Verifying a vertical cell decomposition algorithm

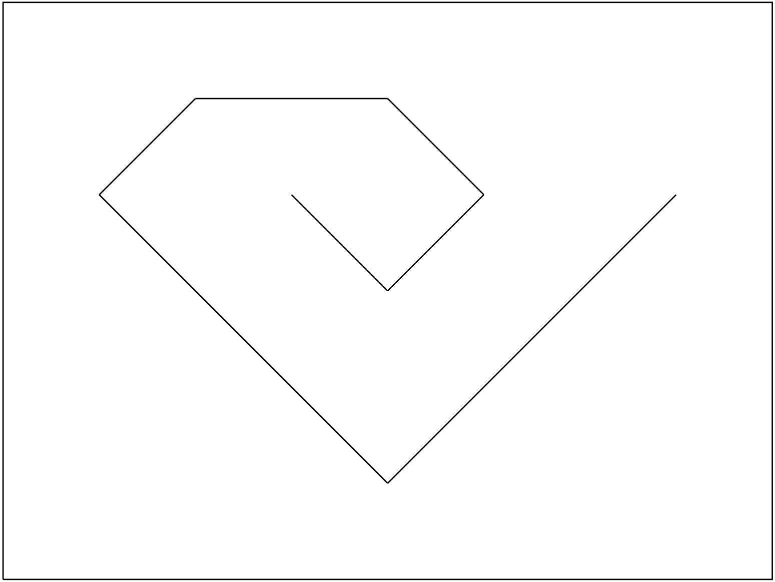
Yves Bertot
Thomas Portet

September 2025

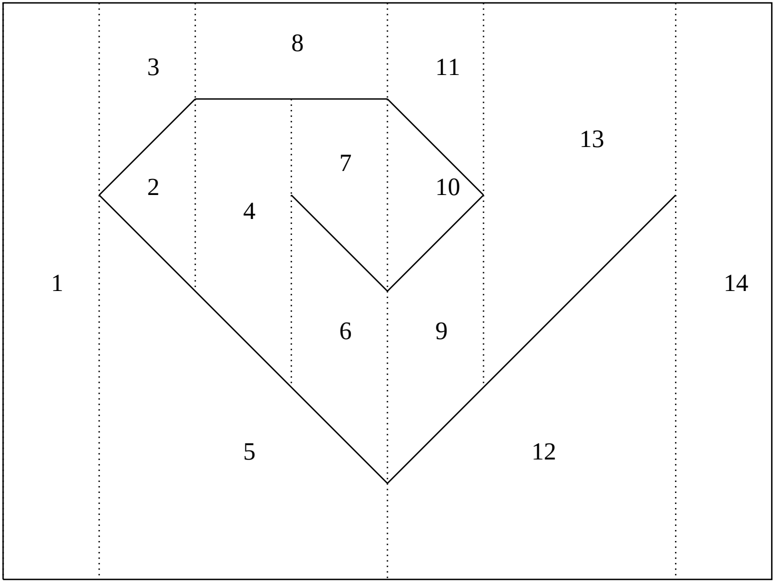
The overall picture

- Find a path between obstacles
- Obstacles are described by straight line segments
- Decompose the working area into simple cells
    - Each cell is safe
    - Each cell is convex
    - Each cell is non-empty
- Moving from cells to neighbors is safe
    - Cells have doors

Example



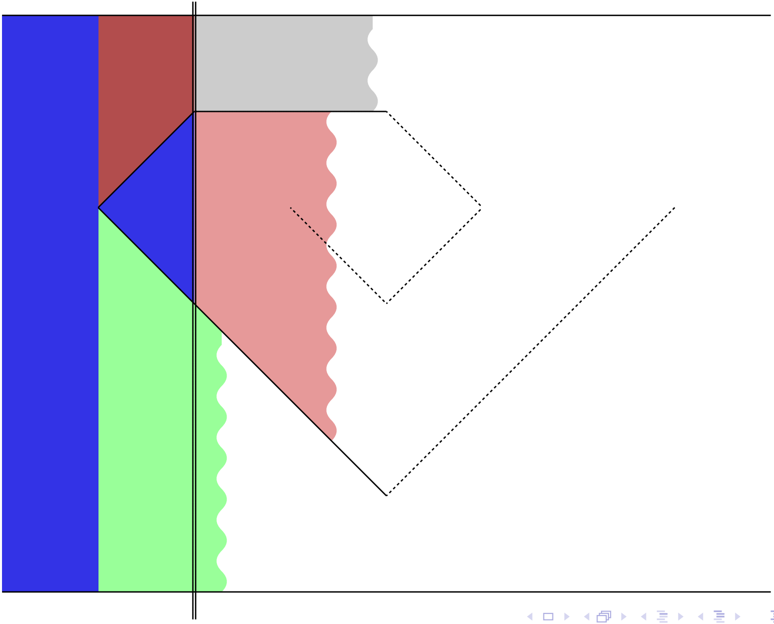
Example : results



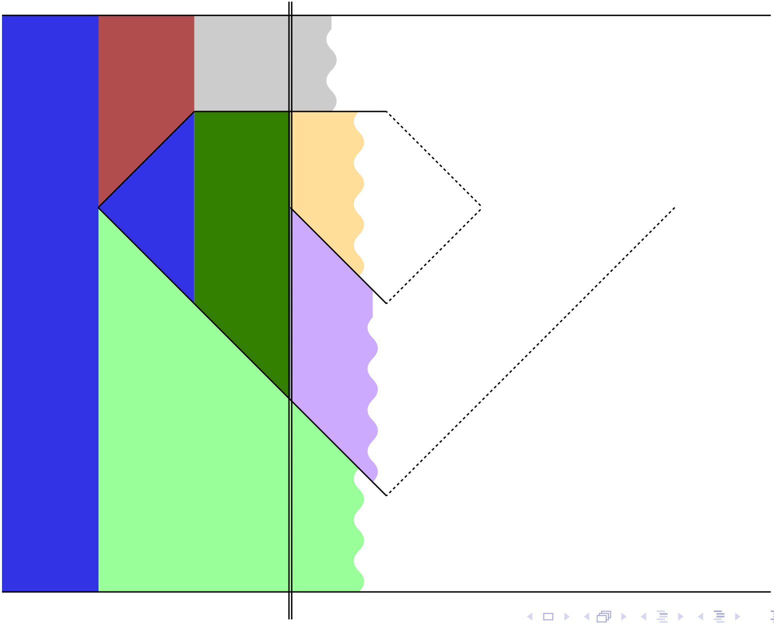
Vertical cell decomposition

- Use a vertical sweep line moving left to right
- Stop each time one meets an event (e.g. an edge tip)
- maintain a vertically ordered sequence of incomplete cells
  - Complete all incomplete cells in contact with the event
  - Create new incomplete cells for edges starting at this event
- Simplifying assumptions
  - No vertical edges
  - Edges do not cross

Intermediate position for vertical cell decomposition (1)



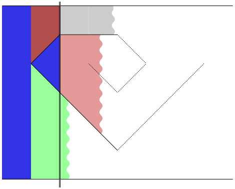
Intermediate position for vertical cell decomposition (2)

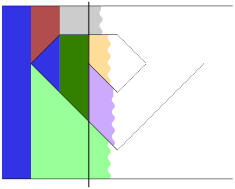


Naive approach to cell generation

- Maintain a sequence of incomplete cells
    - In code and the article, they are called “open cells”
- Compute incomplete cells in contact with the current event
- Complete these cells
- Create new incomplete cells starting at the current event

Illustration



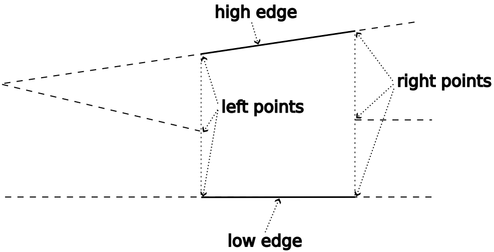


- Event in the middle of the pink area
- Incomplete cells are green, pink, grey (ascending order)
- Contact cell: the pink cell
- New complete cell: complete the pink cell at the event, obtain a dark green cell in the middle
- New incomplete cells: light purple and yellow

Difficulty with vertically aligned events

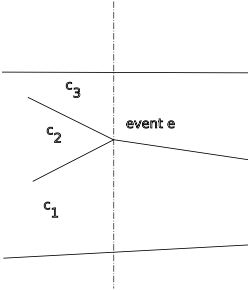
- Width of complete cells : horizontal distance between events
  - Vertically aligned events yield empty cells, if handled naively
- Empty cells are a nuisance
- Solution: special treatment
  - Keep track of last created incomplete and complete cells
  - Update these cells instead of creating new ones

Well-formed cells



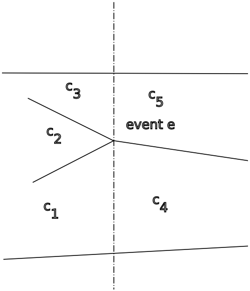
- All cells have a high, a low edge, and a left point sequence
- complete cells have a right point sequence
- Point sequences go from the high edge to the low edge
- Points sequences describe all (known) unsafe points

Non-vertically aligned events



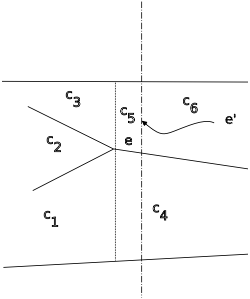
- The lower edge of the lower contact cell finishes further right
- Also for the higher edge of the higher
- All other contact have low and high edge meeting at the event
- The event may have outgoing edges
- Before processing event e, cells c_1, c_2, and c_3 are *incomplete*
- When processing event e, these cells receive a right side at the sweep line

Non-vertically aligned events: new incomplete cells



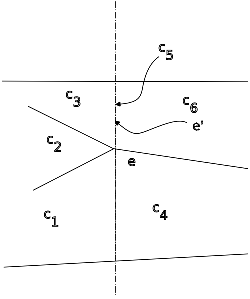
- Two new incomplete cells are created (one outgoing edge)
- Their left side is at the sweep line

Non-vertically aligned events: next event



- Cell c_5 is completed when processing e'
- e' is safe for c_3

Vertically aligned events



- Cell c_5 is completed when processing e'
- There is no need for c_5
- e' must be recorded as unsafe in c_3
- Other unsafe points on the left side of c_5 must be recorded as unsafe for c_6

Basic concepts

- Edges: pairs of points with strict order on first coordinate
- Points above, under, or on edges
- Valid edges for a point
- Edges below edges
- Well-formed cells
- Adjacent cells

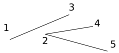
Above or under

- Edge with extremities l and r and an arbitrary point p
- $\begin{vmatrix} 1 & \mathtt{l}_x & \mathtt{l}_y \\ 1 & \mathtt{r}_x & \mathtt{r}_y \\ 1 & \mathtt{p}_x & \mathtt{p}_y \end{vmatrix} > 0 \quad$ if p is in the half plane above the edge.
- edge g_1 is below g_2 if both extremities of g_2 are above g_1 or both extremities of g_1 are under g_2
- Transitivity: two points and one edge, or one point and two edges
  - With vertical constraints
- No transitivity for *edge below*

Proof structure

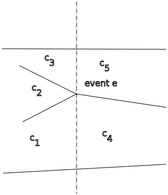
- Assumption concerning the sequence of events
- Properties of cell sequence decomposition
- Logical invariants for the main processing loop
- Main property as a consequence of the invariant

Properties for sequence of events



- The sequence is sorted lexicographically
- Outgoing edges have the left point at the event
- Edges have their right point in the sequence
- Producer code must guarantee this
- Consuming code maintains it easily

The scan state



- a record with 7 fields
- 3 fields compose the sequence of incomplete cells
  - The last created incomplete cell (c_5)
  - Two other fields for prefix and suffix
- 2 fields for the set of complete cells
  - Direct access to the last created complete cell (c_3)
  - Another field for the rest
- The last high edge (top of c_3, c_5)
- One field for the last location of the sweeping line

Invariants of incomplete cell sequences

- Each cell has a low edge below the high edge
- Each cell's high edge is the next cell's low edge
- Each cell's left side is left of the sweep line
- Each cell has a well-formed left-side
  - vertically aligned points,
  - extremities on low and high edges
  - sorted in height
- All edges have their right point in the remaining events
- Each high edge is lower than the higher of all following cells
  - Important because `edge_below` is not transitive

Main proved property

- interior of cells is disjoint from input segments
- points on sides distinct from left and right points are also disjoint from input segments

Key insights

- Incomplete cells are disjoint
- Incomplete cells are disjoint from complete cells
- complete cells are disjoint
- Obstacles are progressively included in the top of all cells

Future improvements

- Remove constraints of edges not crossing
  - Revisit the proof to remove uses of `edge_below`
  - Detect edge crossings incrementally
- Add a field to cells to point to the neighbors
- Understand where efficient numbers can be used
  - For now rational numbers, hope to use floating point numbers
- Provide a solution to allow vertical obstacles
- Add trajectory computations
  - Formal proofs missing

Play with it

`https://stamp.gitlabpages.inria.fr/trajectories`

- Limited computation capability