

Convolution

0.0.1

Generated by Doxygen 1.8.11

Contents

- 1 Deprecated List** **1**

- 2 Data Structure Index** **3**
 - 2.1 Data Structures 3

- 3 File Index** **5**
 - 3.1 File List 5

- 4 Data Structure Documentation** **7**
 - 4.1 CONVOptions Struct Reference 7
 - 4.1.1 Detailed Description 7
 - 4.1.2 Field Documentation 7
 - 4.1.2.1 dataset 7
 - 4.1.2.2 doIO 8
 - 4.1.2.3 nIter 8
 - 4.1.2.4 outdir 8
 - 4.1.2.5 pNI 8
 - 4.1.2.6 prefix 8
 - 4.1.2.7 restartIter 8
 - 4.1.2.8 verbose 8
 - 4.2 CONVPatchDesc Struct Reference 8
 - 4.2.1 Detailed Description 9
 - 4.2.2 Field Documentation 9
 - 4.2.2.1 gAffine 9
 - 4.2.2.2 gl 9

4.2.2.3	gNI	9
4.2.2.4	hDS	9
4.2.2.5	left	10
4.2.2.6	Ini	10
4.2.2.7	nBands	10
4.2.2.8	nodata	10
4.2.2.9	patchI	10
4.2.2.10	pNI	10
4.2.2.11	proj_str	10
4.2.2.12	ul	10
4.3	CONVTiming Struct Reference	10
4.3.1	Detailed Description	11
4.3.2	Field Documentation	11
4.3.2.1	exctime	11
4.3.2.2	itertime	11
4.4	CRS Struct Reference	11
4.4.1	Detailed Description	11
4.4.2	Field Documentation	12
4.4.2.1	m_gtiff	12
4.4.2.2	proj	12
4.5	Grid Struct Reference	12
4.5.1	Detailed Description	12
4.5.2	Field Documentation	12
4.5.2.1	data	12
4.5.2.2	nodata	12
4.5.2.3	rows	13
4.6	Kernel Struct Reference	13
4.7	mem_win Struct Reference	13
4.7.1	Detailed Description	13
4.7.2	Field Documentation	13

4.7.2.1	above_LRows	13
4.7.2.2	left_LCols	14
4.7.2.3	right_LCols	14
4.7.2.4	win	14
4.8	mGrid Struct Reference	14
4.8.1	Detailed Description	14
4.8.2	Field Documentation	14
4.8.2.1	dimids	14
4.8.2.2	dims	15
4.8.2.3	ndims	15
4.8.2.4	nvars	15
4.8.2.5	varids	15
4.9	Raster Struct Reference	15
4.9.1	Detailed Description	15
4.9.2	Field Documentation	15
4.9.2.1	affine	15
4.9.2.2	proj4	15
5	File Documentation	17
5.1	include/CRS.h File Reference	17
5.1.1	Detailed Description	17
5.2	include/Grid.h File Reference	18
5.2.1	Detailed Description	19
5.2.2	Function Documentation	19
5.2.2.1	calc_ridge(float x, float y, float xx, float xy, float yy)	19
5.2.2.2	Grid_alloc(Grid *grid)	20
5.2.2.3	Grid_convolve(const Grid *in_grid, const Kernel *kern, Grid *out_grid)	20
5.2.2.4	Grid_copy(Grid *g1, Grid *g2, int full)	20
5.2.2.5	Grid_dilate(const Grid *in_grid, int i, int j, Kernel *k)	21
5.2.2.6	Grid_downsample(const Grid *in_grid, Grid *out_grid)	21
5.2.2.7	Grid_erode(const Grid *in_grid, int i, int j, Kernel *k)	21

5.2.2.8	<code>Grid_free(Grid *grid)</code>	22
5.2.2.9	<code>Grid_get(int col, int row, const Grid *grid)</code>	22
5.2.2.10	<code>Grid_IDW(const Grid *in_grid, int n_neighbors, float distance, int power, Grid *out_grid)</code>	22
5.2.2.11	<code>Grid_init(Grid *grid)</code>	23
5.2.2.12	<code>Grid_laplace(const Grid *in_grid, Grid *out_grid)</code>	23
5.2.2.13	<code>Grid_Lee(const Grid *grid, int i, int j, int radius, float g_var)</code>	24
5.2.2.14	<code>Grid_mean(const Grid *grid)</code>	24
5.2.2.15	<code>Grid_mean_win(const Grid *grid, int i, int j, int radius)</code>	24
5.2.2.16	<code>Grid_median(const Grid *grid, int i, int j, int radius)</code>	24
5.2.2.17	<code>Grid_read(const char *filePath, Grid *grid)</code>	25
5.2.2.18	<code>Grid_set(int col, int row, Grid *grid, float val)</code>	25
5.2.2.19	<code>Grid_square(const Grid *in_grid, Grid *out_grid)</code>	25
5.2.2.20	<code>Grid_stdev(const float var)</code>	26
5.2.2.21	<code>Grid_subtract(const Grid *in_grid_1, const Grid *in_grid_2, Grid *out_grid)</code>	26
5.2.2.22	<code>Grid_var(const Grid *grid, const float mean)</code>	26
5.2.2.23	<code>Grid_var_win(const Grid *grid, int i, int j, int radius, float mean)</code>	27
5.2.2.24	<code>Grid_zero_cross(const Grid *in_grid, Grid *out_grid)</code>	27
5.2.2.25	<code>ridge(float xx, float xy, float yy)</code>	28
5.2.2.26	<code>sdgd(float x, float y, float xx, float xy, float yy)</code>	28
5.3	<code>src/Activation.c</code> File Reference	28
5.3.1	Detailed Description	29
5.3.2	Function Documentation	29
5.3.2.1	<code>CONVA_atan(double x)</code>	29
5.3.2.2	<code>CONVA_ident(double x)</code>	29
5.3.2.3	<code>CONVA_log(double x)</code>	29
5.3.2.4	<code>CONVA_step(double x)</code>	30
5.3.2.5	<code>CONVA_tanh(double x)</code>	30
5.4	<code>src/conv-fence.c</code> File Reference	30
5.4.1	Detailed Description	31
5.4.2	Function Documentation	31

5.4.2.1	CONV_ExchangeEndFence(void *privateData)	31
5.4.2.2	CONV_ExchangeFence(CONVPatchDesc *patch, float **matrix, CONVTiming *timedata, void *privateData)	31
5.4.2.3	CONV_ExchangeInitFence(CONVPatchDesc *patch, float **m1, float **m2, void *privateData)	32
5.5	src/options.c File Reference	32
5.5.1	Detailed Description	32
5.5.2	Function Documentation	32
5.5.2.1	CONV_Abort(const char str[])	32
5.5.2.2	CONV_ParseArgs(int argc, char **argv, CONVOptions *options)	33
5.6	src/ctiming.c File Reference	33
5.6.1	Detailed Description	34
5.6.2	Function Documentation	34
5.6.2.1	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)	34
5.6.2.2	CONV_TimeExchange(CONVPatchDesc *patch, float **m1, float **m2, int(*exchangeInit)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)	34
5.6.2.3	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)	35
5.6.2.4	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)	35
5.6.2.5	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)	36
5.6.2.6	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)	36
5.7	src/Grid.c File Reference	37
5.7.1	Detailed Description	38
5.7.2	Function Documentation	38

5.7.2.1	<code>calc_ridge(float x, float y, float xx, float xy, float yy)</code>	38
5.7.2.2	<code>Grid_alloc(Grid *grid)</code>	39
5.7.2.3	<code>Grid_convolve(const Grid *in_grid, const Kernel *kern, Grid *out_grid)</code>	39
5.7.2.4	<code>Grid_copy(Grid *g1, Grid *g2, int full)</code>	40
5.7.2.5	<code>Grid_dilate(const Grid *in_grid, int i, int j, Kernel *k)</code>	40
5.7.2.6	<code>Grid_downsample(const Grid *in_grid, Grid *out_grid)</code>	40
5.7.2.7	<code>Grid_erode(const Grid *in_grid, int i, int j, Kernel *k)</code>	41
5.7.2.8	<code>Grid_free(Grid *grid)</code>	41
5.7.2.9	<code>Grid_get(int col, int row, const Grid *grid)</code>	41
5.7.2.10	<code>Grid_IDW(const Grid *in_grid, int n_neighbors, float distance, int power, Grid *out_grid)</code>	42
5.7.2.11	<code>Grid_init(Grid *grid)</code>	42
5.7.2.12	<code>Grid_laplace(const Grid *in_grid, Grid *out_grid)</code>	42
5.7.2.13	<code>Grid_Lee(const Grid *in_grid, int i, int j, int radius, float g_var)</code>	43
5.7.2.14	<code>Grid_mean(const Grid *in_grid)</code>	43
5.7.2.15	<code>Grid_mean_win(const Grid *in_grid, int i, int j, int radius)</code>	43
5.7.2.16	<code>Grid_median(const Grid *in_grid, int i, int j, int radius)</code>	44
5.7.2.17	<code>Grid_read(const char *filePath, Grid *grid)</code>	44
5.7.2.18	<code>Grid_set(int col, int row, Grid *grid, float val)</code>	44
5.7.2.19	<code>Grid_square(const Grid *in_grid, Grid *out_grid)</code>	45
5.7.2.20	<code>Grid_stdev(const float var)</code>	45
5.7.2.21	<code>Grid_subtract(const Grid *in_grid_1, const Grid *in_grid_2, Grid *out_grid)</code>	46
5.7.2.22	<code>Grid_var(const Grid *in_grid, const float mean)</code>	46
5.7.2.23	<code>Grid_var_win(const Grid *in_grid, int i, int j, int radius, float mean)</code>	46
5.7.2.24	<code>Grid_zero_cross(const Grid *in_grid, Grid *out_grid)</code>	47
5.7.2.25	<code>ridge(float xx, float xy, float yy)</code>	47
5.7.2.26	<code>sdgd(float x, float y, float xx, float xy, float yy)</code>	47
5.8	<code>src/Kernel.c</code> File Reference	48
5.8.1	Detailed Description	48
5.8.2	Function Documentation	48
5.8.2.1	<code>Kernel_alloc(Kernel *kern)</code>	48

5.8.2.2	<code>Kernel_free(Kernel *kern)</code>	49
5.8.2.3	<code>Kernel_gauss(float sigma, Kernel *kern)</code>	49
5.8.2.4	<code>Kernel_get(int col, int row, const Kernel *kern)</code>	49
5.8.2.5	<code>Kernel_init(Kernel *kernel)</code>	50
5.8.2.6	<code>Kernel_mean(int radius, Kernel *kern)</code>	50
5.8.2.7	<code>Kernel_set(int col, int row, Kernel *kern, float val)</code>	50
5.8.2.8	<code>Kernel_sobel(int axis, Kernel *kern)</code>	51
5.8.2.9	<code>PixelsNeededForSigma(float sigma)</code>	51
5.9	<code>src/patch.c</code> File Reference	51
5.9.1	Detailed Description	53
5.9.2	Function Documentation	53
5.9.2.1	<code>CONV_AllocateKernel(float ***k, int radius)</code>	53
5.9.2.2	<code>CONV_AllocateLocalMesh(CONVPatchDesc *patch, float ***m1, float ***m2)</code>	53
5.9.2.3	<code>CONV_cleanPatch(CONVPatchDesc *patch, float **matrix)</code>	54
5.9.2.4	<code>CONV_Convolve(CONVPatchDesc *patch, float **mat, float **weights, int radius, int row, int col)</code>	54
5.9.2.5	<code>CONV_Curvature(CONVPatchDesc *patch, float **matrix, int row, int col)</code>	54
5.9.2.6	<code>CONV_FreeKernel(float **kernel)</code>	55
5.9.2.7	<code>CONV_FreeLocalMesh(CONVPatchDesc *patch, float **m1, float **m2)</code>	55
5.9.2.8	<code>CONV_FreePatch(CONVPatchDesc *patch)</code>	55
5.9.2.9	<code>CONV_IDW(CONVPatchDesc *patch, float **matrix, int row, int col, int radius, int power)</code>	55
5.9.2.10	<code>CONV_InitLocalMesh(CONVPatchDesc *patch, float **m1, float **m2)</code>	56
5.9.2.11	<code>CONV_Laplace(CONVPatchDesc *patch, float **matrix, int row, int col)</code>	56
5.9.2.12	<code>CONV_Lee(CONVPatchDesc *patch, float **mat, int i, int j, int radius, float g_var)</code>	57
5.9.2.13	<code>CONV_mixedDerivative(CONVPatchDesc *patch, float **matrix, int row, int col)</code>	58
5.9.2.14	<code>CONV_NN(CONVPatchDesc *patch, float **matrix, int row, int col)</code>	58
5.9.2.15	<code>CONV_ParseDatasetHeader(CONVPatchDesc *patch, const char *dset)</code>	58
5.9.2.16	<code>CONV_PatchCreateDataMeshDesc(CONVOptions *options, CONVPatchDesc *patch)</code>	58
5.9.2.17	<code>CONV_PatchCreateProcessMesh(CONVOptions *options, CONVPatchDesc *patch)</code>	59

5.9.2.18	CONV_PatchCreateProcessMeshWithCart(CONVOptions *options, CONV↔ PatchDesc *patch)	59
5.9.2.19	CONV_PatchMean(CONVPatchDesc *patch, float **mat)	59
5.9.2.20	CONV_PatchVariance(CONVPatchDesc *patch, float **mat, const float mean) .	60
5.9.2.21	CONV_WriteLocalMesh(CONVOptions *opts, CONVPatchDesc *patch, float **mat, int iter)	60
5.10	src/Raster.c File Reference	60
5.10.1	Detailed Description	61
5.10.2	Function Documentation	61
5.10.2.1	Raster_copy(Raster *r1, Raster *r2, int full)	61
5.10.2.2	Raster_downsample(const Raster *r1, Raster *r2)	62
5.10.2.3	Raster_free(Raster *raster)	62
5.10.2.4	Raster_init(Raster *raster)	62
5.10.2.5	Raster_read(char *in_file, Raster *raster)	63
5.10.2.6	Raster_stat(char *in_file, Raster *raster)	63
5.10.2.7	Raster_write(char *out_file, Raster *raster)	63
5.11	src/Util.c File Reference	64
5.11.1	Detailed Description	64
5.11.2	Function Documentation	64
5.11.2.1	absolute(double x)	64
5.11.2.2	almostEqualFloat(float a, float b, int maxUlps)	65
5.11.2.3	byte_swap32(uint32_t a)	65
5.11.2.4	compareFloat(const void *a, const void *b)	65
5.11.2.5	compareInt(const void *a, const void *b)	66
5.11.2.6	gauss(int x, int y, float sigma)	66
5.11.2.7	nibb_swap32(uint32_t a)	66
5.11.2.8	relativeEqualFloat(float a, float b, float maxRelDiff)	67
	Index	69

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:

- [CONVOptions](#)
Processing parameters for identifying mesh size and dataset 7
- [CONVPatchDesc](#)
Descriptor for managing processor mesh, data mesh and raster information 8
- [CONVTiming](#)
Time tracker for profiling communication exchanges 10
- [CRS](#)
Container class for [CRS](#) data Useful for translating between proj and GeoTIFF, or LAS formats 11
- [Grid](#)
Container struct for 2-dimensional array data 12
- [Kernel](#) 13
- [mem_win](#)
Memory window for Asynchronous Remote Memory Access 13
- [mGrid](#)
Container struct for 2-dimensional multivariate array data 14
- [Raster](#)
This class contains basic metadata needed to georeference and project 2D arrays for raster datasets 15

Chapter 3

File Index

3.1 File List

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

include/ Activation.h	??
include/ conv-io.h	??
include/ conv.h	??
include/ CRS.h	17
include/ global.h	??
include/ Grid.h	18
include/ Kernel.h	??
include/ Raster.h	??
include/ Util.h	??
src/ Activation.c	28
src/ conv-fence.c	30
src/ coptions.c	32
src/ ctiming.c	33
src/ Grid.c	
File containing function definitions for pixel arrays	37
src/ Kernel.c	
Functions needed for kernel convolution and window functions	48
src/ patch.c	
Basic patch configuration	51
src/ Raster.c	
File containing function definitions for Raster class	60
src/ Util.c	64

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 [gl](#)
- 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::gl

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::Ini

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

<i>x</i>	First order derivative in x direction
<i>y</i>	First order derivative in y direction
<i>xx</i>	Second-order derivative in x direction
<i>xy</i>	Second-order diagonal derivative
<i>yy</i>	Second-order derivative in y direction

See also

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

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

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

<i>grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>kern</i>	Kernel instance
<i>out_grid</i>	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

<i>g1</i>	Source Grid instance
<i>g2</i>	Destination Grid instance
<i>full</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>k</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

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

Returns

Minimum value within structure kernel

5.2.2.8 void Grid_free (Grid * grid)

Free memory allocated for [Grid](#) instance.

Parameters

<i>Source</i>	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

<i>col</i>	Column index for pixel
<i>row</i>	Row index for pixel
<i>grid</i>	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

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

Returns

0

5.2.2.11 void Grid_init (Grid * grid)

Initialize a [Grid](#) instance.

Parameters

<i>grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	Neighborhood search radius
<i>g_var</i>	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

<i>in_grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	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

<i>filePath</i>	File path to GDAL-compliant dataset
<i>grid</i>	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

<i>col</i>	Column index of pixel
<i>row</i>	Row index of pixel
<i>grid</i>	Destination Grid instance
<i>val</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

<i>var</i>	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

<i>in_grid</i> _↔ <i>_1</i>	Source Grid to be subtracted from
<i>in_grid</i> _↔ <i>_2</i>	Source Grid to subtract
<i>out_grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>mean</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	Neighborhood search radius
<i>mean</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

<i>xx</i>	Second-order x derivative
<i>xy</i>	Second-order diagonal derivative
<i>yy</i>	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

<i>x</i>	First-order x derivative
<i>y</i>	First-order y derivative
<i>xx</i>	Second-order x derivative
<i>xy</i>	Second-order diagonal derivative
<i>yy</i>	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 from a performance 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

<code>privateData</code>	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

<code>patch</code>	Local Patch instance
<code>matrix</code>	Source 2-D array
<code>timedata</code>	Timing tracker
<code>privateData</code>	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 from a performance penalty since each send is reliant on a matching receive. As the number of processes grows, this will likely cause a deadlock.

Parameters

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

Returns

MPI_SUCCESS if success else MPI_ERR

5.5 src/options.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

<i>str</i>	Error message
------------	---------------

Returns

void

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

Parse command line arguments.

Parameters

<i>argc</i>	Argument count
<i>argv</i>	Array of arguments
<i>options</i>	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_Timeliterations](#) (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)(CONVPatchDesc *, float **, float **, void *), int(*exchange)(CONVPatchDesc *, float **, CONVTiming *, void *), int(*exchangeEnd)(void *), CONVTiming *timedata)

Time the exchange 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

<i>patch</i>	Local Patch instance
<i>m1</i>	First 2-D array
<i>m2</i>	Second 2-D array
<i>exchangeInit</i>	Communication initialization function
<i>exchange</i>	Communication function
<i>exchangeEnd</i>	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 exchange between patches.

Parameters

<i>patch</i>	Local Patch instance
--------------	----------------------

Parameters

<i>m1</i>	First 2-D array
<i>m2</i>	Second 2-D array
<i>exchangeInit</i>	Communication initialization function
<i>exchange</i>	Communication function
<i>exchangeEnd</i>	Communication finalization function
<i>timedata</i>	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

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

Returns

0

5.6.2.4 `int CONV_Timeliterations (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

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

Parameters

<i>exchangeInit</i>	Exchange initialization function
<i>exchange</i>	HaloExchange function
<i>exchangeEnd</i>	Exchange finalization function
<i>timedata</i>	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

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

<i>patch</i>	Local Patch instance
<i>nIter</i>	Number of iterations

Parameters

<i>m1</i>	First 2-D array
<i>m2</i>	Second 2-D array
<i>exchangeInit</i>	Communication initialization function
<i>exchange</i>	Communication function
<i>exchangeEnd</i>	Communication finalization function
<i>timedata</i>	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

<i>x</i>	First order derivative in x direction
<i>y</i>	First order derivative in y direction
<i>xx</i>	Second-order derivative in x direction
<i>xy</i>	Second-order diagonal derivative
<i>yy</i>	Second-order derivative in y direction

See also

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

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

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

<i>grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>kern</i>	Kernel instance
<i>out_grid</i>	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

<i>g1</i>	Source Grid instance
<i>g2</i>	Destination Grid instance
<i>full</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>k</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

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

Returns

Minimum value within structure kernel

5.7.2.8 void Grid_free (Grid * *grid*)

Free memory allocated for [Grid](#) instance.

Parameters

<i>Source</i>	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

<i>col</i>	Column index for pixel
<i>row</i>	Row index for pixel
<i>grid</i>	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

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

Returns

0

5.7.2.11 `void Grid_init (Grid * grid)`

Initialize a [Grid](#) instance.

Parameters

<i>grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	Neighborhood search radius
<i>g_var</i>	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

<i>in_grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	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

<i>filePath</i>	File path to GDAL-compliant dataset
<i>grid</i>	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

<i>col</i>	Column index of pixel
<i>row</i>	Row index of pixel
<i>grid</i>	Destination Grid instance
<i>val</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

<i>var</i>	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

<i>in_grid_1</i>	Source Grid to be subtracted from
<i>in_grid_2</i>	Source Grid to subtract
<i>out_grid</i>	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

<i>in_grid</i>	Source Grid instance
<i>mean</i>	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

<i>in_grid</i>	Source Grid instance
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	Neighborhood search radius
<i>mean</i>	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

<i>in_grid</i>	Source Grid instance
<i>out_grid</i>	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

<i>xx</i>	Second-order x derivative
<i>xy</i>	Second-order diagonal derivative
<i>yy</i>	Second-order y derivative

Returns

Ridge strength factor

5.7.2.26 float sgd (float x, float y, float xx, float xy, float yy)

Second derivative in the gradient direction.

Parameters

<i>x</i>	First-order x derivative
<i>y</i>	First-order y derivative
<i>xx</i>	Second-order x derivative
<i>xy</i>	Second-order diagonal derivative
<i>yy</i>	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

<i>kern</i>	Destination Kernel instance
-------------	---

Returns

1 if success, else -1

5.8.2.2 void Kernel_free (Kernel * kern)

Free memory allocated to [Kernel](#).

Parameters

<i>kern</i>	Kernel instance to free
-------------	---

Returns

void

5.8.2.3 int Kernel_gauss (float sigma, Kernel * kern)

Generate gaussian kernel.

Parameters

<i>sigma</i>	Sigma defining the gaussian
<i>kern</i>	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

<i>col</i>	Column index
<i>row</i>	Row index
<i>kern</i>	Source Kernel instance

Returns

If success pixel value, else 0

5.8.2.5 void Kernel_init (Kernel * kernel)

Initialize a [Kernel](#) instance.

Parameters

<i>kernel</i>	Target 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

<i>radius</i>	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

<i>row</i>	Row index
<i>kern</i>	Kernel Instance
<i>val</i>	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

<i>axis</i>	X if 0, else Y
<i>kern</i>	Destination Kernel instance

Returns

0

5.8.2.9 int PixelsNeededForSigma (float *sigma*)

Calculate the diameter necessary to represent a gaussian.

Parameters

<i>sigma</i>	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 $f_{x,y}$ 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
$$\hat{I}_s = \bar{I}_s + k_s(I_s - \bar{I}_s)$$
- 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

<i>k</i>	Pointer for array
<i>radius</i>	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

<i>Destination</i>	Patch instance
<i>m1</i>	First array (used as source array in processing)
<i>m2</i>	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

<i>patch</i>	Target patch instance
<i>matrix</i>	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

<i>patch</i>	Source patch instance
<i>mat</i>	Source 2-d array
<i>weights</i>	2-d Weight array
<i>radius</i>	Convolution Kernel radius
<i>row</i>	Row index
<i>col</i>	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

<i>patch</i>	Source Patch instance
<i>matrix</i>	Source 2-D array
<i>row</i>	Row index
<i>col</i>	Column index

Returns

Curvature value

5.9.2.6 int CONV_FreeKernel (float ** *kernel*)

Free memory allocated for kernel array.

Parameters

<i>kernel</i>	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

<i>path</i>	Local patch instance
<i>m1</i>	First array
<i>m2</i>	Second array

Returns

0

5.9.2.8 void CONV_FreePatch (CONVPatchDesc * *patch*)

Free memory allocated to a Patch instance.

Parameters

<i>patch</i>	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

<i>patch</i>	Source patch instance
<i>matrix</i>	Source 2-D array
<i>row</i>	Row index
<i>col</i>	Column index
<i>radius</i>	Neighborhood search radius
<i>power</i>	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

<i>patch</i>	Destination patch instance
<i>m1</i>	First array
<i>m2</i>	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

<i>patch</i>	Source Patch instance
<i>matrix</i>	Source 2-D array
<i>row</i>	Row index
<i>col</i>	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

$$\hat{I}_s = \bar{I}_s + k_s(I_s - \bar{I}_s)$$

Where: \bar{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

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

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

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

Where $\text{var}(z')$ is the intensity variance and \bar{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

<i>patch</i>	Target patch instance
<i>mat</i>	Source 2-d Array to read from
<i>i</i>	Column index
<i>j</i>	Row index
<i>radius</i>	Neighborhood search radius
<i>g_var</i>	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

<i>patch</i>	Source Patch instance
<i>matrix</i>	Source 2-D array
<i>row</i>	Row index
<i>col</i>	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

<i>patch</i>	target Patch instance
<i>matrix</i>	Source 2-d array
<i>row</i>	Row index
<i>col</i>	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

<i>patch</i>	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

<i>options</i>	Mesh creation options
<i>patch</i>	Destination Patch instance

Returns

0

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

Create a patch mesh of processors.

Parameters

<i>options</i>	Options for mesh
<i>path</i>	Target patch definition

Returns

0

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

Create cartesian patch processor mesh.

Parameters

<i>options</i>	Options for mesh creation
<i>patch</i>	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

<i>patch</i>	Target Patch instance
<i>mat</i>	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

<i>patch</i>	Target patch instance
<i>mat</i>	Source 2-D array to read from
<i>mean</i>	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

<i>opts</i>	Mesh creation options
<i>patch</i>	Source patch instance
<i>mat</i>	Source 2-D array to write
<i>iter</i>	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 constructor for `Raster` instances, may be full or shallow copy.

Parameters

<i>r1</i>	Source <code>Raster</code> instance
<i>r2</i>	Destination <code>Raster</code> instance
<i>full</i>	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

<i>r1</i>	Source raster instance
<i>r2</i>	Destination raster instance

Returns

1 if succeeded, 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

<i>raster</i>	Raster instance to deallocate
---------------	---

Returns

void

5.10.2.4 `void Raster_init (Raster * raster)`

Initialize a new [Raster](#) instance.

`void Raster_init`

Parameters

<i>raster</i>	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

<i>in_file</i>	File path to GDAL-compliant raster
<i>raster</i>	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

<i>in_file</i>	File path to GDAL compliant dataset
<i>raster</i>	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

<i>out_file</i>	Destination file path
<i>raster</i>	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

<i>x</i>	Input double
----------	--------------

Returns

$abs(x)$

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

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%20html>

Parameters

<i>maxUlp</i>	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

<i>a</i>	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

<i>a</i>	Pointer to first value
<i>b</i>	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

<i>a</i>	pointer to first integer
<i>b</i>	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

<i>x</i>	Column index
<i>y</i>	Row index
<i>sigma</i>	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

<i>a</i>	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

<i>a</i>	first float
<i>b</i>	second float
<i>maxRelDiff</i>	Equality threshold

Returns

1 if the difference is less than threshold, else 0

Index

- above_LRows
 - mem_win, [13](#)
- absolute
 - Util.c, [64](#)
- Activation.c
 - CONVA_atan, [29](#)
 - CONVA_ident, [29](#)
 - CONVA_log, [29](#)
 - CONVA_step, [30](#)
 - CONVA_tanh, [30](#)
- affine
 - Raster, [15](#)
- almostEqualFloat
 - Util.c, [65](#)
- byte_swap32
 - Util.c, [65](#)
- CONV_Abort
 - coptions.c, [32](#)
- CONV_AllocateKernel
 - patch.c, [53](#)
- CONV_AllocateLocalMesh
 - patch.c, [53](#)
- CONV_Convolve
 - patch.c, [54](#)
- CONV_Curvature
 - patch.c, [54](#)
- CONV_ExchangeEndFence
 - conv-fence.c, [31](#)
- CONV_ExchangeFence
 - conv-fence.c, [31](#)
- CONV_ExchangeInitFence
 - conv-fence.c, [32](#)
- CONV_FreeKernel
 - patch.c, [55](#)
- CONV_FreeLocalMesh
 - patch.c, [55](#)
- CONV_FreePatch
 - patch.c, [55](#)
- CONV_IDW
 - patch.c, [55](#)
- CONV_InitLocalMesh
 - patch.c, [56](#)
- CONV_Laplace
 - patch.c, [56](#)
- CONV_Lee
 - patch.c, [57](#)
- CONV_NN
 - patch.c, [58](#)
- CONV_ParseArgs
 - coptions.c, [33](#)
- CONV_ParseDatasetHeader
 - patch.c, [58](#)
- CONV_PatchCreateDataMeshDesc
 - patch.c, [58](#)
- CONV_PatchCreateProcessMesh
 - patch.c, [59](#)
- CONV_PatchCreateProcessMeshWithCart
 - patch.c, [59](#)
- CONV_PatchMean
 - patch.c, [59](#)
- CONV_PatchVariance
 - patch.c, [60](#)
- CONV_TimeClassify
 - ctiming.c, [34](#)
- CONV_TimeExchange
 - ctiming.c, [34](#)
- CONV_TimeFillNull
 - ctiming.c, [35](#)
- CONV_TimeIterations
 - ctiming.c, [35](#)
- CONV_TimePreProcess
 - ctiming.c, [36](#)
- CONV_TimeProcess
 - ctiming.c, [36](#)
- CONV_WriteLocalMesh
 - patch.c, [60](#)
- CONV_cleanPatch
 - patch.c, [54](#)
- CONV_mixedDerivative
 - patch.c, [57](#)
- CONVA_atan
 - Activation.c, [29](#)
- CONVA_ident
 - Activation.c, [29](#)
- CONVA_log
 - Activation.c, [29](#)
- CONVA_step
 - Activation.c, [30](#)
- CONVA_tanh
 - Activation.c, [30](#)
- CONVOptions, [7](#)
 - dataset, [7](#)
 - doIO, [7](#)
 - nIter, [8](#)
 - outdir, [8](#)
 - pNI, [8](#)
 - prefix, [8](#)

- restartIter, 8
- verbose, 8
- CONVPatchDesc, 8
 - gAffine, 9
 - gNI, 9
 - gl, 9
 - hDS, 9
 - left, 9
 - Ini, 10
 - nBands, 10
 - nodata, 10
 - pNI, 10
 - patchI, 10
 - proj_str, 10
 - ul, 10
- CONVTiming, 10
 - exchtime, 11
 - itertime, 11
- CRS, 11
 - m_gtiff, 12
 - proj, 12
- calc_ridge
 - Grid.c, 38
 - Grid.h, 19
- compareFloat
 - Util.c, 65
- compareInt
 - Util.c, 66
- conv-fence.c
 - CONV_ExchangeEndFence, 31
 - CONV_ExchangeFence, 31
 - CONV_ExchangeInitFence, 32
- coptions.c
 - CONV_Abort, 32
 - CONV_ParseArgs, 33
- ctiming.c
 - CONV_TimeClassify, 34
 - CONV_TimeExchange, 34
 - CONV_TimeFillNull, 35
 - CONV_Timeliterations, 35
 - CONV_TimePreProcess, 36
 - CONV_TimeProcess, 36
- data
 - Grid, 12
- dataset
 - CONVOptions, 7
- dimids
 - mGrid, 14
- dims
 - mGrid, 14
- doIO
 - CONVOptions, 7
- exchtime
 - CONVTiming, 11
- gAffine
 - CONVPatchDesc, 9
- gNI
 - CONVPatchDesc, 9
- gauss
 - Util.c, 66
- gl
 - CONVPatchDesc, 9
- Grid, 12
 - data, 12
 - nodata, 12
 - rows, 12
- Grid.c
 - calc_ridge, 38
 - Grid_IDW, 41
 - Grid_Lee, 43
 - Grid_alloc, 39
 - Grid_convolve, 39
 - Grid_copy, 39
 - Grid_dilate, 40
 - Grid_downsample, 40
 - Grid_erode, 40
 - Grid_free, 41
 - Grid_get, 41
 - Grid_init, 42
 - Grid_laplace, 42
 - Grid_mean, 43
 - Grid_mean_win, 43
 - Grid_median, 44
 - Grid_read, 44
 - Grid_set, 44
 - Grid_square, 45
 - Grid_stdev, 45
 - Grid_subtract, 45
 - Grid_var, 46
 - Grid_var_win, 46
 - Grid_zero_cross, 46
 - ridge, 47
 - sdgd, 47
- Grid.h
 - calc_ridge, 19
 - Grid_IDW, 22
 - Grid_Lee, 23
 - Grid_alloc, 20
 - Grid_convolve, 20
 - Grid_copy, 20
 - Grid_dilate, 21
 - Grid_downsample, 21
 - Grid_erode, 21
 - Grid_free, 22
 - Grid_get, 22
 - Grid_init, 23
 - Grid_laplace, 23
 - Grid_mean, 24
 - Grid_mean_win, 24
 - Grid_median, 24
 - Grid_read, 25
 - Grid_set, 25
 - Grid_square, 25
 - Grid_stdev, 26

- Grid_subtract, 26
- Grid_var, 26
- Grid_var_win, 27
- Grid_zero_cross, 27
- ridge, 28
- sdgd, 28
- Grid_IDW
 - Grid.c, 41
 - Grid.h, 22
- Grid_Lee
 - Grid.c, 43
 - Grid.h, 23
- Grid_alloc
 - Grid.c, 39
 - Grid.h, 20
- Grid_convolve
 - Grid.c, 39
 - Grid.h, 20
- Grid_copy
 - Grid.c, 39
 - Grid.h, 20
- Grid_dilate
 - Grid.c, 40
 - Grid.h, 21
- Grid_downsample
 - Grid.c, 40
 - Grid.h, 21
- Grid_erode
 - Grid.c, 40
 - Grid.h, 21
- Grid_free
 - Grid.c, 41
 - Grid.h, 22
- Grid_get
 - Grid.c, 41
 - Grid.h, 22
- Grid_init
 - Grid.c, 42
 - Grid.h, 23
- Grid_laplace
 - Grid.c, 42
 - Grid.h, 23
- Grid_mean
 - Grid.c, 43
 - Grid.h, 24
- Grid_mean_win
 - Grid.c, 43
 - Grid.h, 24
- Grid_median
 - Grid.c, 44
 - Grid.h, 24
- Grid_read
 - Grid.c, 44
 - Grid.h, 25
- Grid_set
 - Grid.c, 44
 - Grid.h, 25
- Grid_square
 - Grid.c, 45
 - Grid.h, 25
- Grid_stdev
 - Grid.c, 45
 - Grid.h, 26
- Grid_subtract
 - Grid.c, 45
 - Grid.h, 26
- Grid_var
 - Grid.c, 46
 - Grid.h, 26
- Grid_var_win
 - Grid.c, 46
 - Grid.h, 27
- Grid_zero_cross
 - Grid.c, 46
 - Grid.h, 27
- hDS
 - CONVPatchDesc, 9
- include/CRS.h, 17
- include/Grid.h, 18
- itertime
 - CONVTiming, 11
- Kernel, 13
- Kernel.c
 - Kernel_alloc, 48
 - Kernel_free, 49
 - Kernel_gauss, 49
 - Kernel_get, 49
 - Kernel_init, 50
 - Kernel_mean, 50
 - Kernel_set, 50
 - Kernel_sobel, 50
 - PixelsNeededForSigma, 51
- Kernel_alloc
 - Kernel.c, 48
- Kernel_free
 - Kernel.c, 49
- Kernel_gauss
 - Kernel.c, 49
- Kernel_get
 - Kernel.c, 49
- Kernel_init
 - Kernel.c, 50
- Kernel_mean
 - Kernel.c, 50
- Kernel_set
 - Kernel.c, 50
- Kernel_sobel
 - Kernel.c, 50
- left
 - CONVPatchDesc, 9
- left_LCols
 - mem_win, 13
- Ini

- CONVPatchDesc, 10
- m_gtiff
 - CRS, 12
- mGrid, 14
 - dimids, 14
 - dims, 14
 - ndims, 15
 - nvars, 15
 - varids, 15
- mem_win, 13
 - above_LRows, 13
 - left_LCols, 13
 - right_LCols, 14
 - win, 14
- nBands
 - CONVPatchDesc, 10
- nIter
 - CONVOptions, 8
- ndims
 - mGrid, 15
- nibb_swap32
 - Util.c, 66
- nodata
 - CONVPatchDesc, 10
 - Grid, 12
- nvars
 - mGrid, 15
- outdir
 - CONVOptions, 8
- pNI
 - CONVOptions, 8
 - CONVPatchDesc, 10
- patch.c
 - CONV_AllocateKernel, 53
 - CONV_AllocateLocalMesh, 53
 - CONV_Convolve, 54
 - CONV_Curvature, 54
 - CONV_FreeKernel, 55
 - CONV_FreeLocalMesh, 55
 - CONV_FreePatch, 55
 - CONV_IDW, 55
 - CONV_InitLocalMesh, 56
 - CONV_Laplace, 56
 - CONV_Lee, 57
 - CONV_NN, 58
 - CONV_ParseDatasetHeader, 58
 - CONV_PatchCreateDataMeshDesc, 58
 - CONV_PatchCreateProcessMesh, 59
 - CONV_PatchCreateProcessMeshWithCart, 59
 - CONV_PatchMean, 59
 - CONV_PatchVariance, 60
 - CONV_WriteLocalMesh, 60
 - CONV_cleanPatch, 54
 - CONV_mixedDerivative, 57
- patchI
 - CONVPatchDesc, 10
- PixelsNeededForSigma
 - Kernel.c, 51
- prefix
 - CONVOptions, 8
- proj
 - CRS, 12
- proj4
 - Raster, 15
- proj_str
 - CONVPatchDesc, 10
- Raster, 15
 - affine, 15
 - proj4, 15
- Raster.c
 - Raster_copy, 61
 - Raster_downsample, 61
 - Raster_free, 62
 - Raster_init, 62
 - Raster_read, 62
 - Raster_stat, 63
 - Raster_write, 63
- Raster_copy
 - Raster.c, 61
- Raster_downsample
 - Raster.c, 61
- Raster_free
 - Raster.c, 62
- Raster_init
 - Raster.c, 62
- Raster_read
 - Raster.c, 62
- Raster_stat
 - Raster.c, 63
- Raster_write
 - Raster.c, 63
- relativeEqualFloat
 - Util.c, 67
- restartIter
 - CONVOptions, 8
- ridge
 - Grid.c, 47
 - Grid.h, 28
- right_LCols
 - mem_win, 14
- rows
 - Grid, 12
- sdgd
 - Grid.c, 47
 - Grid.h, 28
- src/Activation.c, 28
- src/Grid.c, 37
- src/Kernel.c, 48
- src/Raster.c, 60
- src/Util.c, 64
- src/conv-fence.c, 30
- src/coptions.c, 32

src/ctiming.c, [33](#)
src/patch.c, [51](#)

ul

 CONVPatchDesc, [10](#)

Util.c

 absolute, [64](#)

 almostEqualFloat, [65](#)

 byte_swap32, [65](#)

 compareFloat, [65](#)

 compareInt, [66](#)

 gauss, [66](#)

 nibb_swap32, [66](#)

 relativeEqualFloat, [67](#)

varids

 mGrid, [15](#)

verbose

 CONVOptions, [8](#)

win

 mem_win, [14](#)