Package: openvdbr (via r-universe)

November 16, 2026

Type Package

Title Sparse Volumes with an 'OpenVDB' Runtime Interface

Version 13.1.0-2

Description Creates, edits, and samples sparse volumetric grids using 'OpenVDB' <https://www.openvdb.org/>. Supports signed distance fields, constructive solid geometry, conversion between triangle meshes and volumes, file exchange, and a native runtime interface.

License Apache License (== 2.0)

Copyright file inst/COPYRIGHTS

Depends R (>= 4.3.0)

Imports Rcpp

LinkingTo Rcpp

Suggests testthat (>= 3.2.3)

SystemRequirements GNU make, CMake (>= 3.24), zlib

Encoding UTF-8

NeedsCompilation yes

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make zlib1g-dev

Repository https://tylermorganwall.r-universe.dev

Date/Publication 2026-11-14 15:04:01 UTC

RemoteUrl https://github.com/tylermorganwall/openvdbr

RemoteRef HEAD

RemoteSha c612775db9be88275f95c5f228c115b4f5d2255e

Contents

vdb_clone 2
vdb_csg 3
vdb_fog 3
vdb_from_mesh 4
vdb_get 5
vdb_grid 5
vdb_info 6
vdb_messages 7
vdb_read 7
vdb_sample 8
vdb_set 9
vdb_sphere 10
vdb_to_mesh 11
vdb_version 11
vdb_write 12

Index **13**

`vdb_clone` *Copy or release a native grid*

Description

Copy or release a native grid

Usage

```
vdb_clone(grid)

vdb_close(grid)
```

Arguments

`grid`	An `openvdbr_grid` handle.

Details

Closing a handle also closes every R alias of that handle. The handle remains identifiable, but further operations report that it is closed.

Value

`vdb_clone()` returns an independent deep copy. `vdb_close()` releases the native storage and invisibly returns `NULL`; repeated closes are safe.

Examples

```
original = vdb_grid()
copy = vdb_clone(original)
vdb_set(copy, c(0, 0, 0), 1)
vdb_get(original, c(0, 0, 0))
vdb_close(copy)
```

---

vdb_csg *Combine level-set surfaces*

Description

Combine level-set surfaces

Usage

```
vdb_csg(a, b, operation = c("union", "intersection", "difference"))
```

Arguments

a	First float-valued level-set grid.
b	Second float-valued level-set grid with the same transform and band width.
operation	Default "union". One of "union", "intersection" or "difference" (subtract b from a).

Details

These are OpenVDB's level-set CSG operations. They combine signed distances; they do not resample mismatched grids or rebuild distance bands.

Value

A new level-set grid. Both inputs remain unchanged.

---

vdb_fog *Convert a signed distance field into a fog volume*

Description

Convert a signed distance field into a fog volume

Usage

```
vdb_fog(grid)
```

Arguments

grid	A float-valued level-set grid.

Details

This produces a bounded density field suitable for volumetric rendering. It retains the source transform and name and leaves the input intact.

Value

A new fog-volume grid with density zero outside and density approaching one inside. The transition spans the input's interior narrow band.

---

vdb_from_mesh *Convert a triangle mesh into a sparse level set*

Description

Convert a triangle mesh into a sparse level set

Usage

```
vdb_from_mesh(
  vertices,
  triangles,
  voxel_size = 0.1,
  half_width = 3,
  name = "surface"
)
```

Arguments

vertices	Numeric matrix with three columns of world-space vertex coordinates.
triangles	Integer matrix with three columns of one-based vertex indices.
voxel_size	Default `0.1`. Positive voxel width in world units.
half_width	Default `3`. Narrow-band half width in voxels, at least one.
name	Default `"surface"`. Name of the resulting grid.

Details

Use a closed, consistently oriented surface for meaningful interior signs. OpenVDB accepts world-space vertices at float precision; this function does not repair nonmanifold or self-intersecting geometry.

Value

A float-valued level-set grid.

Examples

```
vertices = rbind(c(-1, -1, -1), c(1, -1, -1), c(0, 1, -1), c(0, 0, 1))
triangles = rbind(c(1, 3, 2), c(1, 2, 4), c(2, 3, 4), c(3, 1, 4))
pyramid = vdb_from_mesh(vertices, triangles, voxel_size = 0.2)
vdb_sample(pyramid, c(0, 0, 0))
```

---

vdb_get *Read exact voxel values and activation flags*

Description

Read exact voxel values and activation flags

Usage

```
vdb_get(grid, xyz)
```

Arguments

grid	An openvdbr_grid handle with float or double values.
xyz	A coordinate triple or three-column matrix of zero-based integer voxel indices. Negative indices are allowed.

Value

A list containing numeric value and logical active vectors.

---

vdb_grid *Create a sparse scalar volume*

Description

Store values only where they differ from a shared background. This is useful for smoke density, temperature fields, signed distances and other volumes whose bounding box would waste memory as a dense R array.

Usage

```
vdb_grid(
  background = 0,
  voxel_size = 1,
  origin = c(0, 0, 0),
  name = "density",
  type = c("float", "double")
)
```

Arguments

background	Default 0. Value returned outside stored voxels.
voxel_size	Default 1. Positive world-space width of a voxel.
origin	Default c(0, 0, 0). World-space position of index (0, 0, 0).
name	Default "density". Grid name used in VDB files.
type	Default "float". Storage precision: "float" or "double".

Details

Voxel indices are zero-based spatial coordinates and can be negative. Handles have reference semantics: assigning a handle to another variable shares the same grid. Use vdb_clone() for an independent copy. Handles cannot be saved with saveRDS(); use vdb_write() to persist the data.

Value

An openvdbr_grid external pointer, released automatically by garbage collection or explicitly by vdb_close().

Examples

```
cloud = vdb_grid(voxel_size = 0.5)
vdb_set(cloud, rbind(c(-1, 0, 0), c(0, 0, 0), c(1, 0, 0)), c(0.2, 1, 0.2))
vdb_sample(cloud, c(0.25, 0, 0))
vdb_info(cloud)
```

---

vdb_info *Inspect a sparse grid*

Description

Inspect a sparse grid

Usage

```
vdb_info(grid)
```

Arguments

grid	An openvdbr_grid handle.

Value

A list with the grid name, value type, class, active voxel count, estimated native memory in bytes, inclusive index bounds, voxel widths, world-space origin and scalar background. Empty grids have zero bounds; check empty before using them. Counts are doubles to support large grids. Non-scalar backgrounds are reported as NaN.

vdb_messages *Retrieve buffered library diagnostics*

Description

Collect warnings and informational diagnostics from the bundled libraries, including messages produced on worker threads. Operation failures still raise ordinary R errors (or return a status to native consumers).

Usage

```
vdb_messages()
```

Details

The queue is global to this provider. It retains up to 128 messages of 511 bytes each, then reports a count of dropped messages. This bounds memory on error paths and avoids console calls from worker threads. Native consumers can drain the same queue with `diagnostic_next()`.

Value

A character vector of pending messages, removed from the queue.

vdb_read *List or read grids in a VDB file*

Description

VDB files can contain several named sparse volumes, each with its own value type, metadata and spatial transform. Inspect their names before selecting a density, temperature or signed-distance grid.

Usage

```
vdb_read(filename, grid = 1)

vdb_grid_names(filename)
```

Arguments

`filename`	Path to a VDB file.
`grid`	Default 1. Grid name or one-based position in `vdb_grid_names()`.

Details

The bundled OpenVDB implementation includes Blosc and zlib compression. All upstream registered grid types can be read and written without conversion. Editing, scalar sampling and meshing currently support float and double scalar grids; vector and point grids can be inspected and passed through.

Value

`vdb_grid_names()` returns character names without loading voxel data. `vdb_read()` returns a native grid handle. Its selected grid is fully loaded and independent of the source file, which can subsequently be moved or removed.

Examples

```
file = tempfile(fileext = '.vdb')
shape = vdb_sphere(radius = 1, voxel_size = 0.2, name = 'surface')
density = vdb_fog(shape)
vdb_write(shape, file)
vdb_grid_names(file)
restored = vdb_read(file, 'surface')
vdb_sample(restored, c(0, 0, 0))
unlink(file)
```

---

vdb_sample *Sample a sparse volume at arbitrary positions*

Description

Sample a sparse volume at arbitrary positions

Usage

```
vdb_sample(
  grid,
  xyz,
  space = c("world", "index"),
  interpolation = c("linear", "nearest")
)
```

Arguments

`grid`	An `openvdbr_grid` handle with float or double values.
`xyz`	A coordinate triple or three-column numeric matrix.
`space`	Default `"world"`. Use `"world"` coordinates transformed by the grid, or continuous `"index"` coordinates.
`interpolation`	Default `"linear"`. Trilinear interpolation or `"nearest"`.

Details

Sampling includes stored inactive values, which are significant for signed distance fields. Trilinear interpolation uses eight neighboring voxels. The complete transform of an imported grid is respected, including rotations.

Value

A numeric vector with one sampled value per row.

`vdb_set` *Edit sparse voxel values*

Description

Edit sparse voxel values

Usage

```
vdb_set(grid, xyz, values, active = TRUE)
```

Arguments

`grid`	An `openvdbr_grid` handle with float or double values.
`xyz`	A numeric vector of length three or a matrix with three columns. Each row is a zero-based integer coordinate `(x, y, z)` in index space.
`values`	Finite voxel values, one per coordinate or a single value to repeat.
`active`	Default `TRUE`. Logical activation flags, one per coordinate or a single flag to repeat.

Details

Inactive voxels can retain values different from the background. Level sets use this to represent their negative interiors compactly. Duplicate coordinates are applied in input order, so the last value wins. Validation occurs before editing; allocation failures may leave a partial edit.

Value

The input handle, invisibly, after modifying it in place.

vdb_sphere *Create a sphere as a signed distance field*

Description

A level set stores distance to a surface in a narrow band. Values are negative inside and positive outside, with distant values clamped to the band width. This supports smooth surface operations without allocating a dense volume.

Usage

```
vdb_sphere(
  radius = 1,
  center = c(0, 0, 0),
  voxel_size = 0.1,
  half_width = 3,
  name = "surface"
)
```

Arguments

radius	Default 1. Positive sphere radius in world units.
center	Default c(0, 0, 0). Sphere center in world coordinates.
voxel_size	Default 0.1. Positive voxel width in world units.
half_width	Default 3. Narrow-band half width in voxels, at least one.
name	Default "surface". Name of the resulting grid.

Value

A float-valued level-set grid.

Examples

```
# A snowman silhouette made from two sparse level sets.
body = vdb_sphere(1, voxel_size = 0.15)
head = vdb_sphere(0.6, center = c(0, 1.2, 0), voxel_size = 0.15)
snowman = vdb_csg(body, head)
axis = seq(-1.5, 2, length.out = 60)
points = as.matrix(expand.grid(x = axis, y = axis, z = 0))
section = matrix(vdb_sample(snowman, points), length(axis))
graphics::contour(axis, axis, section, levels = 0, asp = 1, drawlabels = FALSE)
```

vdb_to_mesh *Extract a triangle mesh from a scalar volume*

Description

Extract a triangle mesh from a scalar volume

Usage

```
vdb_to_mesh(grid, isovalue = 0, adaptivity = 0)
```

Arguments

`grid`	A float or double scalar grid.
`isovalue`	Default `0`. Surface value; zero for a level set, often `0.5` for fog.
`adaptivity`	Default `0`. Surface simplification between zero and one.

Value

A list of `vertices` (three-column world-space numeric matrix) and `triangles` (three-column, one-based integer indices). OpenVDB quads are triangulated with their original winding preserved.

Examples

```
surface = vdb_to_mesh(vdb_sphere(voxel_size = 0.2), adaptivity = 0.2)
graphics::plot(surface$vertices[, c(1, 2)], asp = 1, pch = '.',
               xlab = 'x', ylab = 'y')
```

vdb_version *OpenVDB runtime version and native interface*

Description

OpenVDB runtime version and native interface

Usage

```
vdb_version()

vdb_api()
```

Details

Native consumers can use `LinkingTo: openvdbr` for the small C headers and `Imports: openvdbr` to ensure the provider is loaded. On R's main thread, call `openvdbr_api_get_v1()` from `openvdbr/openvdbr_r.h`. It retrieves `openvdbr_get_api_v1` using `R_GetCCallable()`. No OpenVDB linker flags or OpenVDB C++ headers are needed by the consumer.

The table uses opaque handles and caller-owned error buffers. Read-only sampling calls can run concurrently with separate output buffers. Mutating or closing a grid must not overlap any access to that grid. File handles require exclusive access. Runtime lookup and all R API calls must remain on R's main thread. Explicitly unloading the provider DLL while handles or consumers remain alive is unsupported.

Value

`vdb_version()` returns the bundled OpenVDB version. `vdb_api()` returns an external pointer to the version-one C function table, retaining the package namespace for the lifetime of the handle.

`vdb_write` *Write sparse grids to a VDB file*

Description

Write sparse grids to a VDB file

Usage

```
vdb_write(grids, filename, overwrite = FALSE)
```

Arguments

`grids`	A native grid or a list of native grids. Each grid retains its own name, value type, metadata and transform.
`filename`	Output path, including a `.vdb` extension if desired.
`overwrite`	Default `FALSE`. Allow replacing an existing file.

Value

The output filename, invisibly.

Index

vdb_api *(vdb_version)*, 11
vdb_clone, 2
vdb_clone(), *6*
vdb_close *(vdb_clone)*, 2
vdb_close(), *6*
vdb_csg, 3
vdb_fog, 3
vdb_from_mesh, 4
vdb_get, 5
vdb_grid, 5
vdb_grid_names *(vdb_read)*, 7
vdb_grid_names(), *7*
vdb_info, 6
vdb_messages, 7
vdb_read, 7
vdb_sample, 8
vdb_set, 9
vdb_sphere, 10
vdb_to_mesh, 11
vdb_version, 11
vdb_write, 12
vdb_write(), *6*