Smooth trajectories in straight line mazes

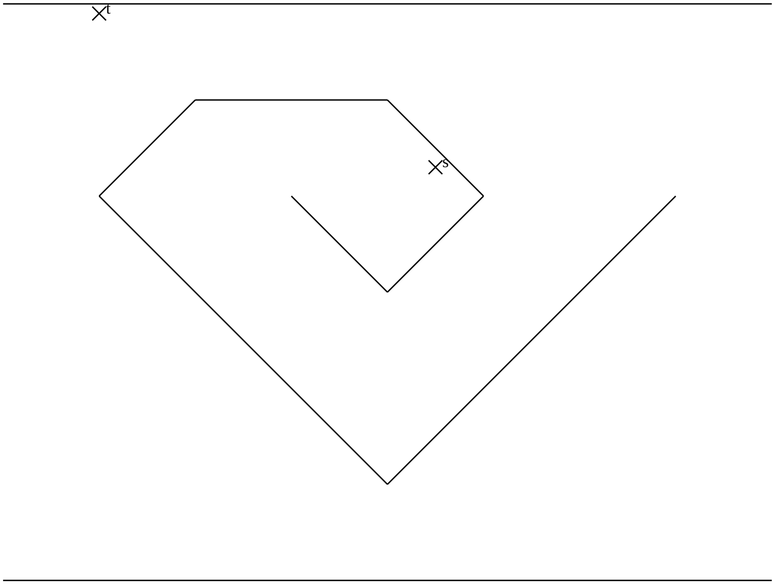
Yves Bertot
Joint work with Thomas Portet, Quentin Vermande

April 2023

The game

- Find a smooth path in a maze
- Decompose the problem
  - Find a discrete approximation of the problem
  - Construct a broken line (non-smooth path)
  - smoothen the angles
- Prove the correctness of the algorithm
  - Prove the absence of collision
  - work in progress
  - Ideally one should also prove that a path is found as soon as one exists

Example



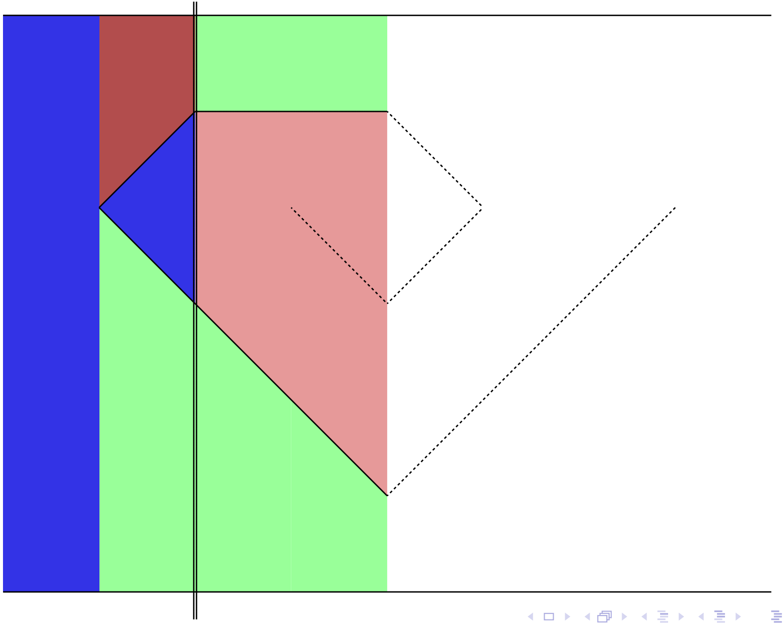
Cell decomposition

- Decompose the space into simple cells
- Each cell is convex
- Each cell is free of obstacles
- Each cell may have neighbours where moving is safe

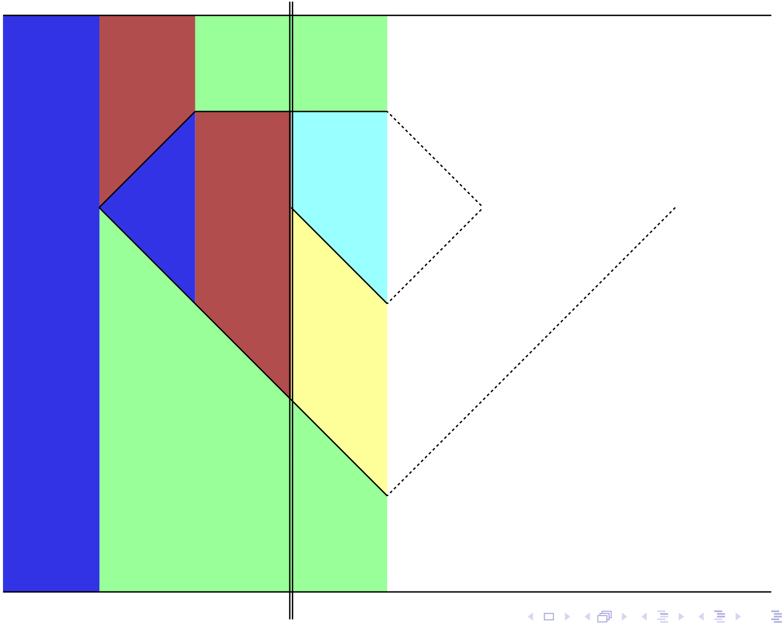
Vertical cell decomposition

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

Intermediate position for vertical cell decomposition (1)



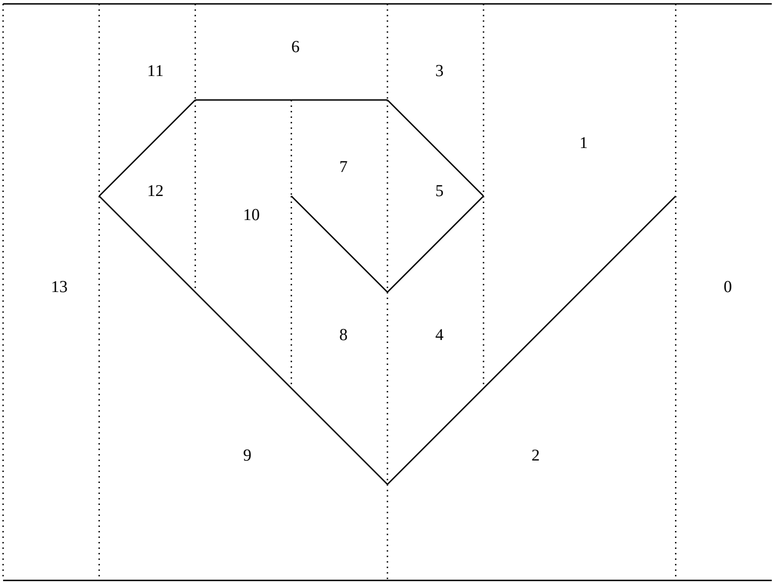
Intermediate position for vertical cell decomposition (2)



Difficulty with vertically aligned events

- Closed cells may be degenerate
  - Left and right side are in contact
- Solution: special treatment
  - Add points to the right side of last closed cell
  - Add points to the left side of last opened cell

Vertical cell decomposition example



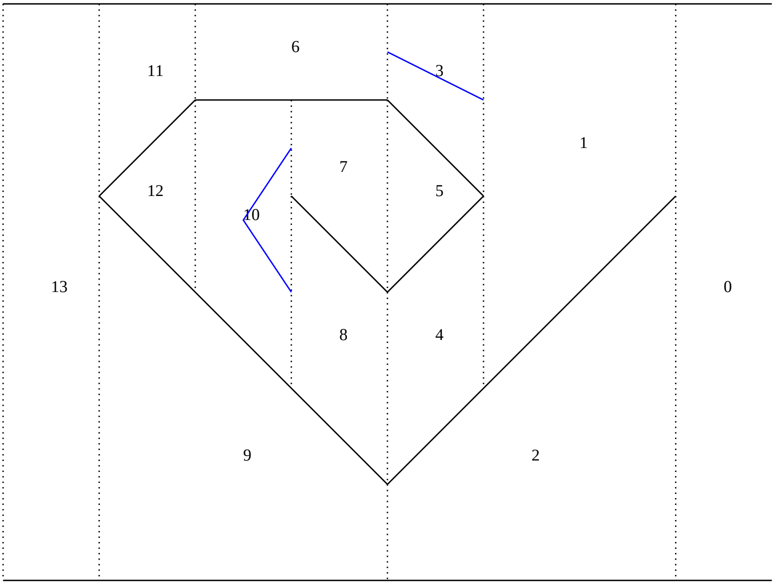
Cell assumptions

- Vertical edges are safe passages between two cells
- Moving directly left-edge right-edge is safe
  - and vice-versa
- Moving from a left-edge to the cell center is safe
  - similarly for a right-edge
  - moving from left-edge to left-edge is safe by going through the cell center

Finding a path in the cell graph

- A discrete path from cell to cell is found by breadth-first search
- Connected components of the graph are defined by polygons
- Special care for points that are already on the common edge of two cells

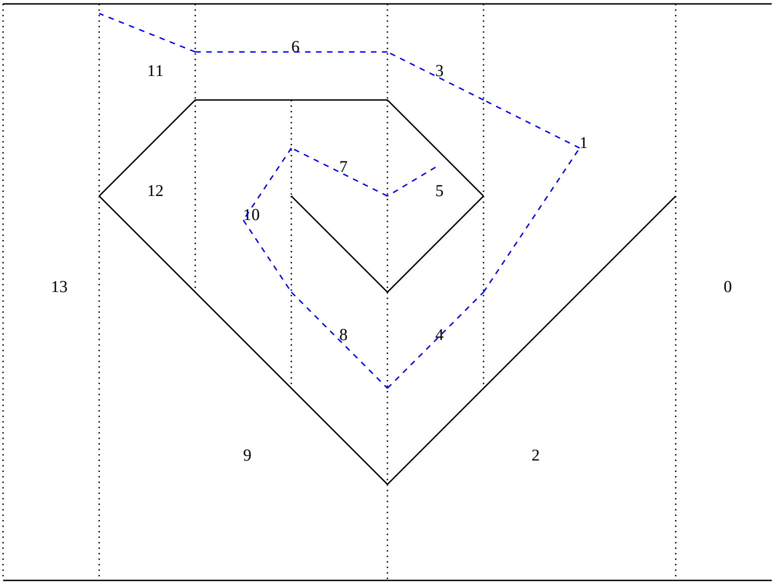
Two examples of elementary safe paths



Making a broken line path between points

- Find the cells containg the points
- Find a discrete path between cell names
- Make a path from vertical edge midpoint to vertical edge midpoint
- Connect the source and target point to the first and last vertical edge midpoints
  - Unless the source or targets are themselves on a vertical edge

broken line safe path between points



Making corners smooth

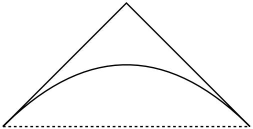
- Angles would require a robot to stop to turn
- rounded bends makes it possible to keep moving
- First approximation: no limit on steering radius
- Using quadratic Bezier curves for this purpose

The basics of quadratic Bézier curves

- Bezier curves are given by a set of control points (3 for a quadratic curve)
- Points on the curves are obtained by computing weighted barycenters
  - The curve is enclosed in the convex hull of the control points
- Given control points $a_0, a_1, \ldots, a_{n-1}, a_n$, a_0, a_1 is tangent to the curve in a_0
  - same for a_{n-1}, a_n in a_n

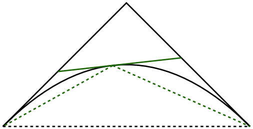
Bezier curve illustration

- Straight edge tips of this drawing are control points
- The curve is inside the triangle



Plotting the Bezier curve

- Show how the point for ratio 4/9 is computed
- Control points for the two subcurves are given by the new point, the initial starting and end points, and the solid green straight edge tip



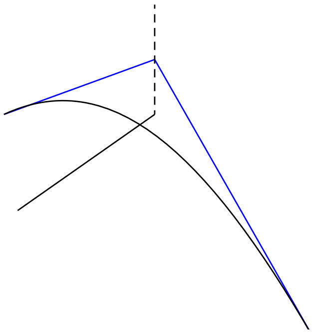
Using Bezier curves for smoothing

- Add extra points in the middle of each straight line segment
- Uses these extra points as first and last control points for Bezier curves
- Use the angle point as the middle control point
- Check the Bezier curve for collision and repair if need be

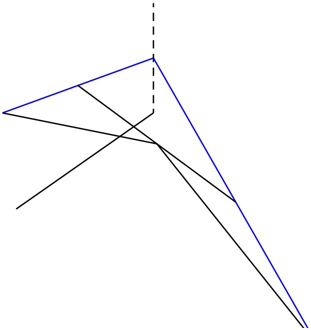
Checking for collision

- Two kinds of angles
- Angles at cell center: in the middle of safe space
  - No risk of collision
- angles at vertical edge midpoint
  - Use dichotomy until a guaranteed result is obtained
  - To compute control points in dichotomy, only half sums are needed

Collision checking, graphically



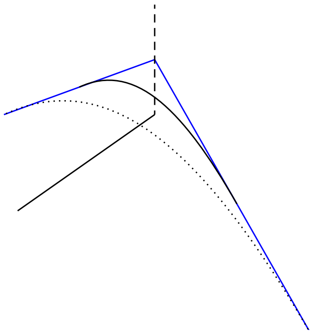
Not passing in the safe zone



Repairing a faulty curve

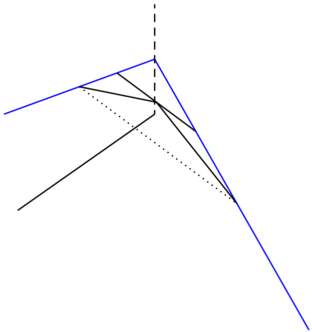
- Simple solution: bring the control points closer to the corner
- Use the first half points computed in the checking phase
- Check and repair again recursively, if need be

Constructing a repaired curve

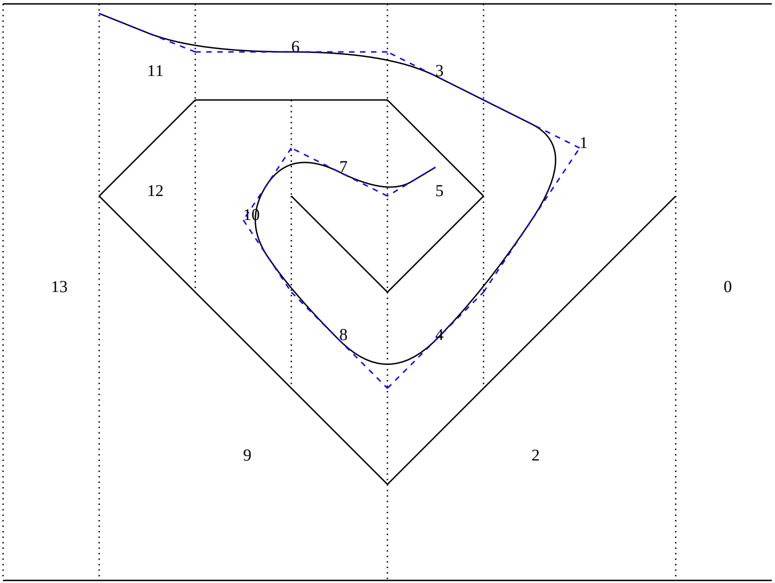


Checking the repaired curve

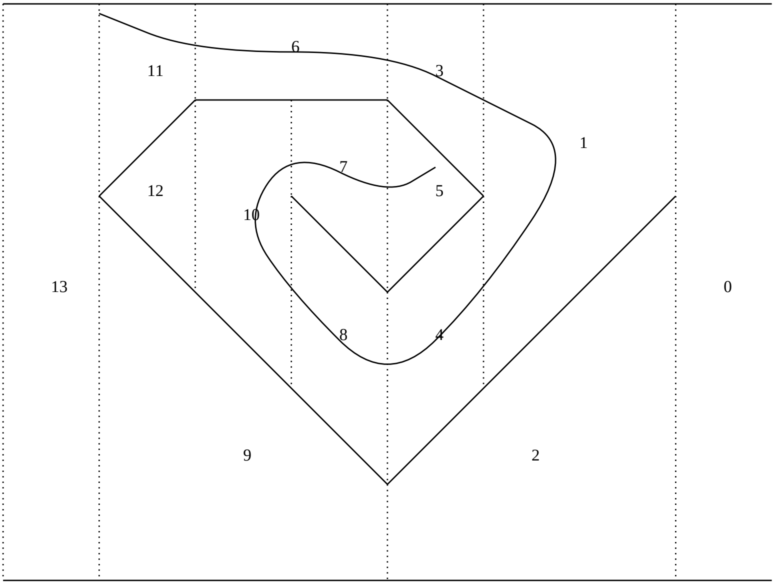
- The one-triangle convex hull is given by the dashed line
- It does not make it possible to conclude
- After dichotomy, the solid lines do



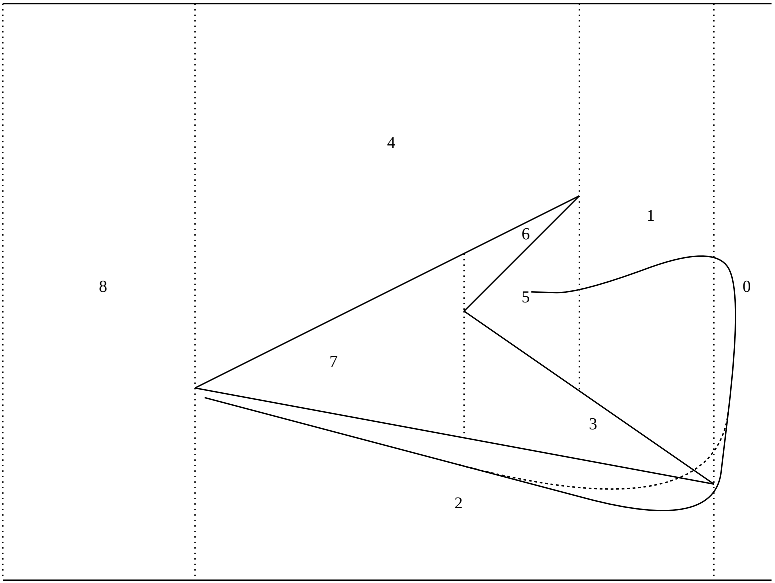
Final trajectories



Final trajectories



Final trajectories: repaired curve example



Proof tools

- Breadth first search (recent development)
- Convex hulls (Pichardie & B. 2001)
  - Orientation predicate
  - Collision between two segments (recent development)
- Convex spaces and Bezier Curve
  - Internship by Q. Vermande
  - Using `infotheo`, especially convex and conical spaces (Affeldt & Garrigue & Saikawa 2020)
- Bernstein Polynomials (B. & Guilhot & Mahboubi, 2010, Zsido 2013)

Key proof features

- Replaced absence collision by guarantees to travel inside a safe subset
  - interior of cells (2-dimensional subsets)
  - interior of doors (1-dimensional subsets)
- Safe paths from cell centers to all doors to other cells
- Safe path from any door on the left side to a door on the right side of a cell
  - This requires cells to have distinct left and right sides
- Bezier curves that cross doors are monotonic in the first coordinate
  - It is enough to prove that the door is passed correctly
- work in progress

Two uses of dichotomy

- In the algorithm, dichotomy at midpoints
  - Obtain triangles that hug the curve close enough
  - Obtain guarantee that any intersection with the vertical line is within the door
  - Does not obtain unicity
- In the proof, dichotomy at the exact value
  - Proves that the door is passed only once

Cell properties

- Two edges for the low and high side
  - These edges do not cross
- Two sequences of points for the left and right side
  - Non-empty
  - Vertically aligned points,
  - Sorted with respect to their second coordinates
  - First and last point must be on low and high edges
- Left and right side must be at distinct first coordinate

Further work

- This is proof-of-concept, not satisfactory for practical use
  - Path from middle of door to middle of door is too naive
  - Bezier Curve do not guarantee pleasant dynamics
- Should consider Clothoids
- Should improve Coq to facilitate plotting parameterized curves
  - Current approach by generating postcript programs from algorithm data
  - Rely on Postscript's Bezier curves (slides 14, 26, 27, 28)