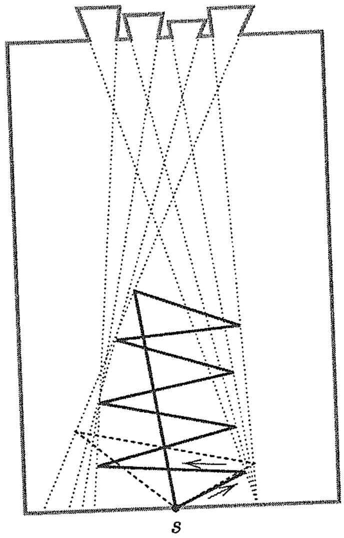
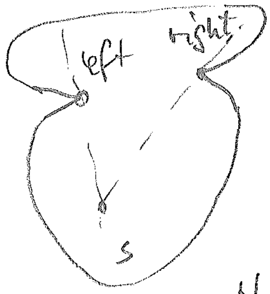
Exploring cuts in clockwise order doesn't work in non-rectilinear polygons.




<u>Idea</u> Explore left and right reflex vertices separately.




<u>Problem</u>

How to know about left vertex v ?


<u>Idea</u> Partition left/right reflex vertices into groups

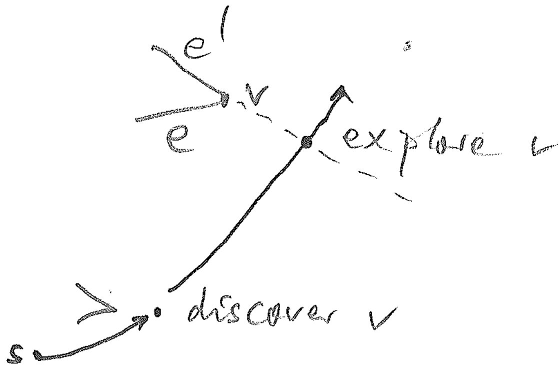
start with right vertices

→ (6 ...

<u>notations</u>:

reflex vertex v <u>discovered</u> : (part of) edge e has been seen

... <u>explored</u> : (part of) edge e' has been seen, too



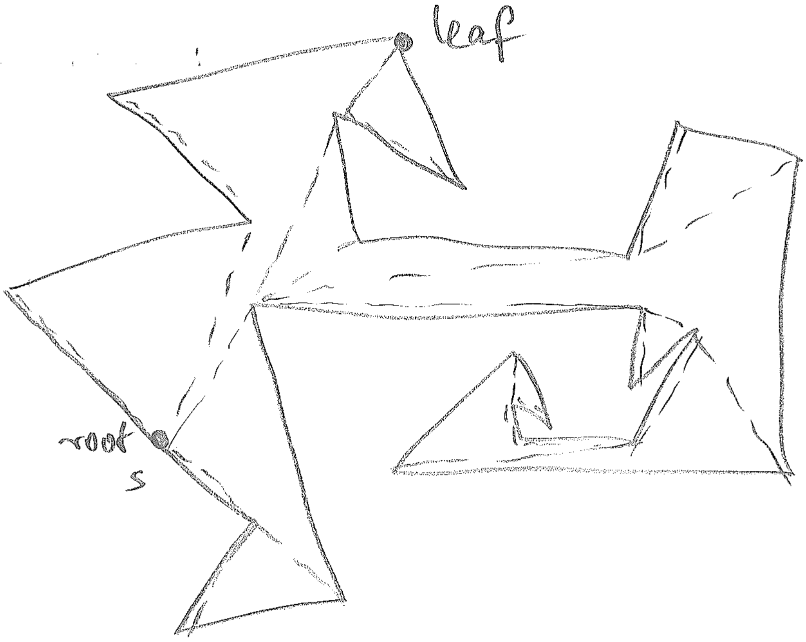

CP : robot's current position

TargetList : List of discovered, but not yet explored, right vertices, sorted in clockwise order on ...
(initially: all right vertices discoverable, but not explorable, from s)

BasePoint : point where group exploration starts (initially = CP)

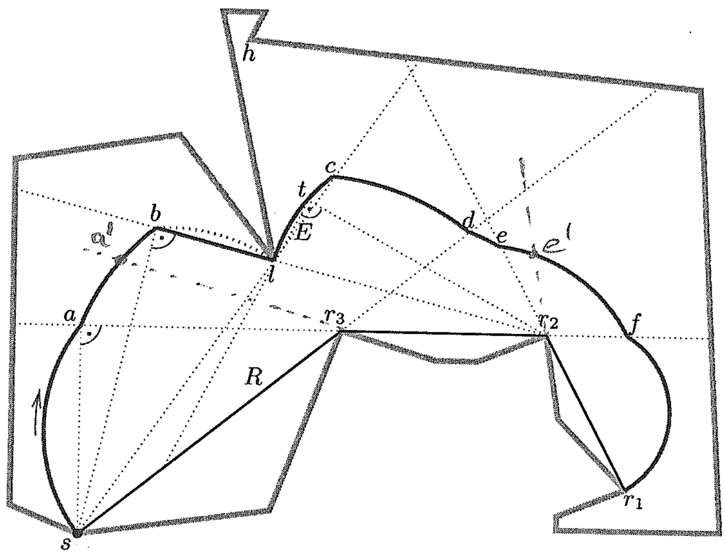
... another useful structure:

Shortest path tree SPT(s)
= tree made of all shortest paths from s to the vertices of P




internal nodes
= reflex verti

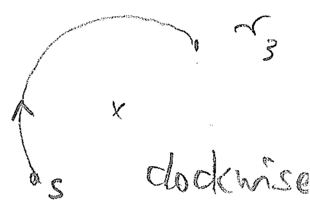
```
procedure ExploreRightVertex (inout TargetList, inout ToDoList);
  BasePoint := CP;
  Target := First (TargetList);
  if Target not visible then
    walk on shortest path from BasePoint to Target
    until Target becomes visible;
  Back := last vertex before CP on shortest path from BasePoint to CP;
  walk clockwise along circ (Back, Target)
   while maintaining TargetList and ToDoList
    whenever First (TargetList) changes let Target := First (TargetList);
    whenever Back becomes invisible update Back;
    exceptions for walking along the circle:
      if the boundary of P blocks the walk on the current circle then
        walk clockwise along the boundary
        until the circular walk is again possible;
      if Target is becoming invisible then
        walk towards Target
        until the blocking vertex is reached;
  until Target is fully explored;
end ExploreRightVertex;
```




## discussion of example:

initially, BasePoint = CP = s,
TargetList = $\{r_3\}$, ToDoList = $\emptyset$

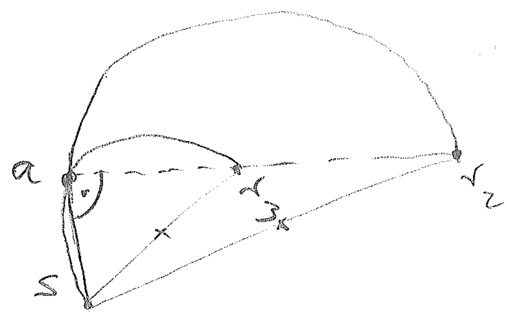
Target = $r_3$ is visible
Back = s
robot starts following circ$(s, r_3)$



at point a, right reflex vertex $r_2$ is discovered
comes before $r_3$ in clockwise order

⇒ Target = $r_2$
robot now follows circ $(s, r_2)$

observe: circ $(s, r_2)$ does pass through a
(because of Thales' theorem)




at point a', $r_3$ gets fully discovered, removed from Target li[st]
at point $b_1$ Target $r_2$ would become invisible

⇒ robot walks towards Target $r_2$ until blocking
vertex l is reached

robot now follows circ $(l, r_2)$ because
Back = l = last vertex of shortest path
from BasePoint s to CP

at point c, BasePoint s becomes visible again

⇒ robot now follows circ $(s, r_2)$ because
Back = s

at point , BasePoint s becomes invisible
robot follows circ $(r_3, r_2)$ because
Back = $r_3$ = last vertex on shortest path
from BasePoint s to CP

at e, right reflex vertex $r_1$ is discovered
comes before $r_2$ in clockwise order

=) Target = $r_1$
robot follows circ ($r_2$, $r_1$)

at e', $r_2$ becomes fully explored and removed from Target List

at f, $r_3$ becomes invisible

robot now follows circ ($r_2$, $r_1$) because
Back = $r_2$ = last vertex on shortest path from BasePoint s to CP

at $r_1$, Target $r_1$ becomes fully explored.

DONE !

<u>Observation</u> Robot reaches cut of Target at the same point as shortest path from BasePoint

<u>Remarks</u>

(i) right reflex vertices $r_2$, $r_1$ were added to Target List because shortest paths from s to $r_2$, $r_1$ did not contain left turns

(ii) right reflex vertex h was <u>not</u> added to Target List, because shortest path s – h makes left turn at l.

reflex vertex h will be listed later on (To Do Li...

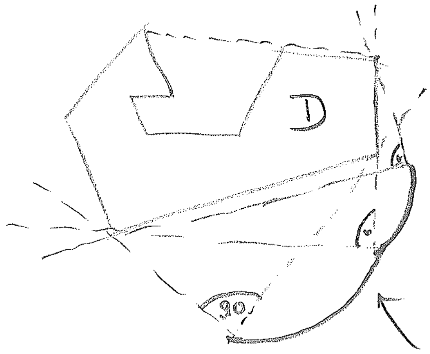
→ (6...

<u>Question</u>: How long is path generated by ExploreRightVertex as compared to shortest path from s to $r_1$ (in example) ?

need structural property for analysis:

## Photographer's path $\hat{=}$ angle hull

suppose photographer moves around object to find best viewpoint; at all times object fits exactly into 90° lens angle




only convex hull*) of object matters, CH(D)

circular arcs (Thales)

photographer's path, AH(D)

Lemma: In the free plane,

$$|AH(D)| \le \frac{\pi}{2} \cdot |CH(D)| \le \frac{\pi}{2} |D|,$$

where $|.|$ denotes the perimeter. (without proof)

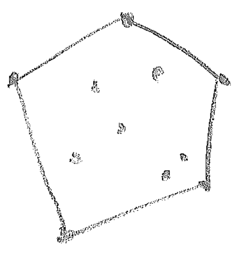
Example:


perimeter(D) = 2
perimeter(AH(D)) = $\pi$

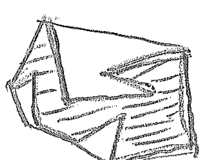
We are interested in a constrained case where object D is located in simple polygon P

---

*) convex hull $CH(D) := \bigcap_{D \subseteq C \subseteq \mathbb{R}^2,\ C \text{ convex}} C$



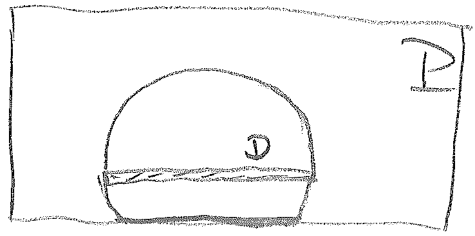
,



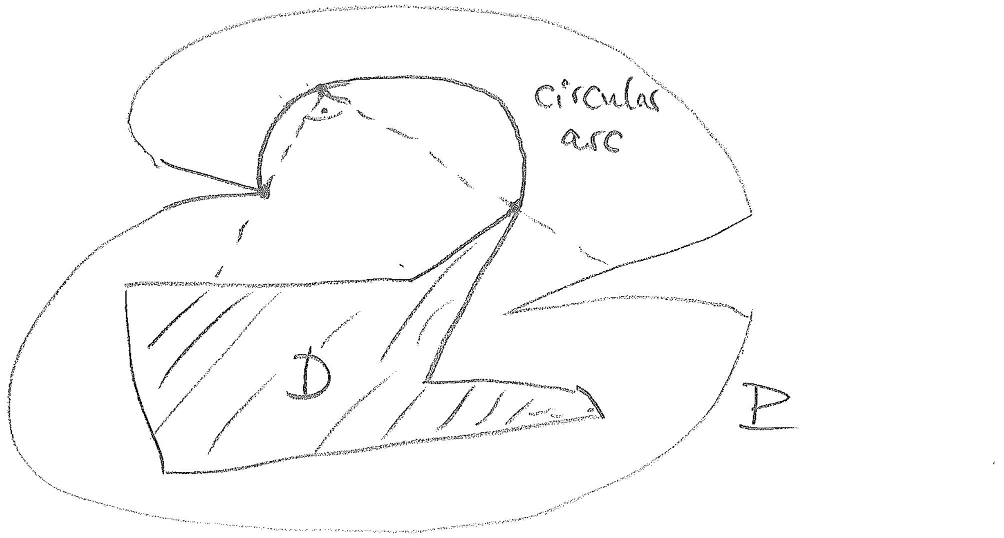
examples

Here, photographer's path may

- follow a polygon's edge ("too close")




- be constrained by reflex polygon vertex touching 90° wedge from the outside

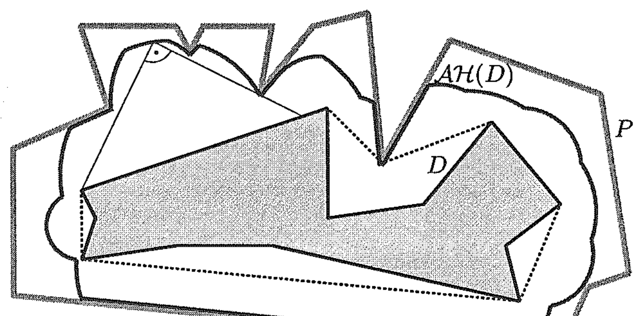



more precisely,

<u>Def:</u> Let D be a connected compact set in the exterior of a simple polygon P. Then,

$$AH_P(D) := \{ z \in P \mid z \text{ can see two points of } D \text{ at a } 90° \text{ angle} \}$$

is called the angle hull of D in P.
(Its boundary: photographer's path.)



Similar to previous case: boundary of angle hull depends only on relative convex hull $RCH_P(D)$ of $D$ in $P$ (rubber band encircling $D$ in $P$).

<u>Theorem A</u> $|AH(D)| \leq 2 \cdot |RCH(D)| \leq 2 \cdot |D|$, and this bound can be attained.

<u>Proof</u> Later.

Consequence:

<u>Lemma 2</u> Path generated by ExploreRightVertex from BasePoint to cut of Target explored is at most twice as long as shortest path.

<u>Proof</u> Path generated is part of $\partial AH$ (shortest path), (except for straight segments leading to blocking vertices, which are replaced by longer circular segments in AH).

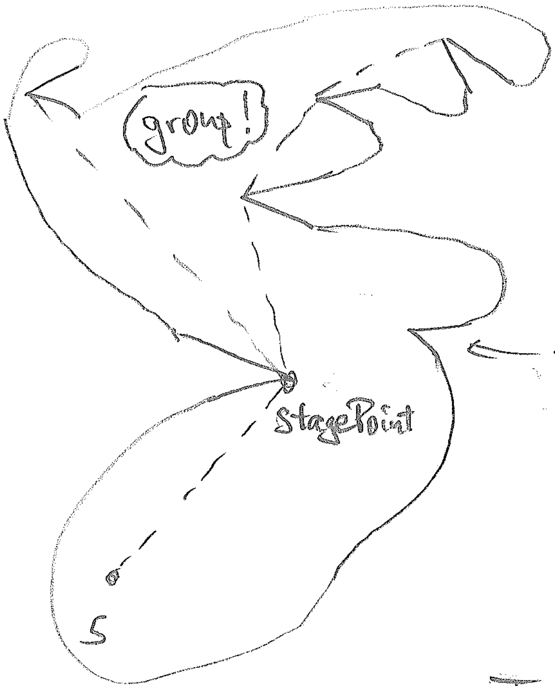
Symmetric to ExploreRightVertex: ExploreLeftVertex

Next: exploring a group of right vertices.

**procedure** *ExploreRightGroup* ( **in** *TargetList*, **out** *ToDoList* );
  *StagePoint* := *CP*;
  *ToDoList* := empty list;
  **while** *TargetList* is not empty **do**
    *ExploreRightVertex* (*TargetList*, *ToDoList*);
      (* *CP* is now on the cut, *C*, of the last target. *)
    **walk** to the point on *C* that is closest to *StagePoint*
      **while** maintaining *TargetList* and *ToDoList*;
  **walk** on the shortest path back to *StagePoint*;
**end** *ExploreRightGroup*;

In ExploreRightGroup:

StagePoint: initially, $= s$
in general, a left reflex vertex of the shortest path tree of $s$, also visited by $W_{opt}$




On calling ExploreRightGroup:

- ToDoList $= \emptyset$
- TargetList contains sorted list of unexplored right vertices, whose shortest paths from StagePoint makes only right turns
among them all right children of Stage[Point]

During execution of ExploreRightVertex (or walk):

- all newly discovered right vertices are added to TargetList whose shortest paths from StagePoint contain only right turns
- all explored right vertices are added to ToDoList that have a left child
(children of explored vertices are known!)

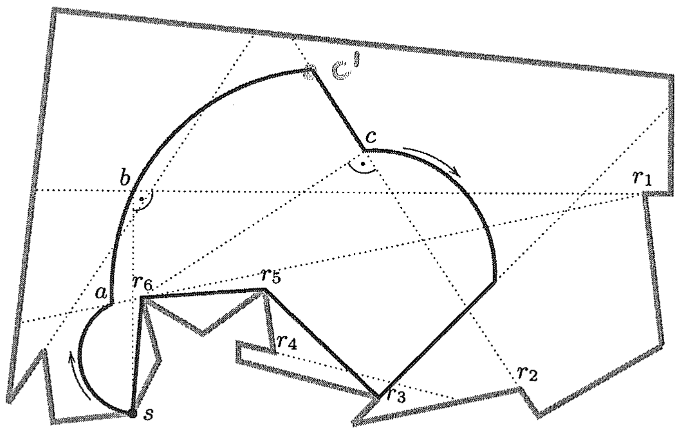
On returning from ExploreRightVertex:

- robot adjusts its position on the cut of last Target (will become useful later)
adjusted points become BasePoints

On termination of ExploreRightGroup:

- all right vertices initially in TargetList, and all their purely right descendants in the shortest path tree of s have been explored.

Example:




during exploration of $r_6$, vertex $r_1$ is discovered, at a

at b, exploration of Target $r_1$ is completed.

ExploreRightVertex returns,

no adjustment necessary because b is closest point to StagePoint s on cut of $r_1$

Target List: $r_6$, $r_1$ removed, $r_2$, $r_5$ added

next call to ExploreRightVertex explores $r_2$, finishes at c'

robot walks along cut to c = closest point to StagePoint s

next call to ExploreRightVertex explores $r_3$, robot walks along cut to $r_2$ = closest point to s while doing so, $r_4$ gets explored

robot returns on shortest path to StagePoint, s.