IROS 2009 Tutorial: Filtering and Planning in I-Spaces PART 5: PLANNING

Steven M. LaValle

October 20, 2009

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Overview of Topics

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

1. An I-space view of planning, starting with temporal filters
2. General planning issues
3. Maze searching
4. Gap navigation trees
5. Landmark-based navigation
6. Bug algorithms
7. Sensorless manipulation

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

From filters to planning

Using Filters For Planning

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Let $\mathcal{I}$ be any I-space.

Assume a filter

$$\iota_k = \phi(\iota_{k-1}, u_{k-1}, y_k)$$

is given.

Let $G \subset \mathcal{I}$ be a *goal region*.

Starting from ι_0, what sequence of actions u_1, u_2, . . ., will lead to some future I-state $\iota_k \in G$?

Using Filters For Planning

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

The future may be unpredictable.

Introduce an *I-state dependent* plan:

$$\pi : \mathcal{I} \rightarrow U$$

Using a filter ϕ, the execution of a plan can be expressed as

$$\iota_k = \phi(\iota_{k-1}, y_k, \pi(\iota_{k-1}))$$

The I-space $\mathcal{I}$ is just a sort of “C-space” that is being explored.

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

General issues

Main Issues

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

The following issues arise repreatedly in planning:

1. **Predictability**
2. **Reachability**
3. **Optimality**
4. **Computability**

Predictability

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Are the effects of actions predictable in the I-space $\mathcal{I}$?

If **yes**, then a *path* through the I-space is obtained.

Example: Sensorless manipulation
Example: Visibility-based pursuit evasion

By analogy to path planning in C-space:

1. Combinatorial planning in I-space
2. Sampling-based planing in I-space

If **no**, then information feedback is critical
It is like feedback planning (or control) in C-space, but instead over I-space

Reachability

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Reachability:
Is the goal region $G \subset \mathcal{I}$ even reachable from the initial I-state?

Do there even exist actions that will take us to G?

Does there exist a plan that can reach G?

With unpredictability, is G *guaranteed* to be reached, over all possible disturbances?

A more basic question is whether the goal can even be adequately expressed in $\mathcal{I}$.

Optimality

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Perhaps many plans can reach G

What criteria should be formulated to compare plans?

Which plans are the best, or optimal with respect to criteria?

Do optimal plans even exist?

Computability

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Given a description of the problem, can an algorithm be determined that automatically computes a useful plan?

Sometimes a clever human designs the plan (e.g. bug algorithms)

What is the algorithmic complexity of computing a solution plan?

What is the implementation difficulty of computing a solution plan?

Important Generic Examples

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

State feedback: I-space is $\mathcal{I} = X$ and plan is $\pi : X \rightarrow U$

Open loop: $\mathcal{I} = \mathbb{N}$ and $\pi : \mathbb{N} \rightarrow U$
π can be written as $(u_1, u_2, u_3, \ldots)$

Sensor feedback: $\mathcal{I} = Y$ and $\pi : Y \rightarrow U$

History feedback $\mathcal{I} = \mathcal{I}_{hist}$ and $\pi : \mathcal{I}_{hist} \rightarrow U$

Recall the previous filters over these 4 I-spaces.

Now we move from *passive* to *active*.

Overall Process

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Based on the task, an overall approach that leads to planning:

1. Design the system, which includes the environment, bodies, and sensors.

Overall Process

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Based on the task, an overall approach that leads to planning:

1. Design the system, which includes the environment, bodies, and sensors.
2. Define the models, which provide the state space X, the sensor mapping h, and the state transition function f.

Overall Process

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Based on the task, an overall approach that leads to planning:

1. Design the system, which includes the environment, bodies, and sensors.
2. Define the models, which provide the state space X, the sensor mapping h, and the state transition function f.
3. Select an I-space $\mathcal{I}$ for which a filter ϕ can be practically computed.

Overall Process

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Based on the task, an overall approach that leads to planning:

1. Design the system, which includes the environment, bodies, and sensors.
2. Define the models, which provide the state space X, the sensor mapping h, and the state transition function f.
3. Select an I-space $\mathcal{I}$ for which a filter ϕ can be practically computed.
4. Take the desired goal, expressed over X, and convert it into an expression over $\mathcal{I}$.

Overall Process

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Based on the task, an overall approach that leads to planning:

1. Design the system, which includes the environment, bodies, and sensors.
2. Define the models, which provide the state space X, the sensor mapping h, and the state transition function f.
3. Select an I-space $\mathcal{I}$ for which a filter ϕ can be practically computed.
4. Take the desired goal, expressed over X, and convert it into an expression over $\mathcal{I}$.
5. Compute a plan π over $\mathcal{I}$ that achieves the goal in terms of $\mathcal{I}$.

Really, all steps should be considered together.

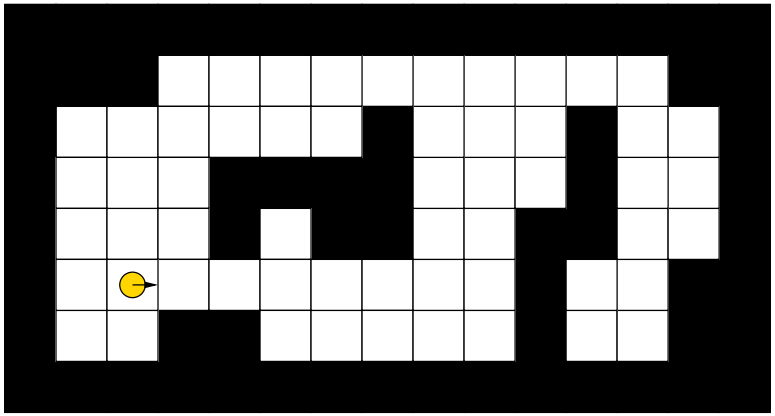
Might have to backtrack.

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Maze searching

Maze Searching

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation



Each $E \in \mathcal{E}$ is a bounded set of white tiles.

$$X \subset \mathbb{Z} \times \mathbb{Z} \times D \times \mathcal{E}$$

Actions: 1) Rotate 90 degrees CCW; 2) Move foward one tile.

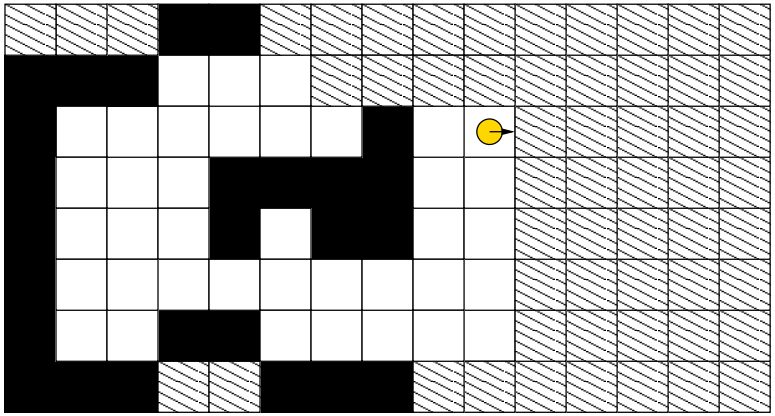
Task: Make a plan that systematically searches all white tiles.
For example, find a hidden treasure.

Maze Searching: Simple Mapping

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Could try $\mathcal{I} = \mathrm{pow}(\mathbb{Z} \times \mathbb{Z} \times D \times E)$.
Too large!

Instead, maintain I-states B (known black tiles) and W (known white tiles).



All other tiles assumed “unknown”.

I-space $\mathcal{I}$ is all ways to partition $\mathbb{Z} \times \mathbb{Z}$ into connected “white”, “black”, and “unknown” tiles.
Linear space required for an I-state (filter memory).

Maze Searching: Blum and Kozen 1978

From filters to planning

General issues

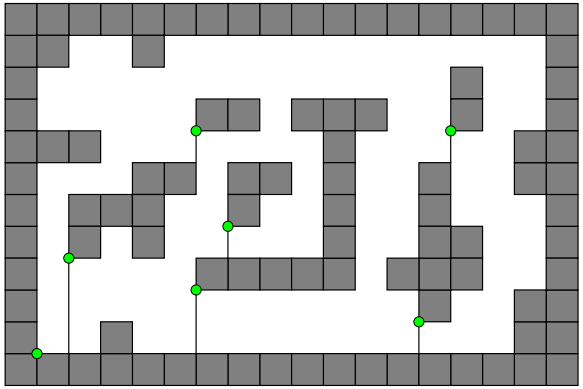
Maze searching

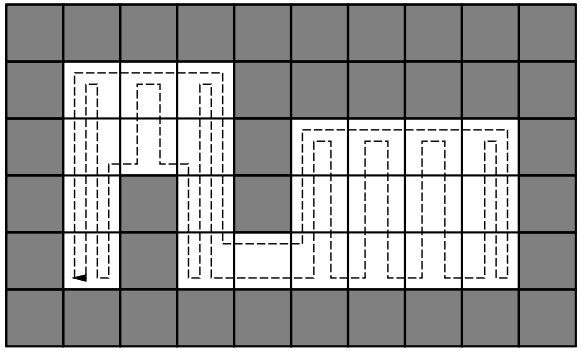
Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation





I-state: Latitude (integer) and orientation (two bits)

Only logarithmic space required: Not enough for a “map”.

They found an I-space that is much smaller than the set of all maps.

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Gap navigation trees

Make Gap Navigation Trees Active

From filters to planning

General issues

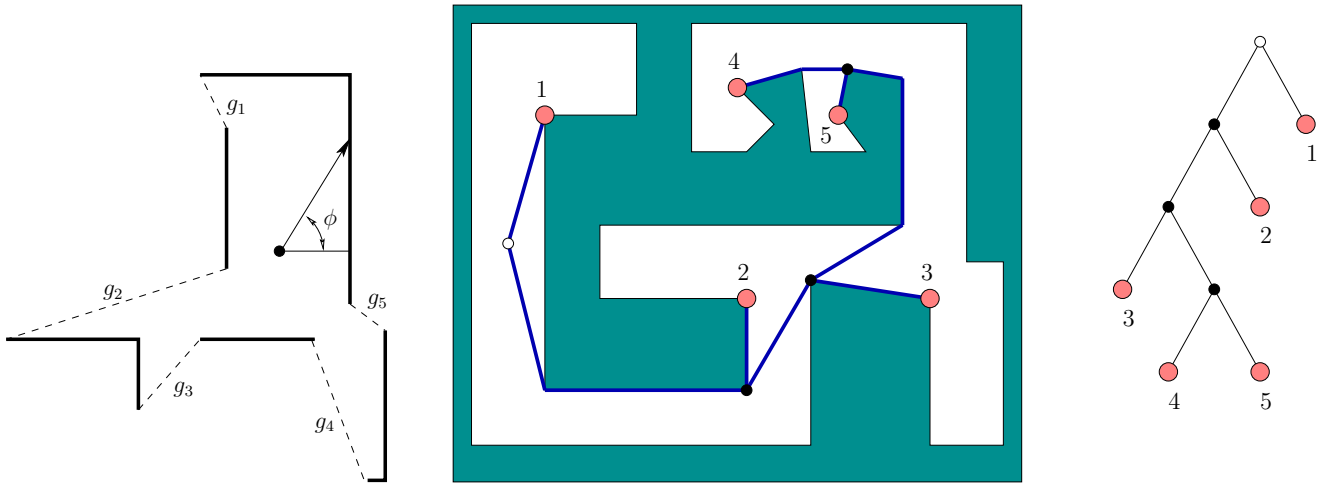
Maze searching

Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation



For gap navigation trees, two active tasks:

1. Full exploration of the environment
2. Distance-optimal navigation to retrieve objects

Tovar, Murrieta, LaValle, *IEEE Trans. Robotics*, 2007.

Introduce an Action

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

A gap-chasing action is introduced:

Move the robot toward a gap g until a critical event occurs.

One of two events must occur:

1. The gap g splits into two gaps g' and g''.
2. The gap g disappears.

Full Exploration

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

If a gap ever appears, mark it as *primitive*.

This is an extension to the filter I-state.

1. Mark all gaps in the initial tree as *non-primitive*.
2. Let $k = 1$.
3. Chase any gap g that is a non-primtive leaf.
4. If g *disappears*, then go to Step 6.
5. If g *splits*, then chase one of its children.
6. Unless all leaves are primitive, increment k and go to Step 3.

At the end, all leaves are primitive and the environment has been fully explored.

Simple Illustration

From filters to planning

General issues

Maze searching

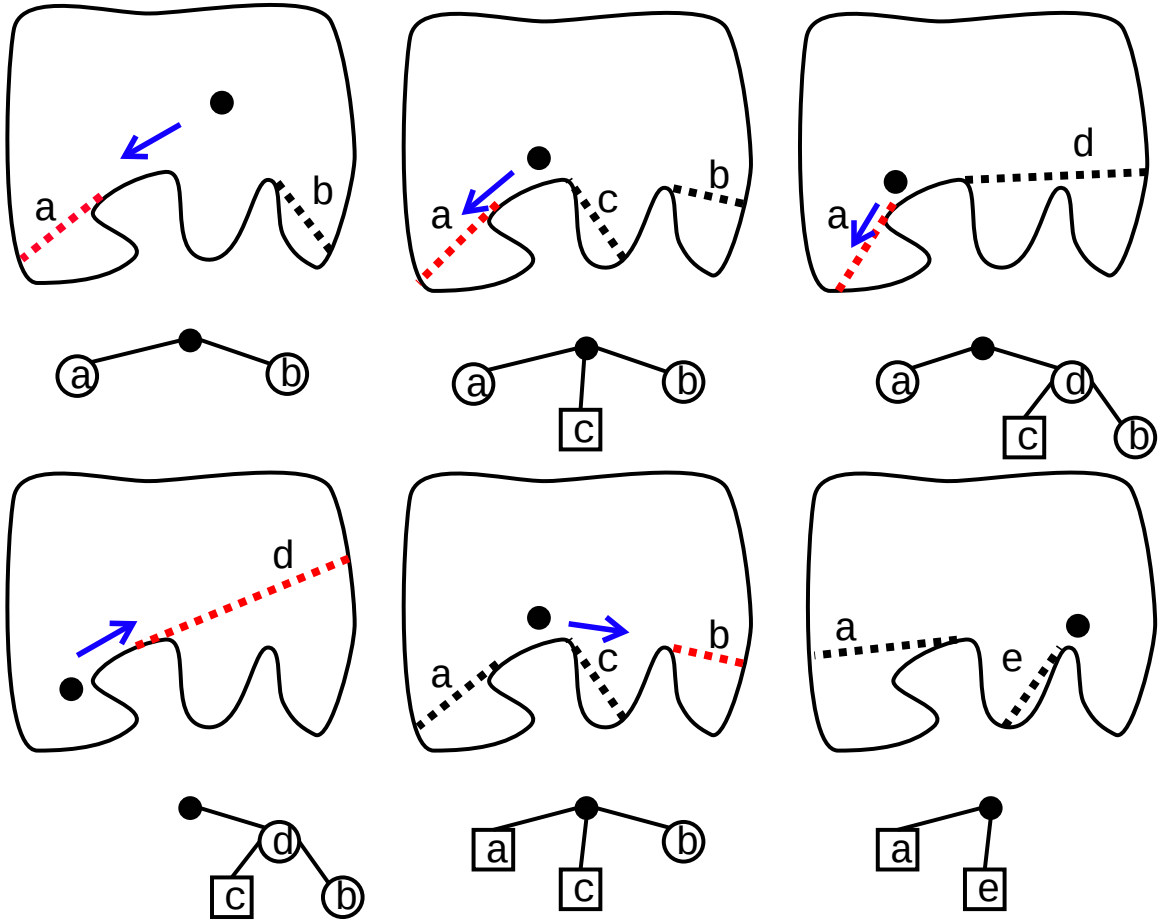
Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation

Chase every non-primitive leaf:



Eventually, all leaves become primitive.

Optimal Object Retrieval

From filters to planning

General issues

Maze searching

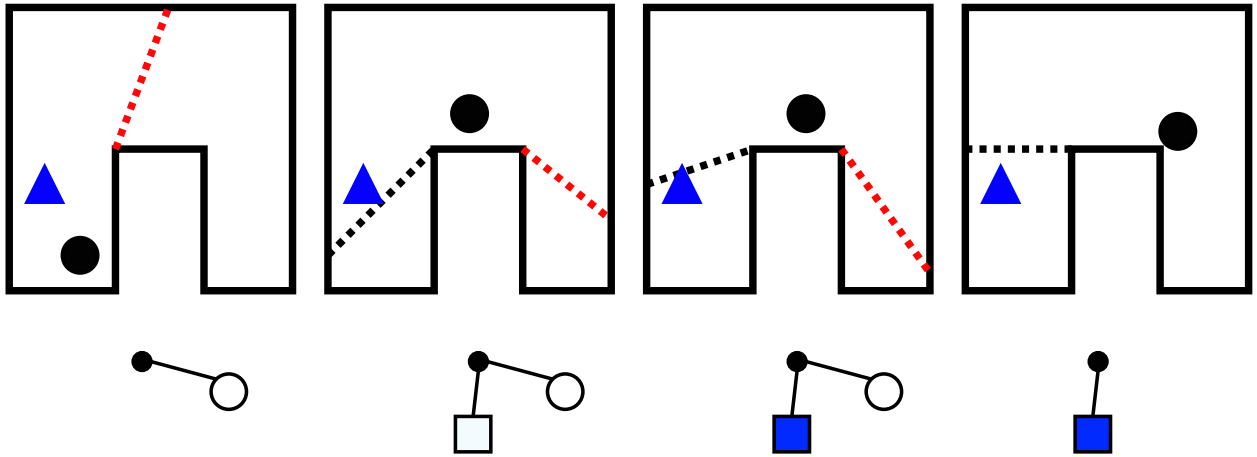
Gap navigation trees

Learning convex hulls of landmarks

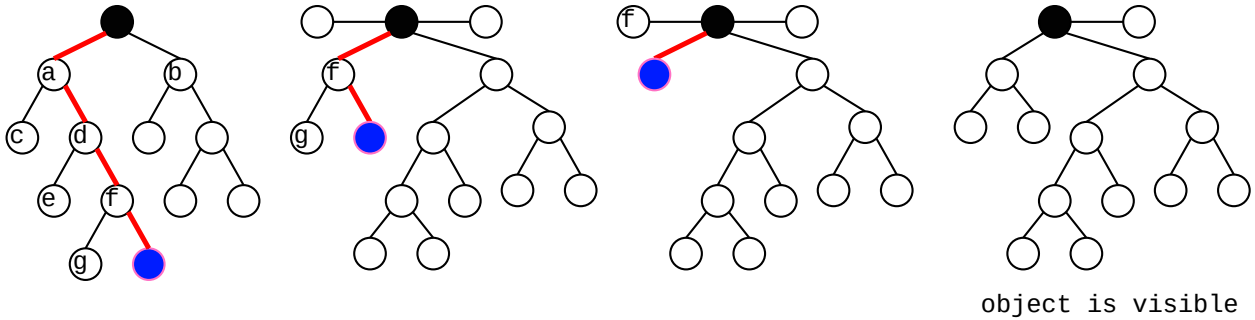
Bug algorithms

Sensorless manipulation

There are no coordinates.



Objects hide behind gaps.



Chase the appropriate sequence of gaps.

Possible Current States

From filters to planning

General issues

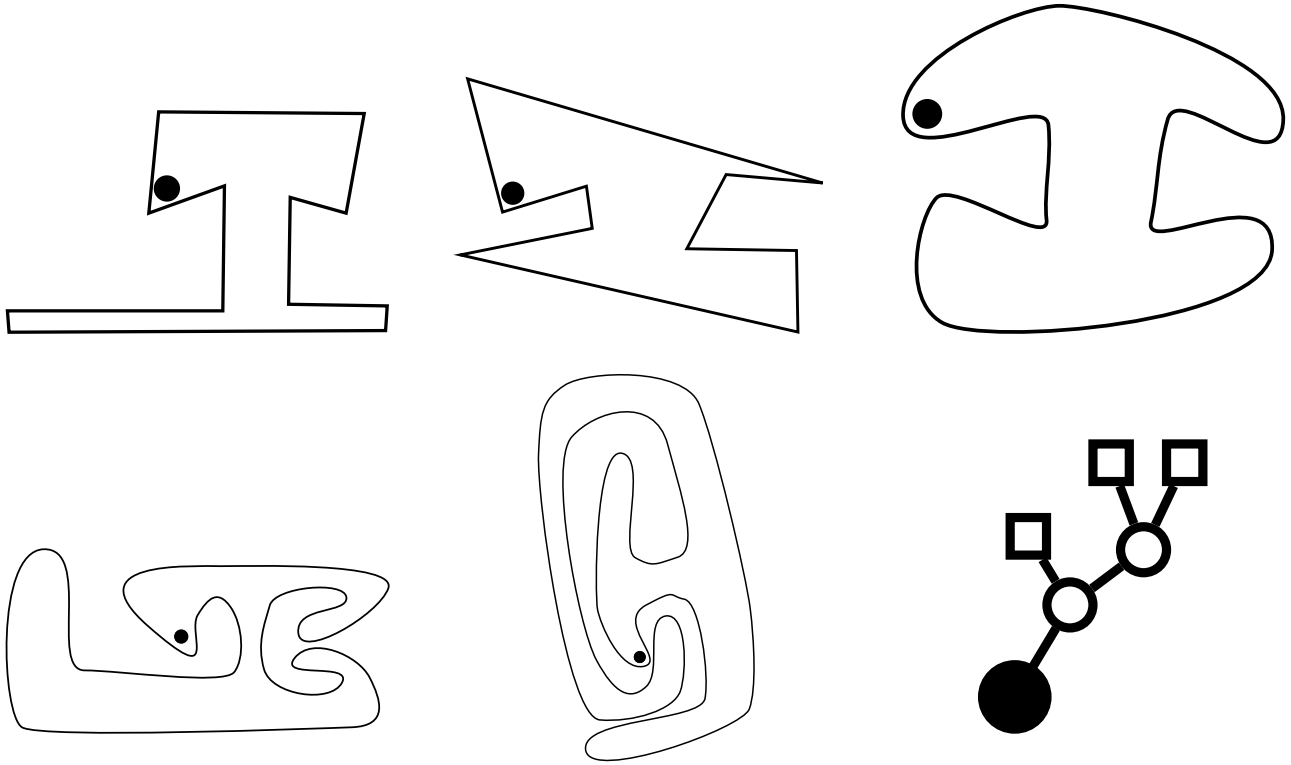
Maze searching

Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation



Many configuraton-environment pairs have the same tree.

The robot does not have to distinguish!

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Learning convex hulls of landmarks

Recall: Cyclic Permutation Sensor

From filters to planning

General issues

Maze searching

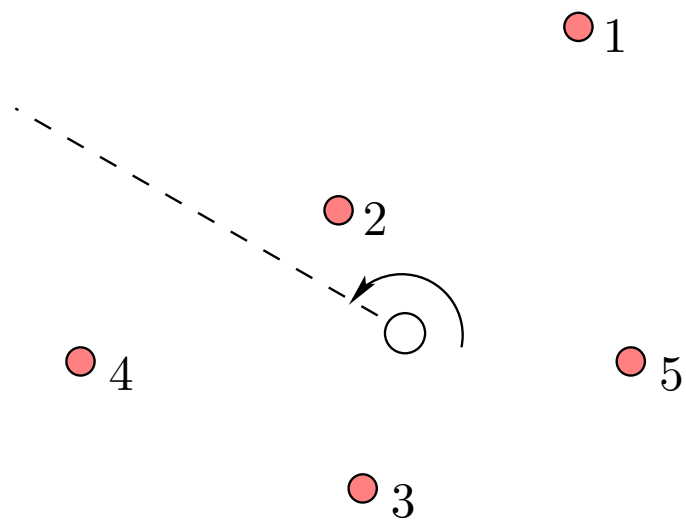
Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation

Relation: “is to the left of in counterclockwise order”



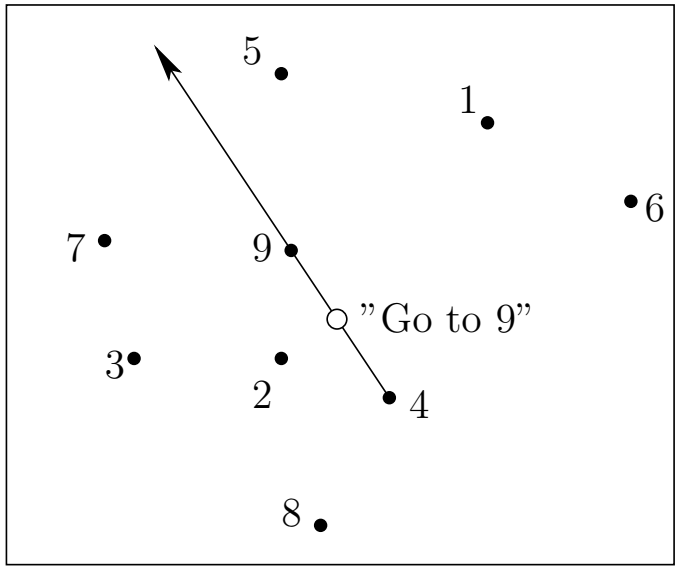
Observation: $y = (1, 2, 4, 3, 5)$

Making an Active Version

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Tovar, Freda, LaValle, 2007.

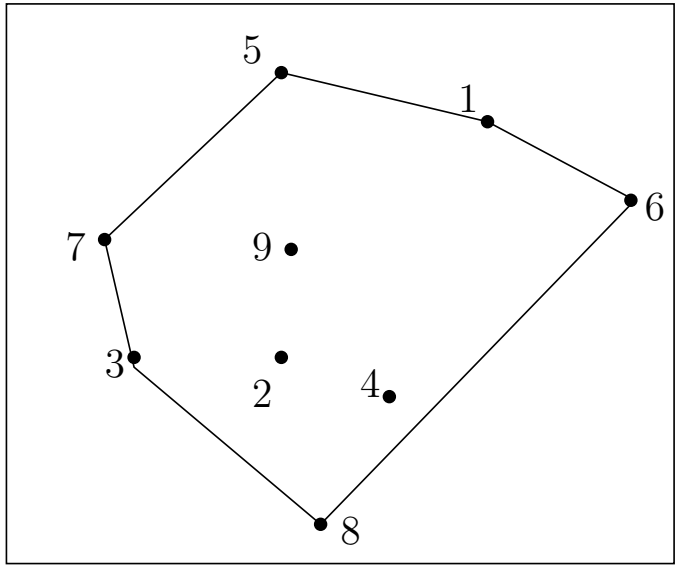
- Landmark locations are unknown
- Introduce action: "Go to landmark i"
- Can notice which landmarks are "to the left" of the path.



Sense that $(6, 1, 5)$ is to the right of $(7, 2, 3, 8)$.

Learning the Arrangement

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation



By visiting all pairs, the filter can learn:

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

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Bug algorithms

Recall Bug Algorithms

From filters to planning

General issues

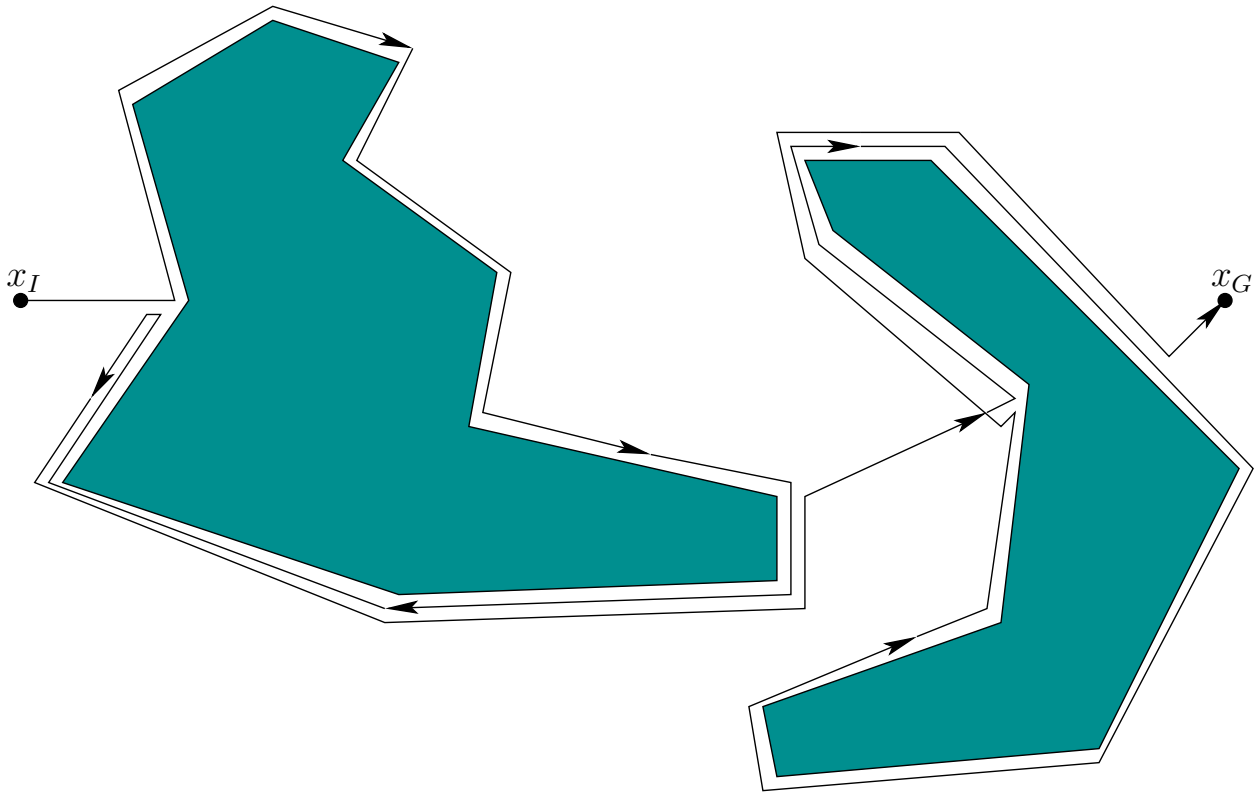
Maze searching

Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation



Navigate without being given an initial map E

Lumelsky, Stepanov, 1987; Kamon, Rivlin, Rimon, 1997; many others...

Intensity Bug

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Taylor, LaValle, ICRA 2009

- The plane contains unknown obstacles with piecewise analytic boundary.
- Each obstacle boundary has finite length.
- A *tower* sends a constant signal.
- Robot has very limited sensors.
- Command the robot so that it reaches the tower.

Sensors

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

- **Boundary (or Contact) sensor:**
  Indicate whether or not robot is on the boundary.
- **Tower alignment/gradient sensor:**
  Indicate whether robot is facing the tower (or intensity gradient).
- **Transformed signal intensity sensor:**
  Observe the value of $m(p - p_t)$.

Regarding m:

- m has only only local maximum, at the tower.
- The function m itself is not given.
- Level sets of m may be symmetric (circles) or asymmetric.

Actions

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

There are three possible actions:

u_{fwd}: Go straight until either ∂E is hit, tower is hit, or local intensity maximum detected.

u_{fol}: Follow ∂E until local maximum detected.

u_{ori}: Rotate until facing tower (or local gradient).

An Plan Designed by Humans

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Illustration of the Plan

Guaranteed to converge; upper bound on distance shown.

Information Spaces

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

State space: $X \subset \mathbb{R}^2 \times S^1 \times \mathcal{E}$

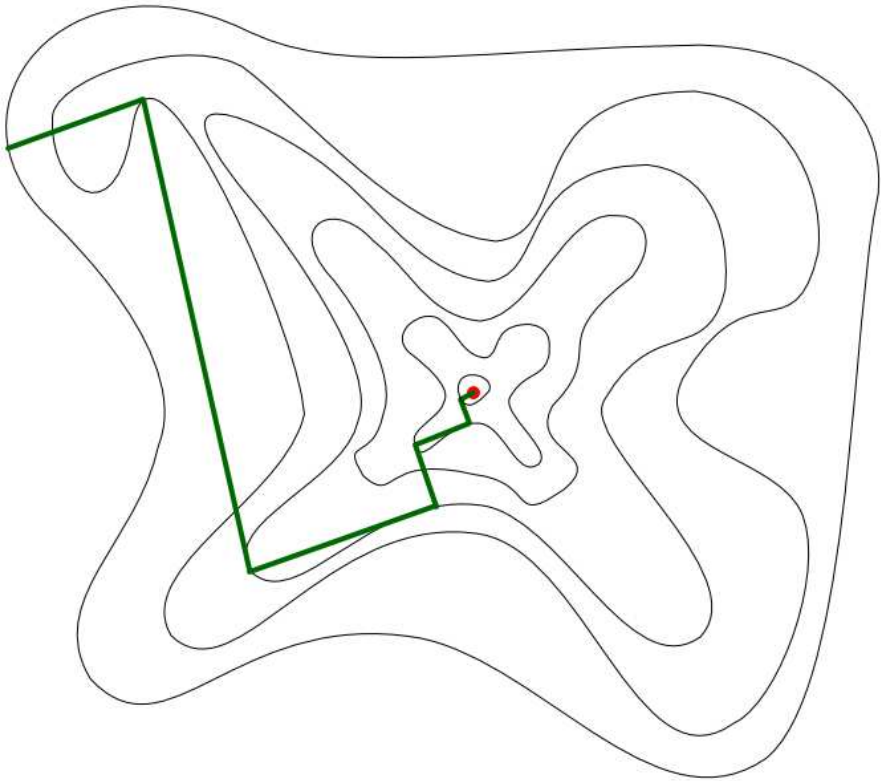
I-space: $\mathcal{I} = Y^3 \subset \mathbb{R}^3$

I-state components:

1. Current observation
2. Observation when obstacle was last contacted
3. Observation just prior to application of u_{fwd}

Multiple Iterations in the Interior

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

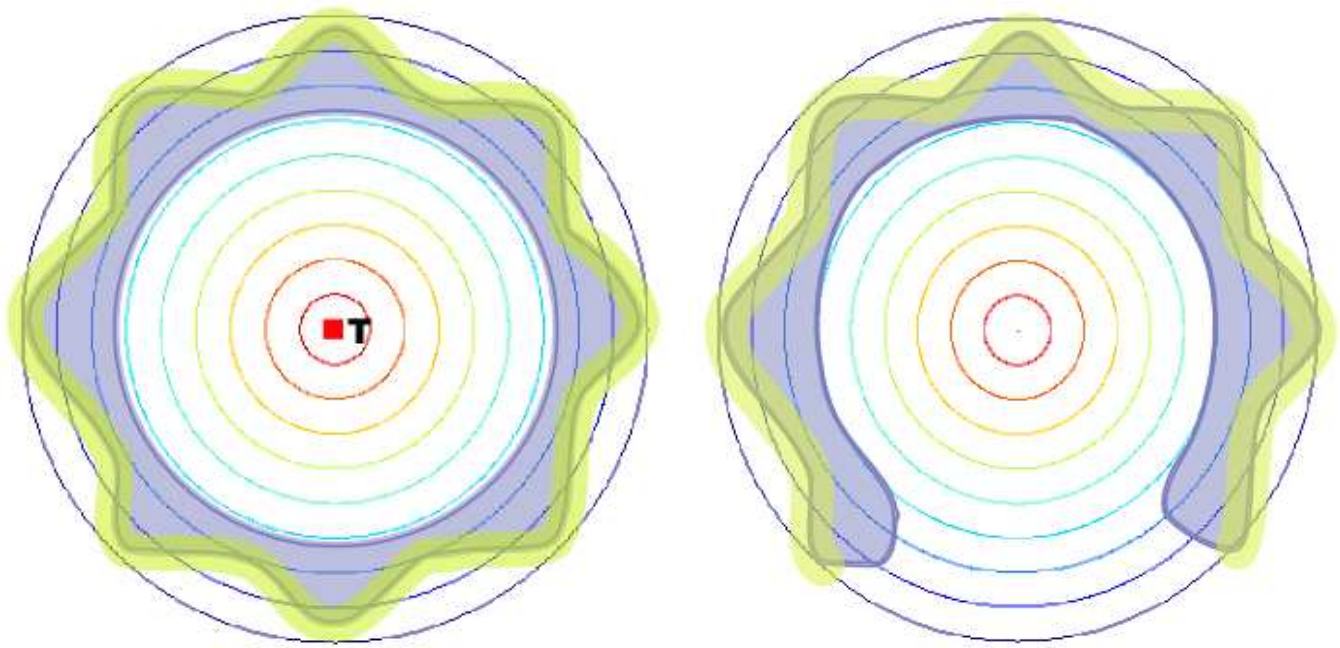


- Equivalent to Steepest Descent with Line Search.
- Result: Convergence is obtained, but distance bound depends on properties of m.

Decidability

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Proposition: Using its sensors and motion primitives, it is impossible for the robot to determine whether the tower is reachable, in other words whether $p_t \in E$.



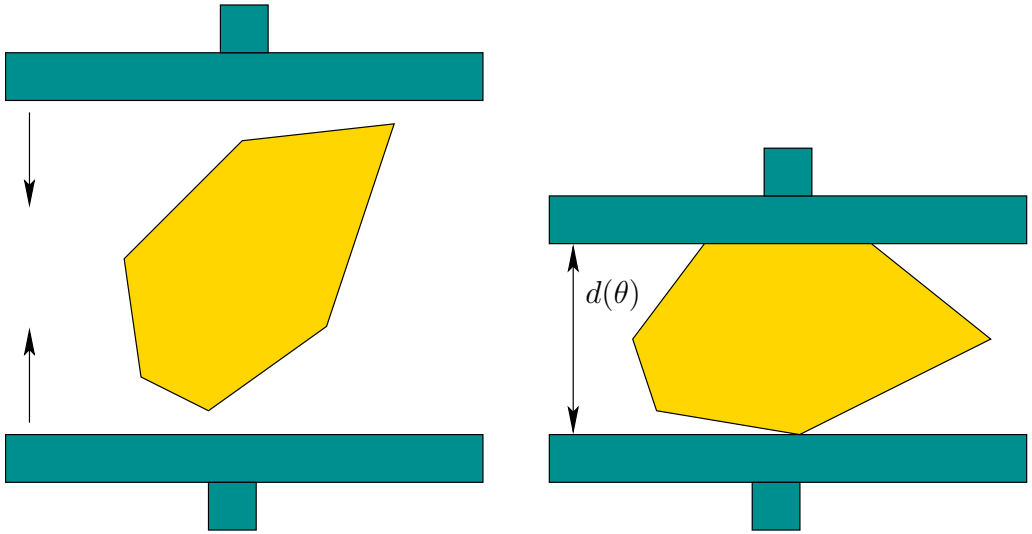
From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Sensorless manipulation

Sensorless Manipulation

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

Try to force parts into a known orientation



Mason, Goldberg, 1990

$$X = S^1 \quad \mathcal{I} = \mathrm{pow}(X)$$

Plan: $\pi = (u_1, u_2, \ldots, u_n)$
A sequence of squeeze operations

Sensorless Manipulation

From filters to planning

General issues

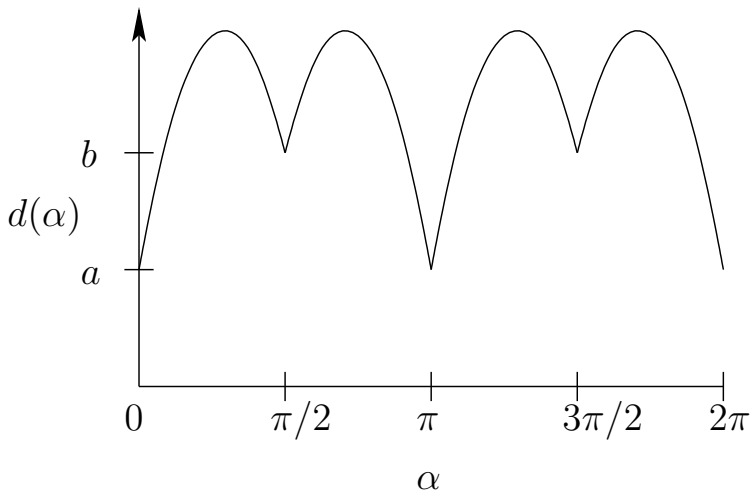
Maze searching

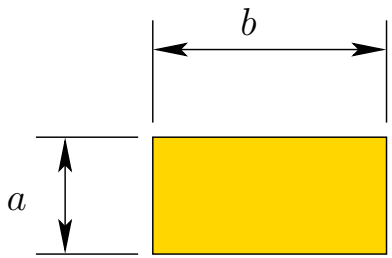
Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation





Consider the “diameter” as a function of orientation.

Sensorless Manipulation

From filters to planning

General issues

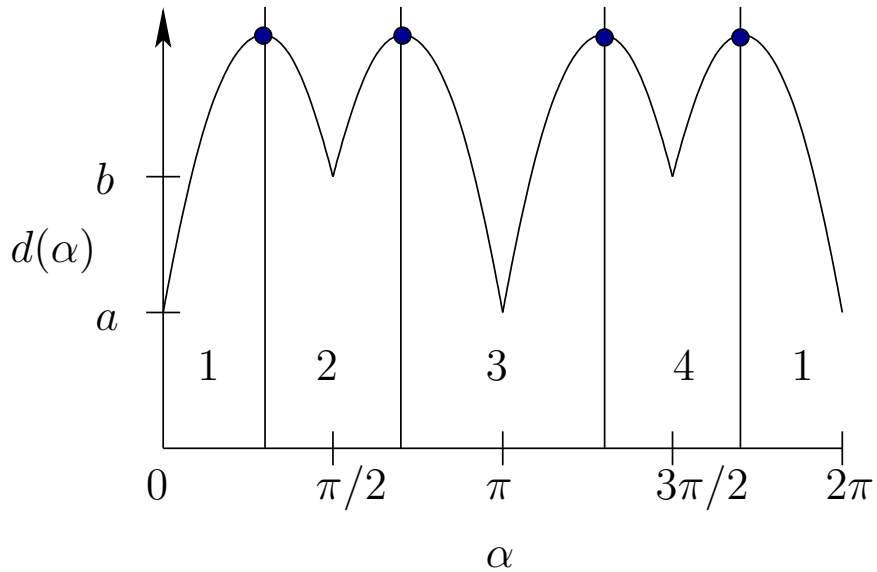
Maze searching

Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation



There are four regions of attraction.

This causes a funneling effect when actions are applied.

Sensorless Manipulation

From filters to planning

General issues

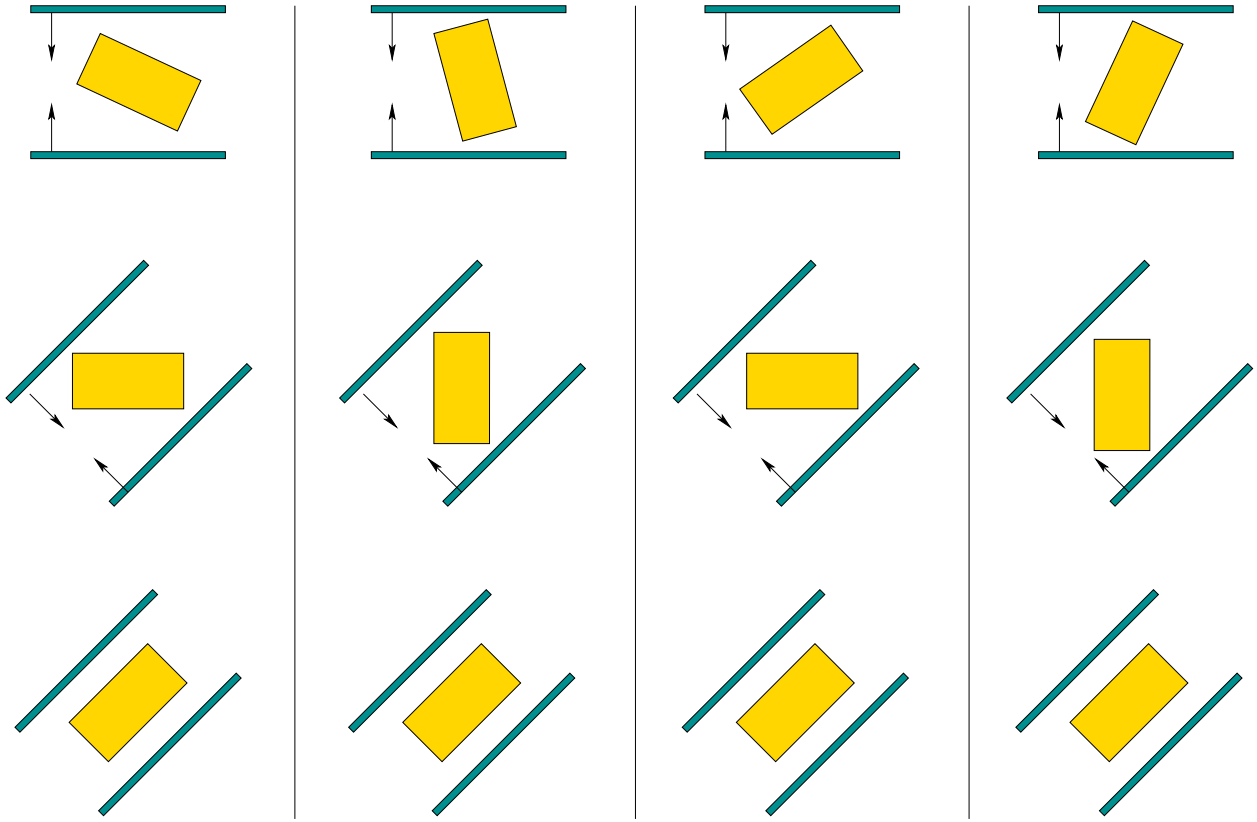
Maze searching

Gap navigation trees

Learning convex hulls of landmarks

Bug algorithms

Sensorless manipulation



A computed plan that applies two squeeze actions

Part 5 Summary

From filters to planning
General issues
Maze searching
Gap navigation trees
Learning convex hulls of landmarks
Bug algorithms
Sensorless manipulation

- Use filters to make I-space transitions
- Plan directly in the I-space
- General planning issues
- Need to design virtual sensors, filters, and planning around a task
- Several examples were shown

Although several examples of nice reduced-complexity I-spaces have been found, we have barely scratched the surface...