Landmark-Based Navigation

In $\mathbb{R}^3$, a finite set $L \subseteq \mathbb{R}^2$ of

$n$ labeled landmarks.

A point robot moves in $\mathbb{R}^2$ ($\mathbb{E} = \mathbb{R}^2$, $\mathbb{E} = \mathbb{R}^2 + S'$)

$X = \mathbb{R}^{2n+2}$

Landmark positions unknown
Robot's position unknown

Sensor: Cyclic permutation of landmarks $h: X \rightarrow Y$

$\{3, 2, 5, \ldots, 10, 7, 6, 1, \ldots, 10, 7, 6, 5, 2, 3\}$

Go to landmark $i$

$y: h(x) = (1, 10, 5, 2, 3, 6, 7, 9, 8, 4)$

Cyclic
Task: Determine which landmarks are on the convex hull boundary of $L$.

More generally, for any $L' \subseteq L$, which landmarks are in the hull of $L'$?

Key observation

1. 
2. 
3.

Go to 2 (from 1)

Infer that 3 is to the left

Simple strategy: Travel between all pairs of landmarks $O(n^2)$ time

$O(n^2 \log n) -$ Tovar, Freda, LaValle

Cont. Math.
A Sensor-Centric Theory of Computation
Jason O'Kane, PhD Thesis, July 2007
"On Comparing the Power of Robots"

Recall from theory of computation
A problem is encoded as a set of strings called \( L \), a language.

A machine should accept strings in \( L \) and reject all others.

\( \in \Sigma^* \)

Diagram:
- **Machine**: FA - Finite automaton
- Pushdown automaton
- **TM**: Turing machine
Regarding a problem (a language):

1) Decidability (solvable?)
2) Complexity (if yes, how costly?)

Establishing equivalences of machines

- Can solve the same problems
- Ex. Adding more tapes to a TM

Compare the power of machines to solve problems

- Some machines can solve more than others

FA < PDA < TM

Ex. Palindromes, redivider, s11ppuvaknuippins
We want the same kinds of concepts for sensing/actuation systems as robotic systems.

Instead of TM

\[ \Sigma^* = \{ \epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \} \]

\[ \mathcal{L} = \{ \epsilon, 1, 01, 10, 000, \ldots \} \]

1D point robot with perfect sensing

Can a robotic system simulate a TM?
Adapt existing theory of computation?

- Measuring uncertainty? Precision physics?
- Too biased toward writing everything down in advance
- Arbitrary discretizations that obscure the true complexity

On-line algorithms