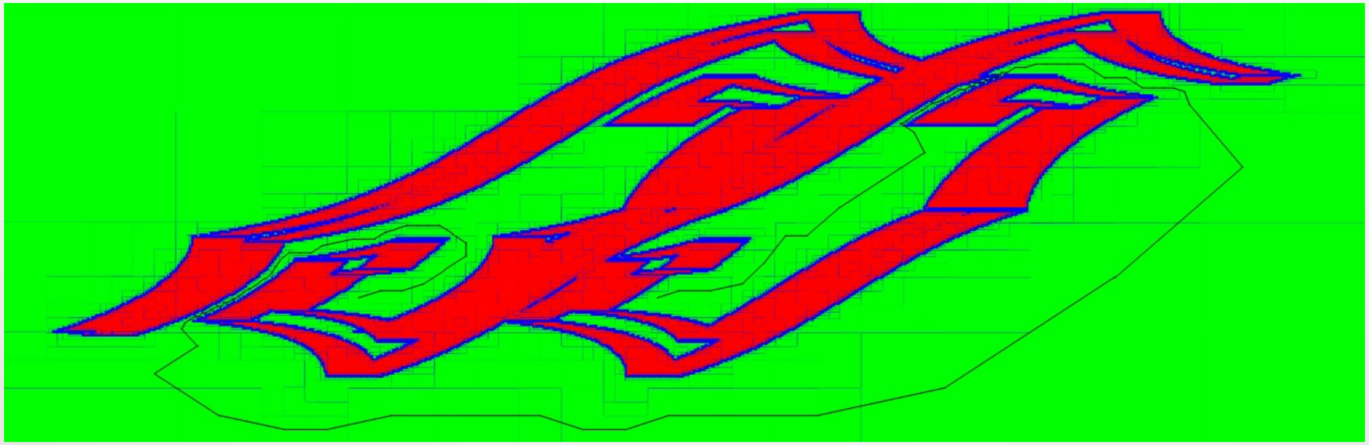
PATH PLANNING USING INTERVALS AND GRAPHS

The proposed approach uses :

- Interval analysis for characterizing the feasible configuration space S, by sub-pavings (union of boxes).

- Graph algorithms for finding short feasible paths.



Path Planning approach

The configuration space (C-space) is represented by a set of non-overlapping boxes (**Paving**).

S is the subset of the configuration space (C-space) corresponding to feasible configuration of the object.

The approach is based on the subdivision of the C-space in 3 groups of boxes :

- Those that have been proved to be inside S,
- Those that have been proved to be outside S
- Those for which nothing has been proved.

Interval analysis is used to prove that a given box is inside or outside S.

Graph discretization of S

1) We use SIVIA, a subdivision algorithm to obtain a characterization of S.

A powerful inclusion test with Interval analysis is used to prove that a box is inside or outside a set S given by nonlinear inequalities.

An *inclusion test* for the Boolean function (or *test*) $t : \mathbb{R}^n \rightarrow \{0,1\}$ is a function $[t] : \mathbb{IR}^n \rightarrow \mathbb{IB}$ such that for all boxes $[\vec{p}] \in \mathbb{IR}^n$,

$$\begin{aligned} [t]\left([\vec{p}]\right) = 1 \quad &\Rightarrow \quad \forall \vec{p} \in [\vec{p}]\,,\ t(\vec{p}) = 1, \\ [t]\left([\vec{p}]\right) = 0 \quad &\Rightarrow \quad \forall \vec{p} \in [\vec{p}]\,,\ t(\vec{p}) = 0. \end{aligned} \tag{2.4}$$

Graph discretization of S

2) We build the graph associated with this characterization of S.

Paving P		**Graph G**
Subbox [p_i] of P	$\leftrightarrow$	Vertices v_i of G
[p_i] and [p_j] are neighbors in P		Edge $v_i v_j$ exists in G

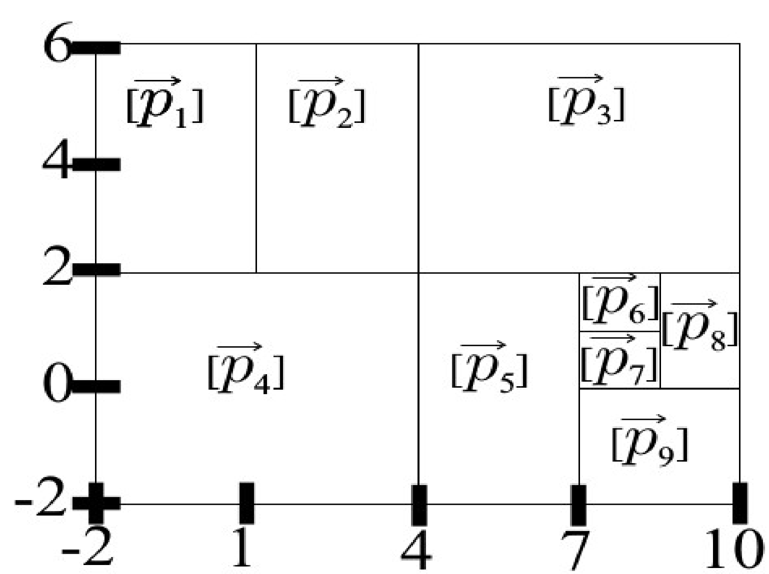


Figure 1: A paving with 9 boxes

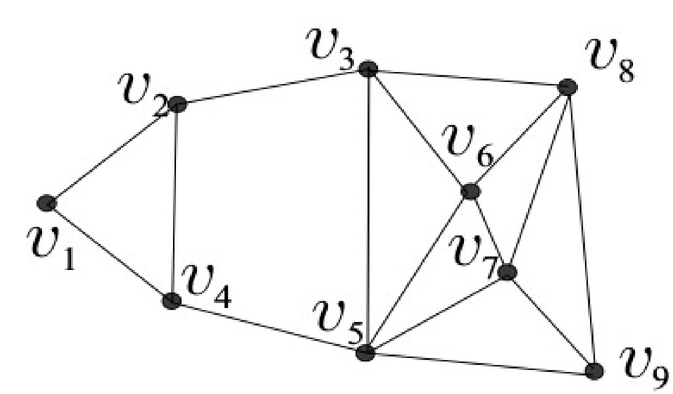


Figure 2: The graph $\mathcal{G}$ associated with the paving of Figure 1

Algorithms for finding a feasible path

- Two algorithms able to find a feasible path from a to b are given:

  - The first one characterizes S and then finds a feasible path.

  - The second algorithm, which is much more efficient, searches only the regions of the C-space that may lead to a good feasible path.

- Both return a box path, i.e., a list of adjacent boxes, such that these boxes are inside S.
Then it is necessary to find a feasible point path, l, from a to b of the moving object through the configuration space.

Algorithms for finding a feasible path

$\textsc{FeasiblePath1}\left([t]\,, \vec{a}, \vec{b}, [\vec{p}_0]\,, \varepsilon\right)$

If $[t]\,(\vec{a}) \neq 1$ or $[t]\,(\vec{b}) \neq 1$, return ("Error: $\vec{a}$ and $\vec{b}$ should be feasible");

If $\vec{a} \notin [\vec{p}_0]$ or $\vec{b} \notin [\vec{p}_0]$, return ("Error: $\vec{a}$ and $\vec{b}$ should belong to $[\vec{p}_0]$");

Stack $= \{[\vec{p}_0]\}$; $\Delta\mathcal{P} = \emptyset$; $\mathcal{P}^- = \emptyset$;

While Stack $\neq \emptyset$;

Pop into $[\vec{p}]$;

If $[t]([\vec{p}]) = 1$, $\mathcal{P}^- = \mathcal{P}^- \cup \{[\vec{p}]\}$;

If $[t]([\vec{p}]) = [0,1]$ and width$([\vec{p}]) \leq \varepsilon$, $\Delta\mathcal{P} = \Delta\mathcal{P} \cup \{[\vec{p}]\}$;

If $[t]([\vec{p}]) = [0,1]$ and width$([\vec{p}]) > \varepsilon$,

Bisect$([\vec{p}])$ and stack the two resulting boxes;

EndWhile;

$\mathcal{P}^+ = \mathcal{P}^- \cup \Delta\mathcal{P}$; $\mathcal{G}^+ = \text{Graph}(\mathcal{P}^+)$; $\mathcal{G}^- = \text{Graph}(\mathcal{P}^-)$;

$v_a = \text{vertex}([\vec{p}_a])$, where $[\vec{p}_a] \in \mathcal{P}^+$ and $\vec{a} \in [\vec{p}_a]$;

$v_b = \text{vertex}([\vec{p}_b])$, where $[\vec{p}_b] \in \mathcal{P}^+$ and $\vec{b} \in [\vec{p}_b]$;

$\mathcal{L}^+ = \textsc{ShortestPath}\,(\mathcal{G}^+, v_a, v_b)$; If $\mathcal{L}^+ = \emptyset$, return ("No path");

If $v_a \notin \mathcal{G}^-$ or $v_b \notin \mathcal{G}^-$, return ("Failure");

$\mathcal{L}^- = \textsc{ShortestPath}\,(\mathcal{G}^-, v_a, v_b)$; If $\mathcal{L}^- \neq \emptyset$, return $\mathcal{L}^-$ else return ("Failure");

Algorithms for finding a feasible path

$\textsc{FeasiblePath2}\left([t]\,,\vec{a},\vec{b},[\vec{p}_0]\right)$

If $[t]\,(\vec{a}) \neq 1$ or $[t]\,(\vec{b}) \neq 1$, return ("Error: $\vec{a}$ and $\vec{b}$ should be feasible");

If $\vec{a} \notin [\vec{p}_0]$ or $\vec{b} \notin [\vec{p}_0]$, return ("Error: $\vec{a}$ and $\vec{b}$ should belong to $[\vec{p}_0]$");

Denote by $\mathcal{P}$ the paving containing the single box $[\vec{p}_0]$;

Repeat

- $\mathcal{P}^+ = \mathrm{Subpaving}(\mathcal{P},\ 1 \in [t]([\vec{p}]))$; $\mathcal{G}^+ = \mathrm{Graph}(\mathcal{P}^+)$;
- $v_a = \mathrm{vertex}([\vec{p}_a])$, where $[\vec{p}_a] \in \mathcal{P}^+$ and $\vec{a} \in [\vec{p}_a]$;
- $v_b = \mathrm{vertex}([\vec{p}_b])$, where $[\vec{p}_b] \in \mathcal{P}^+$ and $\vec{b} \in [\vec{p}_b]$;
- $\mathcal{L}^+ = \textsc{ShortestPath}\ (\mathcal{G}^+, v_a, v_b)$;
- If $\mathcal{L}^+ = \emptyset$, return ("No path");
- $\mathcal{P}^- = \mathrm{Subpaving}(\mathcal{P},[t]([\vec{p}]) = 1)$; $\mathcal{G}^- = \mathrm{Graph}(\mathcal{P}^-)$;
- If $v_a \in \mathcal{G}^-$ and $v_b \in \mathcal{G}^-$, $\mathcal{L}^- = \textsc{ShortestPath}\ (\mathcal{G}^-, v_a, v_b)$;
- If $\mathcal{L}^- \neq \emptyset$, return $\mathcal{L}^-$;
- $\mathcal{C} = \{[\vec{p}] \in \mathcal{P}^+ |\ \mathrm{vertex}([\vec{p}]) \in \mathcal{L}^+$ and $[t]\,([\vec{p}]) = [0,1]\}$;
- Bisect all subboxes of $\mathcal{C}$, thus obtaining a new paving $\mathcal{P}$;

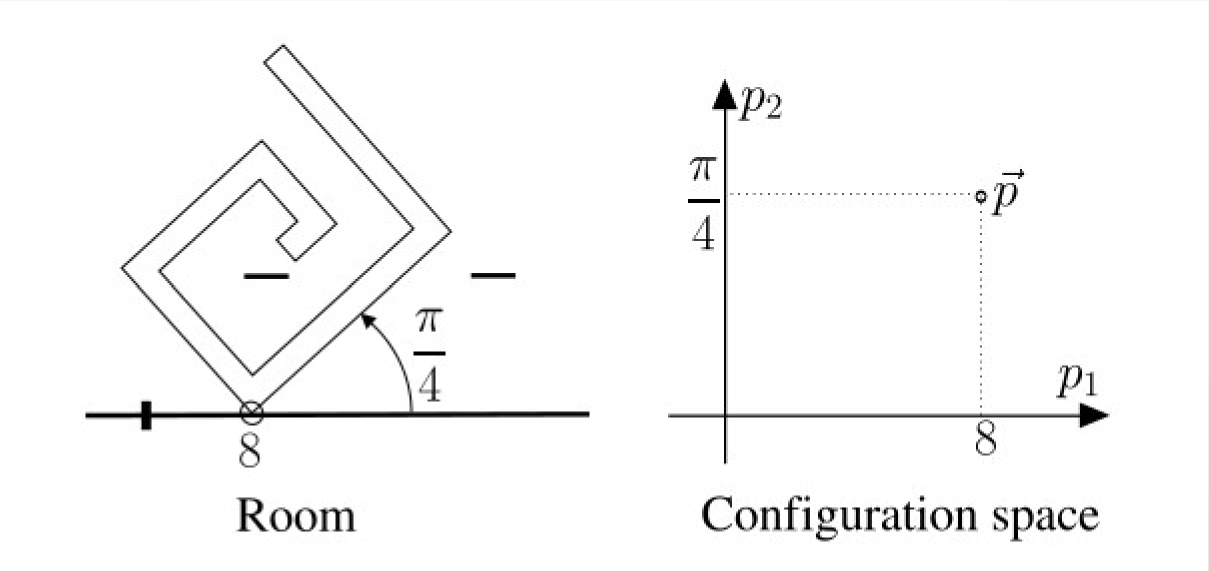
Until False.

2D Example

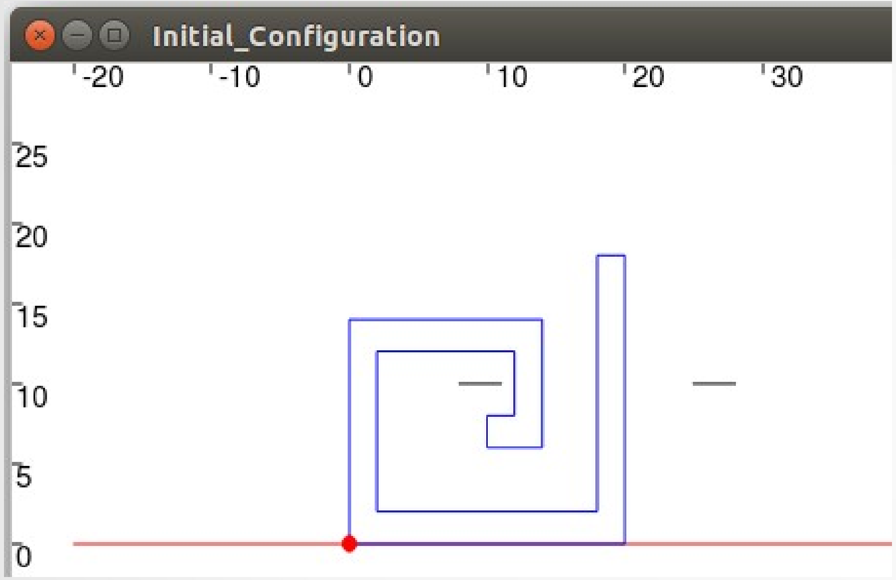
The object to be moved is a nonconvex polygon.
The red vertex of the object is constrained to stay on the horizontal line with equation y = 0.

The configuration of the object is represented by a two dimensional vector, Vp = (p1,p2).

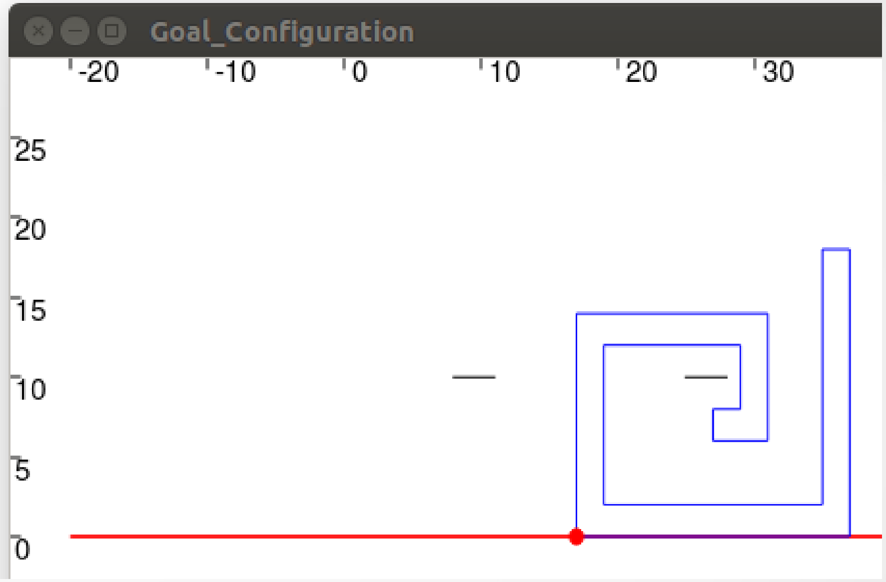
- p1 = x-coordinate of the red vertex.
- p2 = heading angle of the object.



2D Example / Reference



Vp = (0,0)



Vp = (17,0)

Demonstration

L. Jaulin (2001). Path planning using intervals and graphs. Reliable Computing, issue 1, volume 7, 1-15