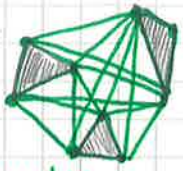
Today : - Computing the Visibility Graph.
- Shortest paths for a Translating Polygonal Robot.

• Computing the Visibility Graph

Let S be a set of disjoint polygonal obstacles in the plane with n edges in total

focus: ($S^* = S \cup \{p_{start}, p_{goal}\}$ isolated vertices in S^* do cause problems in our analysis)



Visibility Graph (S)

Algorithm: Visibility Graph (S)

Input: set S of polygonal obstacles
Output: visibility graph (S)

(1) Initialize a graph $G=(V,E)$ with V the set of vertices of S, and $E=\emptyset$

Key part → (*) (2) For all $v \in V$ compute VisibleVertices (v,S) ⇝ $O(n \log n)$
and an arc (v,w) to E for each w in this set.

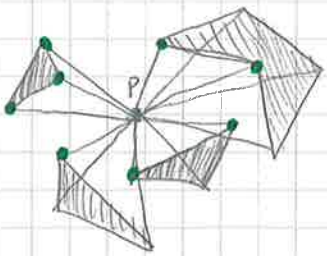
(3) Return G.

As we will see below, step (2) takes $O(n \log n)$ time.
Doing this for all n vertices, we get:

Theorem The visibility graph of a set S of disjoint polygonal obstacles with n edges in total can be computed in $O(n^2 \log n)$ time

• Computing Visible Vertices

Let p be a point that does not lie in the interior of any obstacle.



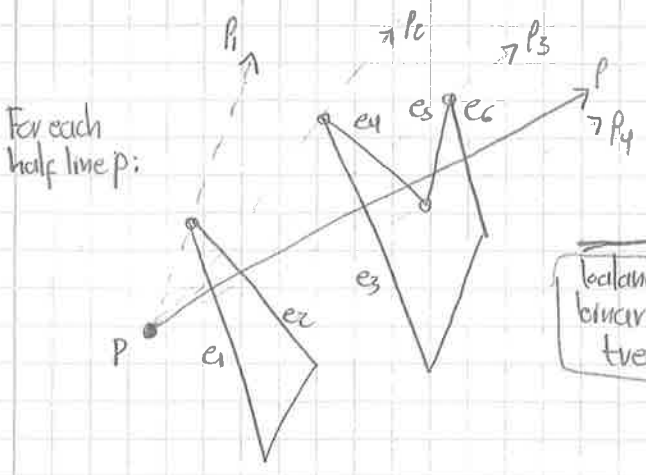
Visible Vertices (p,S)

Want to compute VisibleVertices (p,S) of all obstacle vertices that are visible from p.

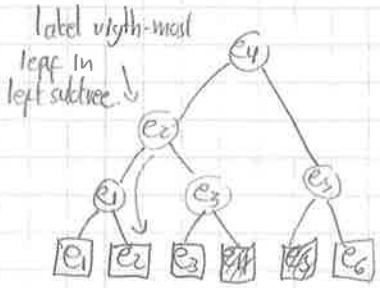
Idea: make a half line

and rotate in clockwise order
order vertices of S according to the order in which they are touched (angle)
keep track of obstacle edges crossed in a balanced search tree.
use this to determine if the new appearing vertices are visible from p.

For each half line p:



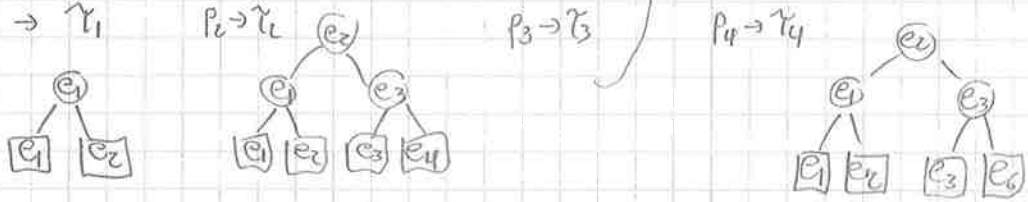
balanced binary search tree $\mathcal{T}$



↳ leaves are the edges crossed by the half line in order

Note: The tree $\mathcal{T}$ changes when p is rotated through a new vertex v

$p_1 \to \mathcal{T}_1$ $p_2 \to \mathcal{T}_2$ $p_3 \to \mathcal{T}_3$ $p_4 \to \mathcal{T}_4$



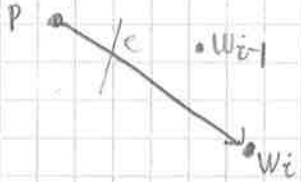
Edges clockwise from v are added to the tree.
" counter-clockwise " " " removed from the tree.

Determining if w_i is visible from p:

(0)

If $\overrightarrow{pw_i}$ intersects interior of polygon containing w_i
→ w_i not visible

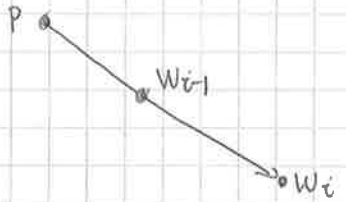
(1)



If w_{i-1} is not in the half line $p_i = \overrightarrow{pw_i}$

Then it suffices to check if the left-most edge of the tree $\mathcal{T}_i$ (first edge crossed by the half line) is crossed by p_i

(2)

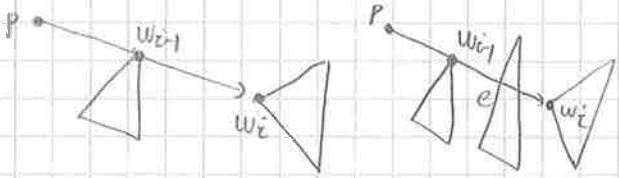


$O(\log n)$ ⇝

If w_{i-1} is not visible then w_i is not visible

If w_{i-1} is visible, search in $\mathcal{T}_{i-1}$ for an edge e that intersects $\overline{w_{i-1}w_i}$

If e exists → w_i not visible
otherwise → w_i visible.



Checking this visibility test takes $O(\log n)$ time.
using the balanced tree structure $\mathcal{T}$.

Algorithm: Visible Vertices (p, S)

Input: Set S of polygonal obstacles
point p not in the interior of any obstacle

Output: The set of all obstacle vertices visible from p.

O(nlogn) ← (1) Sort obstacle vertices according to clockwise angle → $w_1, \dots, w_n$

(2) Let ρ be the half-line parallel to positive x-axis starting at p

Find obstacle edges properly crossed by ρ and store them in a balanced search tree $\mathcal{T}$ in the order in which they are intersected

O(nlogn) ← (3) For $i \leftarrow 1$ to n

O(logn) ← Check if w_i is visible from p
Update the tree $\mathcal{T}$.

(4) Return set of visible vertices

Complexity of Visible Vertices (p, S): O(nlogn) as we wanted

⇒ Complexity of Visibility Graph (S): $\boxed{O(n^2 \log n)}$

Shortest Paths for a Translating Polygonal Robot

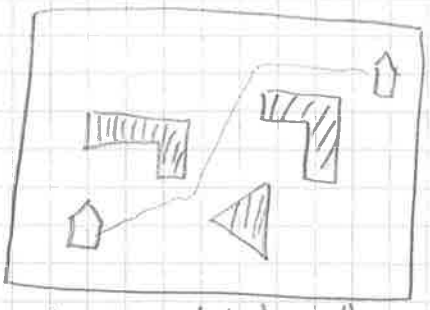
Algorithm:

$O(n \log^2 n)$ ← (1) Compute forbidden space. (complexity $O(n)$)

$O(n^2 \log n)$ ← (2) Compute visibility graph of the forbidden space union p_{start}, p_{goal}.

$O(n \log n + k)$ ← (3) Use Dijkstra's algorithm to find a shortest path, if it exists.
$\downarrow$
$O(n^2)$

Total complexity : $O(n^2 \log n)$



S set of obstacles with n edges in total
R convex robot of constant complexity

Theorem Let R convex robot of constant complexity that can translate among a set of polygonal obstacles with n edges in total. A shortest collision-free path for R from a given start position to a goal position can be computed in $O(n^2 \log n)$ time

Better algorithms not computing the full visibility graph:

Mitchell : $O(n^{8/5+\varepsilon})$
$O(n^{3/2+\varepsilon})$

Hershberger-Suri : $O(n \log n)$