Package ‘velox’

December 1, 2017

Type Package
Title Fast Raster Manipulation and Extraction
Version 0.2.0
Date 2017-11-30
Author Philipp Hunziker
Maintainer Philipp Hunziker <hunzikp@gmail.com>

BugReports https://github.com/hunzikp/velox/issues
Description C++ accelerated raster manipulation and extraction.
Depends R (>= 3.0.2)
License GPL (>= 2)
Imports methods, Rcpp (>= 0.11.6), raster, rgeos, rgdal, sp, sf (>= 0.5-5)
LinkingTo Rcpp, BH (>= 1.65.0-1)
NeedsCompilation yes
SystemRequirements C++11
LazyData TRUE
RoxygenNote 5.0.1
Suggests testthat
Repository CRAN
Date/Publication 2017-12-01 09:54:01 UTC

R topics documented:

bg_intersects ......................................................... 2
boost ............................................................... 3
BoostBoxGrid-class ............................................. 5
BoostFactory ....................................................... 5
BoostGeometries-class ........................................ 5
BoostGrid-class ................................................... 5
**bg_intersects**

Tests whether two BoostObjects intersect (element-wise).

**Description**

Tests whether two BoostObjects intersect (element-wise).
Usage

bg_intersects(obj1, obj2)

## S4 method for signature 'BoostMultiPolygons,BoostBoxGrid'
bg_intersects(obj1, obj2)

## S4 method for signature 'BoostMultiLines,BoostBoxGrid'
bg_intersects(obj1, obj2)

## S4 method for signature 'BoostMultiPolygons,BoostPointGrid'
bg_intersects(obj1, obj2)

## S4 method for signature 'BoostMultiPolygons,BoostGeometries'
bg_intersects(obj1, obj2)

## S4 method for signature 'BoostMultiLines,BoostGeometries'
bg_intersects(obj1, obj2)

## S4 method for signature 'BoostMultiPoints,BoostGeometries'
bg_intersects(obj1, obj2)

Arguments

- obj1: A BoostObject.
- obj2: A BoostObject.

Value

A list, with list element i an integer vector with the indices j for which intersects(x[i],y[j]) is TRUE.

Examples

```r
pts = boost(sf::st_sfc(sf::st_point(c(.5,.5)),
          sf::st_point(c(1.5, 1.5)),
          sf::st_point(c(2.5, 2.5))))
pol = boost(sf::st_sfc(sf::st_polygon(
          list(rbind(c(0,0), c(2,2), c(2,2), c(0,2), c(0,0))))))
bg_intersects(pol, pts)
```

---

**boost**

*Cast a sfc object as a BoostObject*

**Description**

boost creates a BoostObject from a sfc or VeloxRaster object.
Usage

boost(x, ...)

## S3 method for class 'sfc_MULTIPOLYGON'
boost(x, ...)

## S3 method for class 'sfc_POLYGON'
boost(x, ...)

## S3 method for class 'sfc_MULTILINESTRING'
boost(x, ...)

## S3 method for class 'sfc_LINestring'
boost(x, ...)

## S3 method for class 'sfc_MULTIPOINT'
boost(x, ...)

## S3 method for class 'sfc_POINT'
boost(x, ...)

## S3 method for class 'VeloxRaster'
boost(x, box = FALSE, ...)

Arguments

x An sfc object.

... Currently not used.

box Boolean. If TRUE and x is a VeloxRaster object, returns a BoxGrid instead of a PointGrid.

Value

A BoostObject object.

Examples

## Make sfc_POINT
sfc <- sf::st_sfc(sf::st_point(c(0,1)))
## Cast to BoostPoints
boostPoints <- boost(sfc)
**BoostBoxGrid-class**

*An S4 class for storing Boost box grids in C++*

**Description**

This is a class for storing Rcpp pointers to C++ BoxGrid objects.

---

**BoostFactory**

*Rcpp pointer to BoostFactory*

**Description**

Rcpp pointer to BoostFactory.

---

**BoostGeometries-class**

*An S4 virtual class for storing Boost geometry collections in C++*

**Description**

This is a virtual class for storing Rcpp pointers to C++ GeometryCollection objects.

---

**BoostGrid-class**

*An S4 virtual class for storing Boost grids in C++*

**Description**

This is a virtual class for storing Rcpp pointers to C++ grid objects.

---

**BoostMultilines-class**

*An S4 class for storing Boost multiline collections in C++*

**Description**

This is a class for storing Rcpp pointers to C++ MultiLineCollection objects.
BoostMultiPoints-class

An S4 class for storing Boost multipoint collections in C++

Description

This is a class for storing Rcpp pointers to C++ MultiPointCollection objects.

BoostMultiPolygons-class

An S4 class for storing Boost multipolygon collections in C++

Description

This is a class for storing Rcpp pointers to C++ MultiPolygonCollection objects.

BoostObject-class

A S4 class for storing Boost objects in C++

Description

This is a virtual class for storing Rcpp pointers to C++ GeometryCollection and GridCollection objects.

Slots

- geomcollection Rcpp pointer.
- crs An object of class sf::crs, storing the coordinate reference system info.
- precision A numeric scalar.

BoostPointGrid-class

An S4 class for storing Boost point grids in C++

Description

This is a class for storing Rcpp pointers to C++ PointGrid objects.
BoxGrid

**Rcpp pointer to BoxGrid**

**Description**

Rcpp pointer to BoxGrid.

---

**length,BoostGeometries-method**

**BoostGeometries Length**

**Description**

Returns the length (number of Geometries) of a BoostGeometries object.

**Usage**

```r
## S4 method for signature 'BoostGeometries'
length(x)
```

**Arguments**

- `x` A BoostGeometries object.

**Value**

An integer scalar.

---

**MultiLineCollection**

**Rcpp pointer to MultiLineCollection**

**Description**

Rcpp pointer to MultiLineCollection.

---

**MultiPointCollection**

**Rcpp pointer to MultiPointCollection**

**Description**

Rcpp pointer to MultiPointCollection.
MultiPolygonCollection

Rcpp pointer to MultiPolygonCollection

Description

Rcpp pointer to MultiPolygonCollection.

plot,BoostGeometries,ANY-method

Plot BoostGeometries

Description

Plot a BoostGeometries object using the sf plotting function.

Usage

```r
## S4 method for signature 'BoostGeometries,ANY'
plot(x, y, ...)
```

Arguments

- `x` A BoostGeometries object.
- `y` Not used.
- `...` Passed to `sf::plot`.

Value

Void.

PointGrid

Rcpp pointer to PointGrid

Description

Rcpp pointer to PointGrid.
unboost

Cast a BoostGeometries object as a sfc object

Description

unboost creates a sfc object from a BoostGeometries object. Note that all sfc objects created by unboost are of type MULTI.

Usage

```r
unboost(x, ...)    
```

## S3 method for class 'BoostMultiPolygons'
unboost(x, ...)

## S3 method for class 'BoostMultiLines'
unboost(x, ...)

## S3 method for class 'BoostMultiPoints'
unboost(x, ...)

Arguments

- `x` A BoostGeometries object.
- `...` Currently not used.

Value

A sfc object.

Examples

```r
## Make sfc_MULTIPOINT
sfc <- sf::st_sfc(sf::st_multipoint(cbind(0,1)))
## Cast to BoostPoints
boostPoints <- boost(sfc)
## Unboost
sfc2 <- unboost(boostPoints)
print(identical(sfc, sfc2))
```
velox  

Create a VeloxRaster object

Description

velox creates a VeloxRaster object.

Usage

velox(x, extent = NULL, res = NULL, crs = NULL)

Arguments

x  
A Raster* object, matrix, list of matrices, list of VeloxRaster objects, or character string pointing to a GDAL-readable file.

extent  
An extent object or a numeric vector of length 4. Required if x is a matrix or list of matrices, ignored otherwise.

res  
The x and y resolution of the raster as a numeric vector of length 2. Required if x is a matrix or list of matrices, ignored otherwise.

crs  
Optional. A character string describing a projection and datum in the PROJ.4 format. Ignored if x is a Raster* object.

Details

Creates a VeloxRaster object. Note that VeloxRaster objects are Reference Class objects and thus mutable. Hence, the usual R copy on modify semantics do not apply.

Note that if x is a list of VeloxRasters, the extent and crs attributes are copied from the first list element.

Value

A VeloxRaster object.

Examples

## Create VeloxRaster from list of matrices
mat1 <- matrix(1:100, 10, 10)
mat2 <- matrix(100:1, 10, 10)
mat.ls <- list(mat1, mat2)
vx <- velox(mat.ls, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
VeloxRaster-class  

A Reference Class for velox rasters

Description

A Reference Class for velox rasters

Fields

- **rasterbands**: A list of matrices containing the raster data
- **dim**: Raster dimensions
- **extent**: Raster extent
- **res**: Raster resolution
- **nbands**: Number of raster bands
- **crs**: Coordinate reference system (Proj4 string)

Methods

- **aggregate(factor, aggtype = c("sum", "mean", "min", "max", "median"))**: See `VeloxRaster_aggregate`
- **as.matrix(band = 1)**: See `VeloxRaster_as.matrix`
- **as.RasterBrick(assign_data_type = FALSE)**: See `VeloxRaster_as.RasterBrick`
- **as.RasterLayer(band = 1, assign_data_type = FALSE)**: See `VeloxRaster_as.RasterLayer`
- **as.RasterStack(assign_data_type = FALSE)**: See `VeloxRaster_as.RasterStack`
- **col2im(mat, wrow, wcol, band, rowframe = 0, colframe = 0, rowstride = 1, colstride = 1)**: See `VeloxRaster_col2im`
- **crop(x)**: See `VeloxRaster_crop`
- **drop(bands)**: See `VeloxRaster_drop`
- **extract(sp, fun = NULL, df = FALSE, small = FALSE, legacy = FALSE)**: See `VeloxRaster_extract`
- **extract_points(sp)**: See `VeloxRaster_extract_points`
- **getCoordinates()**: See `VeloxRaster_getCoordinates`
- **get_data_type()**: See `VeloxRaster_get_data_type`
- **im2col(wrow, wcol, band, padval = 0, rowframe = 0, colframe = 0, rowstride = 1, colstride = 1)**: See `VeloxRaster_im2col`
- **meanFocal(weights, bands = 1)**: See `VeloxRaster_meanFocal`
- **medianFocal(wrow, wcol, bands = 1)**: See `VeloxRaster_medianFocal`
- **rasterize(spdf, field, band = 1, background = NULL, small = FALSE)**: See `VeloxRaster_rasterize`
- **sumFocal(weights, bands = 1)**: See `VeloxRaster_sumFocal`
- **write(path, overwrite = FALSE)**: See `VeloxRaster_write`
Examples

## Make VeloxRaster objects using the 'velox' function

```r
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
class(vx)
```

### VeloxRaster_aggregate  Aggregate

**Description**

Aggregates a VeloxRaster object to a lower resolution.

**Arguments**

- `factor`: A numeric vector of length 1 or 2 indicating the aggregation factor in the x and y dimensions. Must be positive integers > 1.
- `aggtype`: A character string indicating the aggregation type. See Details.

**Details**

`aggtype` must be one of the following: "sum", "mean", "min", "max", "median".

**Value**

Void.

**Examples**

```r
## Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Aggregate
vx$aggregate(factor=c(2,2), aggtype='sum')
```
VeloxRaster_as.matrix  Cast a VeloxRaster band as a matrix

Description

as.matrix creates a matrix from a VeloxRaster band.

Arguments

band  Integer indicating the VeloxRaster band to be transformed.

Value

A matrix.

Examples

## Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Cast to matrix
vx.mat <- vx$as.matrix(band=1)
identical(mat, vx.mat)

VeloxRaster_as.RasterBrick  Cast a VeloxRaster as a RasterBrick object

Description

as.RasterBrick creates a RasterBrick object from a VeloxRaster.

Arguments

assign_data_type
  Boolean indicating whether the dataType attribute of the returned RasterBrick should be set. If TRUE, the dataType attribute is set to the smallest possible data type.

Value

A RasterBrick object.
## Examples

```r
## Make VeloxRaster with two bands
mat1 <- matrix(1:100, 10, 10)
mat2 <- matrix(100:1, 10, 10)
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1),
          crs="+proj=longlat +datum=WGS84 +no_defs")
## Cast to RasterBrick
library(raster)
rs <- vx$as.RasterBrick()
```

---

### VeloxRaster_as.RasterLayer

*Casts a VeloxRaster band as a RasterLayer object*

#### Description

`as.RasterLayer` creates a RasterLayer object from a VeloxRaster band.

#### Arguments

- **band**
  - Integer indicating the VeloxRaster band to be transformed.

- **assign_data_type**
  - Boolean indicating whether the dataType attribute of the returned RasterLayer should be set. If TRUE, the dataType attribute is set to the smallest possible data type.

#### Value

A RasterLayer object.

#### Examples

```r
## Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Cast to RasterLayer
library(raster)
rl <- vx$as.RasterLayer(band=1)
```
**VeloxRaster_as.RasterStack**

*Cast a VeloxRaster as a RasterStack object*

**Description**

`as.RasterStack` creates a RasterStack object from a VeloxRaster.

**Arguments**

- `assign_data_type`  
  Boolean indicating whether the `dataType` attribute of the returned RasterStack should be set. If TRUE, the `dataType` attribute is set to the smallest possible data type.

**Value**

A RasterStack object.

**Examples**

```r
## Make VeloxRaster with two bands
mat1 <- matrix(1:100, 10, 10)
mat2 <- matrix(100:1, 10, 10)
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1), 
            crs="+proj=longlat +datum=WGS84 +no_defs")
## Cast to RasterStack
library(raster)
rs <- vx$as.RasterStack()
```

**VeloxRaster_col2im**

*col2im*

**Description**

Assigns values to a VeloxRaster band from a matrix of flattened image patches. Patch frames, as specified by `rowframe` and `rowframe`, are not assigned. This function is intended to be used with `mat` matrices constructed with the `im2col` function.

**Arguments**

- `mat`  
  The matrix of flattened image patches.
- `wrow`  
  Patch size in the y dimension.
- `wcol`  
  Patch size in the x dimension.
- `band`  
  The band to be assigned.
rowframe  A non-negative integer specifying the size of the frame around the image patches in the y dimension.

colframe  A non-negative integer specifying the size of the frame around the image patches in the x dimension.

rowstride  A positive integer denoting the stride between extracted patches in the y dimension.

colstride  A positive integer denoting the stride between extracted patches in the x dimension.

Value

Void.

Examples

```r
## Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Apply im2col
dim <- vx$im2col(wrow=2, wcol=2, band=1, padval=0, rowframe=1, colframe=1, rowstride=2, colstride=2)
## Apply col2im
vx$col2im(mat=patch.mat, wrow=2, wcol=2, band=1, rowframe=1, colframe=1, rowstride=2, colstride=2)
isTRUE(all.equal(mat, vx$as.matrix()))
```

---

### VeloxRaster_crop

#### Crop a VeloxRaster object

**Description**

Crops a VeloxRaster object

**Arguments**

- `y`  
  An object from which an extent object can be extracted. Usually a sf, Spatial* or Raster* object.

**Details**

Crops a VeloxRaster object to the extent of `y`.

**Value**

Void.
## VeloxRaster_drop

**Delete a raster band from a VeloxRaster**

### Description
Delete a raster band from a VeloxRaster

### Arguments
- **bands**: Numeric vector containing IDs of bands to be dropped.

### Value
Void.

### Examples
```r
# Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")

# Crop
vx$extent
vx$crop(c(0.3,0.7,0.3,0.7))

# Delete band R
vx$drop(bands=RI)
```

## VeloxRaster_extract

**Extract Values Given Polygons**

### Description
Extracts the values of all cells intersecting with a spatial object (line or polygon) sp and optionally applies R function fun.
Arguments

sp
A sf* POLYGON or MULTIPOLYGON object, a sf* LINE or MULTILINE object, a SpatialPolygons* object, or a SpatialLines* object.

fun
An R function. See Details.

df
Boolean. If TRUE, the return value will be a data frame (or list of data frames, see Details), otherwise a matrix (or list of matrices, see Details). If TRUE, a column ID_sp will be added to each data frame containing the ID of the sp object.

small
Boolean. If TRUE and sp contains polygons, then raster values for small (or oddly shaped) polygons that do not intersect with any cell centroid are established by intersecting the small polygon with the entire (boxed) cells.

legacy
Boolean. Whether to use legacy C++ code (pre velox 0.1.0-9007).

Details

If passed, fun must be an R function accepting a numeric vector as its first (and only mandatory) argument, and returning a scalar. If fun is NULL, extract returns a list of matrices, each matrix containing the raster values intersecting with the respective polygon (but see argument df). If sp contains polygons, then cell-polygon intersections are calculated based on cell centroids (but see argument small). If sp contains lines, then regular cell-line intersections are calculated.

Value

If fun is passed: A numeric matrix or data frame (see argument df) with one row per element in sp, one column per band in the VeloxRaster.

Otherwise: A list of numeric matrices or data frames (see argument df), with one list element per element in sp. Each matrix/data frame consists of one column per band in the VeloxRaster, one row per raster cell intersecting with the geometry.

Examples

```r
## Make VeloxRaster with two bands
set.seed(0)
mat1 <- matrix(rnorm(100), 10, 10)
mat2 <- matrix(rnorm(100), 10, 10)
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1),
  crs="+proj=longlat +datum=WGS84 +no_defs")

## Make SpatialPolygons
library(SpatialPolygons)
library(rgeos)
coord <- cbind(0.5, 0.5)
spoint <- SpatialPoints(coords=coord)
spols <- gBuffer(spgeom=spoint, width=0.5)

## Extract
vx$extract(sp=spols, fun=mean)
```
**VeloxRaster_extract_points**

*Extract Values Given Points*

**Description**

Given a set of points, returns all raster values of the cells with which they intersect.

**Arguments**

- **sp**
  
  A SpatialPoints* object or a sf* POINT object.

**Value**

A numeric matrix. One row per element in `sp`, one column per band in the VeloxRaster.

**Examples**

```r
## Make VeloxRaster with two bands
set.seed(0)
mat1 <- matrix(rnorm(100), 10, 10)
mat2 <- matrix(rnorm(100), 10, 10)
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1),
            crs="+proj=longlat +datum=WGS84 +no_defs")

## Make SpatialPoints
library(sp)
library(rgl)
coord <- cbind(runif(10), runif(10))
spoint <- SpatialPoints(coords=coord)

## Extract
vx$extract_points(sp=spoint)
```

---

**VeloxRaster_getCoordinates**

*Get coordinates*

**Description**

Returns a matrix containing the x-y coordinates of all cell center points of a VeloxRaster.

**Value**

A numeric matrix.
Examples

```r
## Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Get coordinate matrix
pmat <- vx$getCoordinates()
```

---

**VeloxRaster_get_data_type**

*Get data type of a VeloxRaster*

---

**Description**

Get data type of a VeloxRaster

**Details**

Note that this method returns the data type of the raster, not the storage mode. Except in special cases, velox stores all raster data as double precision matrices.

**Value**

A character string denoting a GDAL data type.

---

**VeloxRaster_im2col**

*im2col*

---

**Description**

Creates a matrix of flattened image patches from a VeloxRaster band. Order is left-to-right, top-to-bottom. Note that if any(c(rowframe, colframe)>0), the image patches are (partially) overlapping.

**Arguments**

- `wrow`: Patch size in the y dimension.
- `wcol`: Patch size in the x dimension.
- `band`: The band to be flattened.
- `padval`: A padding value.
- `rowframe`: A non-negative integer specifying the size of the frame around the image patches in the y dimension.
- `colframe`: A non-negative integer specifying the size of the frame around the image patches in the x dimension.
rowstride  A positive integer denoting the stride between extracted patches in the y dimension. I.e. only every rowstride-th patch is extracted.

colstride  A positive integer denoting the stride between extracted patches in the x dimension. I.e. only every colstride-th patch is extracted.

Value

A numeric matrix with \((\text{wrow}+2\times\text{rowframe})\times(\text{wcol}+2\times\text{colframe})\) columns.

Examples

```r
## Make VeloxRaster
mat <- matrix(1:100, 10, 10)
vx <- velox(mat, extent=c(0,1,0,1), res=c(0.1,0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Apply im2col
patch.mat <- vx$im2col(wrow=2, wcol=2, band=1, padval=0,
    rowframe=1, colframe=1, rowstride=2, colstride=2)
dim(patch.mat)
```

---

### VeloxRaster\_meanFocal  \textit{Mean focal}

Description

Applies a mean filter with weights matrix \textit{weights} to a VeloxRaster.

Arguments

- \textbf{weights}  A numeric matrix of weights. Both dimensions must be uneven.
- \textbf{bands}  Numeric vector indicating bands where filter is applied.

Details

Padding is currently not implemented.

Value

\textbf{Void}.

Examples

```r
## Make VeloxRaster with two bands
mat1 <- matrix(1:100, 10, 10)
mat2 <- matrix(100:1, 10, 10)
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1),
    crs="+proj=longlat +datum=WGS84 +no_defs")
## Mean focal
weights <- matrix(1, 5, 5)
```
### VeloxRaster\_medianFocal

**Median focal**

#### Description

Applies a median filter of dimension \( wcol \times wrow \) to a VeloxRaster.

#### Arguments

- **wrow** y dimension of filter. Must be uneven integer.
- **wcol** x dimension of filter. Must be uneven integer.
- **bands** Numeric vector indicating bands where filter is applied.

#### Details

Padding is currently not implemented.

#### Value

Void.

#### Examples

```r
## Make VeloxRaster with two bands
mat1 <- matrix(1:100, 10, 10)
mat2 <- matrix(100:1, 10, 10)
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1),
            crs="+proj=longlat +datum=WGS84 +no_defs")
## Median focal
vx$medianFocal(wrow=5, wcol=5, bands=c(1,2))
```
**VeloxRaster_rasterize**

**Rasterize Polygons or Lines**

**Description**

Rasterizes a collection of polygons or lines, i.e. assigns the values in the `field` column of the SPDF to the raster cells intersecting with the respective geometry.

**Arguments**

- `spdf`: A sf POLYGON or MULTIPOLYGON object, a sf LINE or MULTILINE object, a SpatialPolygonsDataFrame object, or a SpatialLinesDataFrame object.
- `field`: A character string corresponding to the name of a numeric column in `spdf`.
- `band`: A positive integer denoting the ID of the band where the rasterized values are written.
- `background`: Optional. A numeric value assigned to all background cells.
- `small`: Boolean. If TRUE and `spdf` contains polygons, then intersections for small (or oddly shaped) polygons that do not intersect with any cell centroid are established by intersecting the small polygon with the entire (boxed) cells.

**Details**

Note that rasterization is performed sequentially. Hence, cells being contained by multiple polygons are assigned the value of the last polygon in the `spdf` object. If argument `small = TRUE`, then the values of small polygons are assigned last.

If `spdf` contains polygons, then cell-polygon intersections are calculated based on cell centroids (but see argument `small`). If `spdf` contains lines, then regular cell-line intersections are calculated.

**Value**

Void.

**Examples**

```r
## Make VeloxRaster
mat <- matrix(0, 10, 10)
vx <- velox(mat, extent=c(0, 1, 0, 1), res=c(0.1, 0.1), crs="+proj=longlat +datum=WGS84 +no_defs")
## Make SpatialPolygonsDataFrame
library(sp)
library(rgeos)
coord <- cbind(0.5, 0.5)
spoint <- SpatialPoints(coords=coord)
spols <- gBuffer(spgeom=spoint, width=0.25)
spdf <- SpatialPolygonsDataFrame(Sr=spols, data=data.frame(value=1), match.ID=FALSE)
## Rasterize, set background to -1
vx$rasterize(spdf=spdf, field="value", background=-1)
```
**VeloxRaster_sumFocal**  
*Sum focal*

**Description**  
Applies a focal sum with weights matrix `weights` to a VeloxRaster.

**Arguments**  
- `weights`: A numeric matrix of weights. Both dimensions must be uneven.  
- `bands`: Numeric vector indicating bands where filter is applied.

**Details**  
Padding is currently not implemented.

**Value**  
Void.

**Examples**  
```r  
# Make VeloxRaster with two bands  
mat1 <- matrix(1:100, 10, 10)  
mat2 <- matrix(100:1, 10, 10)  
vx <- velox(list(mat1, mat2), extent=c(0,1,0,1), res=c(0.1,0.1),  
  crs="+proj=longlat +datum=WGS84 +no_defs")  
# Sum focal  
weights <- matrix(1, 5, 5)  
vx$sumFocal(weights=weights, bands=c(1,2))
```

---

**VeloxRaster_write**  
*Write a VeloxRaster to disk as a GeoTiff file*

**Description**  
Write a VeloxRaster to disk as a GeoTiff file

**Arguments**  
- `path`: Output filename as character string.  
- `overwrite`: Boolean indicating whether target file should be overwritten.

**Value**  
Void.
Description

Extract a subset of geometries from a BoostGeometries object.

Usage

```r
## S4 method for signature 'BoostMultiPolygons,ANY,ANY'
x[i]
## S4 method for signature 'BoostMultiLines,ANY,ANY'
x[i]
## S4 method for signature 'BoostMultiPoints,ANY,ANY'
x[i]
```

Arguments

- `x` A BoostGeometries object.
- `i` An integer vector index.

Value

A BoostGeometries object.
Index

[, BoostMultilines, ANY, ANY-method, 25
  ([, BoostMultiPolygons, ANY, ANY-method, 25
    [, BoostMultiPoints, ANY, ANY-method, 25
      ([, BoostMultiPolygons, ANY, ANY-method, 25
        bg_intersects, 2
        bg_intersects, BoostMultilines, BoostBoxGrid-method
          (bg_intersects), 2
        bg_intersects, BoostMultilines, BoostGeometries-method
          (bg_intersects), 2
        bg_intersects, BoostMultiPolygons, BoostBoxGrid-method
          (bg_intersects), 2
        bg_intersects, BoostMultiPolygons, BoostGeometries-method
          (bg_intersects), 2
        bg_intersects, BoostMultiPolygons, BoostPointGrid-method
          (bg_intersects), 2
        boost, 3
        BoostBoxGrid-class, 5
        BoostFactory, 5
        BoostGeometries-class, 5
        BoostGrid-class, 5
        BoostMultilines-class, 5
        BoostMultiPoints-class, 6
        BoostMultiPolygons-class, 6
        BoostObject-class, 6
        BoostPointGrid-class, 6
        BoxGrid, 7
        length, BoostGeometries-method, 7
        MultiLineCollection, 7
        MultiPointCollection, 7
        MultiPolygonCollection, 8
        plot, BoostGeometries, ANY-method, 8

PointGrid, 8
unboost, 9
velox, 10
VeloxRaster (VeloxRaster-class), 11
VeloxRaster-class, 11
VeloxRaster_aggregate, 11, 12
VeloxRaster_as_matrix, 11, 13
VeloxRaster_asRasterBrick, 11, 13
VeloxRaster_asRasterLayer, 11, 14
VeloxRaster_asRasterStack, 11, 15
veloxraster_col2im, 11, 15
veloxraster_crop, 11, 16
veloxraster_drop, 11, 17
veloxraster_extract, 11, 17
veloxraster_extract_points, 11, 19
veloxraster_getCoordinates, 11, 19
veloxraster_getdata_type, 11, 20
veloxraster_im2col, 11, 20
veloxraster_medianFocal, 11, 21
veloxraster_rasterize, 11, 22
veloxraster_rasterize, 11, 23
veloxraster_sumFocal, 11, 24
veloxraster_write, 11, 24

26