VKarPhase

Richard J. Mathar

Generated by Doxygen 1.8.2

Mon Oct 15 2012 11:04:45

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Frame	3
FrameSet	4
Point	8

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Data Structure Documentation

4.1 Frame Class Reference

Public Member Functions

- Frame ()
- Frame (int dimx, int dimy)
- Frame (int dimx, int dimy, const Point &pt)
- int area () const
- bool inside (const Frame &windo) const
- bool covers (int x, int y) const

Data Fields

- int nx
- int ny
- Point posit

4.1.1 Detailed Description

A rectangular array subdivided in square pixels. The content of the array is a description of a 2D integer rectangular coordinate space. No actual data of some function of these pixels is stored.

Author

R. J. Mathar

Since

2010-01-01

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Frame::Frame ( )

Default ctor. This is equivalent to creating a field with no pixels.

4.1.2.2 Frame::Frame (int *dimx*, int *dimy*)

Ctor of a frame anchored at the origin with known dimensions. The frame ranges from 0<=x<dimx and 0<=y<dimy.

Parameters

dimx	The number of pixels in the x direction
dimy	The number of pixels in the y direction

4.1.2.3 Frame::Frame (int *dimx*, int *dimy*, const Point & *pt*)

General ctor of a frame anchored at the origin with known dimensions.

Parameters

dimx	The number of pixels in the x direction
dimy	The number of pixels in the y direction
pt	The position of the llx and lly point of the frame in a larger scope.

4.1.3 Member Function Documentation

4.1.3.1 int Frame::area ( ) const

The integer area in units of pixels squared.

4.1.3.2 bool Frame::covers (int *x*, int *y*) const

Test if the pixel is inside this window.

Parameters

x	The x-coordinate of the pixel
y	The y-coordinate of the pixel

Returns

true if the pixel specified is one of the pixels inside this array. x coordinates equal to posit.x and y coordinates equatl to posit.y are considered to be outside for this purpose.

4.1.3.3 bool Frame::inside (const Frame & *windo*) const

Test if this is entirely inside another frame.

Parameters

windo	The reference array .

Returns

True if this array is completely covered by windo; false if there is no or only partial overlap.

4.1.4 Field Documentation

4.1.4.1 int Frame::nx

Number of pixels in the x direction

4.1.4.2 int Frame::ny

Number of pixels in the y direction

4.1.4.3 Point Frame::posit

Position of the origin (lower left x and lower left y coordinate) on a larger canvas with more than one frame.

4.2 FrameSet Class Reference

Public Member Functions

- FrameSet ()
- FrameSet (const Frame &windo)
- FrameSet & operator+= (const Frame &windo)
- FrameSet & operator+= (const FrameSet &windo)
- void boundbox (int bb[4]) const
- int area () const
- void nonoverl ()
- int covers (int x, int y) const
- void loc (int idx, int widx[3]) const

Data Fields

- std::vector< Frame > wd
- std::vector< int > areas

Private Member Functions

- int covers (std::vector< int > &xedgvec, std::vector< int > &yedgvec, int xidx, int yidx) const

4.2.1 Detailed Description

A rectangular pixelized array with double values in windowed regions. The class contains rectangular non-overlapping subregions on a commensurate common pixel basis. Values in the interstitial regions between these windows are not stored. The main functionality is an efficient storage scheme for a big rectangular region (super-array) with internal stripe and sub-block geometry.

Author

R. J. Mathar

Since

2010-10-31

4.2.2 Constructor & Destructor Documentation

4.2.2.1 FrameSet::FrameSet ( )

Ctor with no windows at all.

4.2.2.2 FrameSet::FrameSet (const Frame & *windo*)

Ctor with a single rectangular window.

Parameters

windo	The sole window to represent the geometry.

4.2.3 Member Function Documentation

4.2.3.1 int FrameSet::area ( ) const

Total number of pixels squared.

Returns

The sum over all areas of the subwindows. This is usually smaller than the prodcut of the edge lengths of the bounding box because here the interstitial area is not included.

4.2.3.2 void FrameSet::boundbox (int *bb[4]*) const

Compute the bounding box.

Parameters

bb	On return, the values with the llx, lly, urx and ury integer coordinates of the maximum rectangular frame. The urx and ury coordinates are (in the usual C/C++ sense) the first integers outside any of the sub-frames.

4.2.3.3 int FrameSet::covers (int *x,* int *y*) const

4.2.3.4 int FrameSet::covers (std::vector< int > & *xedgvec,* std::vector< int > & *yedgvec,* int *xidx,* int *yidx*) const `[private]`

Parameters

xedgvec	
yedgvec	
xidx	
yidx	

Returns

4.2.3.5 void FrameSet::loc (int *idx,* int *widx[3]*) const

Match the index with a subframe number and 2D index in the subframe

Parameters

`in`	*idx*	The flattened index between 0 and the area.
`out`	*widx*	The subframe widx[0], and the x and y coordinate within the subframe in widx[1..2].

4.2.3.6 void FrameSet::nonoverl ( )

Merge the existing frames. The set of frames is split into non-overlapping frames, and these are merged into larger non-overlapping rectangular frames.

4.2.3.7 FrameSet & FrameSet::operator+= (const Frame & *windo*)

Add the area of another rectangular frame to this set.

Parameters

windo	The information on the additional rectangle covered.

4.2.3.8 FrameSet & FrameSet::operator+= (const FrameSet & *wset*)

Add the areas of another set of frames to this . The rectangles defined by the argument are added to this set. The argument may contain partially overlapping areas. The internal representation is remodeled to describe a set of non-overlapping areas.

Parameters

wset	The information on the additional rectangle covered.

4.2.4 Field Documentation

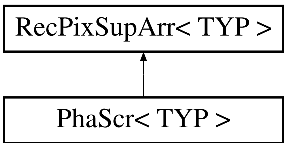
4.2.4.1 std::vector<int> FrameSet::areas

4.2.4.2 std::vector<**Frame**> FrameSet::wd

The subwindows.

4.3 PhaScr< TYP > Class Template Reference

Inheritance diagram for PhaScr< TYP >:



Public Member Functions

- PhaScr (const RecPixSupArr< TYP > &fr, const double pscal)
- void plot (std::ostream &os, const FrameSet &fs)

Data Fields

- double pixscal
- double eval

4.3.1 Detailed Description

template<class TYP>class PhaScr< TYP >

Author

R. J. Mathar

Since

2010-10-31

4.3.2 Constructor & Destructor Documentation

4.3.2.1 template<class TYP > PhaScr< TYP >::PhaScr (const RecPixSupArr< TYP > & *fr*, const double *pscal*)

Ctor with the information on pixel frames and physical size.

Parameters

fr	The arrangement of pixel frames.
pscal	The scale parameter, meters per pixel.

4.3.3 Member Function Documentation

4.3.3.1 template< class TYP > void **PhaScr**< TYP >::plot (std::ostream & *os*, const **FrameSet** & *fs*)

Parameters

os	
fs	

4.3.4 Field Documentation

4.3.4.1 template< class TYP > double **PhaScr**< TYP >::eval

The eigenvalue. More acurately, if the Karhunen-Loeve equation is

$$\int d^2 r' C_\varphi(r,r') K_j(r') = \mathscr{B}^2 K_j(r)$$

for modes K_j, then the value of this variable is the square root of the eigenvalue, that is, $\mathscr{B}$. One may compare results with the values of the scaled λ^2 shown in [**?**] for outputs containing a single stripe by using the area, Fried parameter and Kolomgorov exponent as documented in the publication.

4.3.4.2 template< class TYP > double **PhaScr**< TYP >::pixscal

pixel scale, meters for each pixel's edge

4.4 Point Class Reference

Public Member Functions

- Point ()
- Point (const int xcoo, const int ycoo)

Data Fields

- int x
- int y

4.4.1 Detailed Description

A point with two Cartesian coordinates in the plane.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Point::Point ( )

Default ctor. Point at the origin.

4.4.2.2 Point::Point (const int *xcoo*, const int *ycoo*)

Ctor from two known Cartesian coordinates.

Parameters

xcoo	The x coordinate.
ycoo	The y coordinate.

4.4.3 Field Documentation

4.4.3.1 int Point::x

Abscissa coordinate.

4.4.3.2 int Point::y

Ordinate coordinate.

4.5 RecPixArr< TYP > Class Template Reference

Public Member Functions

- RecPixArr ()
- RecPixArr (int dimx, int dimy, TYP fvalue=0)
- RecPixArr (int dimx, int dimy, std::valarray< TYP > &fvalue)
- template<class TYPV >
  void set (const TYPV *fval)
- TYP at (int x, int y) const
- int area () const
- void clear ()

Data Fields

- int nx
- int ny
- std::valarray< TYP > val

4.5.1 Detailed Description

template<class TYP>class RecPixArr< TYP >

A rectangular pixelized array with a single double value at each pixel. The content of the array is hardly more than a simple 2D array with a retrieve and store method.

Author

R. J. Mathar

Since

2010-10-31

4.5.2 Constructor & Destructor Documentation

4.5.2.1 template<class TYP > export RecPixArr< TYP >::RecPixArr ( )

Default ctor. This is equivalent to creating a field with no pixels.

4.5.2.2 template<class TYP > export RecPixArr< TYP >::RecPixArr (int *dimx*, int *dimy*, TYP *fvalue* = 0)

Ctor of a flat screen with constant value.

Parameters

`in`	*dimx*	The number of pixels in the x direction
`in`	*dimy*	The number of pixels in the y direction
`in`	*fvalue*	A constant value assigned to all pixels.

4.5.2.3 template<class TYP > export RecPixArr< TYP >::RecPixArr (int *dimx*, int *dimy*, std::valarray< TYP > & *fvalue*)

Ctor of a screen with all pixels known.

Parameters

`in`	*dimx*	The number of pixels in the x direction
`in`	*dimy*	The number of pixels in the y direction
`in`	*fvalue*	The vector of pixel values in standard row[0], row[1],..order

4.5.3 Member Function Documentation

4.5.3.1 template<class TYP > int RecPixArr< TYP >::area ( ) const

The integer area in units of pixels squared.

4.5.3.2 template<class TYP > export TYP RecPixArr< TYP >::at (int *x*, int *y*) const

Read the value of a pixel.

Parameters

x	The 0-based index in the x direction
y	The 0-based index in the y direction

Returns

> The current value at the specified address. Request to read a pixel outside the region defined by the array return zero.

4.5.3.3 template<class TYP > void RecPixArr< TYP >::clear ( )

Shrink to a zero-size (no-pixel) array.

4.5.3.4 template<class TYP > template<class TYPV > template void RecPixArr< TYP >::set (const TYPV ∗ *fval*)

Set all pixels.

Parameters

`in`	*fval*	The values that replace the current frame.

4.5.4 Field Documentation

4.5.4.1 template<class TYP > int RecPixArr< TYP >::nx

Number of pixels in the x direction

4.5.4.2 template<class TYP > int RecPixArr< TYP >::ny

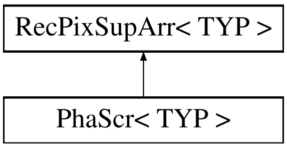
Number of pixels in the y direction

4.5.4.3 template<class TYP > std::valarray<TYP> RecPixArr< TYP >::val

The values at the positions.

4.6 RecPixSupArr< TYP > Class Template Reference

Inheritance diagram for RecPixSupArr< TYP >:



Public Member Functions

- RecPixSupArr ()
- template<class TYPV >
  void set (const TYPV ∗fval, const FrameSet &fs)
- TYP at (int x, int y, const FrameSet &fs) const
- std::valarray< TYP > toValarray (const FrameSet &fs) const

Data Fields

- std::vector< RecPixArr< TYP > > vals

4.6.1 Detailed Description

template<class TYP>class RecPixSupArr< TYP >

A rectangular pixelized array with double values in windowed regions. The class contains rectangular non-overlapping subregions on a commensurate common pixel basis. Values in the interstitial regions between these windows are not stored. The main functionality is an efficient storage scheme for a big rectangular region (super-array) with internal stripe and sub-block geometry. The contents is equivalent to one mode or one phase screen.

Author

R. J. Mathar

Since

2010-10-31

4.6.2 Constructor & Destructor Documentation

4.6.2.1 template<class TYP > **RecPixSupArr< TYP >::RecPixSupArr** ( )

Ctor with no window.

4.6.3 Member Function Documentation

4.6.3.1 template<class TYP > TYP **RecPixSupArr< TYP >::at** (int *x*, int *y*, const **FrameSet** & *fs*) const

Read the value of a pixel.

Parameters

windo	The 0-based subwindow number
x	The 0-based index in the x direction of that subwindow
y	The 0-based index in the y direction of that subwindow

Returns

The current value at the specified address. Requests to read a pixel outside the region defined by the array return zero.

4.6.3.2 template<class TYP > template<class TYPV > template void **RecPixSupArr**< TYP >::set (const TYPV ∗ *fval*, const **FrameSet** & *fs*)

Set all values in all windows.

Parameters

`in`	*fval*	An array with as many values as the area of all subwindows.
`in`	*fs*	The geometry information about the layout of the subwindows.

4.6.3.3 template<class TYP > std::valarray< TYP > **RecPixSupArr**< TYP >::toValarray (const **FrameSet** & *fs*) const

Convert the values of all pixels to a valarray. The representation returned is linearized with successive row[0], row[1] etc

Parameters

`in`	*fs*	The structural information on size and placement of the windows.

Returns

The valarray representation with the pixel values. Pixels in the interstitial regions (outside any windows) are filled with zeros.

4.6.4 Field Documentation

4.6.4.1 template<class TYP> std::vector<**RecPixArr** <TYP> > **RecPixSupArr**< TYP >::vals

The subwindows.

4.7 VKarCorr< TYP > Class Template Reference

Public Member Functions

- VKarCorr (const double rho0, const double ouscale=0., const double kolgamma=2./3.)
- double corr (const double d)
- void genmods (const FrameSet &fs, const double pixscal)
- void genscr (const int Nscr, const int seed, const FrameSet &fs)
- void plot (std::ostream &os, const FrameSet &fs)
- CCfits::FITS ∗ toFITS (const std::string &fname, const FrameSet &fs) const

Data Fields

- double rFried
- double outscal
- std::vector< PhaScr< TYP > > modes
- std::vector< PhaScr< TYP > > screens
- double kolg_exp

Private Attributes

- double Cfact
- double cphi
- double outscalwn

4.7.1 Detailed Description

template<class TYP>class VKarCorr< TYP >

Characterisation of a two dimenisonal turbulence with van-Karman spectrum.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 template<class TYP > VKarCorr< TYP >::VKarCorr (const double *rho0*, const double *ouscale* = `0.`, const double *kolgamma* = `2./3.`)

Ctor with the basic physical characteristics.

Parameters

rho0	The Fried parameter [m]
ouscale	The outer scale length [m] The default is zero, which generates the Kolmogorov spectrum.
kolgamma	The unitless exponent of the power spectrum. The default is 2/3, which yields the Kolmogorov standard slope.

4.7.3 Member Function Documentation

4.7.3.1 template<class TYP > double VKarCorr< TYP >::corr (const double *P*)

Evaluate the correlation function for a distance between two points. The applicable formula of the structure function is [**?** , eq (9)]

$$D_\varphi = -(P/r_0)^{1+\gamma}\frac{1}{\Gamma(-1/2-\gamma/2)(\pi P/L_0)^{1+\gamma}}\left[\Gamma(1/2+\gamma/2)-2(\pi P/L_0)^{1/2+\gamma/2}K_{1/2+\gamma/2}(2\pi P/L_0)\right]$$

with Fried radius r_0, outer scale L_0, Kolmogorov exponent γ. The limit of Kolmogorov turbulence is [**? ?**]

$$D_\varphi = 2c_\varphi(P/r_0)^{1+\gamma}$$

with $g(\gamma)=\sqrt{\pi}\Gamma(-1/2-\gamma/2)/\Gamma(-\gamma/2)$. The value returned is $-1/2$ of this, where sign change and factor mark the transition to the correlation function.

Parameters

P	The distance between the two points in the pupil [m]

Returns

The value of the bivariate phase correlation function at that distance.

4.7.3.2 template<class TYP > void VKarCorr< TYP >::genmods (const FrameSet & *fs*, const double *pixscal*)

Generate the Karhunen-Loeve modes. This populates the array of modes that is part of the object.

Parameters

`in`	*fs*	The parameters of the finite elements (pixels) as a number of frames.
`in`	*pixscal*	The length and width of each pixel in units of meter.

4.7.3.3 template< class TYP > void VKarCorr< TYP >::genscr (const int *Nscr,* const int *seed,* const **FrameSet** & *fs*)

Generate randomized phase screens. This populates the array of screens that is part of the object.

Parameters

`in`	*Nscr*	The number of screens to be generated
`in`	*seed*	The seed of the random number generator. Only the lowest 12 bits are actually used.

4.7.3.4 template< class TYP > void VKarCorr< TYP >::plot (std::ostream & *os,* const **FrameSet** & *fs*)

An ASCII representation of all modes and screens. The Fried parameter and outer scale are printed comment style, i.e., preceded by a `#`, then the modes, then the phase screens. The format is for each mode and each screen a description of parameters in comments (ie, in lines starting with the sharp). Each line is the x and y value of the pixel in meters followed by the phase value of the mode in radians. Note that the modes are not yet multiplied by the (square root of) the eigenvalue. After each pixel row follows an empty lines. Two empty lines separate the modes and screens. This format is exactly the one needed to display the values scanning with gnuplot commands through the data, `gnuplot`

```
gnuplot> set style data lines

gnuplot> set hidden3d

gnuplot> splot "out.asc" index 0

gnuplot> splot "out.asc" index 1

gnuplot> splot "out.asc" index 2

gnuplot> quit
```

Parameters

os	The output stream to be written to.
fs	The frame set description with the geometric paramters.

4.7.3.5 template< class TYP > CCfits::FITS ∗ VKarCorr< TYP >::toFITS (const std::string & *fname,* const **FrameSet** & *fs*) const

Convert the modes and screens to a FITS file. It has been a design choice by the CCfits maintainers for version 2.2 to disallow copy construction on return, so we return a pointer to the local object rather than the object itself. The keywords in the primary header are

- `RFRIED` The Fried radius in units of meter
- `OUTSCA` The outer scale in meter. A negative value means ∞
- `MODES` The number of Karhunen-Lo‘eve modes. Each plane in the FITS cube is one mode. To reach to the phase screens, this number of planes must be skipped.
- `SCREENS` The number of phase screens. This is the number planes in the FITS cube minus the number of modes.
- `KOLGEXP` The Kolmogorov scale exponent.
- `FRLLX` Keywords that start with FR describe the frames in units of pixel coordinates. `LLX`, `LLY`, `URX` and `URY` refer to the lower left x, lower left y, upper right x and upper right y coordinate of a bounding box of a rectangular region.

Other keywords are strictly following the FITS standard and relate the pixel units to meters.

The value reported with each pixel of the modes is already multiplied by the associated eigenvalue. That is, if the eigenvalue equation is

$$\int d^2r' C_\varphi(r,r')K_j(r') = \mathscr{B}_j^2 K_j(r)$$

, the values in the file are $\mathscr{B}_j K_j(r)$ for mode j. (This is a difference to the format of the standard ASCII output.)

Values outside the stripes and not covered by the modes are set to `NULL`.

Parameters

fname	The file name of the FITS file that is to be generated.
fs	The information on the layout of the subwindows.

4.7.4 Field Documentation

4.7.4.1 template<class TYP > double **VKarCorr**< TYP >::Cfact `[private]`

The constant that appears in the relation between the correlation and the factors that depend on outer scale, Fried radius and projected base line.

4.7.4.2 template<class TYP > double **VKarCorr**< TYP >::cphi `[private]`

4.7.4.3 template<class TYP > double **VKarCorr**< TYP >::kolg_exp

The scaling exponent in the power spectrum. This refers to the 3D turbulence, which is 2/3 for the Kolmogorov rate, *not* to the 2D projected case where this becomes a 5/3.

4.7.4.4 template<class TYP > std::vector<**PhaScr**<TYP> > **VKarCorr**< TYP >::modes

The Karhunen Loeve modes.

4.7.4.5 template<class TYP > double **VKarCorr**< TYP >::outscal

The outer scale [m]. If zero, a Kolmogorov power spectrum is used.

4.7.4.6 template<class TYP > double **VKarCorr**< TYP >::outscalwn `[private]`

The outer scale wavenumber, including the factor 2∗pi

4.7.4.7 template<class TYP > double **VKarCorr**< TYP >::rFried

The Fried parameter [m]. The value refers to an unspecified wavelength. It is the one of the instrument, not necessarily the one at 500 nm.

4.7.4.8 template<class TYP > std::vector<**PhaScr**<TYP> > **VKarCorr**< TYP >::screens

Phase screens generated.

5 File Documentation

5.1 Frame.cpp File Reference

Functions

- std::ostream & operator<< (std::ostream &os, const Frame &wd)

5.1.1 Function Documentation

5.1.1.1 std::ostream& operator<< (std::ostream & *os*, const Frame & *wd*)

Display the bounding box of a frame.

Parameters

os	The output stream to print to
wd	The frame to be displayed.

Returns

The updated output stream

5.2 Frame.h File Reference

Data Structures

- class Frame

Functions

- std::ostream & operator<< (std::ostream &os, const Frame &wd)

5.2.1 Function Documentation

5.2.1.1 std::ostream& operator<< (std::ostream & *os*, const Frame & *wd*)

Display the bounding box of a frame.

Parameters

os	The output stream to print to
wd	The frame to be displayed.

Returns

The updated output stream

5.3 FrameSet.cpp File Reference

Functions

- std::ostream & operator<< (std::ostream &os, const FrameSet &fr)

5.3.1 Function Documentation

5.3.1.1 std::ostream& operator<< (std::ostream & *os*, const FrameSet & *fr*)

Write a describtion of a frame set to a stream. The format is (for each of the stripes) a literal f, a 0-based index, and the bracketed bounding box of the subwindow.

Parameters

os	The stream the informatin is written to.
fr	The frame set to be described.

Returns

5.4 FrameSet.h File Reference

Data Structures

- class FrameSet

Functions

- std::ostream & operator<< (std::ostream &os, const FrameSet &fr)

5.4.1 Function Documentation

5.4.1.1 std::ostream& operator<< (std::ostream & *os*, const **FrameSet** & *fr*)

Write a describtion of a frame set to a stream. The format is (for each of the stripes) a literal f, a 0-based index, and the bracketed bounding box of the subwindow.

Parameters

os	The stream the informatin is written to.
fr	The frame set to be described.

Returns

5.5 PhaScr.cpp File Reference

5.6 PhaScr.h File Reference

Data Structures

- class PhaScr< TYP >

5.7 Point.cpp File Reference

5.8 Point.h File Reference

Data Structures

- class Point

5.9 RecPixArr.cpp File Reference

5.10 RecPixArr.h File Reference

Data Structures

- class RecPixArr< TYP >

5.11 RecPixSupArr.cpp File Reference

5.12 RecPixSupArr.h File Reference

Data Structures

- class RecPixSupArr< TYP >

5.13 VKarCorr.cpp File Reference

Macros

- #define M_PI 3.14159265358979323846264338328

Functions

- int dsyev_ (char ∗jobz, char ∗uplo, long int ∗n, double ∗a, long int ∗lda, double ∗w, double ∗work, long int ∗lwork, long int ∗info)
- int dlarnv_ (long int ∗idist, long int ∗iseed, long int ∗n, double ∗x)

5.13.1 Macro Definition Documentation

5.13.1.1 #define M_PI 3.14159265358979323846264338328

5.13.2 Function Documentation

5.13.2.1 int dlarnv_ (long int ∗ *idist,* long int ∗ *iseed,* long int ∗ *n,* double ∗ *x*)

5.13.2.2 int dsyev_ (char ∗ *jobz,* char ∗ *uplo,* long int ∗ *n,* double ∗ *a,* long int ∗ *lda,* double ∗ *w,* double ∗ *work,* long int ∗ *lwork,* long int ∗ *info*)

5.14 VKarCorr.h File Reference

Data Structures

- class VKarCorr< TYP >

5.15 VKarPhase.cpp File Reference

Functions

- int main (int argc, char ∗argv[])

5.15.1 Function Documentation

5.15.1.1 int main (int *argc,* char ∗ *argv[]*)

VKarPhase use and command line options.

```
VKarPhase [-w width] [-l length] [-P pixnum] [-N screenNo] [-r Fried] [-L
outerscl] [-f out.fits] [-s seed] x0 y0 [x1 y1...] > out.asc
```

The comand line options in decreasing order of importance are

- `-w` The width of each individual stripe in meters. The default is 1.8. Note that this the actual coverage will differ from this number because it will be readjusted to a multiple of the pixel scale.

- `-l` The length of each individual stripe in meters. The default is 1.8. Note that this the actual coverage will differ from this number because it will be readjusted to a multiple of the pixel scale.
- `-P` The estimate of the pixel count in each of the screens and modes. The default is 100, which is probably too small for most applications. The time of computation grows with some power of this number, so instead of setting this value to arbitrarily large numbers with no chance to compute in any finite time it is advised to increase this parameter by small factors near 1.5 or 2 to gain exprience of typical run times. The number refers to the total of all pixels (square elements) in the union of the stripes; the interstitial region in between the stripes is not included in this count.
- `-N` The number of phase screens generated. The default is 100.
- `r` The Fried radius in units of meters. The default is 0.15. This is the scaling factor for the seeing and should be set to a value that is reasonable for the observing wavelength of the electromagnetic spectrum.
- `-L` The outer scale radius of the von-Karman spectrum. The default is -1. Negative values trigger the use of the Kolmogorov limit (infinite outer scale).
- `-f` The name of the FITS file to be created. There is no default. Note that existing files will not be overwritten. If this file exists at run time, an associated error message from CCfits ensues.
- `-s` The seed of the pseudo random number generator. The default is 0. To generate additional phase screens after having done this before, the seed must be changed, because this is indeed working with a pseudo random number generator.
- `-g` The scale parameter in the exponent of the 3D structure function. The default is 2/3, the usual Kolmogorov power law.

The command line parameters follow the command line options and are pairs of floating point numbers indicating the lower left coordinates in x and y of the placement of the stripes. The number of stripes is half of the number of these parameters.

If the stripes defined this way overlap by at least one pixel, the number of stripes reported in the FITS files will usually differ, because the program re-distributes the area into non-overlapping rectangular regions to simplify building the covariance matrix.

The standard output is a list of Karhunen-Loeve modes followed by a list of phase screens. The format is for each mode and each screen a description of parameters in comments (ie, in lines starting with the sharp). Each line is the x and y value of the pixel in meters followed by the phase value of the mode in radians. Note that the modes are not yet multiplied by the (square root of) the eigenvalue. After each pixel row follows an empty lines. Two empty lines separate the modes and screens. This format is exactly the one needed to display the values scanning with gnuplot commands through the data, `gnuplot`

```
gnuplot> set style data lines

gnuplot> set hidden3d

gnuplot> splot "out.asc" index 0

gnuplot> splot "out.asc" index 1

gnuplot> splot "out.asc" index 2

gnuplot> quit
```

Generate the modes and store them in the VKarCorr object

Generate the screens and store them in the VKarCorr object

Index