

Package: trowel (via r-universe)

October 15, 2026

Title A Live Constrained Delaunay Mesh

Version 0.0.0.9000

Description A persistent, editable constrained Delaunay triangulation: add and remove points and segments at any time, refine toward quality targets, and read the mesh back as plain tables (vertices, triangles with constraint depth, segments). Built on the 'spade' Rust crate via extendr. The incremental sibling of 'cdtr'.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements Cargo (Rust's package manager), rustc

Config/rextendr/version 0.4.0

Suggests cdtr, testthat (>= 3.0.0)

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL <https://github.com/hypertidy/trowel>

BugReports <https://github.com/hypertidy/trowel/issues>

Config/pak/sysreqs libclang-dev

Repository <https://hypertidy.r-universe.dev>

Date/Publication 2026-10-01 19:09:10 UTC

RemoteUrl <https://github.com/hypertidy/trowel>

RemoteRef HEAD

RemoteSha 32e0fe8329de42c5bf61656a114bd9040d6b49e7

Contents

mesh_edit	2
mesh_new	3

mesh_read	3
mesh_refine	4
mesh_T	5
Trowel	6
Index	7

mesh_edit	<i>Edit a live mesh</i>
-----------	-------------------------

Description

Points are added with [mesh_add_points\(\)](#) (returning stable vertex ids; a point at an existing vertex returns that vertex's id). Segments join two vertex ids; a segment that crosses an existing constraint has the crossing inserted as a new vertex, with attributes interpolated along the new segment. Removing a point removes any segment through it; removing a segment clears its constraint edges and leaves the vertices.

Usage

```
mesh_add_points(m, x, y, PA = NULL)

mesh_add_segments(m, a, b)

mesh_remove_points(m, ids)

mesh_remove_segments(m, seg_ids)
```

Arguments

m	a mesh from mesh_new()
x, y	coordinates of points to add
PA	attribute matrix for the new points (one row per point), or NULL
a, b	vertex ids of segment endpoints
ids	vertex ids to remove
seg_ids	segment ids to remove

Value

vertex ids for `mesh_add_points`, segment ids for `mesh_add_segments`, the mesh invisibly for the removals

mesh_new	Create a live mesh
----------	--------------------

Description

A persistent constrained Delaunay triangulation that accepts edits at any time. Start empty or from the usual three tables: vertices, segments as index pairs, optional per-vertex attributes.

Usage

```
mesh_new(x = NULL, y = NULL, s0 = NULL, s1 = NULL, PA = NULL)
```

Arguments

x, y	vertex coordinates (may be omitted for an empty mesh)
s0, s1	1-based indices into the vertices for constraint segments
PA	numeric matrix of per-vertex attributes (one row per vertex), or NULL; column names become attribute names

Value

a trowel handle (a mutable object: functions that edit it return it invisibly)

Examples

```
x <- c(0, 1, 1, 0, 0.25, 0.75, 0.75, 0.25)
y <- c(0, 0, 1, 1, 0.25, 0.25, 0.75, 0.75)
m <- mesh_new(x, y, 1:8, c(2, 3, 4, 1, 6, 7, 8, 5))
mesh_counts(m)
table(mesh_triangles(m)$depth)
```

mesh_read	Read a live mesh as tables
-----------	----------------------------

Description

Read a live mesh as tables

Usage

```

mesh_vertices(m)

mesh_triangles(m)

mesh_segments(m)

mesh_constraint_edges(m)

mesh_edges(m)

mesh_counts(m)

```

Arguments

`m` a mesh

Value

data frames: `mesh_vertices` (id, x, y, origin, attribute columns; origin is "input", "crossing" or "steiner"), `mesh_triangles` (v0, v1, v2 as vertex ids, depth = constraint crossings from outside), `mesh_segments` (segment, position, vertex: the chain of vertex ids along each live segment), `mesh_constraint_edges` and `mesh_edges` (a, b vertex ids); `mesh_counts` is a list of sizes.

mesh_refine	<i>Refine a live mesh</i>
-------------	---------------------------

Description

Delaunay refinement toward a minimum angle and/or a maximum triangle area, inserting Steiner vertices. Attributes on new vertices are interpolated linearly from the mesh as it was before refinement. Can be called again after further edits.

Usage

```

mesh_refine(
  m,
  min_angle = NULL,
  max_area = NULL,
  max_steiner = Inf,
  keep_constraints = FALSE,
  inner_only = TRUE
)

```

Arguments

m	a mesh
min_angle	minimum angle in degrees (NULL for none; spade caps at 30)
max_area	maximum triangle area (NULL for none)
max_steiner	budget of Steiner vertices (Inf for unlimited)
keep_constraints	if TRUE constraint edges are never split
inner_only	if TRUE (default) the region outside every constraint (depth 0) is left as it was: Steiner vertices that would land there are dropped (vertices on boundary constraint edges stay, since they belong to both sides). Holes count as inside; filter by depth parity yourself if a hole should not be refined. FALSE refines every triangle in the hull.

Value

invisibly, a list with complete (did refinement finish within budget), added (vertex count added) and new_ids

mesh_T	<i>Triangle vertex index matrix</i>
--------	-------------------------------------

Description

Convenience for plotting and for handing to code that wants 1-based row indices into [mesh_vertices\(\)](#) rather than ids.

Usage

mesh_T(m)

Arguments

m	a mesh
---	--------

Value

integer matrix, 3 columns, rows index into mesh_vertices(m)

Trowel

The live mesh.

Description

The live mesh.

Usage

Trowel

Format

An object of class environment of length 12.

Methods

Method new: Create an empty mesh, with optional attribute column names.

Method add_points: Add points; returns their ids (an existing vertex at the same position is reused).

Method add_segments: Add constraint segments between existing vertex ids; crossings with existing constraints are resolved by inserting the crossing point. Returns segment ids.

Method remove_points: Remove vertices by id. Segments through them are removed too.

Method remove_segments: Remove segments by id (constraint flags cleared; vertices stay).

Method refine: Refine the mesh. Any argument may be NA to skip it.

Method vertices: Vertices: id, x, y, origin (0 input, 1 crossing, 2 steiner), attributes.

Method triangles: Triangles as vertex ids (3 columns) plus depth.

Method constraint_edges: Constraint edges currently in the mesh, as vertex id pairs.

Method edges: All edges as vertex id pairs.

Method segments: User segments: segment id and the chain of vertex ids along it.

Index

* datasets

Trowel, [6](#)

`mesh_add_points (mesh_edit)`, [2](#)
`mesh_add_points()`, [2](#)
`mesh_add_segments (mesh_edit)`, [2](#)
`mesh_constraint_edges (mesh_read)`, [3](#)
`mesh_counts (mesh_read)`, [3](#)
`mesh_edges (mesh_read)`, [3](#)
`mesh_edit`, [2](#)
`mesh_new`, [3](#)
`mesh_new()`, [2](#)
`mesh_read`, [3](#)
`mesh_refine`, [4](#)
`mesh_remove_points (mesh_edit)`, [2](#)
`mesh_remove_segments (mesh_edit)`, [2](#)
`mesh_segments (mesh_read)`, [3](#)
`mesh_T`, [5](#)
`mesh_triangles (mesh_read)`, [3](#)
`mesh_vertices (mesh_read)`, [3](#)
`mesh_vertices()`, [5](#)

Trowel, [6](#)