-detector on Laplacian derivatives.

See also

https://en.wikipedia.org/wiki/Zero_crossing

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Note

currently uses 3x3 neighborhood for calculations

Returns

1

5.2.2.25 float ridge (float *xx*, float *xy*, float *yy*)

Calculate Ridge Strength factor.

Parameters

xx	Second-order x derivative
xy	Second-order diagonal derivative
yy	Second-order y derivative

Returns

Ridge strength factor

5.2.2.26 float sdgd (float *x*, float *y*, float *xx*, float *xy*, float *yy*)

Second derivative in the gradient direction.

Parameters

x	First-order x derivative
y	First-order y derivative
xx	Second-order x derivative
xy	Second-order diagonal derivative
yy	Second-order y derivative

Returns

Derivative in direction of gradient

5.3 src/Activation.c File Reference

```
#include <stdio.h>
#include <math.h>
#include <stdint.h>
#include "Util.h"
```

Functions

- double CONVA_ident (double x)

  Identify function, returns self.

- int CONVA_step (double x)

  Binary step function.
- double CONVA_log (double x)

  Soft step (logarithmic) function.
- double CONVA_tanh (double x)

  Hyperbolic Tangent activation function.
- double CONVA_atan (double x)

  Arc tangent activation function.

5.3.1 Detailed Description

Laundry list of available activation functions for neural networks

See also

https://en.wikipedia.org/wiki/Activation_function

5.3.2 Function Documentation

5.3.2.1 double CONVA_atan (double *x*)

Arc tangent activation function.

Parameters

x	input value

Returns

$arctan(x)$

5.3.2.2 double CONVA_ident (double *x*)

Identify function, returns self.

Parameters

x	input value

Returns

x

5.3.2.3 double CONVA_log (double *x*)

Soft step (logarithmic) function.

Parameters

x	input value

Returns

$\frac{1}{1+exp(-x)}$

5.3.2.4 int CONVA_step (double *x*)

Binary step function.

Parameters

x	input value

Returns

1 if positive, else 0

5.3.2.5 double CONVA_tanh (double *x*)

Hyperbolic Tangent activation function.

Parameters

x	input value

Returns

$tanh(x)$

5.4 src/conv-fence.c File Reference

```
#include <mpi.h>
#include "conv.h"
```

Data Structures

- struct mem_win

  memory window for Asynchronous Remote Memory Access

Typedefs

- typedef struct mem_win mem_win

  memory window for Asynchronous Remote Memory Access

Functions

- int CONV_ExchangeInitFence (CONVPatchDesc *patch, float **m1, float **m2, void *privateData)

  Initialize Window and ghost arrays for one-sided communication This function passes ghost arrays using point-to-point communication which will suffer froma peformance penalty since each send is reliant on a matching receive. As the number of processes grows, this will likely cause a deadlock.
- int CONV_ExchangeEndFence (void *privateData)

  Free the windows allocated for this communication.
- int CONV_ExchangeFence (CONVPatchDesc *patch, float **matrix, CONVTiming *timedata, void *privateData)

  Initialize Window and ghost arrays for one-sided communication This function passes ghost arrays to neighbors using a 2-step trick to avoid deadlock and diagonal communication. The left and right arrays are passed, a fence is called to finish communication, then the top and bottom arrays are sent.

5.4.1 Detailed Description

List of workflow specific steps. These will often include communication and iteration

5.4.2 Function Documentation

5.4.2.1 int CONV_ExchangeEndFence (void * *privateData*)

Free the windows allocated for this communication.

Parameters

privateData	pointer to window memory

Returns

0

5.4.2.2 int CONV_ExchangeFence (CONVPatchDesc * *patch*, float ** *matrix*, CONVTiming * *timedata*, void * *privateData*)

Initialize Window and ghost arrays for one-sided communication This function passes ghost arrays to neighbors using a 2-step trick to avoid deadlock and diagonal communication. The left and right arrays are passed, a fence is called to finish communication, then the top and bottom arrays are sent.

Parameters

patch	Local Patch instance
matrix	Source 2-D array
timedata	Timing tracker
privateData	pointer to memory

Returns

MPI_SUCCESS or MPI_ERR

5.4.2.3 int CONV_ExchangeInitFence (CONVPatchDesc * *patch*, float ** *m1*, float ** *m2*, void * *privateData*)

Initialize Window and ghost arrays for one-sided communication This function passes ghost arrays using point-to-point communication which will suffer froma peformance penalty since each send is reliant on a matching receive. As the number of processes grows, this will likely cause a deadlock.

Parameters

patch	Local Patch instance
m1	First 2-D array
m2	Second 2-D array
privateData	Pointer to memory

Returns

MPI_SUCCESS if sucess else MPI_ERR

5.5 src/coptions.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <mpi.h>
#include "conv.h"
#include "conv-io.h"
```

Functions

- int CONV_ParseArgs (int argc, char **argv, CONVOptions *options)

  Parse command line arguments.
- void CONV_Abort (const char str[])

  Abort MPI processes gracefully.

5.5.1 Detailed Description

Command line argument parsing functions

Note

Command line arguments are not guaranteed in the MPI environment to be passed to all processes. To be portable, we must process on rank 0 and distribute results

5.5.2 Function Documentation

5.5.2.1 void CONV_Abort (const char *str[ ]*)

Abort MPI processes gracefully.

Parameters

str	Error message

Returns

void

5.5.2.2 int CONV_ParseArgs (int *argc*, char ** *argv*, CONVOptions * *options*)

Parse command line arguments.

Parameters

argc	Argument count
argv	Array of arguments
options	Mesh creation option object

Returns

0

5.6 src/ctiming.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <math.h>
#include <mpi.h>
#include "conv.h"
#include "conv-io.h"
#include "Util.h"
```

Functions

- int CONV_TimeIterations (CONVPatchDesc *patch, int nIter, int doCheckpoint, float **m1, float **m2, int(*exchangeInit)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

  Exchange values between patches.
- int CONV_TimeExchange (CONVPatchDesc *patch, float **m1, float **m2, int(*exchangeInit)(CONV↩PatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

  Time the exchagne between patches.
- int CONV_TimePreProcess (CONVPatchDesc *patch, int nIter, int doCheckpoint, float **m1, float **m2, int(*exchangeInit)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

Apply Lee Filter and exchange.

- int CONV_TimeFillNull (CONVPatchDesc *patch, int nIter, int doCheckpoint, float **m1, float **m2, int(*exchangeInit)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

  Fill null values using focal mean with a 3x3 window.

- int CONV_TimeProcess (CONVPatchDesc *patch, int nIter, float **m1, float **m2, int(*exchangeInit)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

  Apply Gaussian smoothing to raster.

- int CONV_TimeClassify (CONVPatchDesc *patch, float **m1, float **m2, int(*exchangeInit)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

  Calculate implicit curvature of patch data.

5.6.1 Detailed Description

Communication dependent functions for mesh

5.6.2 Function Documentation

5.6.2.1 int CONV_TimeClassify (CONVPatchDesc * *patch*, float ** *m1*, float ** *m2*, int(*)(CONVPatchDesc *, float **, float **, void *) *exchangeInit*, int(*)(CONVPatchDesc *, float **, CONVTiming *, void *) *exchange*, int(*)(void *) *exchangeEnd*, CONVTiming * *timedata*)

Calculate implicit curvature of patch data.

Parameters

patch	Local Patch instance
m1	First 2-D array
m2	Second 2-D array
exchangeInit	Communication initialization function
exchange	Communication function
exchangeEnd	Communication finalization function

Returns

0

5.6.2.2 int CONV_TimeExchange (CONVPatchDesc * *patch*, float ** *m1*, float ** *m2*, int(*)(CONVPatchDesc *, float **, float **, void *) *exchangeInit*, int(*)(CONVPatchDesc *, float **, CONVTiming *, void *) *exchange*, int(*)(void *) *exchangeEnd*, CONVTiming * *timedata*)

Time the exchagne between patches.

Parameters

patch	Local Patch instance

Parameters

m1	First 2-D array
m2	Second 2-D array
exchangeInit	Communication initialization function
exchange	Communication function
exchangeEnd	Communication finalization function
timedata	Time tracker

Returns

0

5.6.2.3 int CONV_TimeFillNull (CONVPatchDesc * *patch*, int *nIter*, int *doCheckpoint*, float ** *m1*, float ** *m2*, int(*)(CONVPatchDesc *, float **, float **, void *) *exchangeInit*, int(*)(CONVPatchDesc *, float **, CONVTiming *, void *) *exchange*, int(*)(void *) *exchangeEnd*, CONVTiming * *timedata*)

Fill null values using focal mean with a 3x3 window.

Parameters

patch	Local Patch instance
doCheckpoint	Write output?
m1	First 2-D Array
m2	Second 2-D Array
exchangeInit	Communication initialization function
exchange	Communication function
exchangeEnd	Communication finalization function
timedata	Time tracker

Returns

0

5.6.2.4 int CONV_TimeIterations (CONVPatchDesc * *patch*, int *nIter*, int *doCheckpoint*, float ** *m1*, float ** *m2*, int(*)(CONVPatchDesc *, float **, float **, void *) *exchangeInit*, int(*)(CONVPatchDesc *, float **, CONVTiming *, void *) *exchange*, int(*)(void *) *exchangeEnd*, CONVTiming * *timedata*)

Exchange values between patches.

Parameters

patch	Local Patch Instance
nIter	Number of iterations
doCheckpoint	Whether to perform a checkpoint after exchange
m1	First 2-D array
m2	Second 2-D array

Parameters

exchangeInit	Exchange initialization function
exchange	HaloExchange function
exchangeEnd	Exchange finalization function
timedata	Timing tracker

Returns

0

5.6.2.5 int CONV_TimePreProcess (CONVPatchDesc * *patch,* int *nIter,* int *doCheckpoint,* float ** *m1,* float ** *m2,* int(*)(CONVPatchDesc *, float **, float **, void *) *exchangeInit,* int(*)(CONVPatchDesc *, float **, CONVTiming *, void *) *exchange,* int(*)(void *) *exchangeEnd,* CONVTiming * *timedata*)

Apply Lee Filter and exchange.

Parameters

patch	Local Patch instance
nIter	number of iterations
doCheckpoint	Write output?
m1	First 2-D array
m2	Second 2-D array
exchangeInit	Communication initialization function
exchange	Communication function
exchangeEnd	Communication finalization function
timedata	Time tracker

Warning

doCheckpoint and nIter are useless currently

Returns

0

5.6.2.6 int CONV_TimeProcess (CONVPatchDesc * *patch,* int *nIter,* float ** *m1,* float ** *m2,* int(*)(CONVPatchDesc *, float **, float **, void *) *exchangeInit,* int(*)(CONVPatchDesc *, float **, CONVTiming *, void *) *exchange,* int(*)(void *) *exchangeEnd,* CONVTiming * *timedata*)

Apply Gaussian smoothing to raster.

Parameters

patch	Local Patch instance
nIter	Number of iterations

Parameters

m1	First 2-D array
m2	Second 2-D array
exchangeInit	Communication initialization function
exchange	Communication function
exchangeEnd	Communication finalization function
timedata	Time tracker

Returns

0

5.7 src/Grid.c File Reference

File containing function definitions for pixel arrays.

```
#include <stdlib.h>
#include <stdio.h>
#include "Grid.h"
#include "Kernel.h"
#include "Util.h"
#include <math.h>
#include <gdal.h>
#include <string.h>
```

Functions

- void Grid_init (Grid *grid)
  Initialize a Grid instance.
- int Grid_alloc (Grid *grid)
  Allocate pixels for Grid.
- void Grid_free (Grid *grid)
  Free memory allocated for Grid instance.
- int Grid_copy (Grid *g1, Grid *g2, int full)
  Copy constructor for grid instances, shallow or full.
- float Grid_get (int col, int row, const Grid *grid)
  Retrieve a pixel value from Grid.
- int Grid_set (int col, int row, Grid *grid, float val)
  Set the value of a Grid pixel.
- int Grid_zero_cross (const Grid *in_grid, Grid *out_grid)
  Create binary grid of zero-crossings.
- int Grid_laplace (const Grid *in_grid, Grid *out_grid)
  Calculate Laplacian Derivative from Grid.
- float calc_ridge (float x, float y, float xx, float xy, float yy)
  Calculate ridge strength factor.
- float ridge (float xx, float xy, float yy)
  Calculate Ridge Strength factor.

- int Grid_convolve (const Grid *in_grid, const Kernel *kern, Grid *out_grid)
  Convolve a kernel across a grid.
- int Grid_IDW (const Grid *in_grid, int n_neighbors, float distance, int power, Grid *out_grid)
  Perform inverse distance weighting interpolation on grid.
- int Grid_subtract (const Grid *in_grid_1, const Grid *in_grid_2, Grid *out_grid)
  Perform map arithmetic to subtract one Grid from another.
- float Grid_mean (const Grid *in_grid)
  Calculate the mean cell value for the Grid.
- float Grid_var (const Grid *in_grid, const float mean)
  Calculate the cell value variance for a Grid.
- float Grid_stdev (const float var)
  Calculate the standard deviation for a Grid.
- int Grid_square (const Grid *in_grid, Grid *out_grid)
  Calculate the square of a grid Raises all values to an exponent of 2.
- int Grid_read (const char *filePath, Grid *grid)
  Read a Grid from a GDAL-compliant file.
- int Grid_downsample (const Grid *in_grid, Grid *out_grid)
  Downsample a grid to 2x pixel size.
- float Grid_median (const Grid *in_grid, int i, int j, int radius)
  Calculate the neighborhood median value for a given pixel.
- float Grid_erode (const Grid *in_grid, int i, int j, Kernel *k)
  Greyscale Morphologic erosion.
- float Grid_dilate (const Grid *in_grid, int i, int j, Kernel *k)
  Greyscale morphologic dilation.
- float Grid_Lee (const Grid *in_grid, int i, int j, int radius, float g_var)
  Lee filter (Adaptive mean)
- float Grid_mean_win (const Grid *in_grid, int i, int j, int radius)
  Neighborhood mean value.
- float Grid_var_win (const Grid *in_grid, int i, int j, int radius, float mean)
  Window(neighborhood) variance.
- float sdgd (float x, float y, float xx, float xy, float yy)
  Second derivative in the gradient direction.

5.7.1 Detailed Description

File containing function definitions for pixel arrays.

Author

Nathan Casler

Date

15 October 2017

5.7.2 Function Documentation

5.7.2.1 float calc_ridge (float *x*, float *y*, float *xx*, float *xy*, float *yy*)

Calculate ridge strength factor.

Warning

Deprecated Function is currently deprecated in favor of ridge() function

Parameters

x	First order derivative in x direction
y	First order derivative in y direction
xx	Second-order derivative in x direction
xy	Second-order diagonal derivative
yy	Second-order derivative in y direction

See also

https://dsp.stackexchange.com/questions/1714/best-way-of-segmenting-veins-in-leaves

$$-\frac{y^2x'' + 2x'y'xy'' - x^2y''}{y^3}$$

Returns

Ridge strength (float32)

5.7.2.2 int Grid_alloc (Grid * *grid*)

Allocate pixels for Grid.

If the grid instance is a non-null pointer, the grid instance will be freed before it is allocated.

Parameters

grid	Grid defining pixel dimensions

Returns

1 on success, else -1

5.7.2.3 int Grid_convolve (const Grid * *in_grid*, const Kernel * *kern*, Grid * *out_grid*)

Convolve a kernel across a grid.

Parameters

in_grid	Source Grid instance
kern	Kernel instance
out_grid	Destination Grid instance

Returns

0

5.7.2.4 int Grid_copy (Grid * g1, Grid * g2, int full)

Copy constructor for grid instances, shallow or full.

Parameters

g1	Source Grid instance
g2	Destination Grid instance
full	If non-zero, do full copy of pixel data, else leave pixels uninitialized

Returns

0

5.7.2.5 float Grid_dilate (const Grid * in_grid, int i, int j, Kernel * k)

Greyscale morphologic dilation.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
k	Structuring Kernel

Returns

Maximum value masked by structuring kernel

5.7.2.6 int Grid_downsample (const Grid * in_grid, Grid * out_grid)

Downsample a grid to 2x pixel size.

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Warning

This methodology may suffer from edge effects

Returns

0

5.7.2.7 float Grid_erode (const Grid * *in_grid*, int *i*, int *j*, Kernel * *k*)

Greyscale Morphologic erosion.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
k	Structuring kernel

Returns

Minimum value within structure kernel

5.7.2.8 void Grid_free (Grid * *grid*)

Free memory allocated for Grid instance.

Parameters

Source	Grid instance

Returns

void

5.7.2.9 float Grid_get (int *col*, int *row*, const Grid * *grid*)

Retrieve a pixel value from Grid.

Note

This function uses mirroring on borders so out-of-bound indexes are reflected to their internal index values.

Parameters

col	Column index for pixel
row	Row index for pixel
grid	Source Grid instance

Returns

Pixel data value (float32)

5.7.2.10 int Grid_IDW (const Grid * *in_grid*, int *n_neighbors*, float *distance*, int *power*, Grid * *out_grid*)

Perform inverse distance weighting interpolation on grid.

Note

Currently the distance threshold is not utilized

Warning

Deprecated. Current implementation has errors and is replaced by null_fill

Parameters

in_grid	Source Grid instance
n_neighbors	Desired number of neighbors
distance	Maximum distance threshold
power	Weighting coefficient
out_grid	Destination Grid instance

Returns

0

5.7.2.11 void Grid_init (Grid * *grid*)

Initialize a Grid instance.

Parameters

grid	Destination Grid instance

Returns

void

5.7.2.12 int Grid_laplace (const Grid * *in_grid*, Grid * *out_grid*)

Calculate Laplacian Derivative from Grid.

See also

https://en.wikipedia.org/wiki/Laplace_operator

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Warning

May experience edge effects since Grid_get mirrors edges

Returns

1

5.7.2.13 float Grid_Lee (const Grid * *in_grid*, int *i*, int *j*, int *radius*, float *g_var*)

Lee filter (Adaptive mean)

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Neighborhood search radius
g_var	Global variance measure

Returns

Weighted mean value

5.7.2.14 float Grid_mean (const Grid * *in_grid*)

Calculate the mean cell value for the Grid.

Parameters

in_grid	Source Grid instance

Returns

Mean value

5.7.2.15 float Grid_mean_win (const Grid * *in_grid*, int *i*, int *j*, int *radius*)

Neighborhood mean value.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Search neighborhood radius

Returns

Neighborhood mean value

5.7.2.16 float Grid_median (const Grid * *in_grid*, int *i*, int *j*, int *radius*)

Calculate the neighborhood median value for a given pixel.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Search neighborhood radius

Returns

Neighborhood median value

5.7.2.17 int Grid_read (const char * *filePath*, Grid * *grid*)

Read a Grid from a GDAL-compliant file.

Parameters

filePath	File path to GDAL-compliant dataset
grid	Destination Grid instance

Returns

0 on success, -1 on IOError

5.7.2.18 int Grid_set (int *col*, int *row*, Grid * *grid*, float *val*)

Set the value of a Grid pixel.

Parameters

col	Column index of pixel
row	Row index of pixel
grid	Destination Grid instance
val	Specified pixel value

Note

This function will return -1 if column or row indexes are out of bounds.

Returns

1 on succes, else -1

5.7.2.19 int Grid_square (const Grid * *in_grid*, Grid * *out_grid*)

Calculate the square of a grid Raises all values to an exponent of 2.

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Returns

1

5.7.2.20 float Grid_stdev (const float *var*)

Calculate the standard deviation for a Grid.

Warning

Deprecated, literally just applies square root to input

Parameters

var	Variance for Grid

Returns

Standard deviation

5.7.2.21 int Grid_subtract (const Grid * *in_grid_1*, const Grid * *in_grid_2*, Grid * *out_grid*)

Perform map arithmetic to subtract one Grid from another.

Parameters

in_grid_1	Source Grid to be subtracted from
in_grid_2	Source Grid to subtract
out_grid	Destination Grid

Returns

0

5.7.2.22 float Grid_var (const Grid * *in_grid*, const float *mean*)

Calculate the cell value variance for a Grid.

Parameters

in_grid	Source Grid instance
mean	Mean pixel value

Returns

Variance

5.7.2.23 float Grid_var_win (const Grid * *in_grid*, int *i*, int *j*, int *radius*, float *mean*)

Window(neighborhood) variance.

Parameters

in_grid	Source Grid instance
i	Column index
j	Row index
radius	Neighborhood search radius
mean	Neighborhood mean value

Returns

Neighborhood variance value

5.7.2.24 int Grid_zero_cross (const Grid * *in_grid*, Grid * *out_grid*)

Create binary grid of zero-crossings.

A zero-crossing occurs in a neighborhood where both positive and negative values are found. This can be used as an edge-detector on Laplacian derivatives.

See also

https://en.wikipedia.org/wiki/Zero_crossing

Parameters

in_grid	Source Grid instance
out_grid	Destination Grid instance

Note

currently uses 3x3 neighborhood for calculations

Returns

1

5.7.2.25 float ridge (float *xx*, float *xy*, float *yy*)

Calculate Ridge Strength factor.

Parameters

xx	Second-order x derivative
xy	Second-order diagonal derivative
yy	Second-order y derivative

Returns

Ridge strength factor

5.7.2.26 float sdgd (float *x*, float *y*, float *xx*, float *xy*, float *yy*)

Second derivative in the gradient direction.

Parameters

x	First-order x derivative
y	First-order y derivative
xx	Second-order x derivative
xy	Second-order diagonal derivative
yy	Second-order y derivative

Returns

Derivative in direction of gradient

5.8 src/Kernel.c File Reference

Functions needed for kernel convolution and window functions.

```
#include <stdio.h>
#include <stdlib.h>
#include "Kernel.h"
#include <string.h>
#include <math.h>
#include "Util.h"
```

Functions

- void Kernel_init (Kernel *kernel)
  Initialize a Kernel instance.
- int Kernel_sobel (int axis, Kernel *kern)
  Create a 3x3 Sobel Kernel.
- int PixelsNeededForSigma (float sigma)
  Calculate the diameter necessary to represent a gaussian.
- int Kernel_gauss (float sigma, Kernel *kern)
  Generate gaussian kernel.
- int Kernel_mean (int radius, Kernel *kern)
  Generate a mean kernel estimator.
- int Kernel_alloc (Kernel *kern)
  Allocate pixels for Kernel instance.
- void Kernel_free (Kernel *kern)
  Free memory allocated to Kernel.
- float Kernel_get (int col, int row, const Kernel *kern)
  Get pixel value of kernel.
- int Kernel_set (int col, int row, Kernel *kern, float val)
  Set pixel value of kernel col Column index.

5.8.1 Detailed Description

Functions needed for kernel convolution and window functions.

Author

Nathan Casler

Date

16 October 2017

5.8.2 Function Documentation

5.8.2.1 int Kernel_alloc (Kernel * *kern*)

Allocate pixels for Kernel instance.

Parameters

kern	Destination Kernel instance

Returns

1 if success, else -1

5.8.2.2 void Kernel_free (Kernel * *kern*)

Free memory allocated to Kernel.

Parameters

kern	Kernel instance to free

Returns

void

5.8.2.3 int Kernel_gauss (float *sigma*, Kernel * *kern*)

Generate gaussian kernel.

Parameters

sigma	Sigma defining the gaussian
kern	Destination kernel instance

Returns

1

5.8.2.4 float Kernel_get (int *col*, int *row*, const Kernel * *kern*)

Get pixel value of kernel.

Parameters

col	Column index
row	Row index
kern	Source Kernel instance

Returns

If success pixel value, else 0

5.8.2.5 void Kernel_init (Kernel * *kernel*)

Initialize a Kernel instance.

Parameters

kernel	Targer Kernel instance

Returns

void

5.8.2.6 int Kernel_mean (int *radius*, Kernel * *kern*)

Generate a mean kernel estimator.

Warning

Deprecated. Use Grid_mean_win

Parameters

radius	Desired kernel radius Destination Kernel instance

Returns

1

5.8.2.7 int Kernel_set (int *col*, int *row*, Kernel * *kern*, float *val*)

Set pixel value of kernel col Column index.

Parameters

row	Row index
kern	Kernel Instance
val	Desired pixel value

Returns

if success 1, else -1

5.8.2.8 int Kernel_sobel (int *axis*, Kernel * *kern*)

Create a 3x3 Sobel Kernel.

Parameters

axis	X if 0, else Y
kern	Destination Kernel instance

Returns

0

5.8.2.9 int PixelsNeededForSigma (float *sigma*)

Calculate the diameter necessary to represent a gaussian.

Parameters

sigma	Designated sigma for gaussian distribution

Returns

Representative diameter for a gaussian

See also

http://blog.demofox.org/2015/08/19/gaussian-blur

5.9 src/patch.c File Reference

basic patch configuration

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <mpi.h>
#include <stdint.h>
#include "gdal.h"
#include "cpl_conv.h"
#include "ogr_srs_api.h"
#include "Util.h"
#include "conv.h"
```

Functions

- int CONV_PatchCreateProcessMesh (CONVOptions *options, CONVPatchDesc *patch)

  Create a patch mesh of processors.
- int CONV_PatchCreateProcessMeshWithCart (CONVOptions *options, CONVPatchDesc *patch)

  Create cartesian patch processor mesh.
- int CONV_PatchCreateDataMeshDesc (CONVOptions *options, CONVPatchDesc *patch)

  Create data decomposition for patch.
- int CONV_ParseDatasetHeader (CONVPatchDesc *patch, const char *dset)

  Read Raster metadata from GDAL-compliant raster.
- int CONV_AllocateLocalMesh (CONVPatchDesc *patch, float ***m1, float ***m2)

  Allocate memory for Patch's pixel arrays.
- int CONV_AllocateKernel (float ***k, int radius)

  Allocate a kernel array.
- int CONV_FreeLocalMesh (CONVPatchDesc *patch, float **m1, float **m2)

  Free memory allocated for local arrays.
- int CONV_FreeKernel (float **kernel)

  Free memory allocated for kernel array.
- int CONV_InitLocalMesh (CONVPatchDesc *patch, float **m1, float **m2)

  Initialize local mesh Zero initialize arrays set nodata value, nodata initialize halo. Read raster data into m1 array.
- int CONV_cleanPatch (CONVPatchDesc *patch, float **matrix)

  Set outside valid elevation range to nodata Useful inbetween iterations to remove erroneous values.
- float CONV_NN (CONVPatchDesc *patch, float **matrix, int row, int col)

  Nearest neighbor null-filling This function uses a 3x3 neighborhood to fill nulls with a focal mean. Can be applied iteratively to fill voids.
- float CONV_Laplace (CONVPatchDesc *patch, float **matrix, int row, int col)

  Calculate Laplacian of patch pixel.
- float CONV_mixedDerivative (CONVPatchDesc *patch, float **matrix, int row, int col)

  Calculate the mixed derivative f_{xy} of a given patch pixel.
- float CONV_Curvature (CONVPatchDesc *patch, float **matrix, int row, int col)

  Calculate the curvature value for a given patch pixel.
- float CONV_Convolve (CONVPatchDesc *patch, float **mat, float **weights, int radius, int row, int col)

  Convolve a 2-d weight array with a patch pixel.
- float CONV_IDW (CONVPatchDesc *patch, float **matrix, int row, int col, int radius, int power)

  Fill nulls in neighborhood using Inverse Distance Weighting Formula uses Shephards inverse Distance Weighting formula.
- int CONV_WriteLocalMesh (CONVOptions *opts, CONVPatchDesc *patch, float **mat, int iter)

  Write local patch data to a GeoTiff dataset.
- float CONV_PatchMean (CONVPatchDesc *patch, float **mat)

  Calculate the mean for a patch's array data.
- float CONV_PatchVariance (CONVPatchDesc *patch, float **mat, const float mean)

  Calculate variance of patch data.
- float CONV_Lee (CONVPatchDesc *patch, float **mat, int i, int j, int radius, float g_var)

  Apply Lee filter to patch The Lee filter is designed to eliminate speckle noise while preserving edges and point features in radar imagery. Based on a linear speckle noise model and the minimum mean square error (MMSE) design approach, the filter produces the enhanced data according to

  $$\widehat{I}_s = \overline{I}_s + k_s(I_s - \overline{I}_s)$$

  .
- void CONV_FreePatch (CONVPatchDesc *patch)

  Free memory allocated to a Patch instance.

5.9.1 Detailed Description

basic patch configuration

Author

Nathan Casler

Date

17 October 2017 Heavily sampled from advanced MPI tutorials provided by Bill Gropp at ATPESC

See also

will

5.9.2 Function Documentation

5.9.2.1 int CONV_AllocateKernel (float *** *k*, int *radius*)

Allocate a kernel array.

Parameters

k	Pointer for array
radius	Kernel radius

Returns

0 if success

5.9.2.2 int CONV_AllocateLocalMesh (CONVPatchDesc * *patch*, float *** *m1*, float *** *m2*)

Allocate memory for Patch's pixel arrays.

Parameters

Destination	Patch instance
m1	First array (used as source array in processing)
m2	Second array (used as destination in processing) Allocate a C-style 2-D array (array of pointers). Allocate a local mesh with ghost cells and as a contiguous block so that stridec access may be used for left/right edges.

For simplicity all patches have halo cells on all sides, even if the process shares a physical boundary.

Returns

0

5.9.2.3 int CONV_cleanPatch (CONVPatchDesc * *patch*, float ** *matrix*)

Set outside valid elevation range to nodata Useful inbetween iterations to remove erroneous values.

Parameters

patch	Target patch instance
matrix	Array to clean

Returns

number of pixels changed

5.9.2.4 float CONV_Convolve (CONVPatchDesc * *patch*, float ** *mat*, float ** *weights*, int *radius*, int *row*, int *col*)

Convolve a 2-d weight array with a patch pixel.

Parameters

patch	Source patch instance
mat	Source 2-d array
weights	2-d Weight array
radius	Convolution Kernel radius
row	Row index
col	Col index

Returns

Convolved value

5.9.2.5 float CONV_Curvature (CONVPatchDesc * *patch*, float ** *matrix*, int *row*, int *col*)

Calculate the curvature value for a given patch pixel.

Based on the formula for planar implicit curves

$$\kappa = -\frac{f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_x^2f_{yy}}{(f_x^2 + f_y^2)^{3/2}}$$

Parameters

patch	Source Patch instance
matrix	Source 2-D array
row	Row index
col	Column index

Returns

Curvature value

5.9.2.6 int CONV_FreeKernel (float ** *kernel*)

Free memory allocated for kernel array.

Parameters

kernel	Kernel to free

Returns

0

5.9.2.7 int CONV_FreeLocalMesh (CONVPatchDesc * *patch*, float ** *m1*, float ** *m2*)

Free memory allocated for local arrays.

Parameters

path	Local patch instance
m1	First array
m2	Second array

Returns

0

5.9.2.8 void CONV_FreePatch (CONVPatchDesc * *patch*)

Free memory allocated to a Patch instance.

Parameters

patch	Patch instance

Returns

void

5.9.2.9 float CONV_IDW (CONVPatchDesc * *patch*, float ** *matrix*, int *row*, int *col*, int *radius*, int *power*)

Fill nulls in neighborhood using Inverse Distance Weighting Formula uses Shephards inverse Distance Weighting formula.

Note

Implementation currently has errors.

Parameters

patch	Source patch instance
matrix	Source 2-D array
row	Row index
col	Column index
radius	Neighborhood search radius
power	Weighting exponent coefficient

Returns

Weighted value

5.9.2.10 int CONV_InitLocalMesh (CONVPatchDesc * *patch*, float ** *m1*, float ** *m2*)

Initialize local mesh Zero initialize arrays set nodata value, nodata initialize halo. Read raster data into m1 array.

Parameters

patch	Destination patch instance
m1	First array
m2	Second array

Returns

0

5.9.2.11 float CONV_Laplace (CONVPatchDesc * *patch*, float ** *matrix*, int *row*, int *col*)

Calculate Laplacian of patch pixel.

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Parameters

patch	Source Patch instance
matrix	Source 2-D array
row	Row index
col	Column index

Returns

Laplacian value

5.9.2.12 float CONV_Lee (CONVPatchDesc * *patch*, float ** *mat*, int *i*, int *j*, int *radius*, float *g_var*)

Apply Lee filter to patch The Lee filter is designed to eliminate speckle noise while preserving edges and point features in radar imagery. Based on a linear speckle noise model and the minimum mean square error (MMSE) design approach, the filter produces the enhanced data according to

$$\widehat{I}_s = \overline{I}_s + k_s(I_s - \overline{I}_s)$$

.

Where: $\overline{I}_s$ is the local mean value within window η_s and k_s is the adaptive filter coefficient.

$$k_s = 1 - \frac{C_u^2}{C_s^2}$$

And

$$C_s^2 = \frac{1}{|\eta_s|} \sum_{p \in \eta} \frac{(I_p - \overline{I}_s)^2}{(I_p - \overline{I}_s)^2}$$

and C_u^2 is a constant for a given imange and can be determined by

$$C_u^2 = \frac{var(z')}{(\overline{z}')^2}$$

Where $var(z')$ is the intensity variance and $\overline{z}'$is the mean over a homogenous area of the image.

See also

http://www.cs.virginia.edu/~lgs9a/rodinia/heartwall/srad/paper_2.pdf

Parameters

patch	Target patch instance
mat	Source 2-d Array to read from
i	Column index
j	Row index
radius	Neighborhood search radius
g_var	Global variance measure for patch

Returns

Weighted neighborhood mean

5.9.2.13 float CONV_mixedDerivative (CONVPatchDesc * *patch*, float ** *matrix*, int *row*, int *col*)

Calculate the mixed derivative f_{xy} of a given patch pixel.

Parameters

patch	Source Patch instance
matrix	Source 2-D array
row	Row index
col	Column index

Returns

if success Mixed derivative value, else 0

5.9.2.14 float CONV_NN (CONVPatchDesc * *patch*, float ** *matrix*, int *row*, int *col*)

Nearest neighbor null-filling This function uses a 3x3 neighborhood to fill nulls with a focal mean. Can be applied iteratively to fill voids.

Parameters

patch	target Patch instance
matrix	Source 2-d array
row	Row index
col	Column index

Returns

Average of non-null neighbors if found, else nodata

5.9.2.15 int CONV_ParseDatasetHeader (CONVPatchDesc * *patch*, const char * *dset*)

Read Raster metadata from GDAL-compliant raster.

Parameters

patch	Destination Patch instance dset File path to GDAL-compliant raster

Returns

0 if success, else -1

5.9.2.16 int CONV_PatchCreateDataMeshDesc (CONVOptions * *options*, CONVPatchDesc * *patch*)

Create data decomposition for patch.

Parameters

options	Mesh creation options
patch	Destination Patch instance

Returns

0

5.9.2.17 int CONV_PatchCreateProcessMesh (CONVOptions * *options*, CONVPatchDesc * *patch*)

Create a patch mesh of processors.

Parameters

options	Options for mesh
path	Target patch definition

Returns

0

5.9.2.18 int CONV_PatchCreateProcessMeshWithCart (CONVOptions * *options*, CONVPatchDesc * *patch*)

Create cartesian patch processor mesh.

Parameters

options	Options for mesh creation
patch	Destination patch definition

Returns

0

5.9.2.19 float CONV_PatchMean (CONVPatchDesc * *patch*, float ** *mat*)

Calculate the mean for a patch's array data.

Parameters

patch	Target Patch instance
mat	Source 2-D array

Returns

Mean of array data

5.9.2.20 float CONV_PatchVariance (CONVPatchDesc * *patch*, float ** *mat*, const float *mean*)

Calculate variance of patch data.

Parameters

patch	Target patch instance
mat	Source 2-D array to read from
mean	Mean for source array

Returns

Patch variance

5.9.2.21 int CONV_WriteLocalMesh (CONVOptions * *opts*, CONVPatchDesc * *patch*, float ** *mat*, int *iter*)

Write local patch data to a GeoTiff dataset.

Parameters

opts	Mesh creation options
patch	Source patch instance
mat	Source 2-D array to write
iter	Iteration count for use in output dataset name

Returns

0

5.10 src/Raster.c File Reference

File containing function definitions for Raster class.

```
#include <stdlib.h>
#include <math.h>
#include "Grid.h"
#include "Raster.h"
#include "gdal.h"
#include "cpl_string.h"
#include "cpl_conv.h"
#include "ogr_srs_api.h"
#include <string.h>
```

Functions

- void Raster_init (Raster *raster)

  Initialize a new Raster instance.
- int Raster_stat (char *in_file, Raster *raster)

  Read raster metadata into Raster instance, but not pixel data.
- int Raster_read (char *in_file, Raster *raster)

  Read GDAL compliant raster data into a Raster instance.
- int Raster_copy (Raster *r1, Raster *r2, int full)

  Copy constructor for Raster instances, may be full or shallow.
- int Raster_write (char *out_file, Raster *raster)

  Write a raster instance out to GeoTiff file.
- int Raster_downsample (const Raster *r1, Raster *r2)

  Downsample a raster instance so the resolution in x and y are 1/2.
- void Raster_free (Raster *raster)

  De-initialize and free memory allocated for a raster instance.

5.10.1 Detailed Description

File containing function definitions for Raster class.

Author

Nathan Casler

Date

15 October 2017

5.10.2 Function Documentation

5.10.2.1 int Raster_copy (Raster * *r1*, Raster * *r2*, int *full*)

Copy constructor for Raster instances, may be full or shallow.

Copy contructor for Raster instances, may be full or shallow copy.

Parameters

r1	Source Raster instance
r2	Destination Raster instance
full	Copy pixel data if true, else leave empty

Returns

0

5.10.2.2 int Raster_downsample (const Raster * r1, Raster * r2)

Downsample a raster instance so the resolution in x and y are 1/2.

Downsample a Raster instance to 2x the pixel size in x and y dimensions. Downsampling will use a naive average to downsample the pixels, which can cause border effects around edges of image.

Downsampling will use a basic average to downsample the pixels, which can cause border effects around edges of image.

Parameters

r1	Source raster instance
r2	Destination raster instance

Returns

1 if successed, else 0

5.10.2.3 void Raster_free (Raster * raster)

De-initialize and free memory allocated for a raster instance.

De-initialize and free allocated memory from a Raster instance.

Parameters

raster	Raster instance to deallocate

Returns

void

5.10.2.4 void Raster_init (Raster * raster)

Initialize a new Raster instance.

void Raster_init

Parameters

raster	The Raster instance to initialize

Returns

void

5.10.2.5 int Raster_read (char * *in_file*, Raster * *raster*)

Read GDAL compliant raster data into a Raster instance.

Read GDAL-compliant raster data into a Raster instance.

Parameters

in_file	File path to GDAL-compliant raster
raster	Destination Raster instance

Returns

1 if success else -1

5.10.2.6 int Raster_stat (char * *in_file*, Raster * *raster*)

Read raster metadata into Raster instance, but not pixel data.

Read raster metadata into Raster instance, excludes pixel data.

Parameters

in_file	File path to GDAL compliant dataset
raster	Raster object to store metadata

Returns

1 if success else -1

5.10.2.7 int Raster_write (char * *out_file*, Raster * *raster*)

Write a raster instance out to GeoTiff file.

Write a Raster instance out to GeoTiff.

Parameters

out_file	Destination file path
raster	Source raster instance

Returns

0

5.11 src/Util.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <assert.h>
#include <math.h>
#include "Util.h"
#include <stdint.h>
```

Functions

- int almostEqualFloat (float a, float b, int maxUlps)

  Check for near-equality in floating-point numbers This function uses integer comparison to check distance between floats should allow us to determine if the values are close to the nodata value.
- int compareFloat (const void ∗a, const void ∗b)

  Naive floating-point comparison check for use in qsort.
- int compareInt (const void ∗a, const void ∗b)

  Integer comparison check for use in qsort.
- uint32_t nibb_swap32 (uint32_t a)

  Nibble-level endianness swap.
- uint32_t byte_swap32 (uint32_t a)

  Byte-level endianness swap.
- int relativeEqualFloat (float a, float b, float maxRelDiff)

  Relative equality comparison.
- float gauss (int x, int y, float sigma)

  Calculate the gaussian value of a given pixel.
- double absolute (double x)

  Absolute value of double-precision float.

5.11.1 Detailed Description

Utility functions for library

Author

Nathan Casler

Date

17 October 2017

5.11.2 Function Documentation

5.11.2.1 double absolute (double *x*)

Absolute value of double-precision float.

Parameters

x	Input double

Returns

$abs(x)$

5.11.2.2 int almostEqualFloat (float *a*, float *b*, int *maxUlps*)

Check for near-equality in floating-point numbers This function uses integer comparison to check distance between floats should allow us to determine if the values are close to the nodata value.

See also

http://www.cygnus-software.com/papers/comparingfloats/Comparing%20floating%20point%2html

Parameters

maxUlps	the maximum error in terms of Units in the Last Place

Returns

1 if near-equal, else 0

5.11.2.3 uint32_t byte_swap32 (uint32_t *a*)

Byte-level endianness swap.

Note

May be useful if data is saved on machine with different endianness.

Parameters

a	Source integer to swap

Returns

Unsigned 32-bit integer swapped at every 8bits

5.11.2.4 int compareFloat (const void * *a*, const void * *b*)

Naive floating-point comparison check for use in qsort.

Parameters

a	Pointer to first value
b	Pointer to second value

Returns

1 if a $>$ b, 0 if equal, -1 if a $<$ b

5.11.2.5 int compareInt (const void * *a*, const void * *b*)

Integer comparison check for use in qsort.

Parameters

a	pointer to first integer
b	pointer to second integer return 1 if a $>$ b else 0

5.11.2.6 float gauss (int *x*, int *y*, float *sigma*)

Calculate the gaussian value of a given pixel.

Parameters

x	Column index
y	Row index
sigma	Sigma coefficient defining gaussian function

Returns

value of gaussian at (x, y)

5.11.2.7 uint32_t nibb_swap32 (uint32_t *a*)

Nibble-level endianness swap.

Note

Probably few actually applications unless someone changed order of data at the nibble(4bit) level.

Parameters

a	Unsigned 32 bit integer to swap

Returns

Unsigned 32 bit integer where all nibbles have been swapped

5.11.2.8 int relativeEqualFloat (float *a*, float *b*, float *maxRelDiff*)

Relative equality comparison.

Deprecated in favor fo almostEqualFloat due to limitations around 0

Parameters

a	first float
b	second float
maxRelDiff	Equality threshold

Returns

1 if the difference is less than threshold, else 0

Index