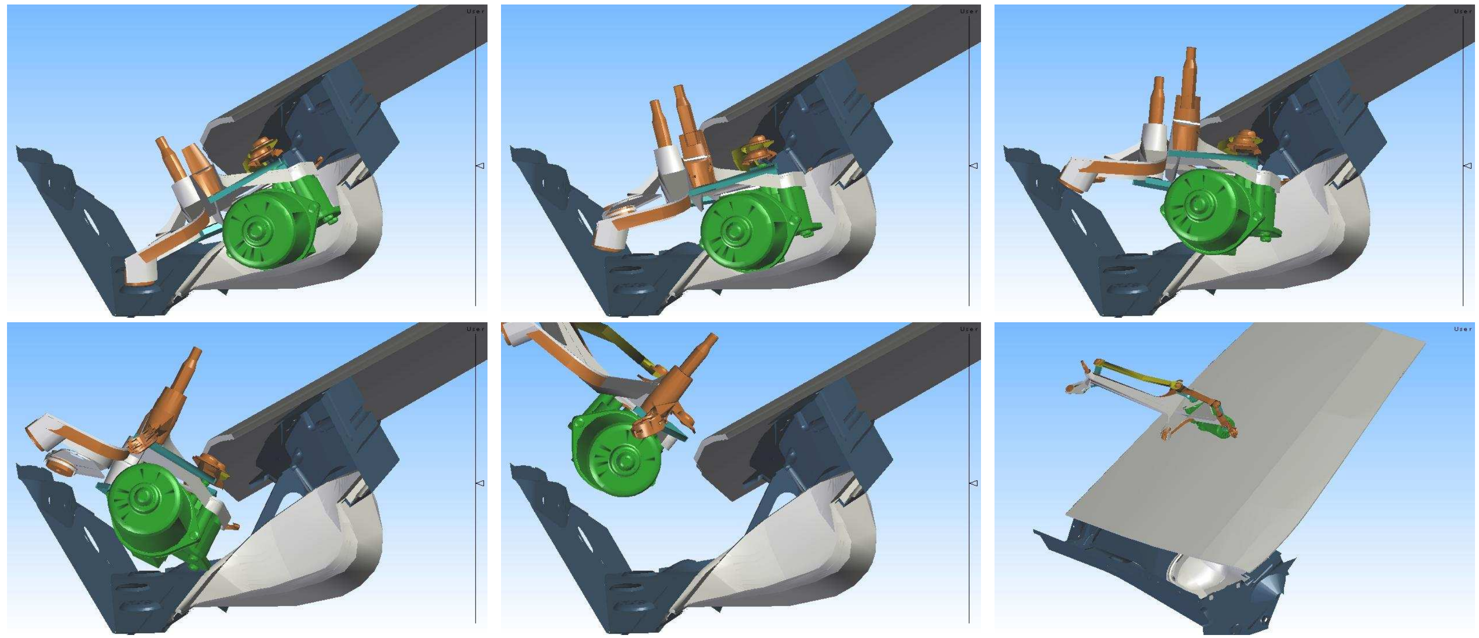
Minimal Sensing Structures for Designing Robot Motions

Steve LaValle, University of Illinois

April 18, 2008

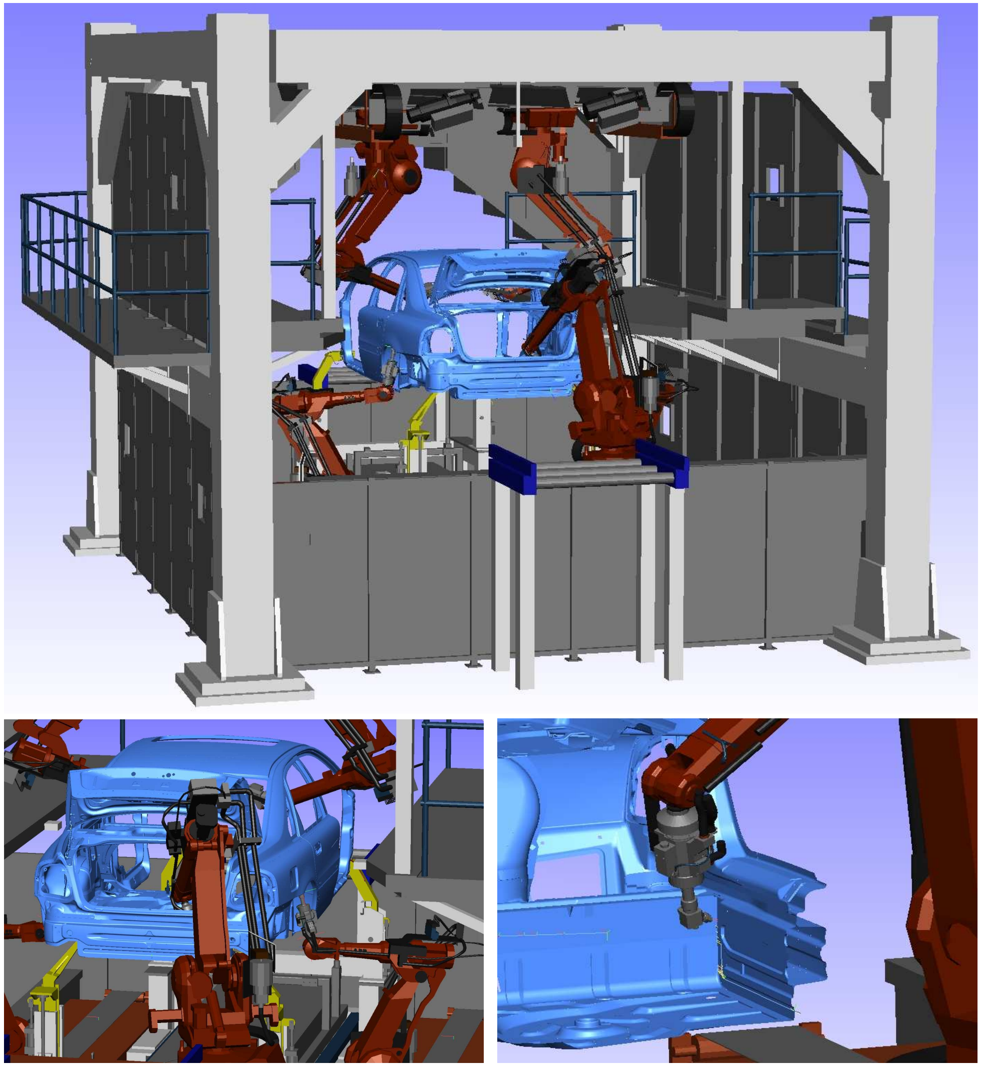
Planning Algorithms Overview

Wiper Motor Assembly

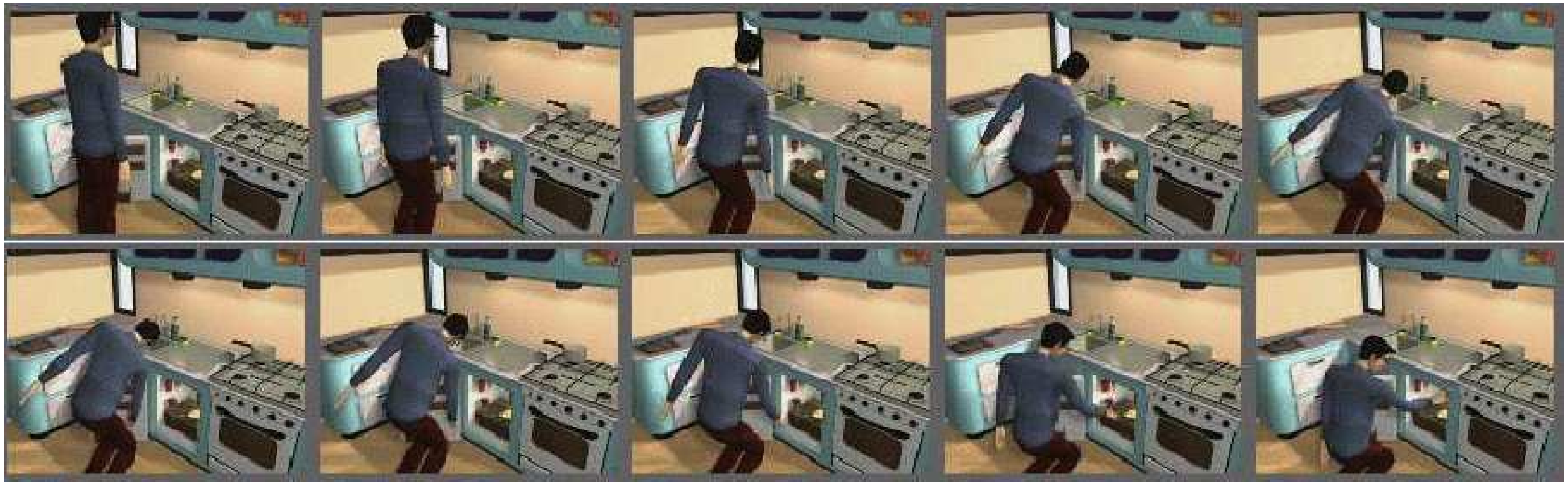


- Kineo CAM and LAAS/CNRS, Toulouse, France
- Integrated into Robcad (eM-Workplace)
- Add-ons for 3D Studio Max, Solidworks
- Direct users: Renault, Airbus, Ford, Optivus, ...

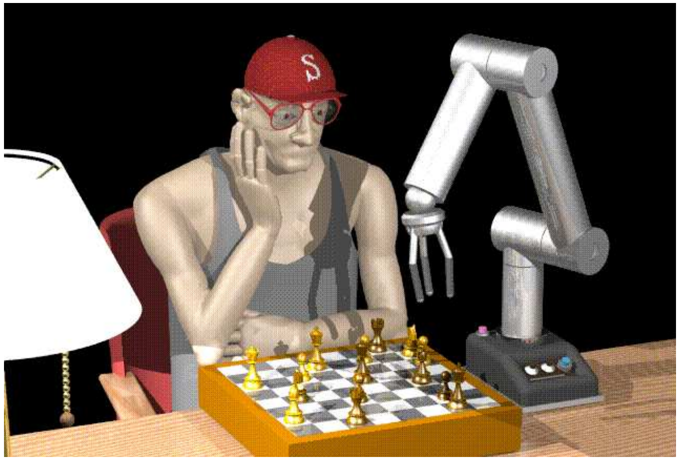
Sealing Cracks at Volvo Cars



Fraunhofer Chalmers Centre and Volvo Cars, Sweden



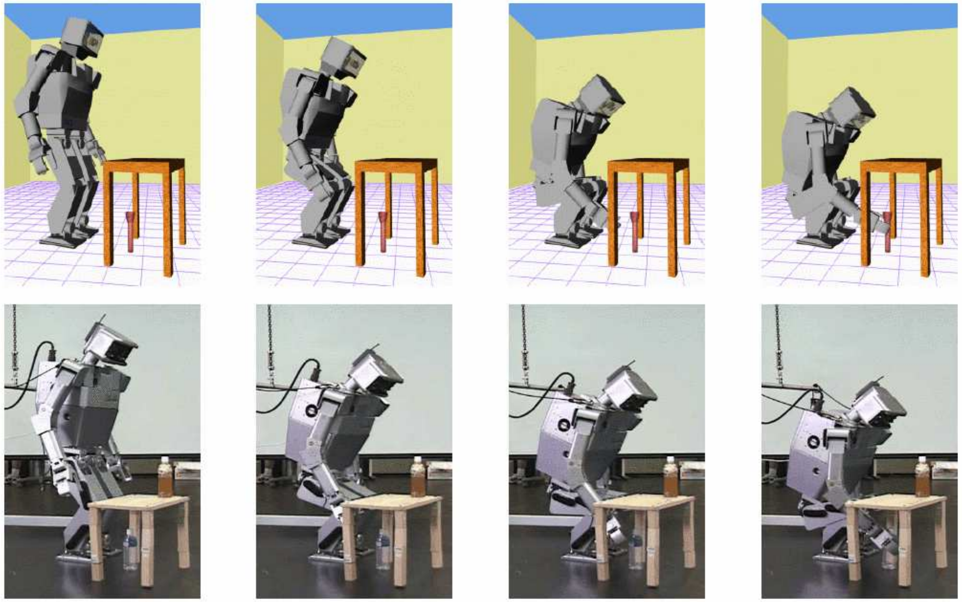
Marcelo Kallman, UC Merced



James Kuffner, CMU

Humanoid Robots

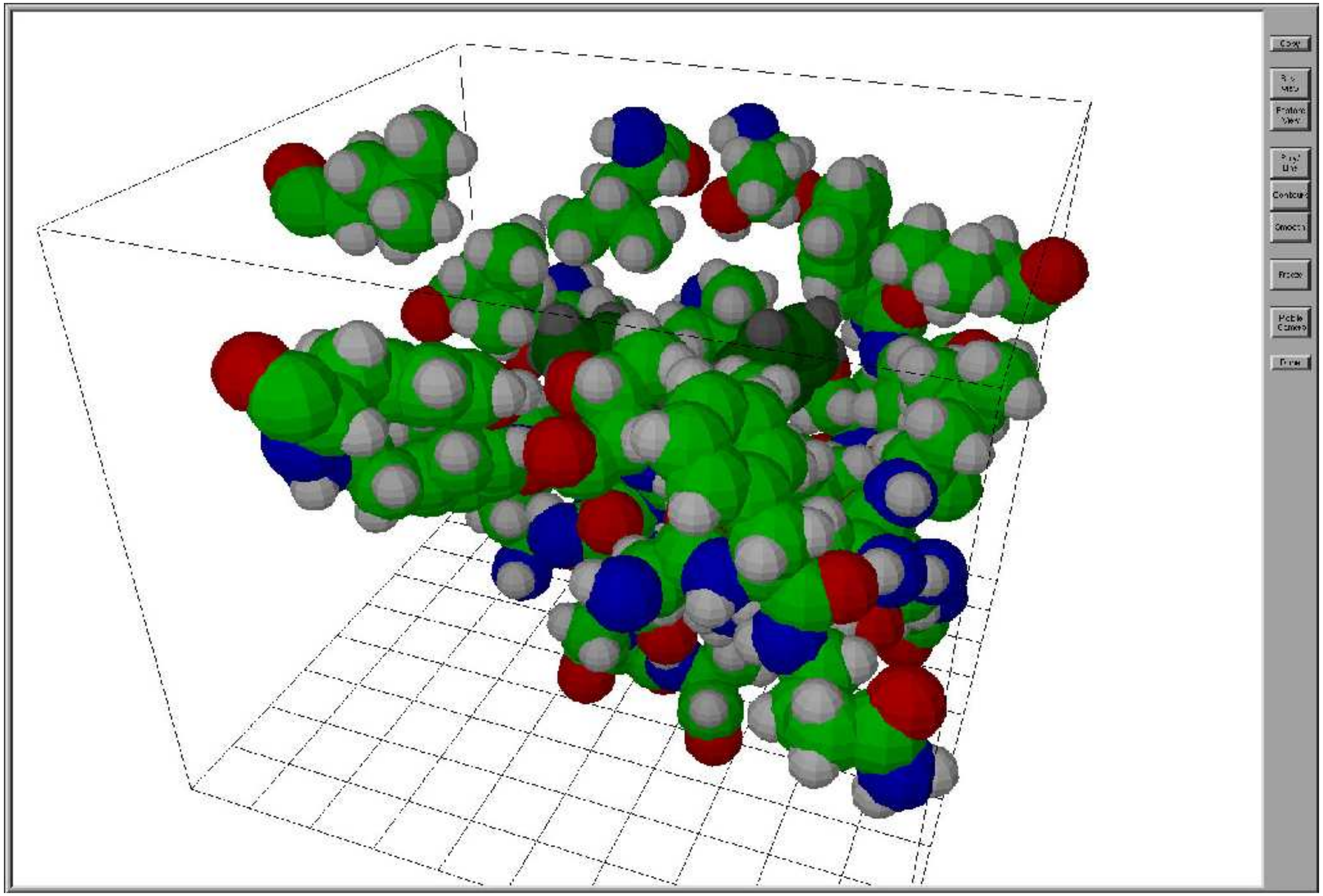
Kagami and H7



Planning

University of Tokyo and AIST

Molecules, Etc.



From Nic Simeon, LAAS/CNRS

Algorithms Need Discretizations

The world is more or less continuous.

Computation is discrete.

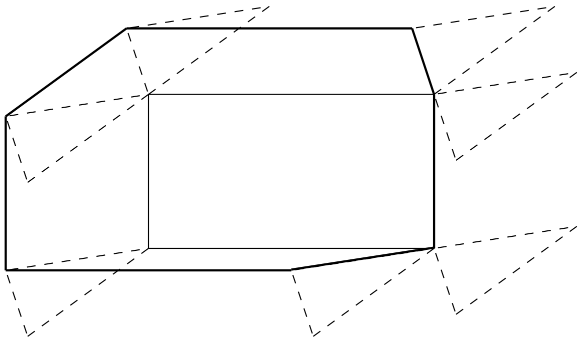
- 1970s: Grids, logic-based planning
- 1980s: Combinatorial motion planning
- 1990s: Sampling-based motion planning

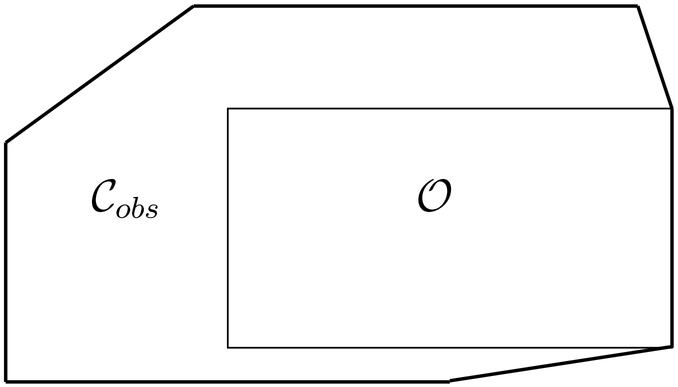
Also: Planning problems are *implicitly* encoded.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

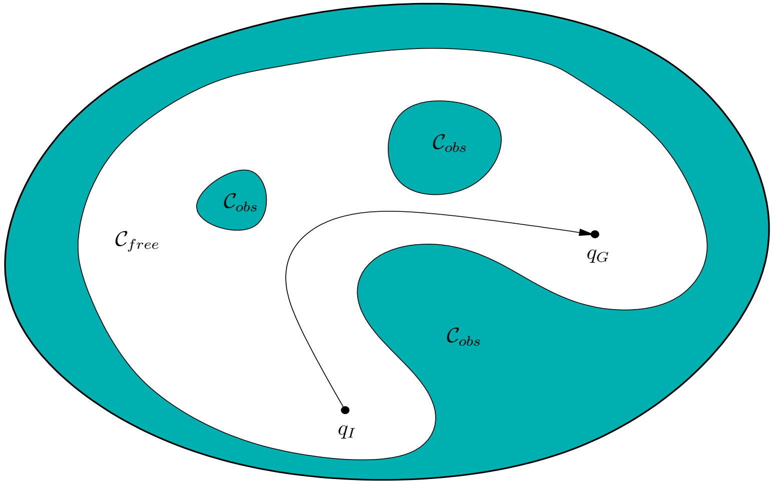
The C-Space Obstacles

Lozano-Perez, 1979



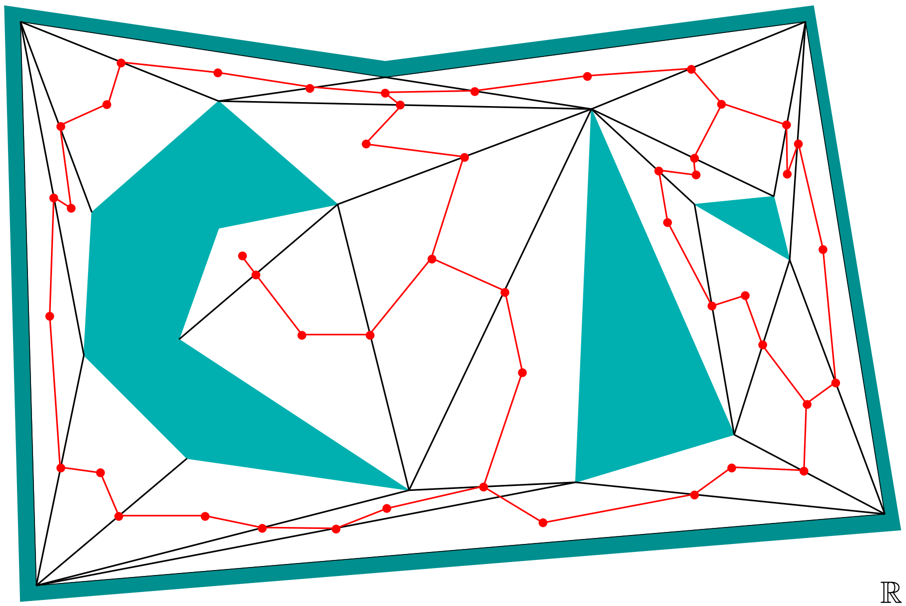


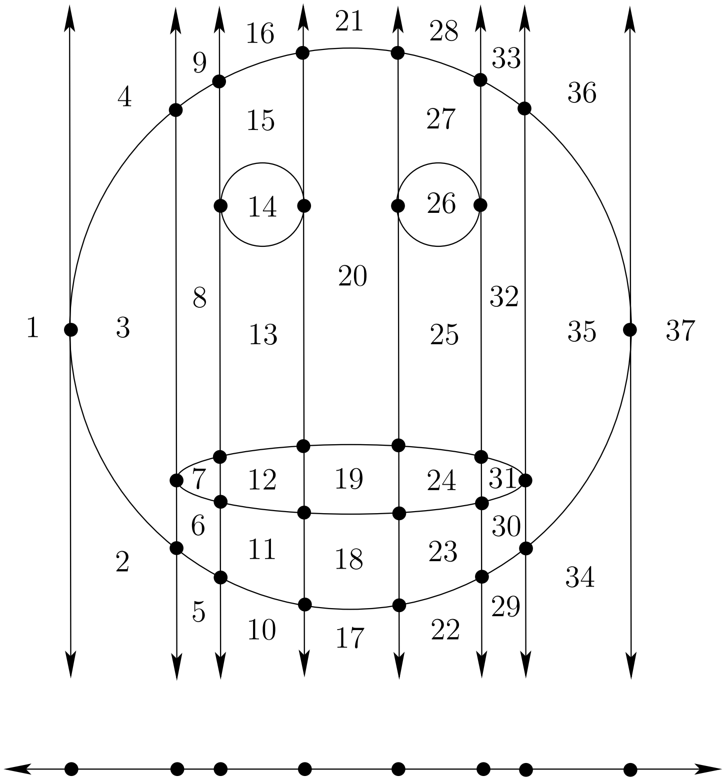
Reasoning about exact geometry



Motion planning progressed after identifying the right spaces.

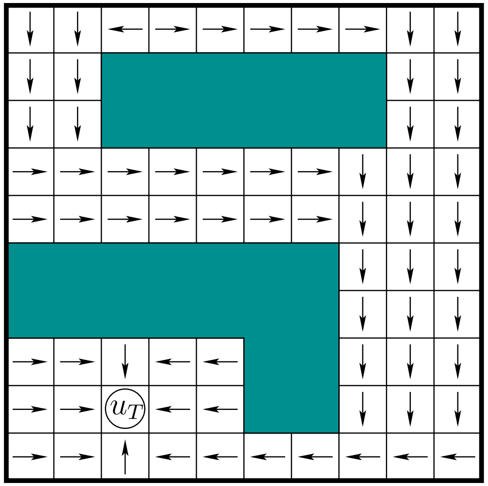
O'Dunlaing, Yap, 1982; Schwartz, Sharir, 1983.

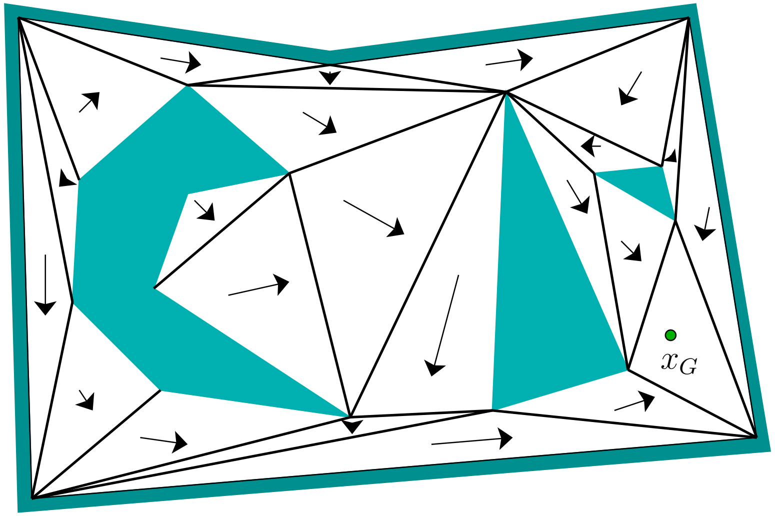


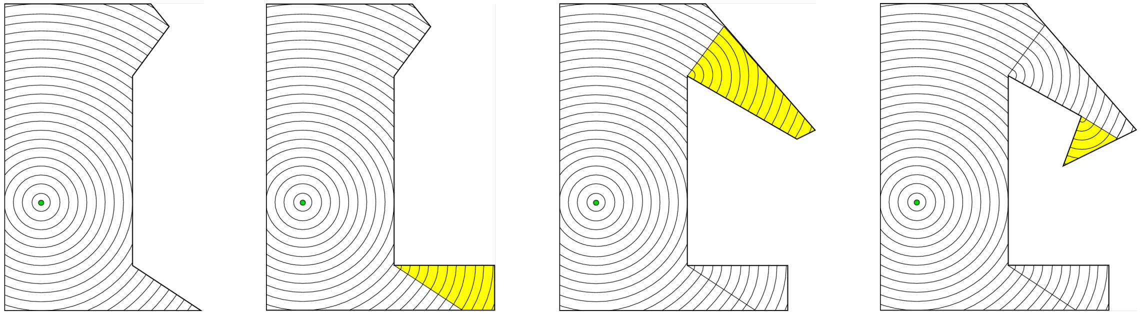


Exact, structure-preserving discretizations.

Beautiful, complete algorithms.







Compute a collision-free velocity field over the C-space.

Generally better than tracking a path.

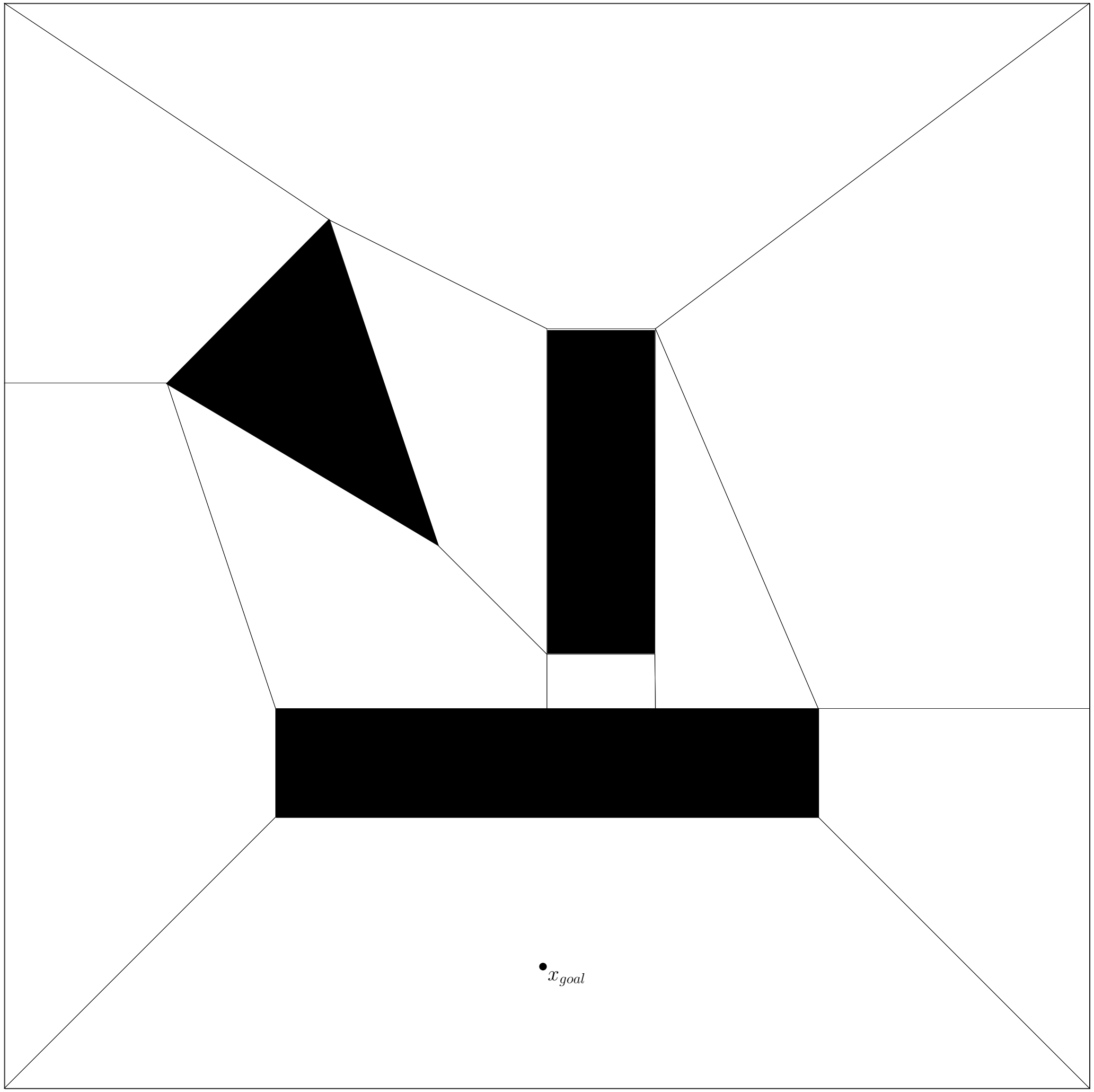
Lindemann, LaValle, CDC 2005; Lindemann, LaValle, RSS, 2006; Lindemann, Hussein, LaValle, CDC 2006.

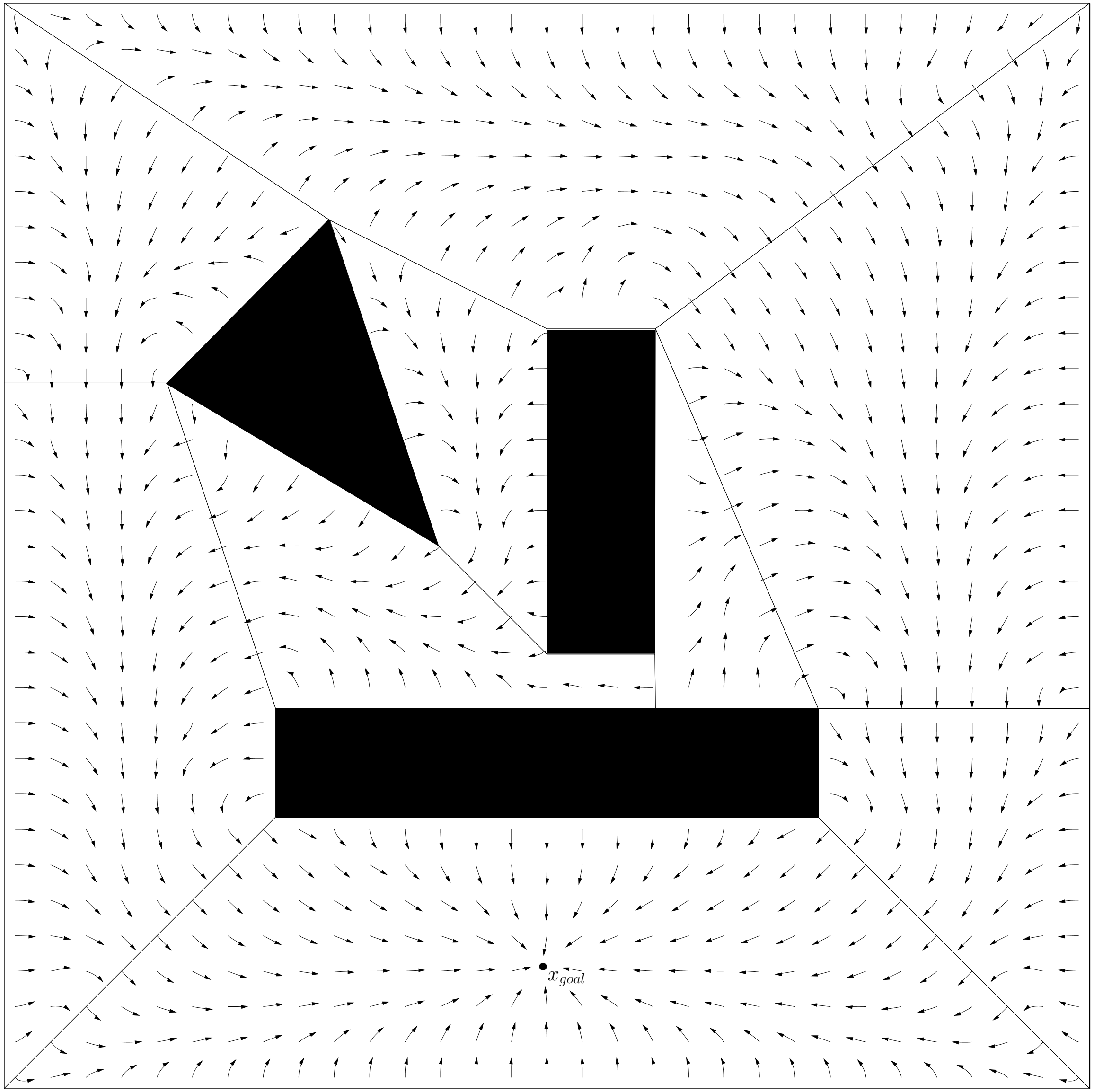
Instead of using the gradient of a navigation function as the vector field, we construct one directly. We do this as follows:

- Partition the space into simple cells.
- Use the cell connectivity graph to determine a high-level motion plan.
- Define local vector fields on each cell which are compatible with the motion plan.
- Appropriately blend the vector fields together to obtain a global vector field.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

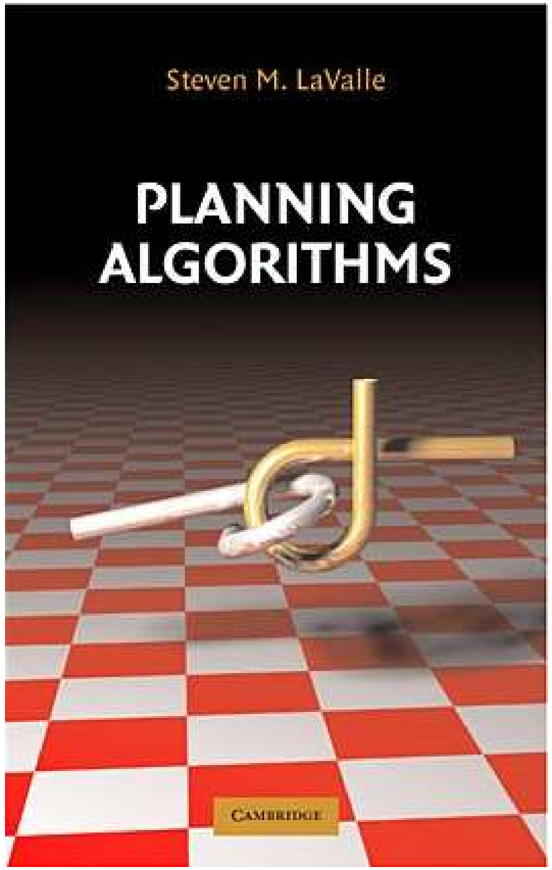
Decomposition




x_{goal}

Free Book

Planning Algorithms Overview
Information Spaces
Gap Navigation Trees
Learning Point Arrangements
Other Problems
Conclusions



PART I
Introductory Material
Chapters 1-2

PART II
Motion Planning
(Planning in Continuous Spaces)
Chapters 3-8

PART III
Decision-Theoretic Planning
(Planning Under Uncertainty)
Chapters 9-12

PART IV
Planning Under Differential Constraints
Chapters 13-15

Free download ($\approx$ 1000 pages): **http://planning.cs.uiuc.edu/**

Also published by Cambridge University Press, May 2006.

Information Spaces

Terminology

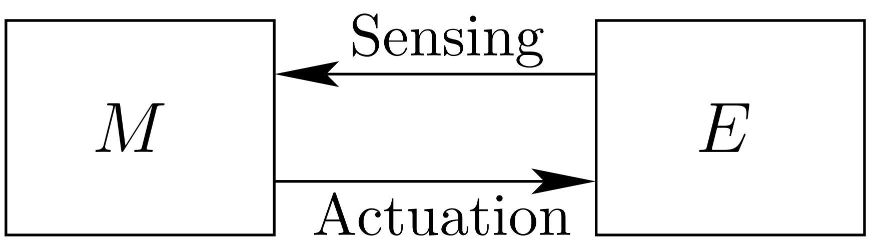
Where have *information spaces* arisen?

Early appearance of concept: H. Kuhn, 1953

- **Extensive form games**
  Unknown state information regarding other players.
- **Stochastic control theory**
  Disturbances in prediction and measurements cause imperfect state information.
- **Robotics/AI**
  Uncertainty due to limited sensing.

Alternative names: belief states, knowledge states, hyperstates

The configuration space is the crucial space for mechanics, motion planning.
The state (phase) space is the crucial space in system theory.



The **information space** is the natural space that arises for autonomous systems with sensing and actuation uncertainties.

The *history I-state* at time t is:

$$\eta_t = (\tilde{u}_t, \tilde{y}_t)$$

with

Input space:	U
Input history:	$\tilde{u}_t : [0, t) \rightarrow U$
Observation space:	Y
Observation history:	$\tilde{y}_t : [0, t] \rightarrow Y$

The *history I-space*, $\mathcal{I}_{hist}$, is the set of all possible η_t for all $t \in [0, \infty)$.

Problems:

- $\mathcal{I}_{hist}$ is enormous!
- How do we know that goals are achieved?

Bring in a State Space

There is a state space, X.

The *state* could represent robot configuration, velocity, environment model, and so on.

Some potential interference from "nature":

Nature input history: $\tilde{\theta}_t : [0, t) \rightarrow \Theta$

Nature observation history: $\tilde{\psi}_t : [0, t] \rightarrow \Psi$

State transition equation: $x' = \Phi(x, \tilde{u}_t, \tilde{\theta}_t)$

Observation equation: $y = h(\tilde{x}_t, \tilde{\psi}_t)$

Initial conditions: η_0 defined, and $\eta_t = (\eta_0, \tilde{u}_t, \tilde{y}_t)$.

How to Use the I-Space

Two approaches:

1. Take all of the information available, and try to estimate the state.
   A feedback plan is expressed as $\pi : X \to U$.

2. Solve the task entirely in terms of an information space.
   A feedback plan can be expressed as $\pi : \mathcal{I}_{hist} \to U$.

The second is more interesting (to me, at least).
Attempt to "live" in the information space!
Estimation is **sufficient**, but often not **necessary**.

Making Derived I-Spaces

Construct *information mappings* (**I-maps**) to transform the I-space:

$$\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{der}$$

Define a plan as $\pi : \mathcal{I}_{der} \rightarrow U$.

Examples:

State estimation:	$\kappa : \mathcal{I}_{hist} \rightarrow X$	$\eta_t \mapsto \hat{x}(t)$
Time feedback:	$\kappa : \mathcal{I}_{hist} \rightarrow [0, \infty)$	$\eta_t \mapsto t$
Sensor feedback:	$\kappa : \mathcal{I}_{hist} \rightarrow Y$	$\eta_t \mapsto y(t)$
Limited memory:	$\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{mem}$	$\eta_t \mapsto \eta_{t-1,t}$
Nondeterministic:	$\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{ndet}$	$\eta_t \mapsto X(t) \subseteq X$
Probabilistic:	$\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{prob}$	$\eta_t \mapsto p(x \mid \eta_t)$
Kalman filter:	$\kappa : \mathcal{I}_{hist} \rightarrow \mathcal{I}_{gauss}$	$\eta_t \mapsto (\mu_t, \Sigma_t)$

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Sufficient I-Maps

Try to "live" in some $\mathcal{I}_{der}$.

We need to make an *information transition equation*.

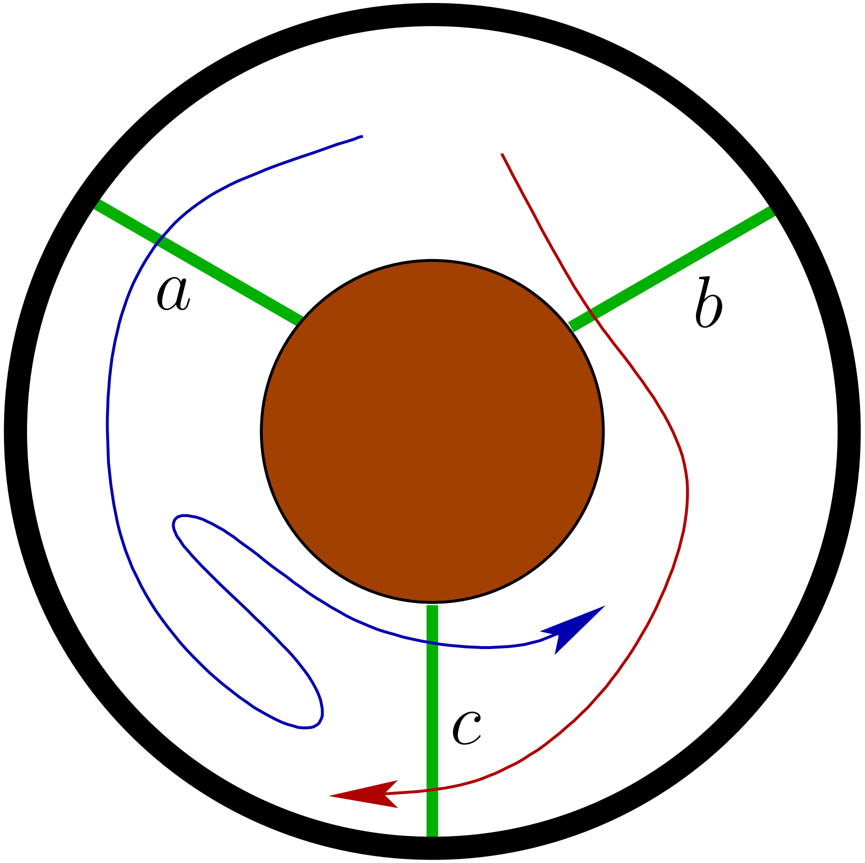
$$\begin{tikzcd}[column sep=huge, row sep=huge]
\mathcal{I}_{hist} \arrow[r, "f_{\mathcal{I}}"] \arrow[d, "\kappa"'] & \mathcal{I}_{hist} \arrow[d, "\kappa"'] \\
\mathcal{I}_{der} \arrow[r, dashed, "f_{\mathcal{I}_{der}}?"] & \mathcal{I}_{der}
\end{tikzcd}$$

$$\text{Stage } k \qquad\qquad \text{Stage } k+1$$

Does the derived I-state contain sufficient information for computing transitions?

Enables the robot's memory to be smaller.

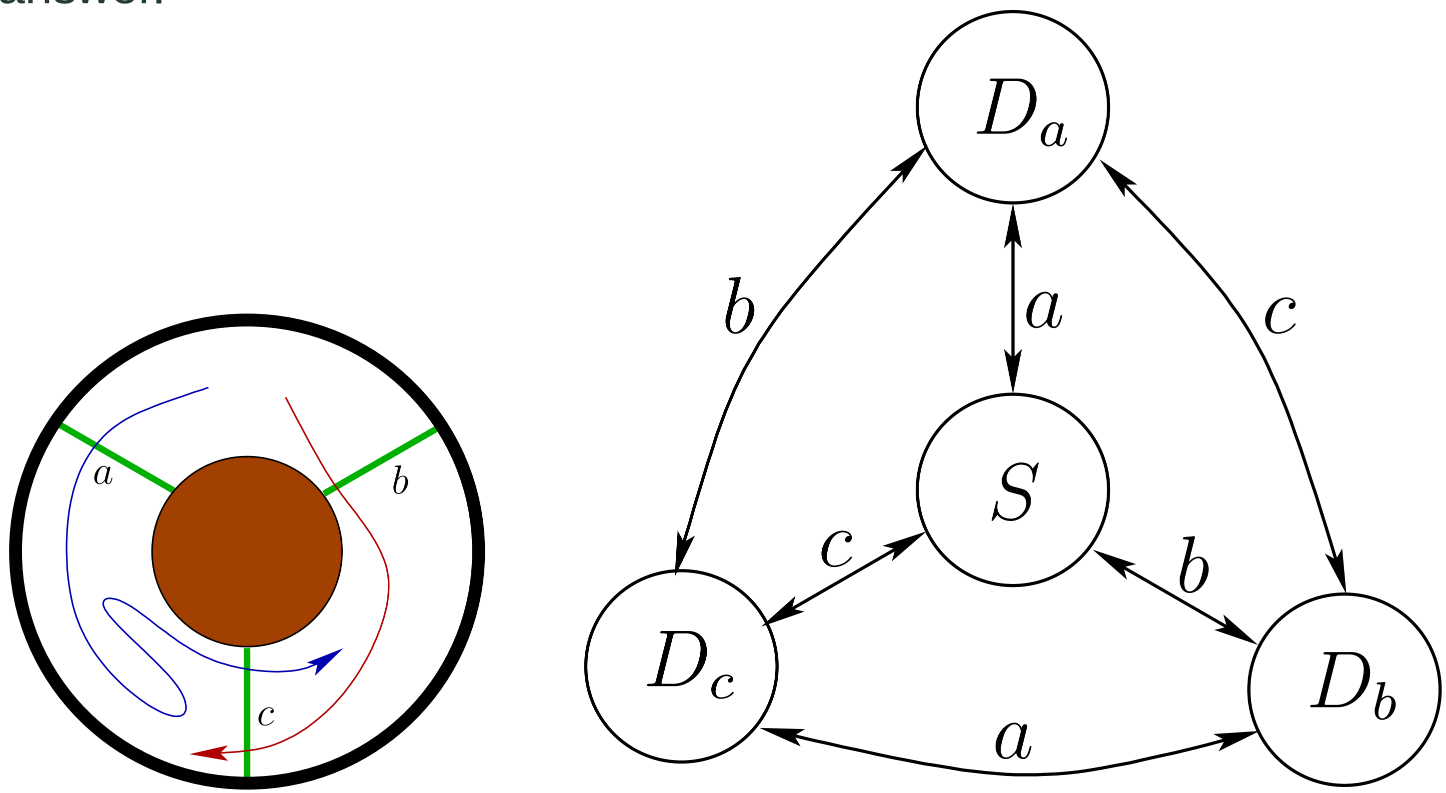
A Simple Inference Problem



History I-state: $abbacbacababab cabcbba$

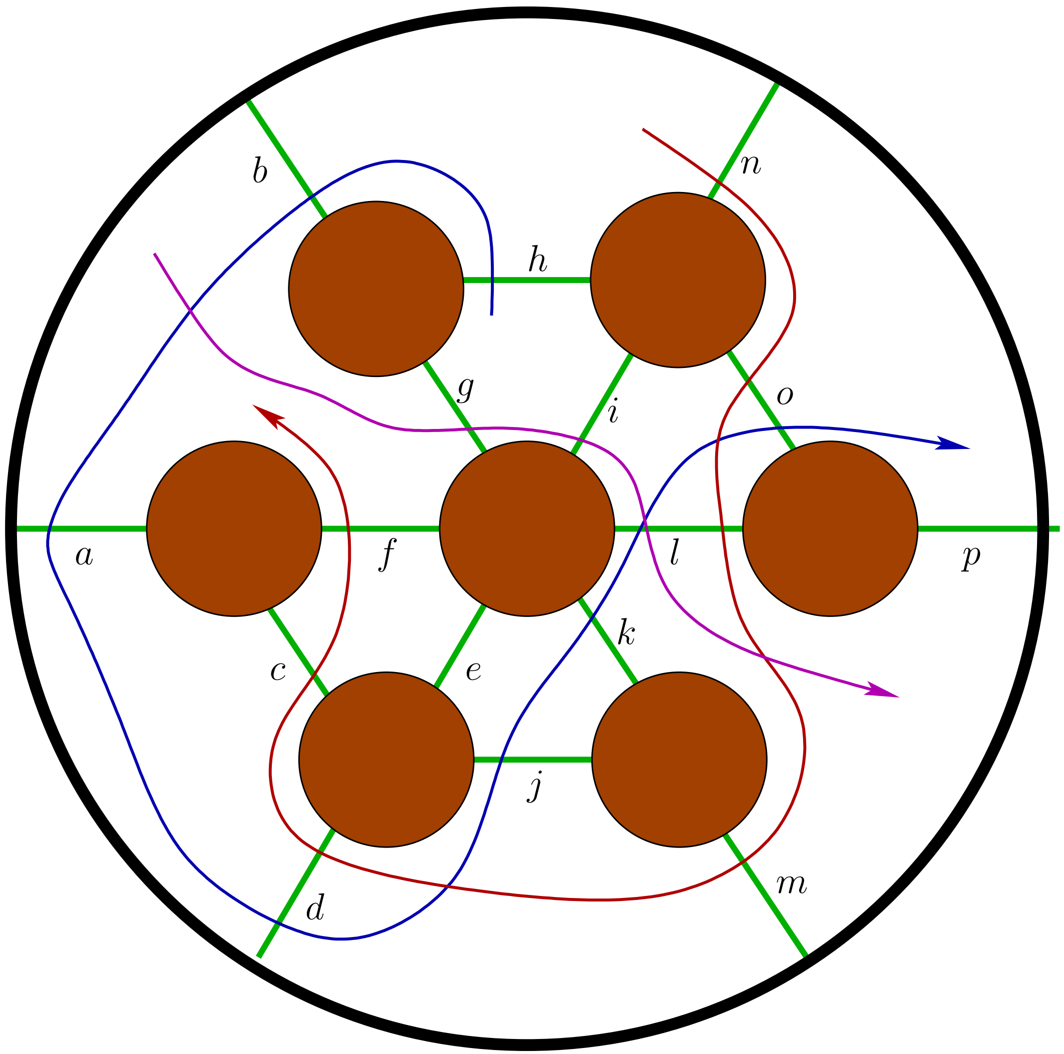
Question: Are the agents in the **same** room?

This two-bit machine can read strings of any length and correctly report the answer.



[Worked out with F. Cohen and B. Tovar]

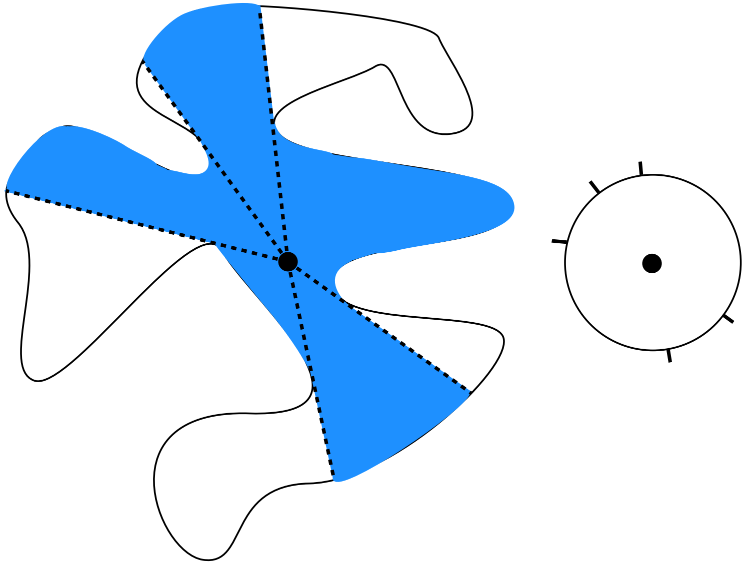
Generalizing the Inference Problem



More holes, more beams, more agents, ...

Gap Navigation Trees

Tovar, Guilamo, Murrieta, LaValle, 2003-2006.



- Bounded contractible planar region with piecewise-analytic boundary
- Robot can only sense depth discontinuities
- Environment representation is not given
- No distance or angular measurements
- No odometry, GPS, or compass
- Motion primitive: **Chase a gap**

Some Work with Similar Motivation

- Sensing only what is needed
  [Erdmann, Mason, 88; Donald, Jennings, 91; Rimon, Canny, 94]
- Minimal representations for manipulation
  [Goldberg, 93]
- Bug algorithms
  [Lumelsky, Stepanov, 87; Kamon, Rivlin, Rimon, 96; Kamon, Rivlin, 97]
- Shortest paths without maps
  [Papadimitriou, Yannakakis, 89]
- Landmark-based navigation
  [Hait, Simeon, Taix, 97; Taylor, Kriegman, 98]
- Efficient updates to the visibility polygon
  [Aronov, Guibas, Teichmann, Zhang, 98]
- On-line target tracking
  [Gonzalez-Banos, Lee, Latombe, 02]

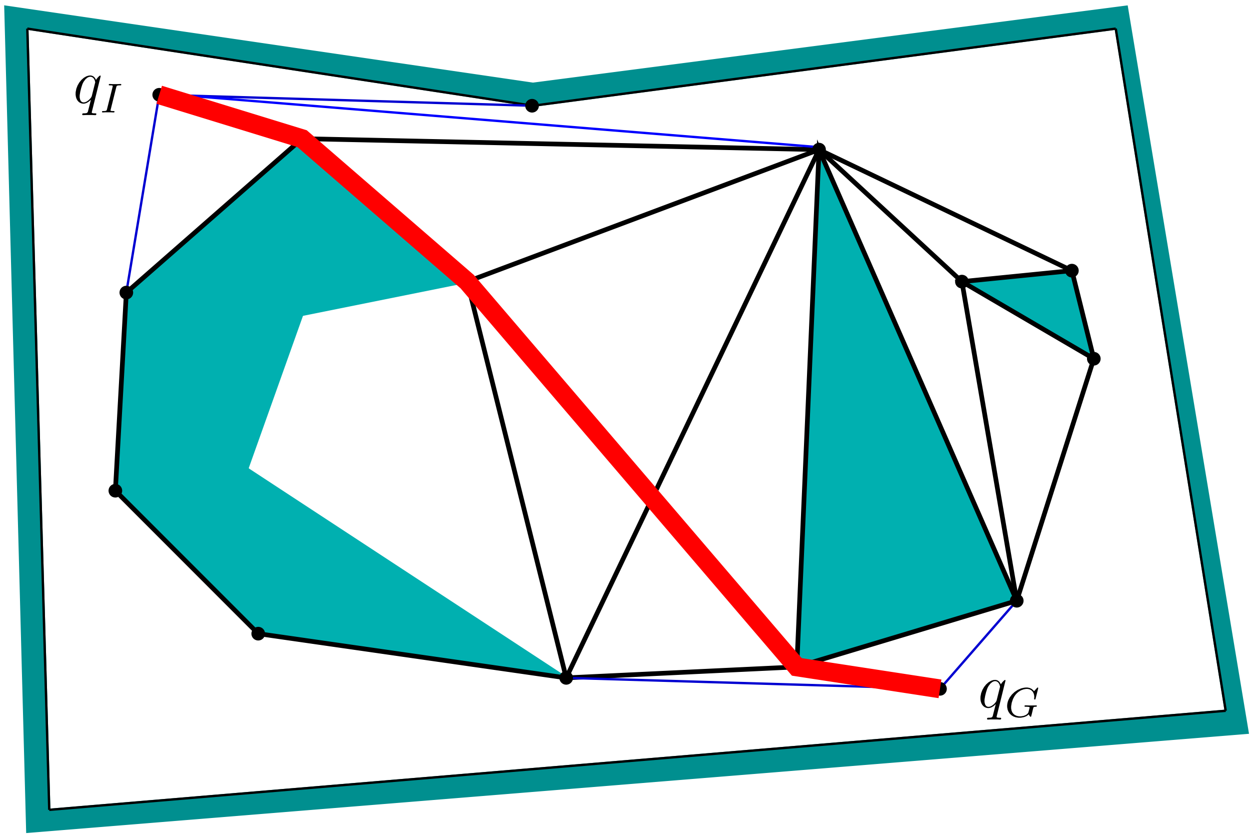
Tasks:

- Optimal planning and navigation to any prescribed location
- Retrieve and deliver static objects optimally

Assumptions:

- Drop the robot into unknown, bounded, simply-connected, piecewise-smooth, planar region.
- Minimal sensing model: gap sensor

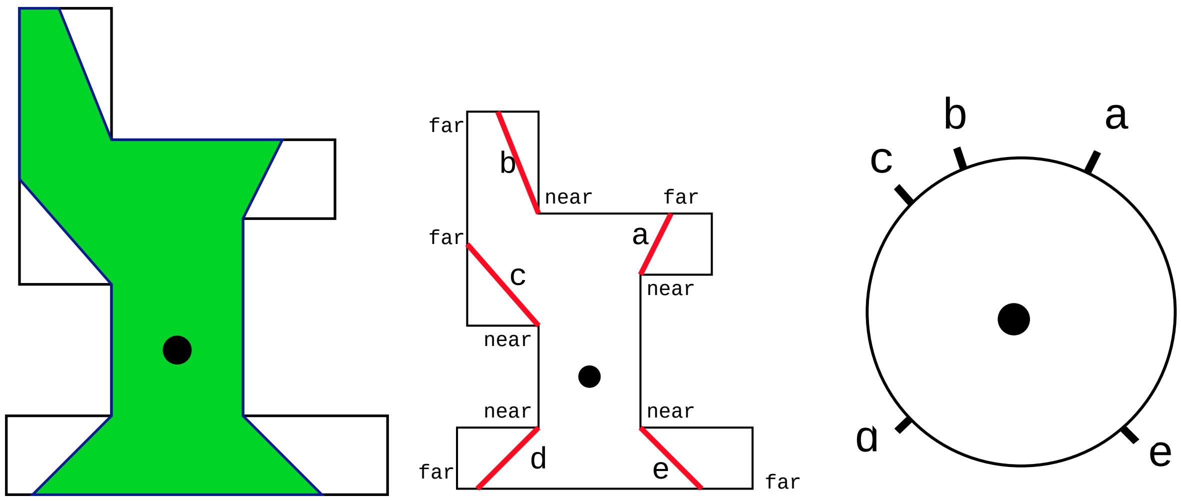
Shortest Paths with Perfect Information



Note that geodesics follow bitangents.

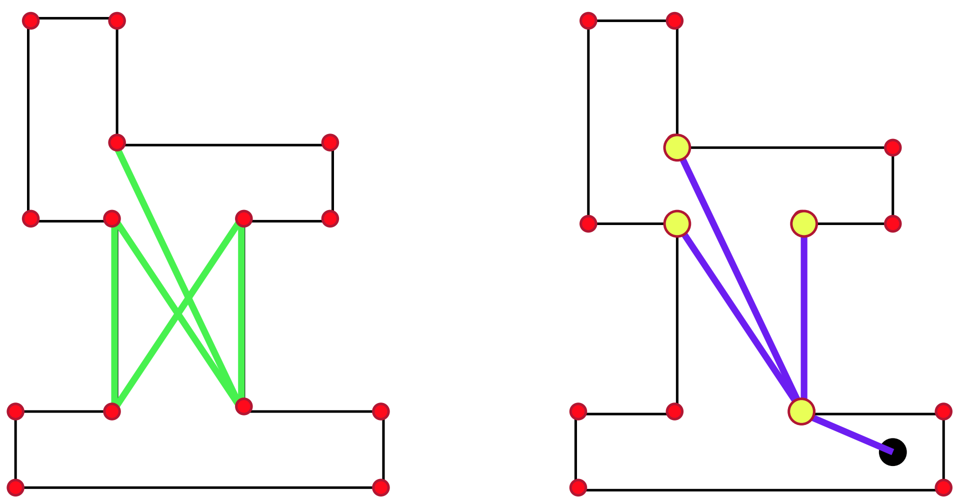
Recall the Minimal Sensing Model

- A ’gap’ is a discontinuity in depth information.
- A ’gap-sensor’ is able to track the gaps at all time.
- Only gap angular order is preserved. Not *exact* angular position.



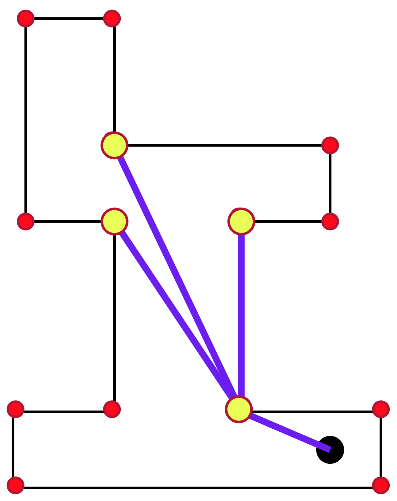
A Visibility Tree from a Fixed Location

- Choosing a source point, compute the shortest path to any other location.
- Paths of the visibility tree belong to the bitangent graph.



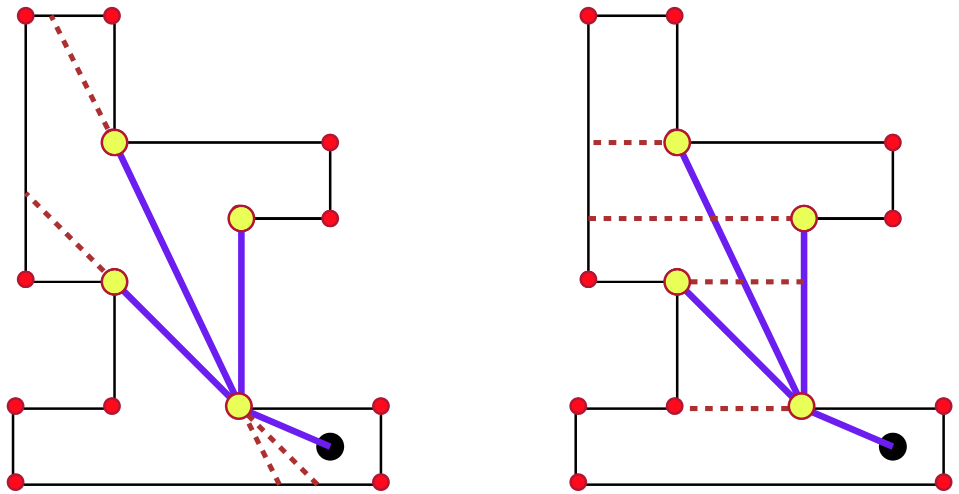
Visibility Tree (cont'd)

- Knowing the visibility tree of the current location, the robot can reach any other location optimally.
- Only useful if perfect localization is assumed.
- Is it possible to obtain the same paths with only online-sensor measurements?



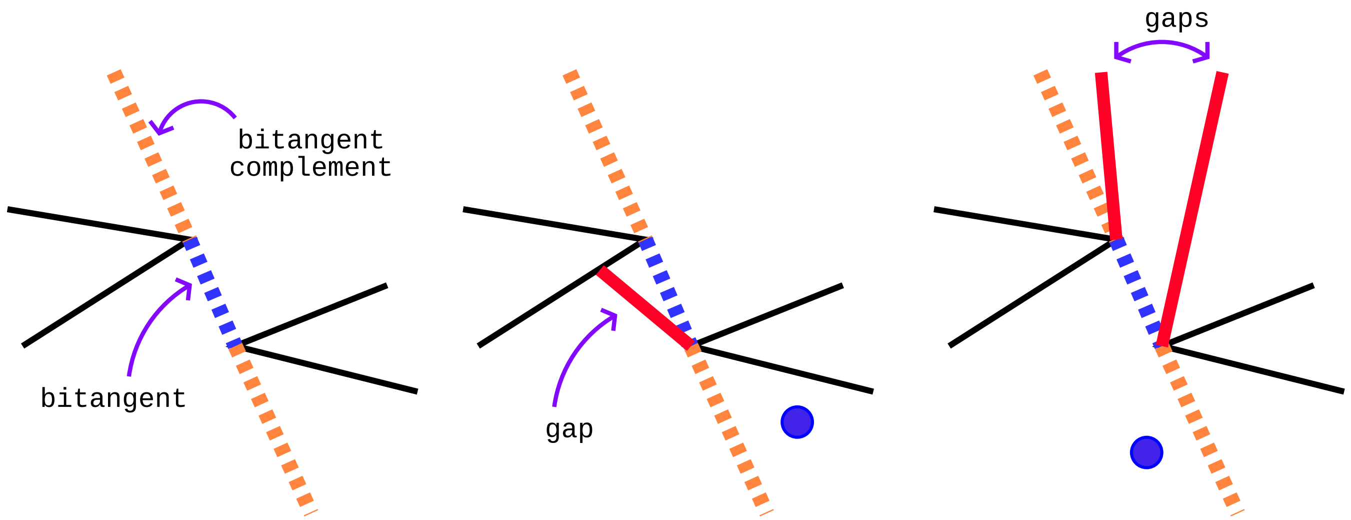
Properties of the Visibility Tree

- A robot traveling in a visibility tree sees *bifurcations* and *dead-ends.*
- These happen when the robot crosses *inflections* and *bitangents complements.*



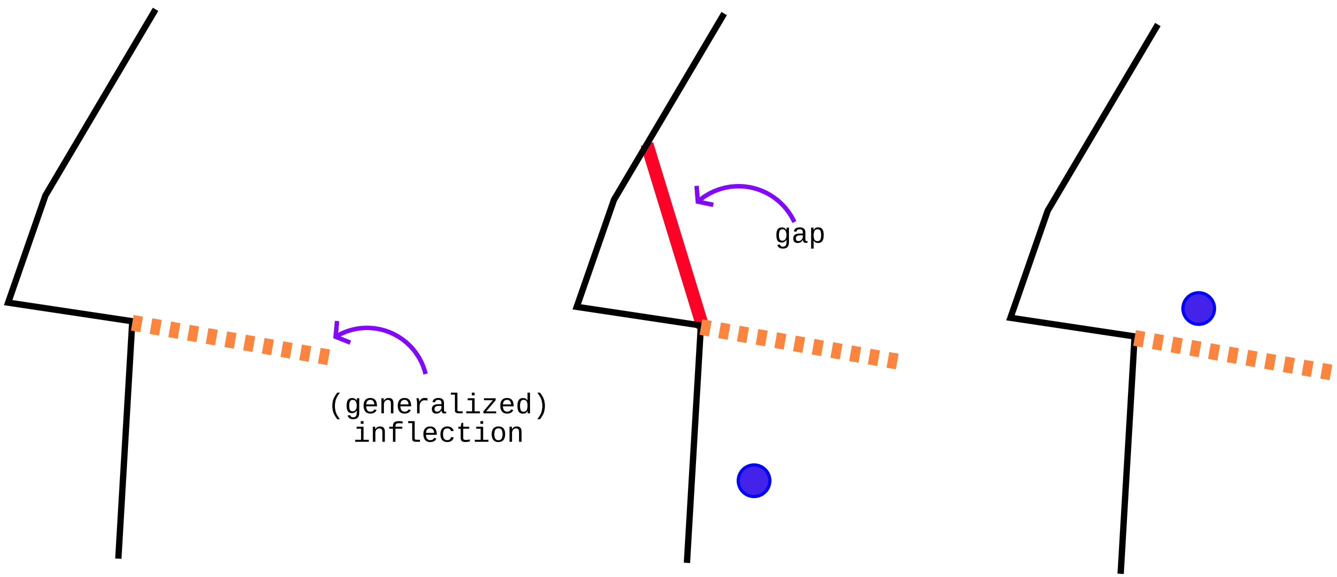
Bitangents

A bitangent is a closed line segment whose supporting line is tangent at two points of the environment boundary.



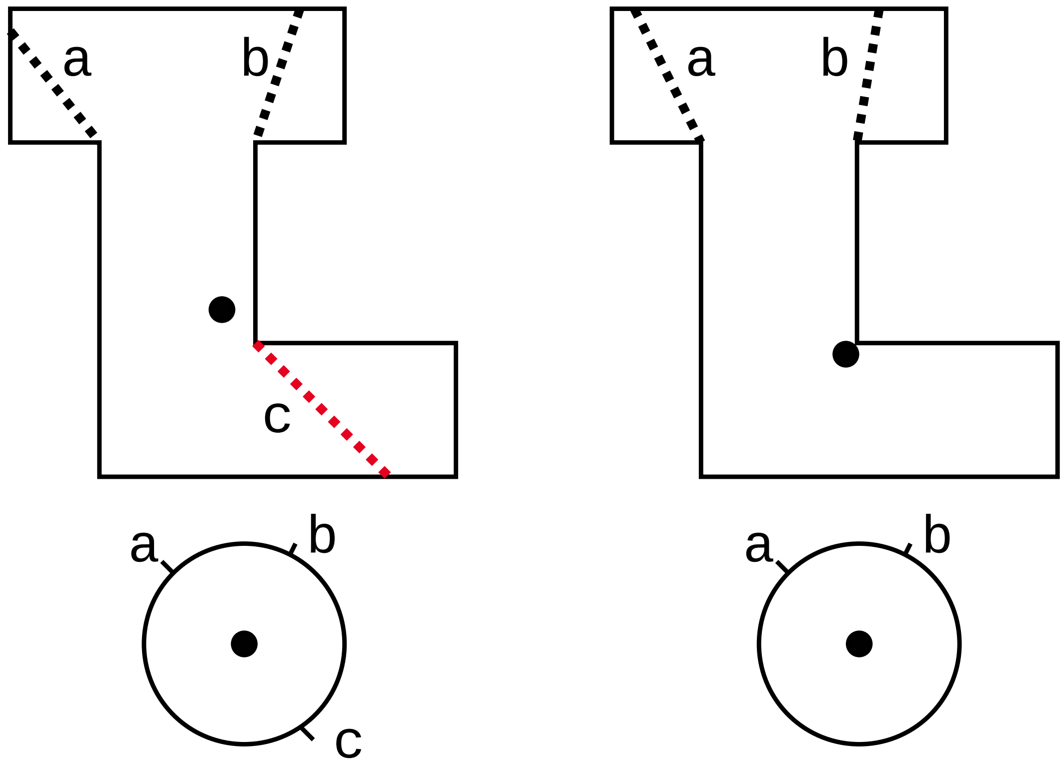
Note: The boundary does not have to be polygonal.

An inflection line is found by extending a ray outward from an inflection point of the environment boundary.

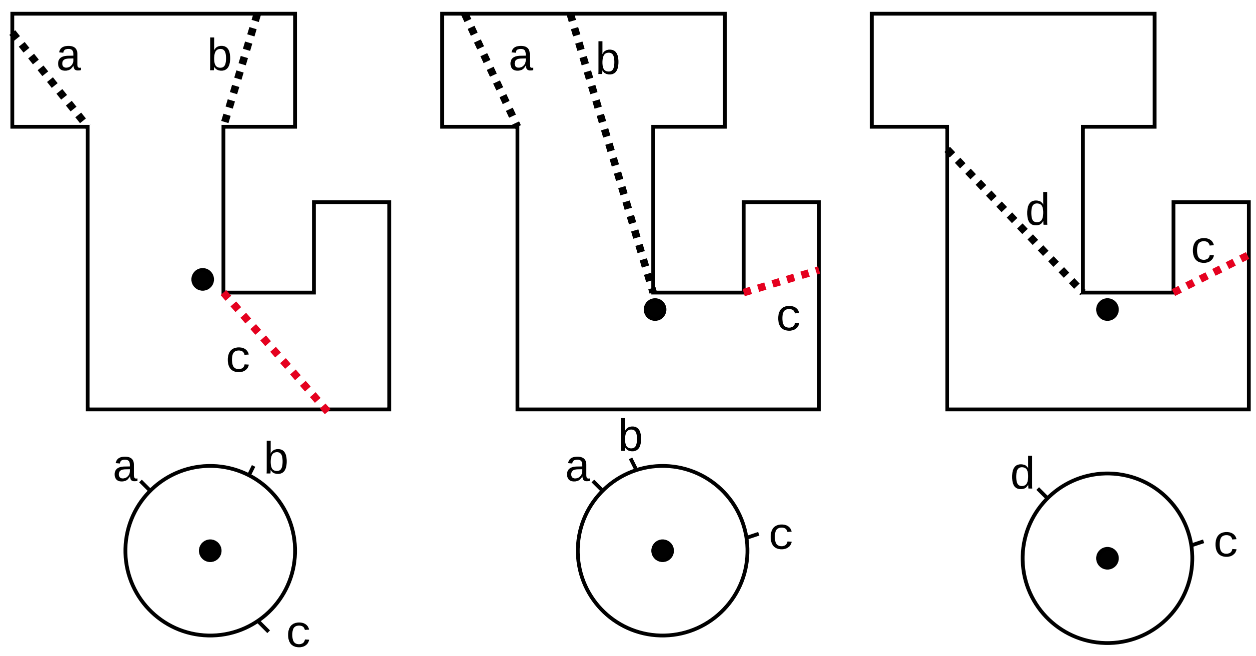


The boundary does not have to be polygonal.

Appearances — Disappearances

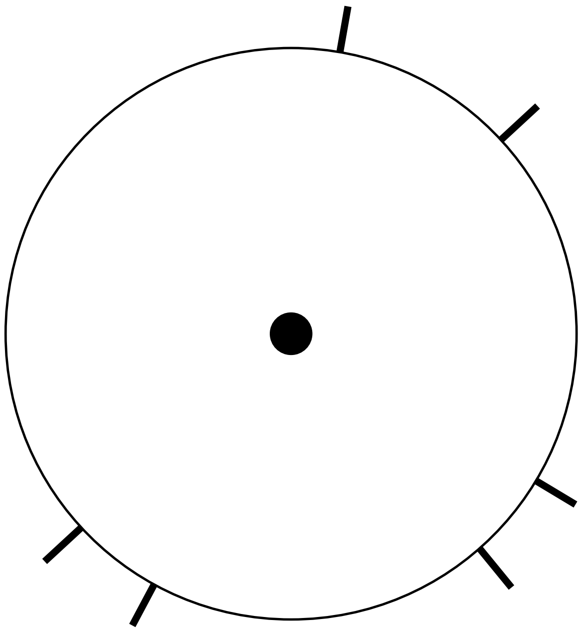


Splits — Merges

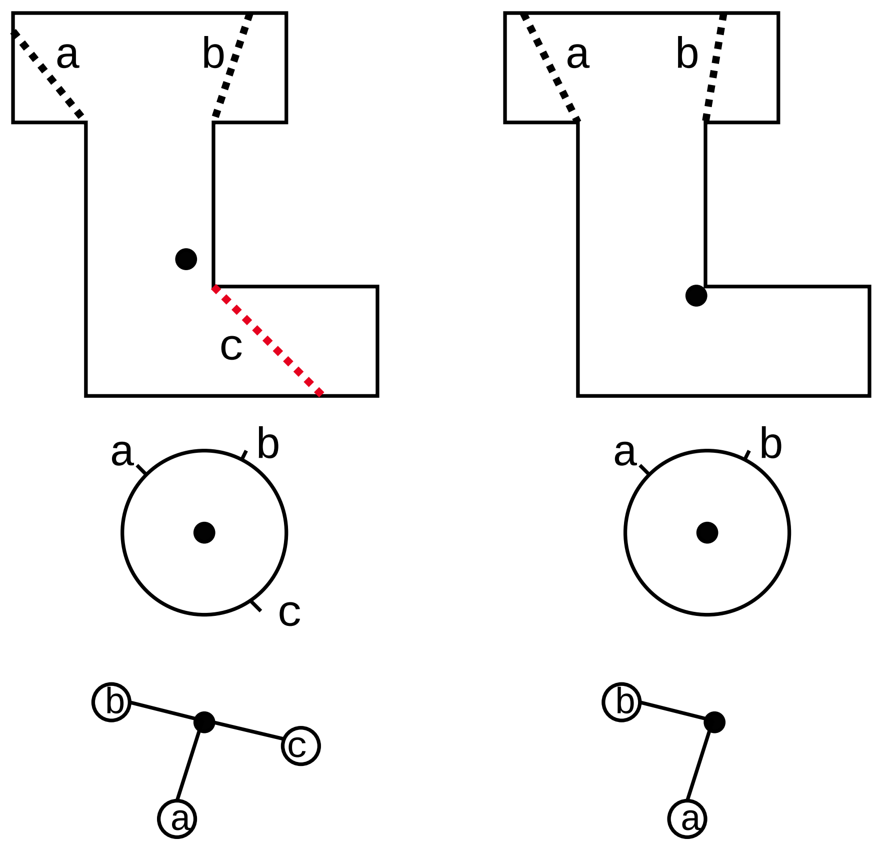


A Tree is Maintained Relative to the Robot

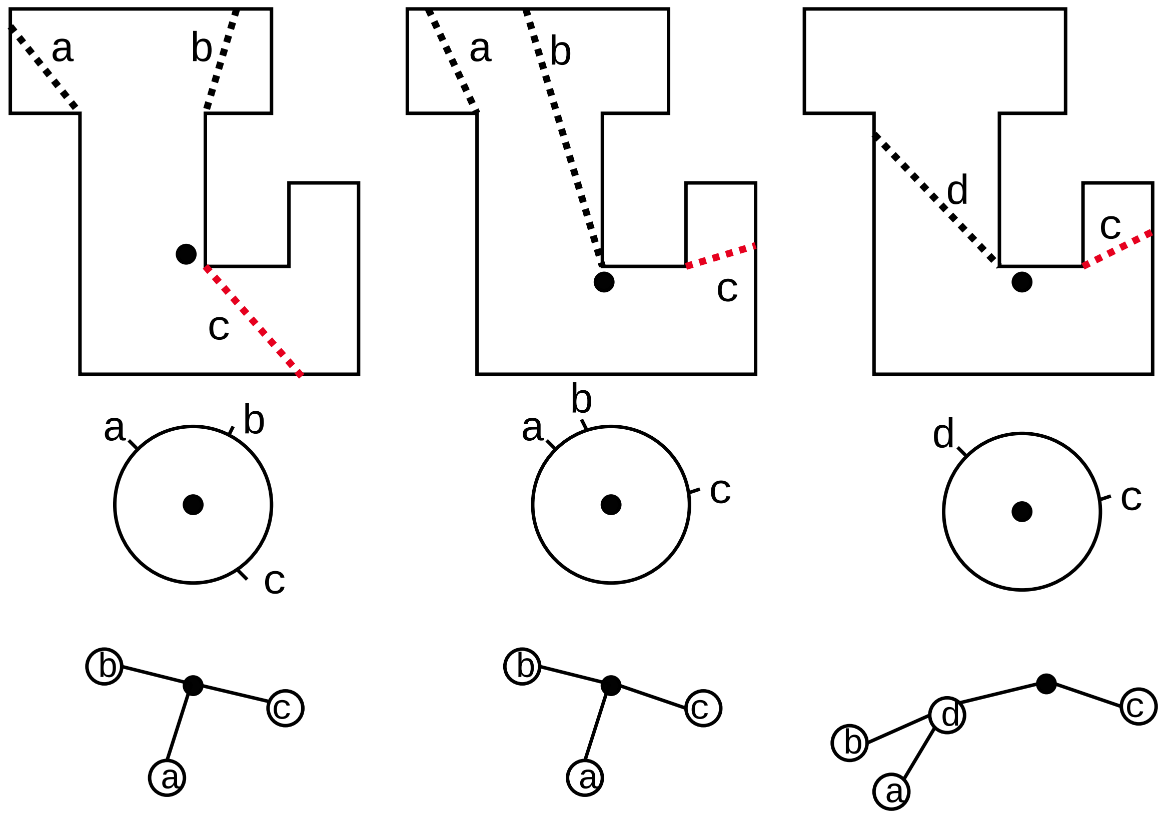
- The tree root moves with the robot.
- Every node in the tree represents a gap.
- Every child of the root represents a gap currently visible.



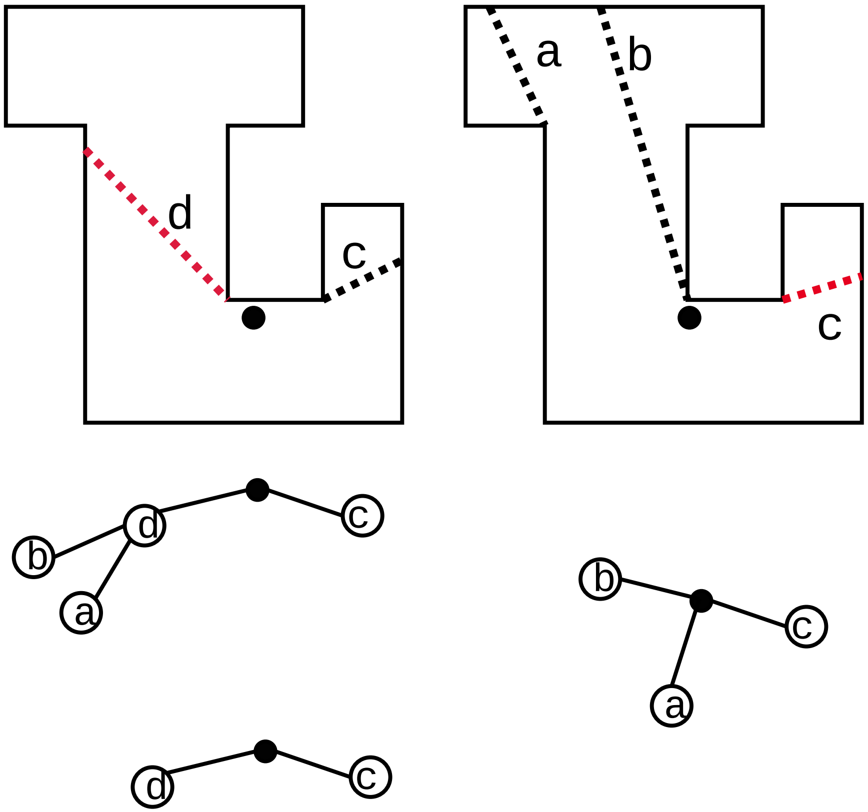
Add or remove a leaf of the root, preserving the angular order



The two gaps (nodes) merging become the child of a new node.

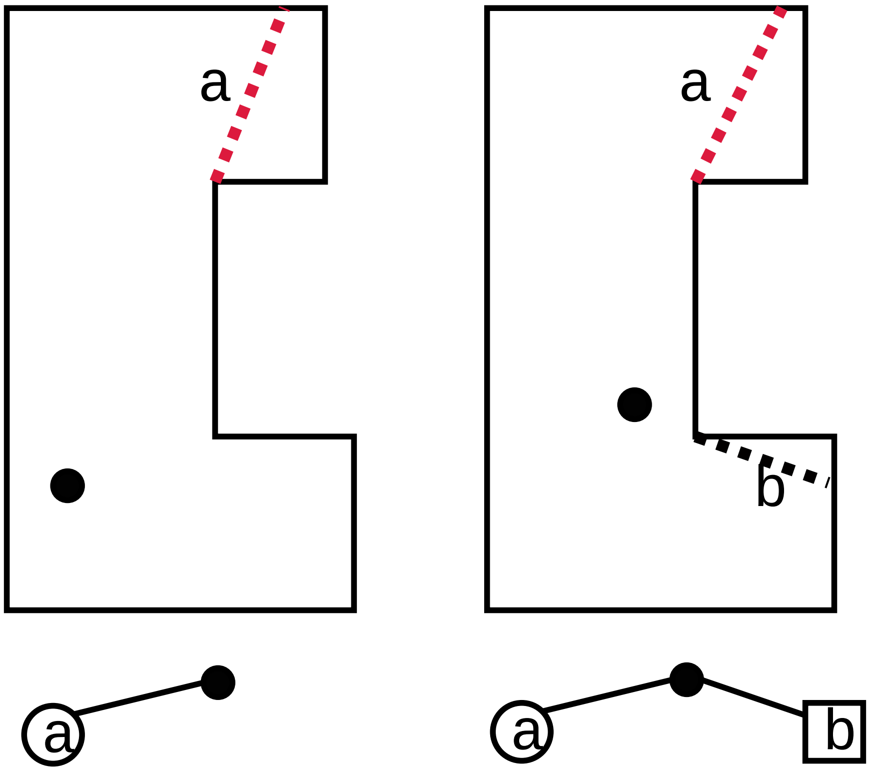


If the splitting node have children, these become children of the root. Otherwise, the node is replaced with the two nodes representing the new gaps.

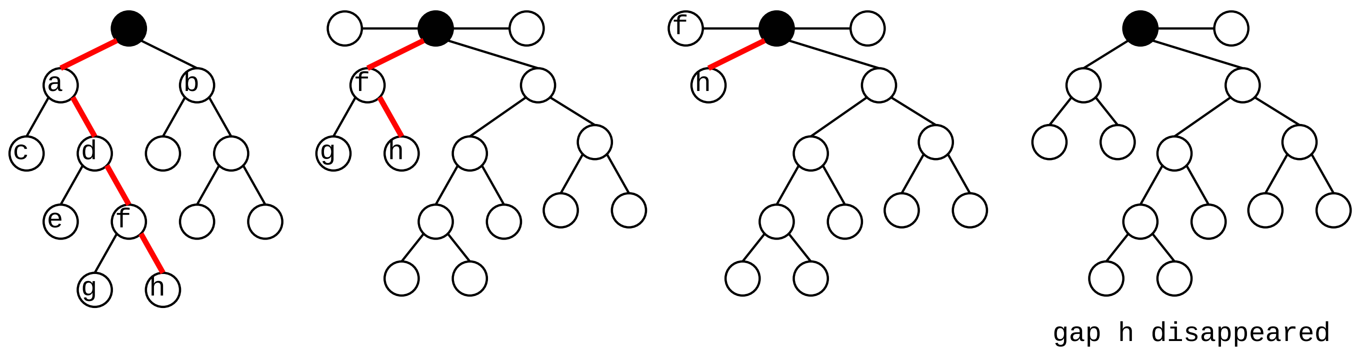


Appearances: Primitive Nodes

Appearances have a special meaning. They generate *primitive nodes*, to indicate *already seen*.



The Control Model



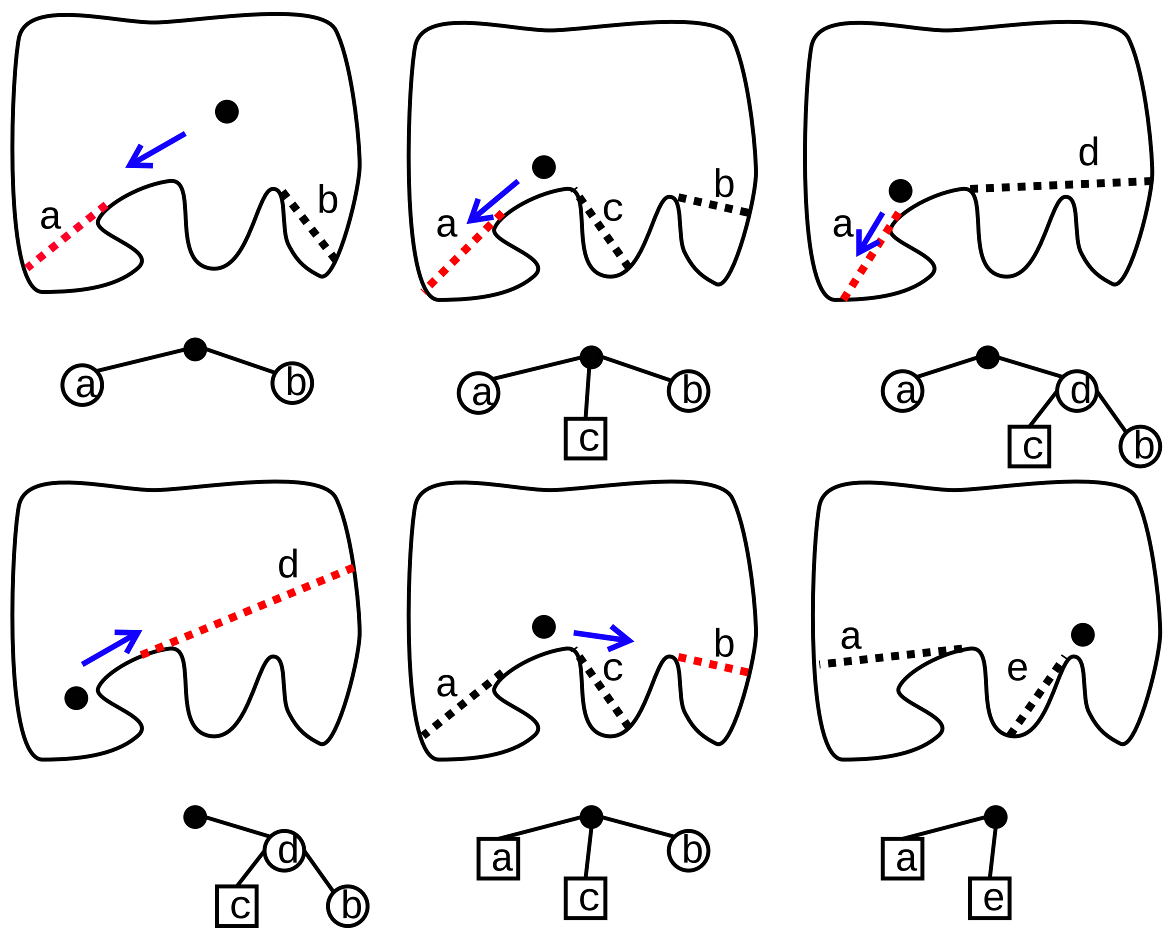
To chase gap h, chase a that will split, and then follow d, and so on, until h disappears.

Keep encoding all of the critical events.

Robust gap chasing: Minguez, Montano, IEEE TRA Feb 2004

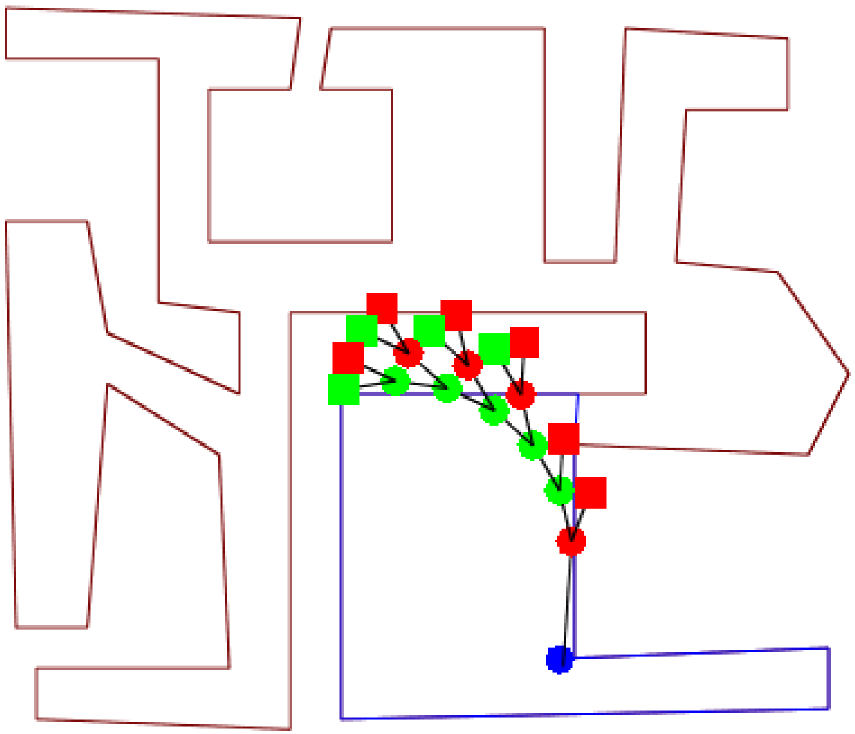
Strategy for Constructing Full Tree

Chase every non-primitive leaf:



Eventually, all leaves become primitive.

Computed Example

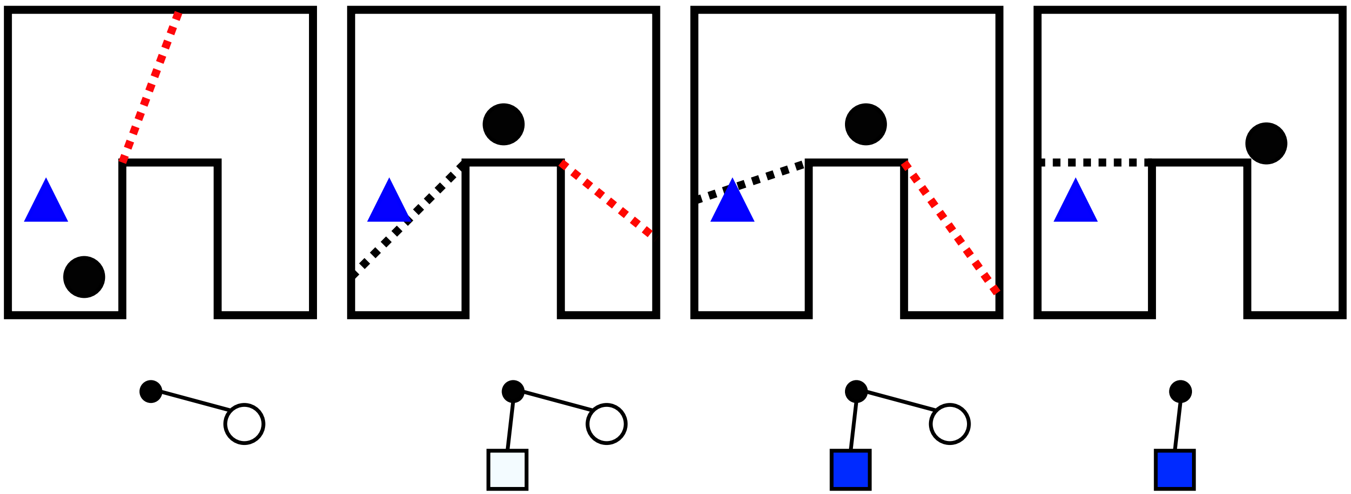


Red means the hidden portion is to the right.

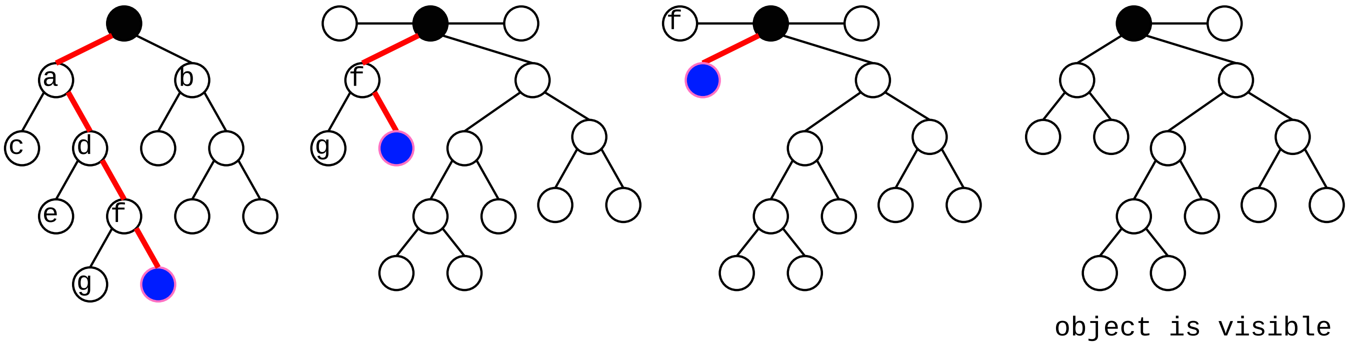
Yellow means to the left.

Retrieving Objects

- There are no “coordinates”; goals are specified by gaps.
- Indicate from where the object becomes visible.
- Associate objects with gaps.
- A gap “merges” with an object in the association.
- To retrieve an object optimally, follow the path to the associated gap



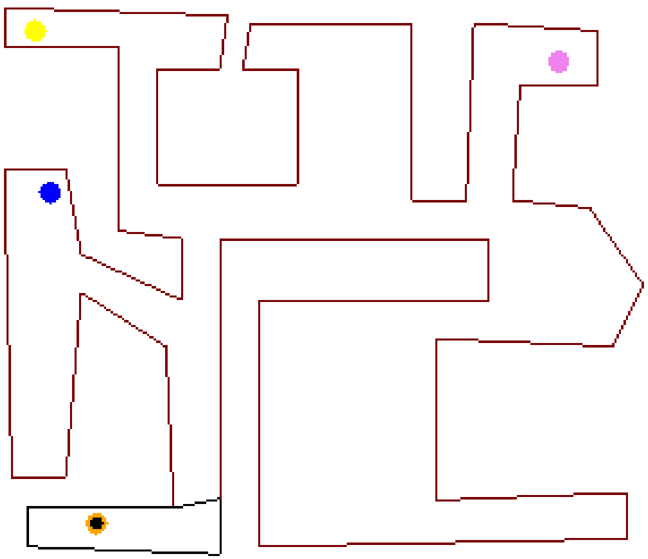
Retrieving Objects (Cont'd)

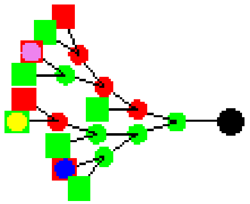


To retrieve the *blue* object, chase the associated gap.

The object is "hidden" behind the gap. The gap encodes the last time it was visible.

A Computed Example of Delivering Objects





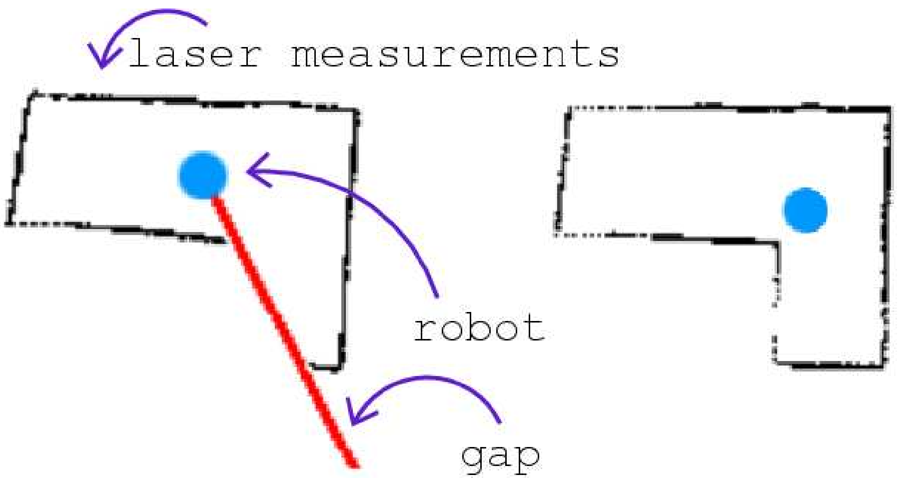
Model Validation on a Real Robot

Pioneer P2-DX, differential drive, two SICK lasers, on-board computations.

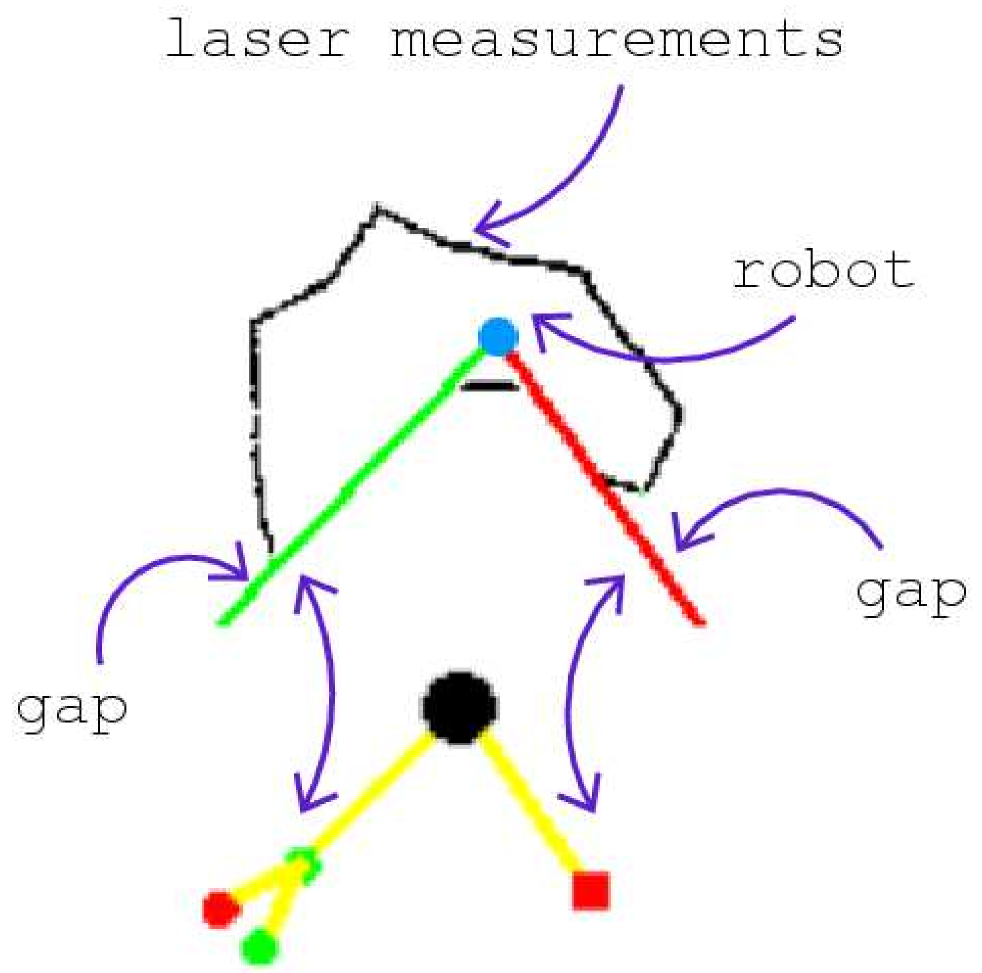
Model Validation on a Real Robot (Cont'd)

Gap disappearing

Gap disappearing



Model Validation on a Real Robot (Cont'd)



What Did That Have to Do With I-Spaces?

Let e be the environment, described as piecewise-analytic, closed curve in $\mathbb{R}^2$.

Let q be the configuration in the environment, $q \in SE(2)$.

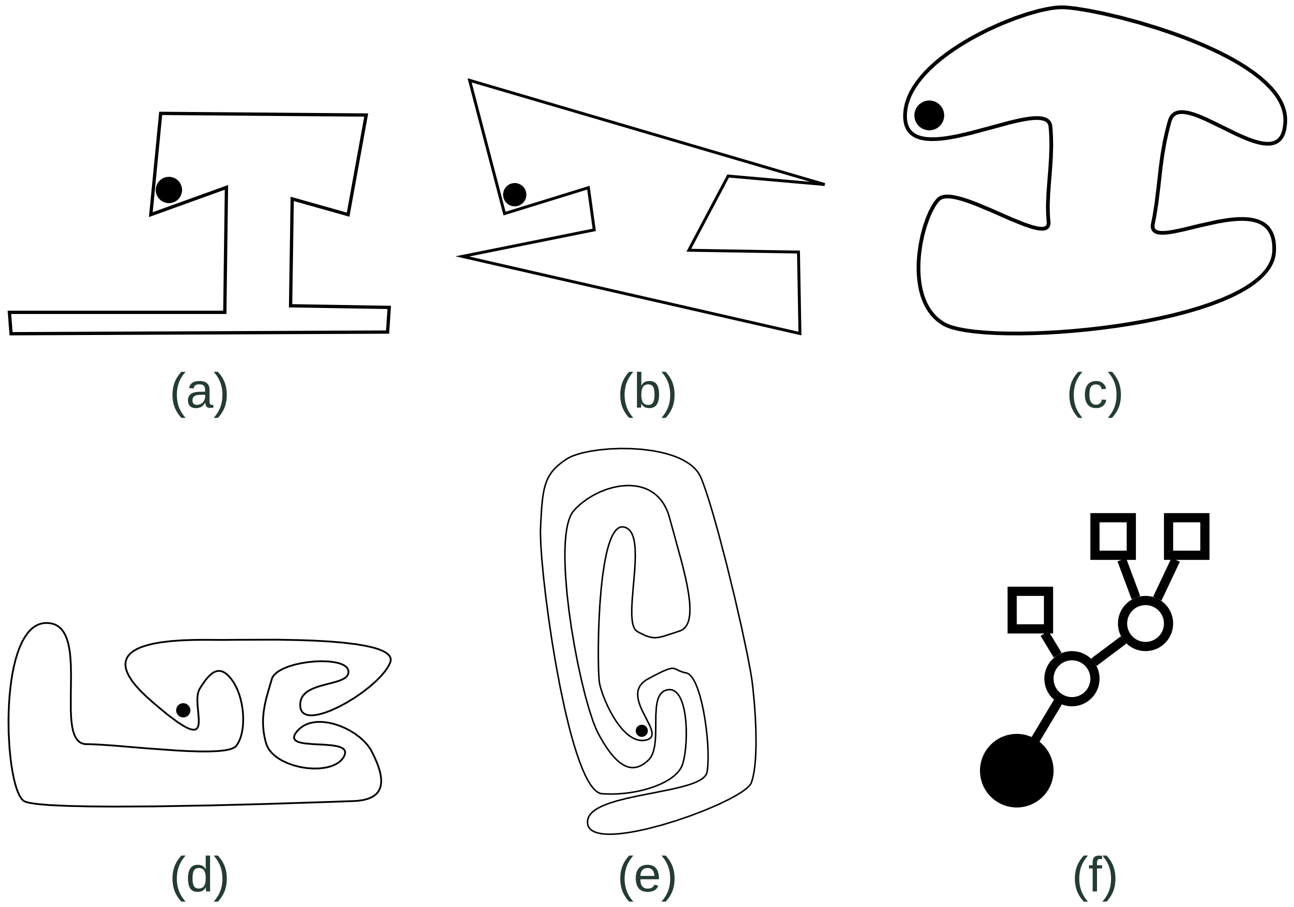
The *state* is (e, q).

Nondeterministic I-states:

$$\{(e, q) \mid e \text{ and } q \text{ are consistent with gap sensor and action histories}\}$$

We are solving tasks without ever knowing the true state.

Distance-Optimal Navigation Without Measuring Distances



Each nondeterministic I-state includes numerous environments and configurations within those environment.

The robot does not have to distinguish!

Extensions and Applications of GNTs

Multiply-Connected Environments:

- The trees can be extended.
- Problem of distinguishing holes
- Problem of knowing when a hole is completely traversed.
- Paths are locally optimal (within homotopy class).

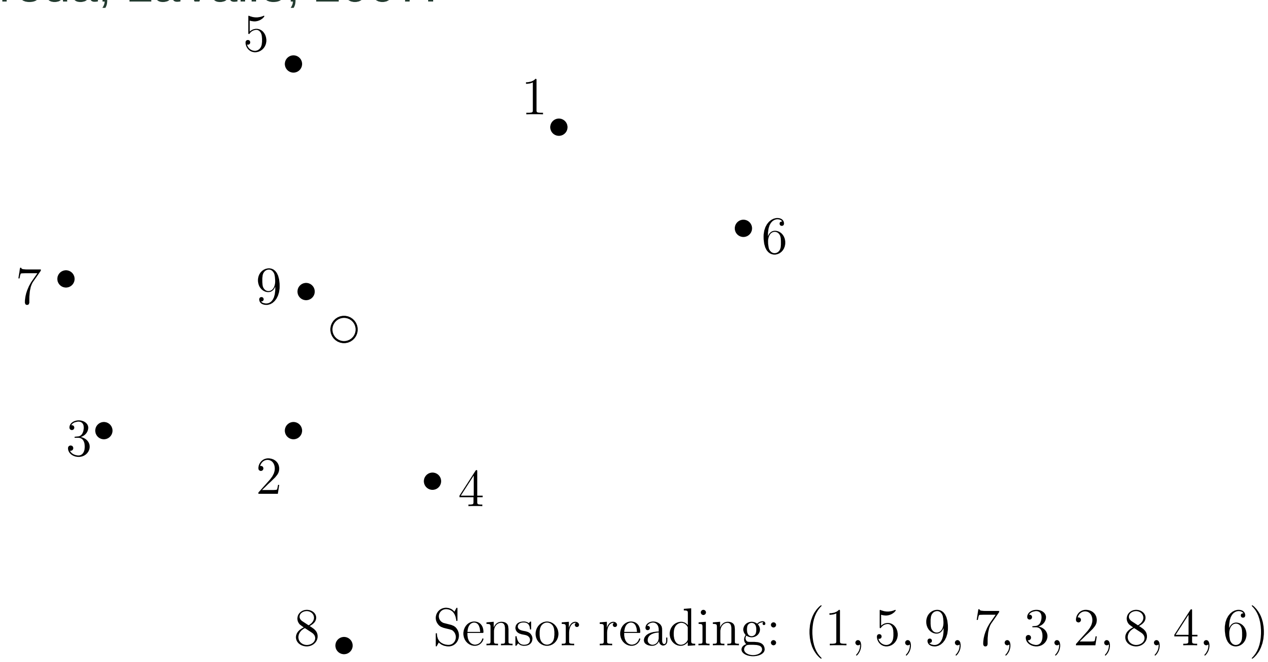
Visibility-Based Pursuit-Evasion:

- Search for evaders using tree-based navigation.
- Maintain binary labels on gaps.

Learning Point Arrangements

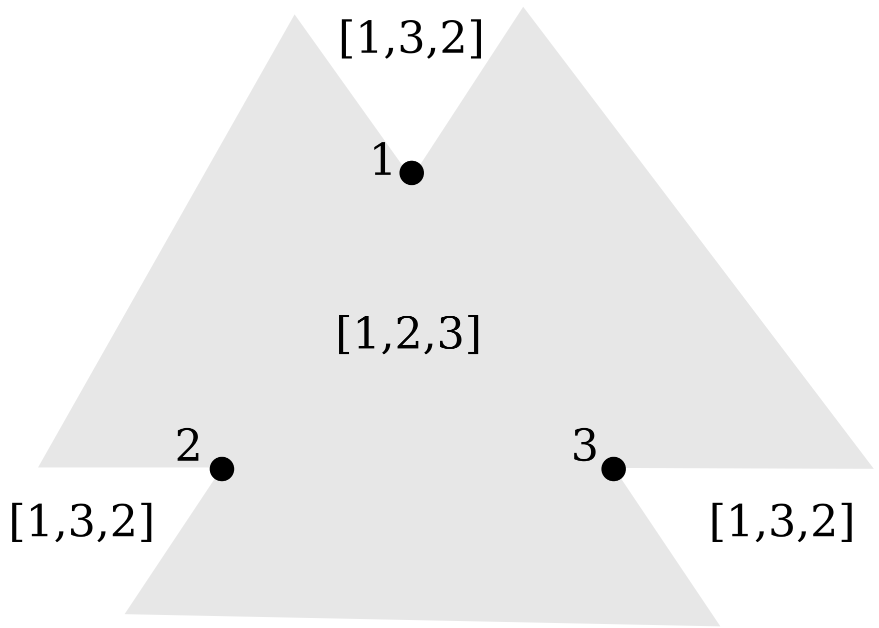
Landmark-Based Navigation Without Distances

Tovar, Freda, LaValle, 2007.



- There are n labeled landmarks in $\mathbb{R}^2$.
- Coordinates are unknown, always.
- Motion command: "Go to landmark i"
- Sensor gives only cyclic permutation

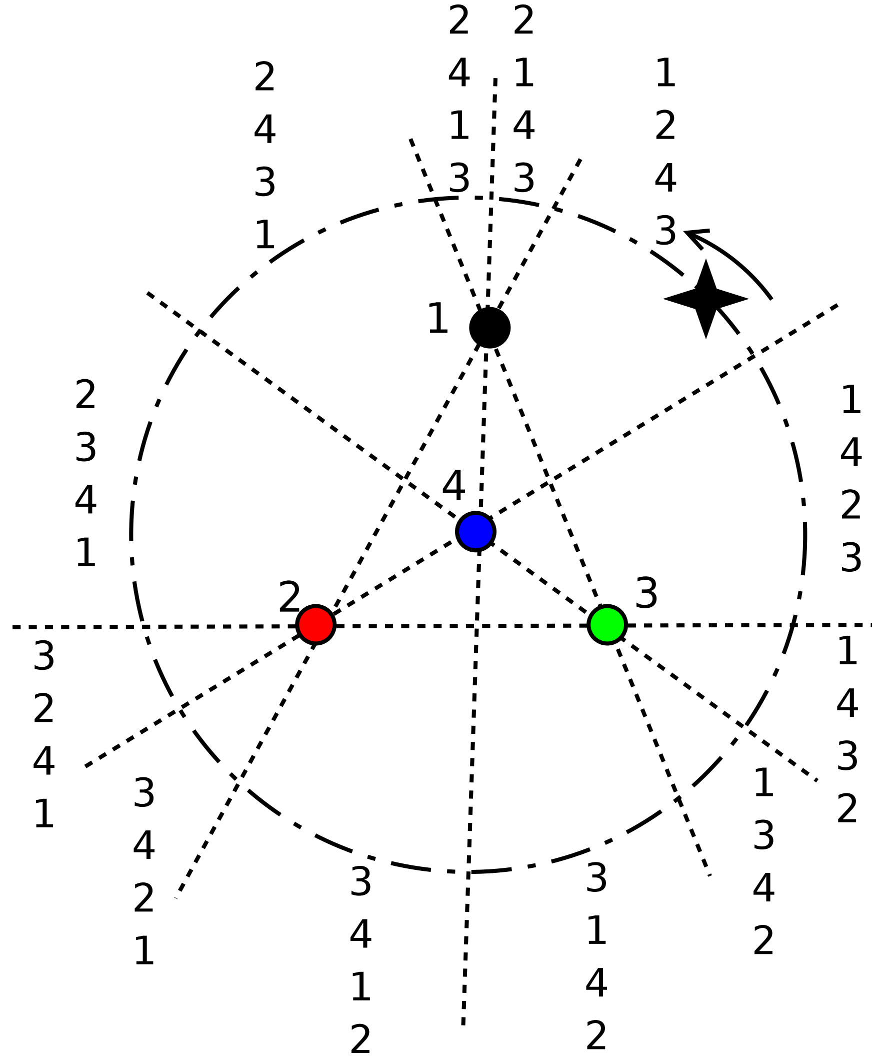
Landmark-Based Navigation Without Distances



We showed that:

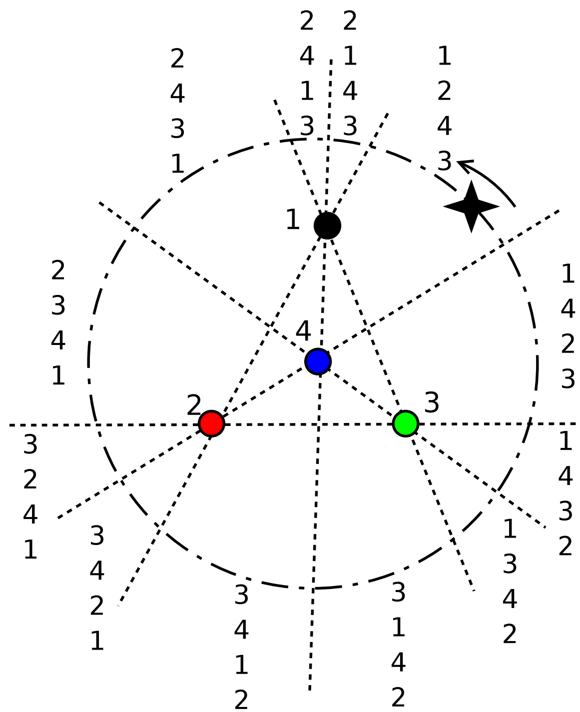
- For any subset $L' \subset L$ of landmarks, the robot can determine which others in L lie in the convex hull of L'.
- Equivalently, the robot can discover the dual arrangement.
- The robot can navigation to any goal specified as a cyclic permutation.

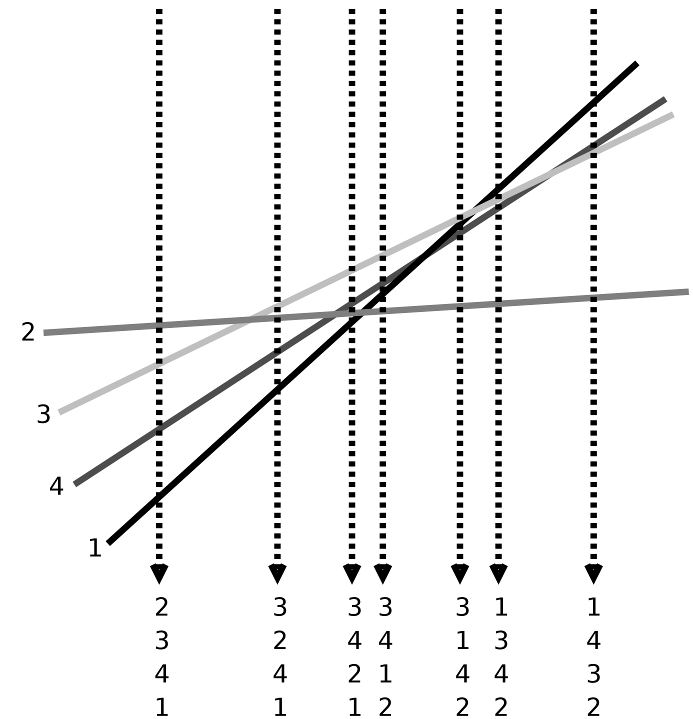
Walking Along a Circle



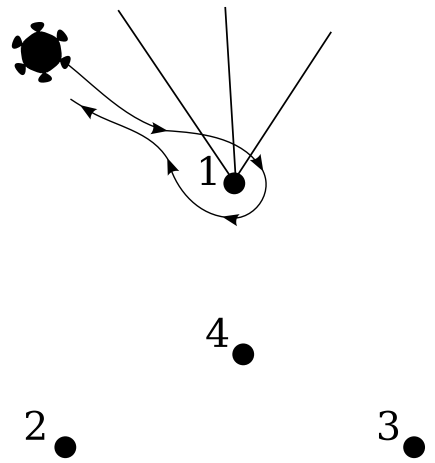
Walking along a circle.

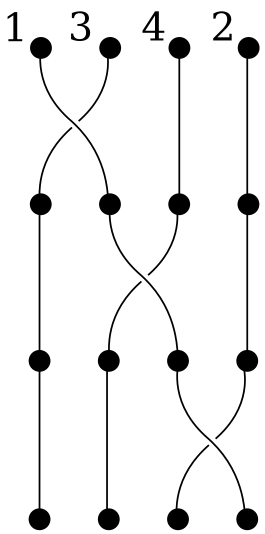
The Dual Line Arrangement





The dual line arrangement.





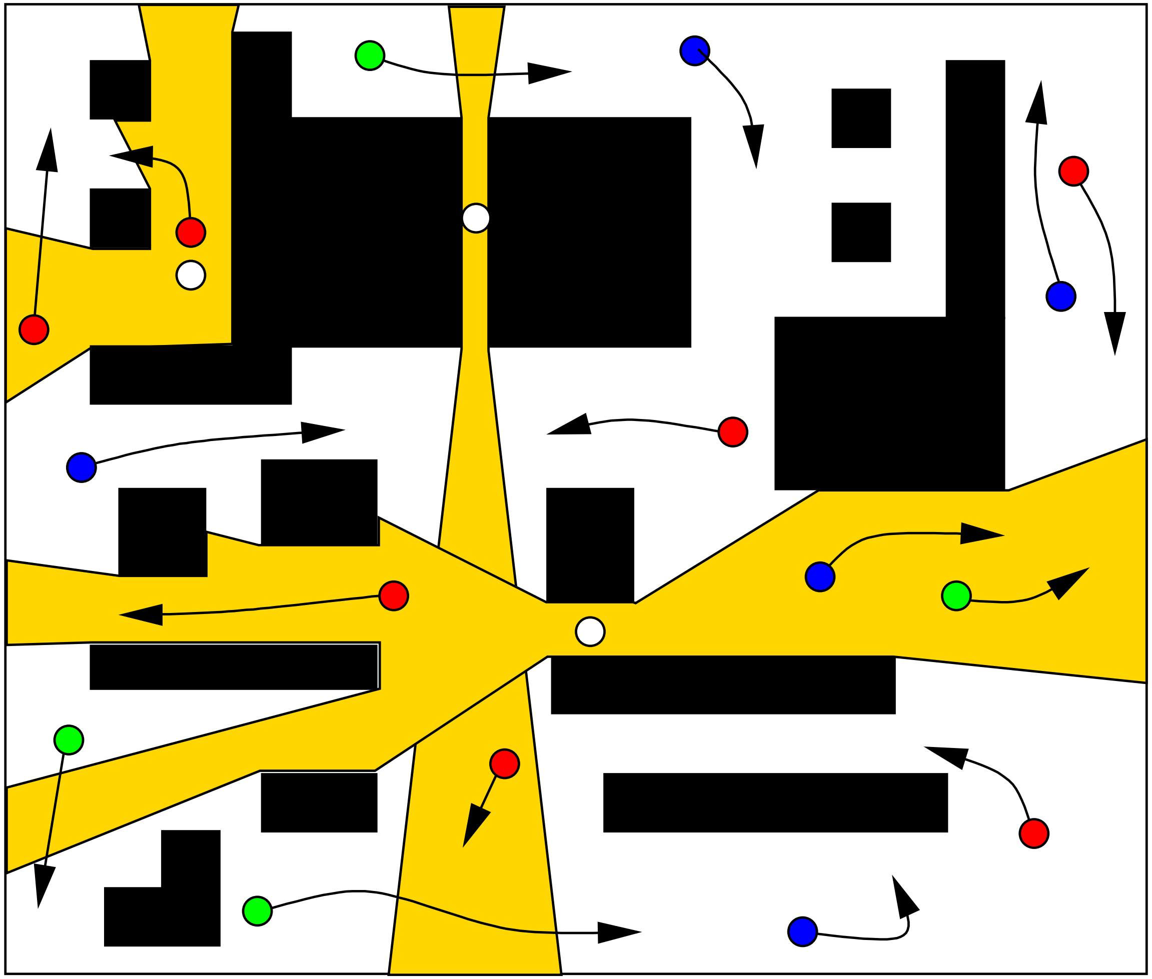
The history I-state actually maps into a *braid group*.

Other Problems

Shadow Information Spaces: Maintaining Team Movements

with Jingjin Yu (UIUC PhD Student)

Searching in a Building or Campus of Buildings



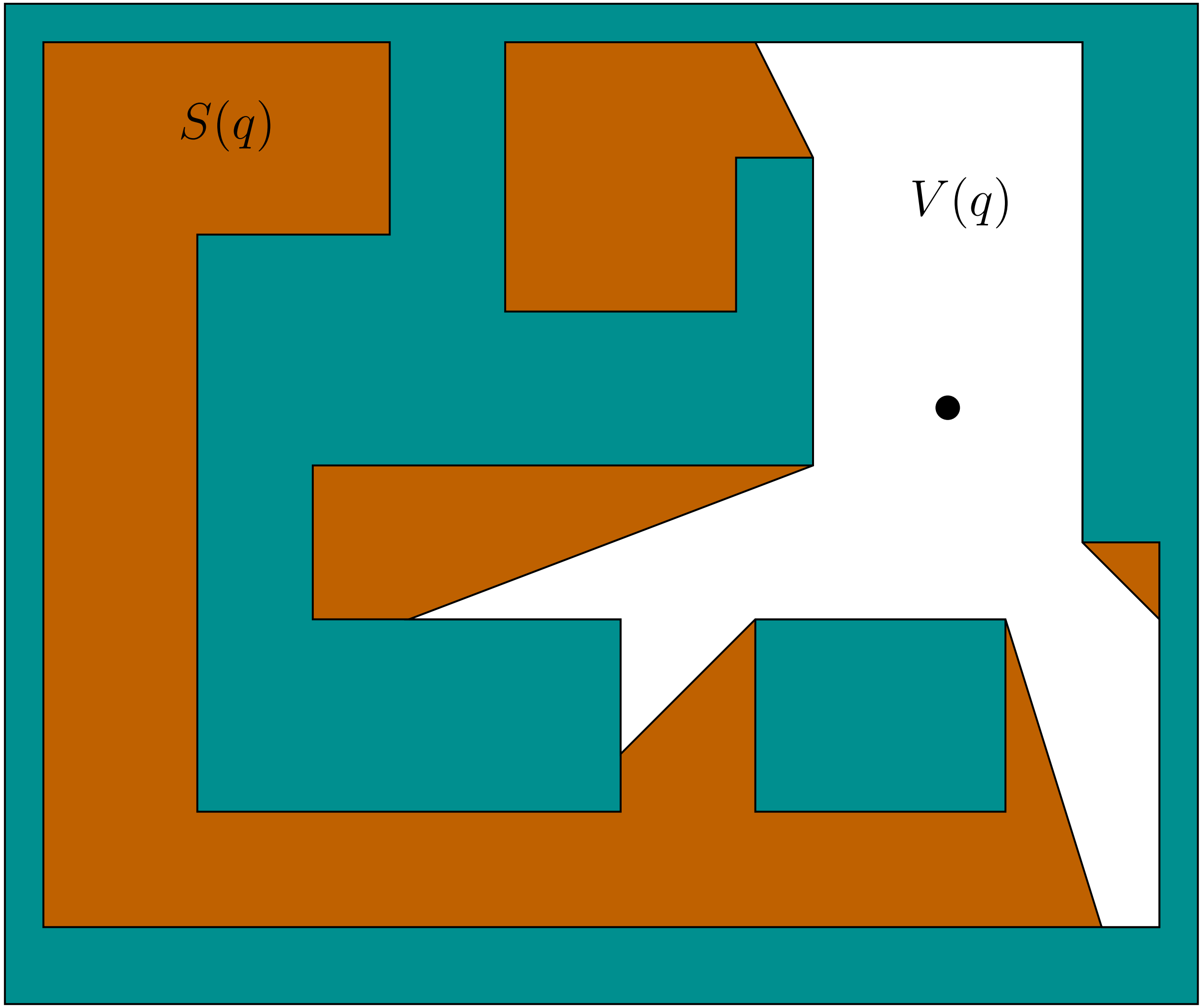
Following People With Helicopters

Common Situation

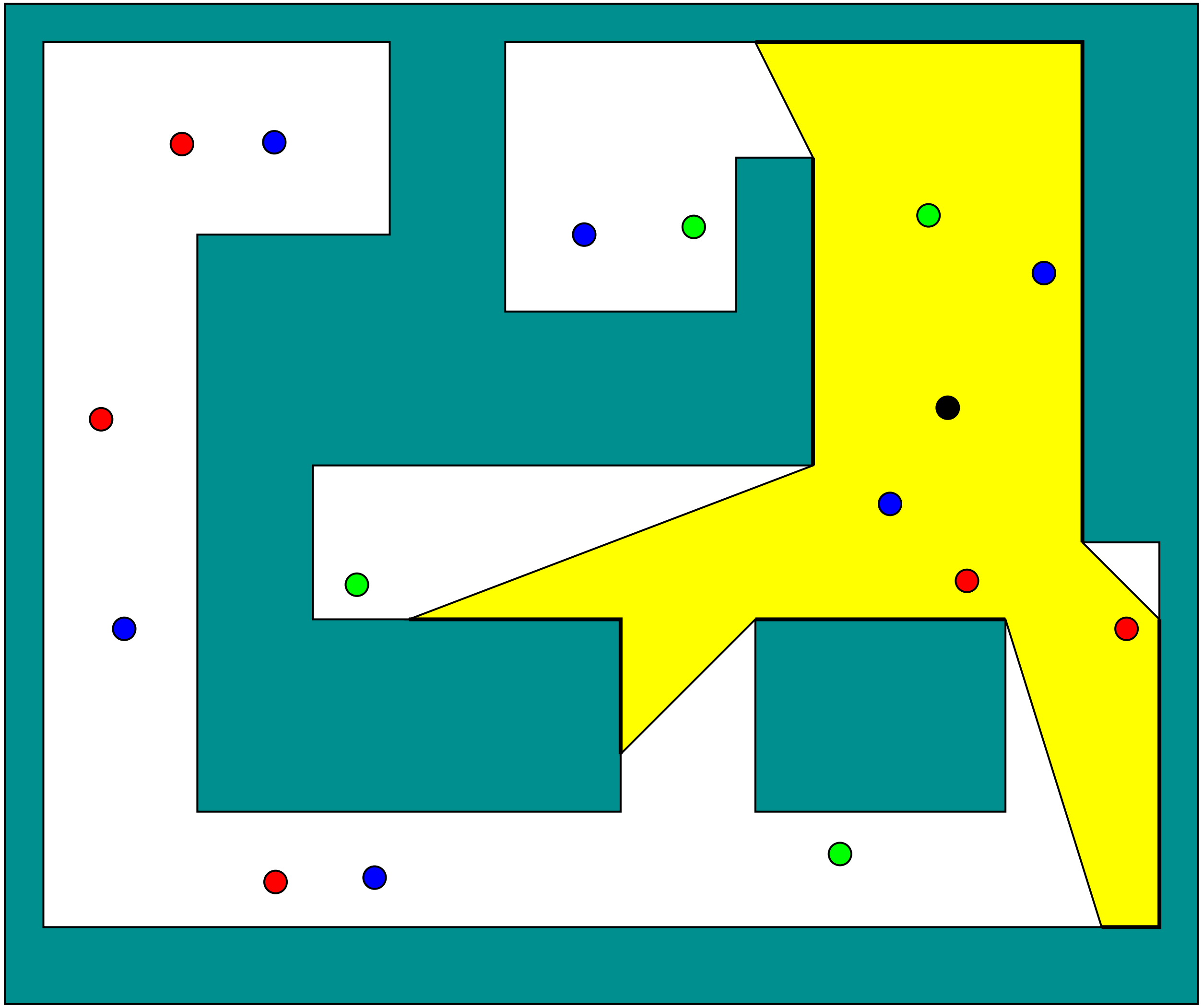
- Robots or people move around carrying sensors
- The sensor field-of-view changes topologically
- Numerous targets or agents pass in and out of view
- Sensors cannot precisely localize or distinguish targets

Inference tasks: Counting, tracking, pursuit-evasion, monitoring team movement, surveillance.

Example: The Shadow Region



Example: Moving Agents and Moving Sensor



Clocks, Chronometers, Horology

“Until the mid 1750s, navigation at sea was an unsolved problem due to the difficulty in calculating longitudinal position. To find their longitude, they needed a portable time standard that would work aboard a ship.”

Why study time sensing uncertainty?

- Clocks are cheap and accurate, but sometimes time error is a serious issue:
  For GPS, 1ns of time error = 30cm of position error
- Understanding *information requirements* leads to better strategies:
  Avoid time coordinates, minimal time dependency, distributed

Reconsidering Time in Control Theory

with Magnus Egerstedt 

“closed-loop” control: $\gamma : X \rightarrow U$

“open-loop” control: $\gamma : T \rightarrow U$

- Time is not special; it is like any other state variable.
- Rename “open-loop” to *perfect time-feedback control*.
- Introduced notion of *strongly open-loop control*: $\gamma : P \rightarrow U$.
- Defined policies in terms of I-spaces over $Z = X \times T$.
- Raises many open questions in reducing time dependencies in control.

Conclusions

General Conclusions

- Information spaces seem to pop up everywhere
- Important to understand minimal information requirements
- Inference problems lead to greater unification

- Try to simplify the I-space and “live” in it.
- Try to understand *information requirements* of tasks.
- Formulate deciability, complexity, and the power of machines in terms of robotic primitives and information spaces.