Policies based on Trajectory Libraries

Martin Stolle

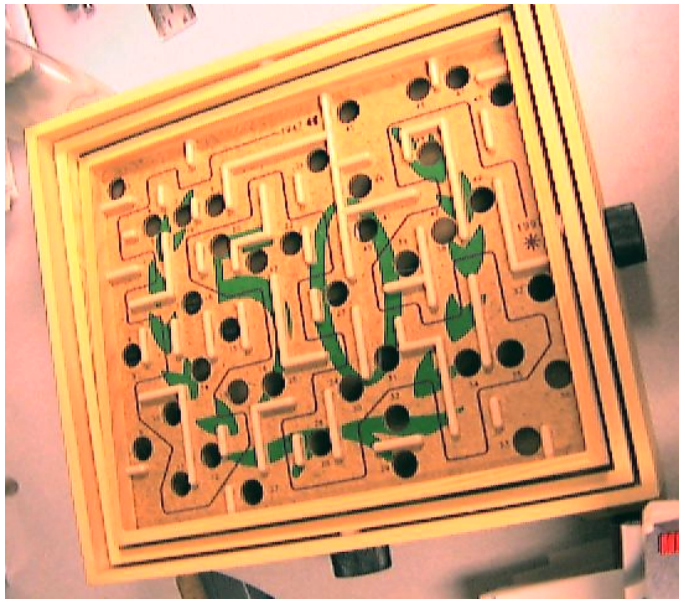
Robotics Institute

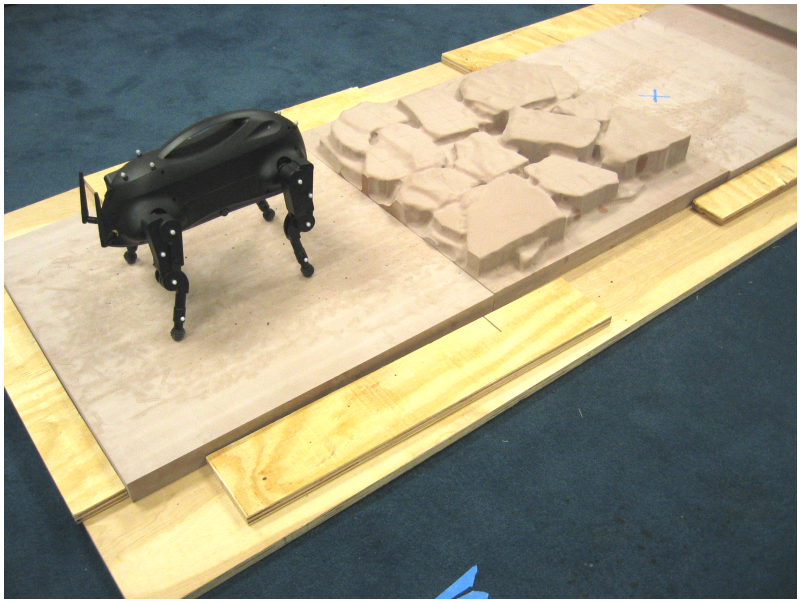
Carnegie Mellon University

http://www.cs.cmu.edu/~mstoll

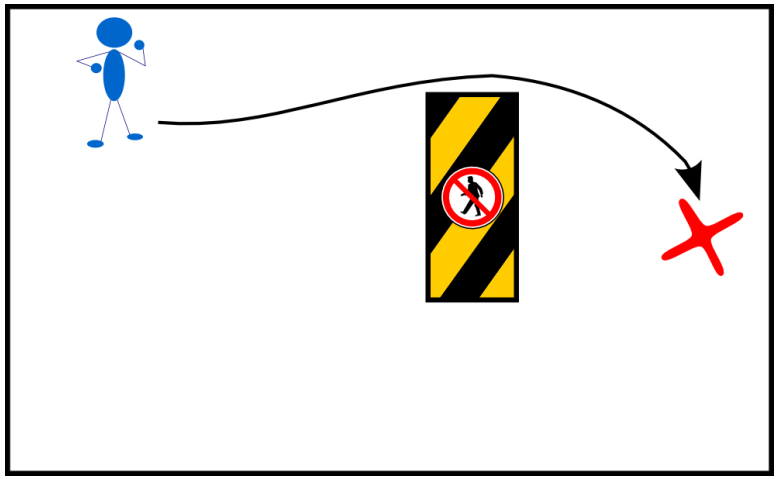
Motivation

- Controlling high dimensional, dynamic systems
- Maze first, Little Dog ongoing

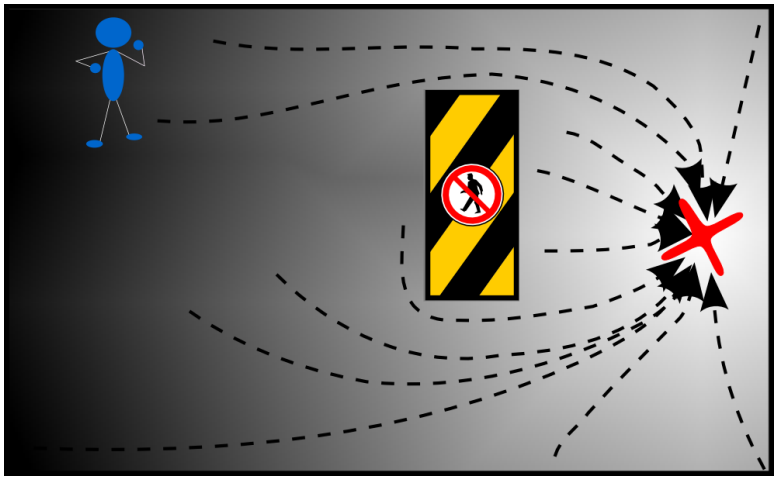




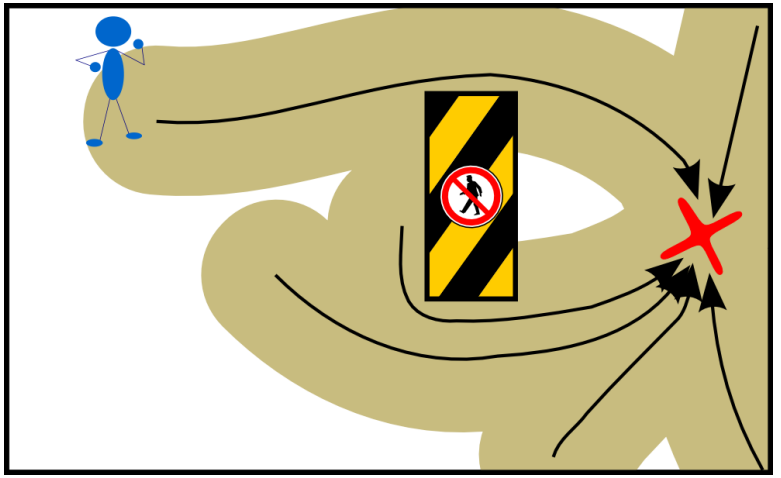
Solutions



trajectory plan



global policy



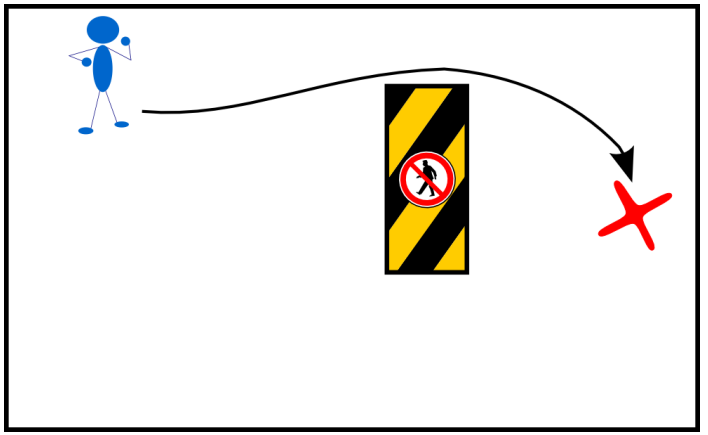
library of trajectories

Key Ideas

- learn: create a library of trajectories
- index by state: select action from any trajectory based on state
- learn: improve library based on experience

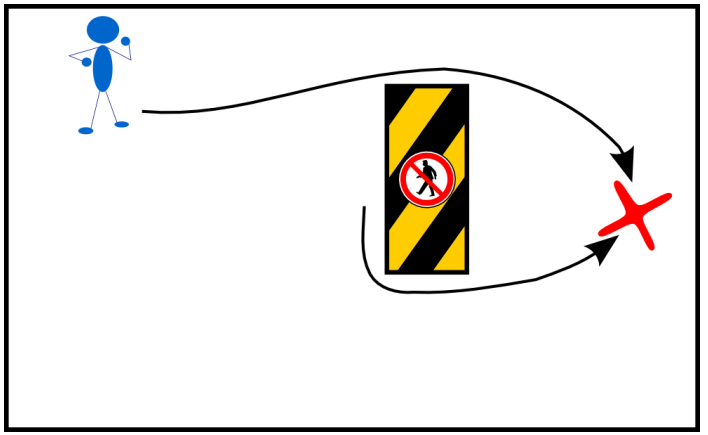
Key Ideas

- **learn: create a library of trajectories**
- index by state: select action from any trajectory based on state
- learn: improve library based on experience



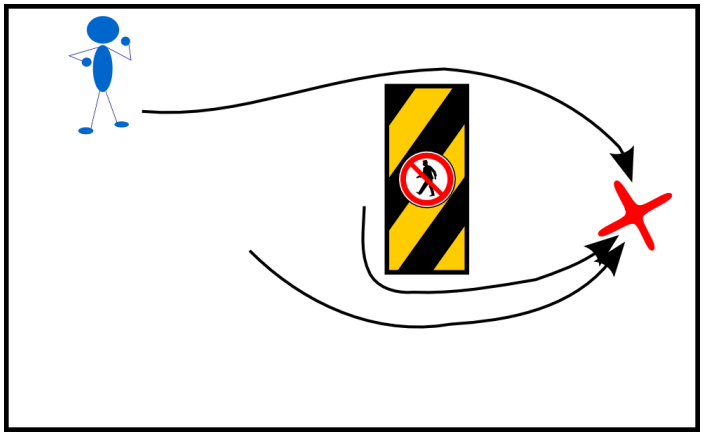
Key Ideas

- **learn: create a library of trajectories**
- index by state: select action from any trajectory based on state
- learn: improve library based on experience



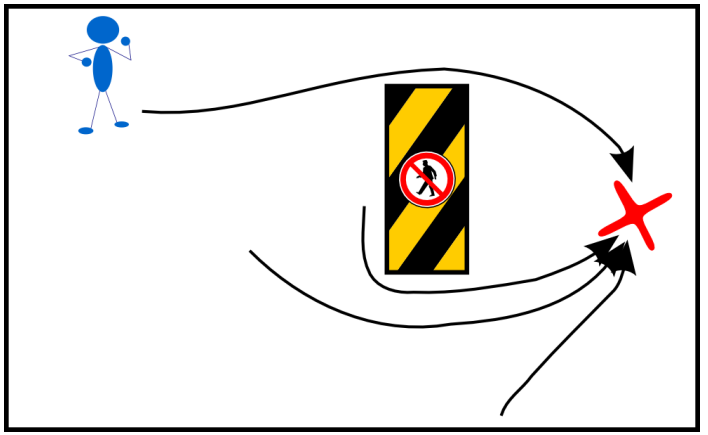
Key Ideas

- **learn: create a library of trajectories**
- index by state: select action from any trajectory based on state
- learn: improve library based on experience



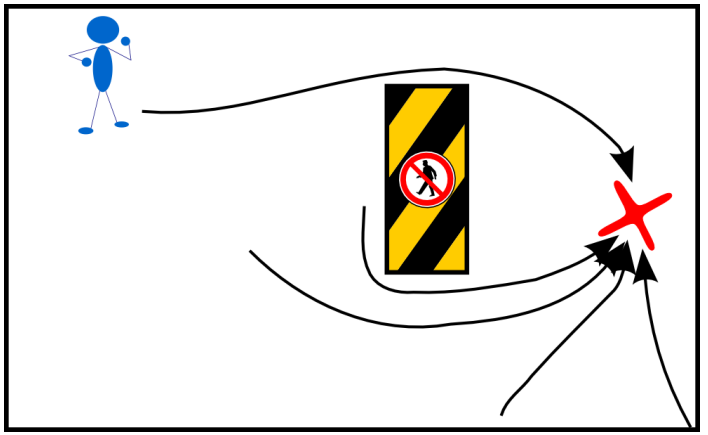
Key Ideas

- **learn: create a library of trajectories**
- index by state: select action from any trajectory based on state
- learn: improve library based on experience



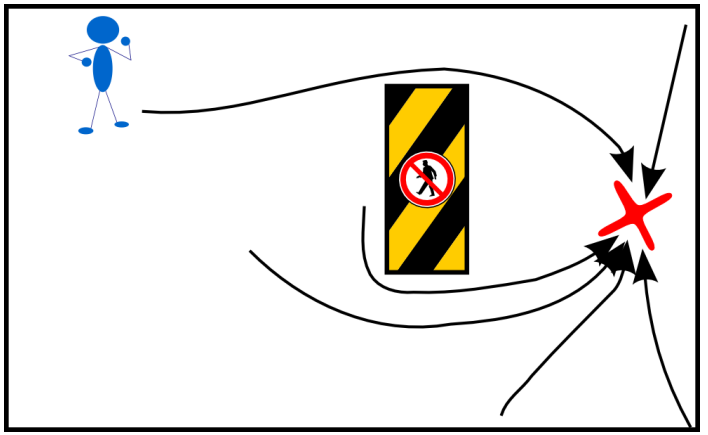
Key Ideas

- **learn: create a library of trajectories**
- index by state: select action from any trajectory based on state
- learn: improve library based on experience



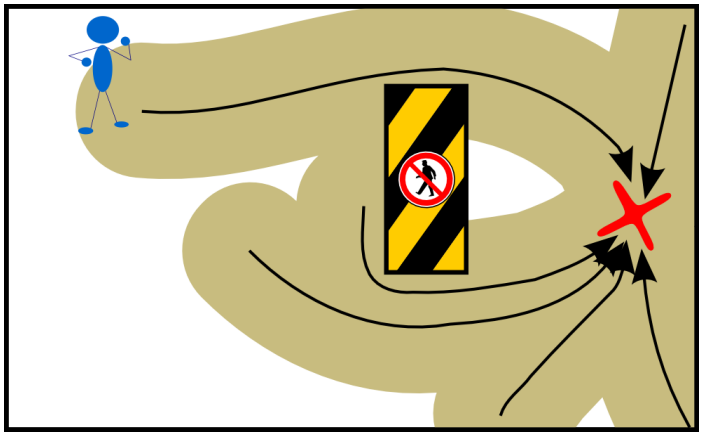
Key Ideas

- **learn: create a library of trajectories**
- index by state: select action from any trajectory based on state
- learn: improve library based on experience



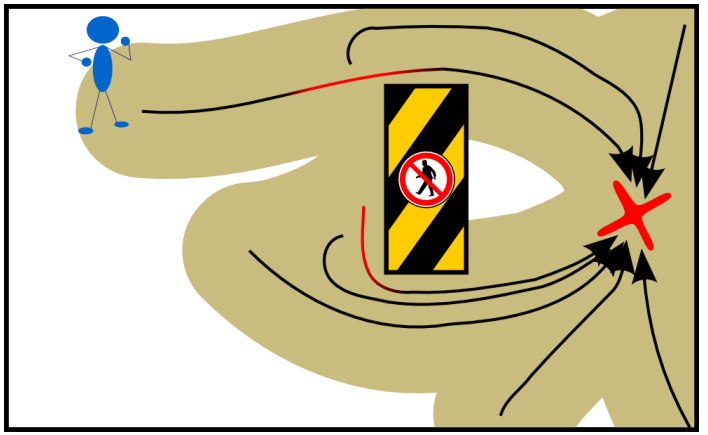
Key Ideas

- learn: create a library of trajectories
- **index by state: select action from any trajectory based on state**
- learn: improve library based on experience



Key Ideas

- learn: create a library of trajectories
- index by state: select action from any trajectory based on state
- **learn: improve library based on experience**

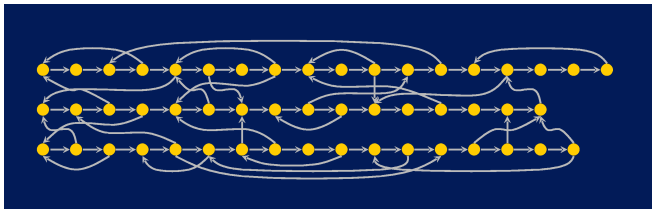


Outline

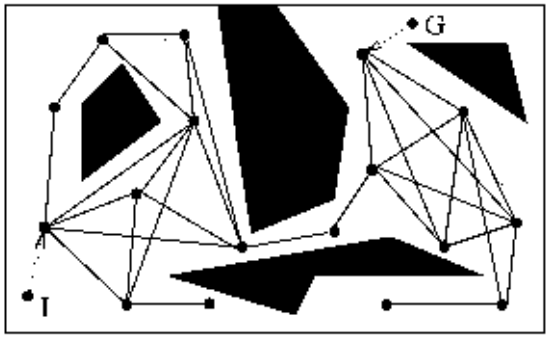
- Motivation
- Related Work
- Marble Maze implementation
- Little Dog implementation
- Learning from Demonstration
- Library Transfer

Existing Libraries

- in graphics [eg. Lee et al. 2002]
  - motion graph: library of character motions
  - blending of motions [eg. Safonova 2006]

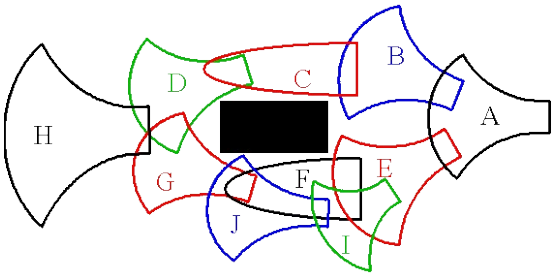


- in planning
  - persistent stores [Grossman 1992]
  - probabilistic road maps [eg. Kavraki et al. 1996]

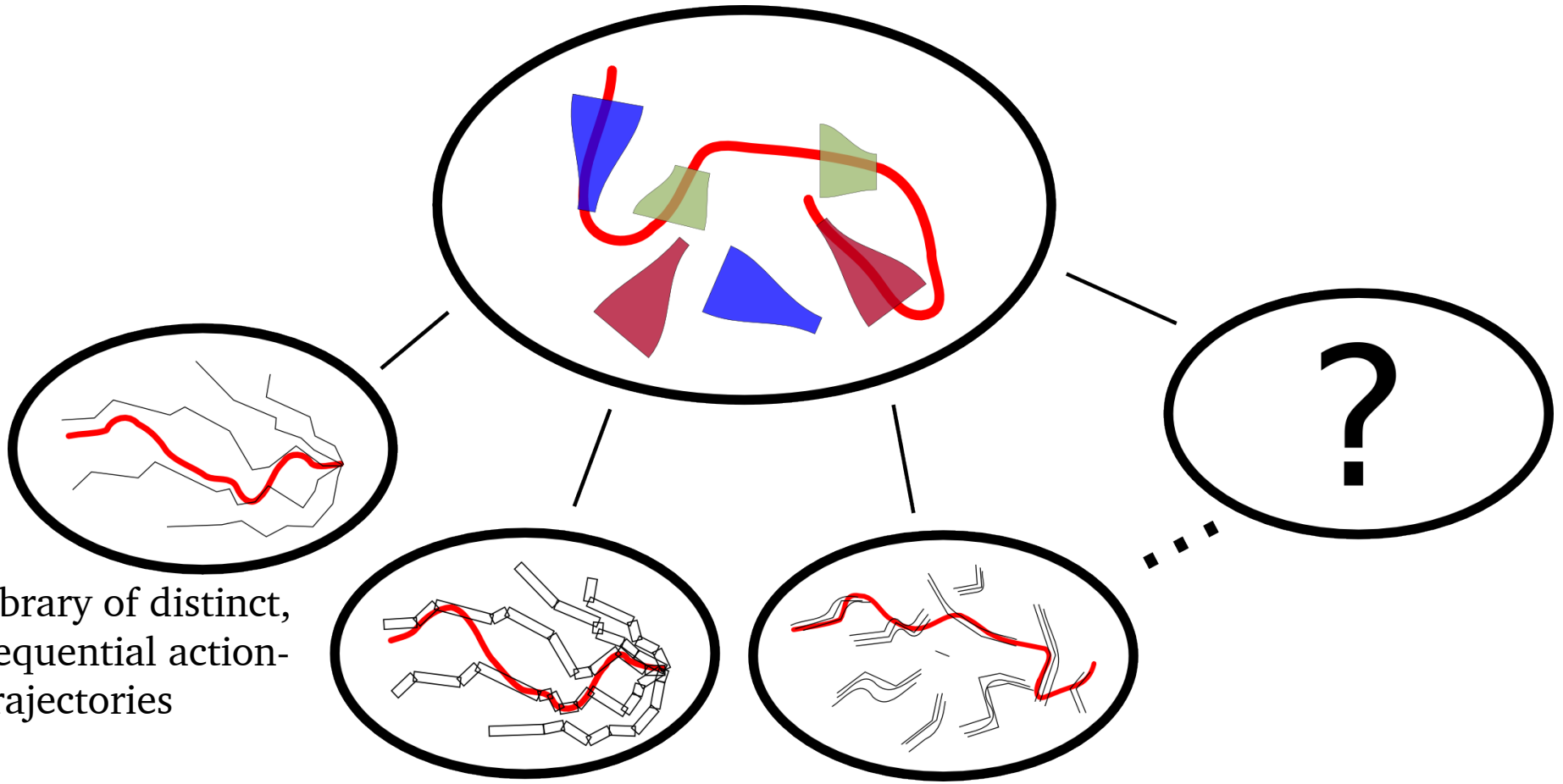


Existing Libraries

- in control
  - hybrid helicopter control [Frazzoli 2001]
  - composition of local potential functions [eg. Conner 2003]
  - local trajectory optimizers [Atkeson 1994]



Kinds of Libraries

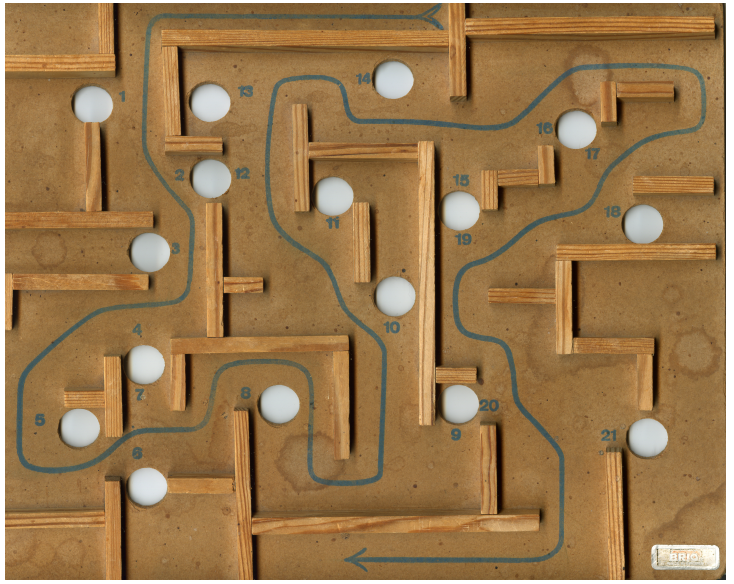


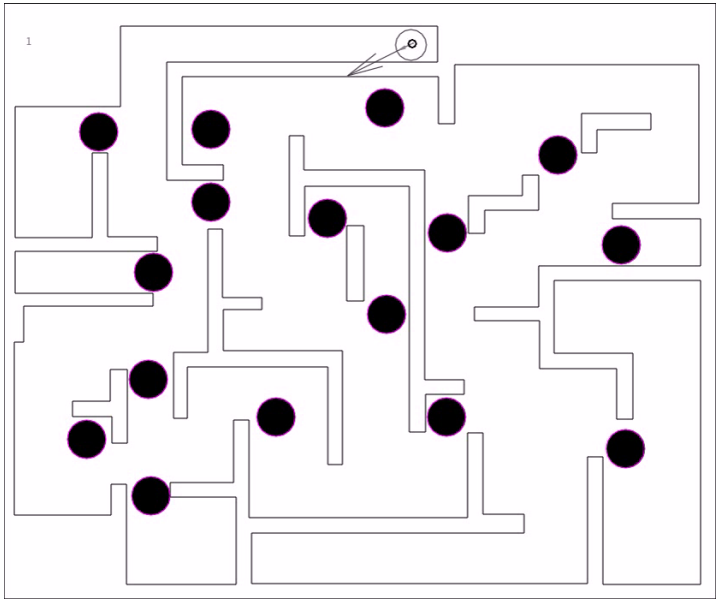
library of distinct, sequential action-trajectories

library of distinct, sequential subpolicies

library of trajectory fragments

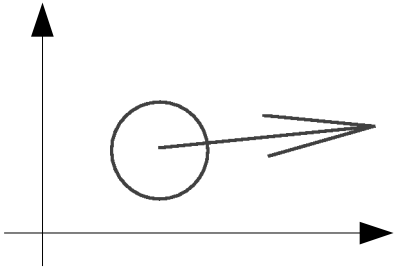
Marble Maze



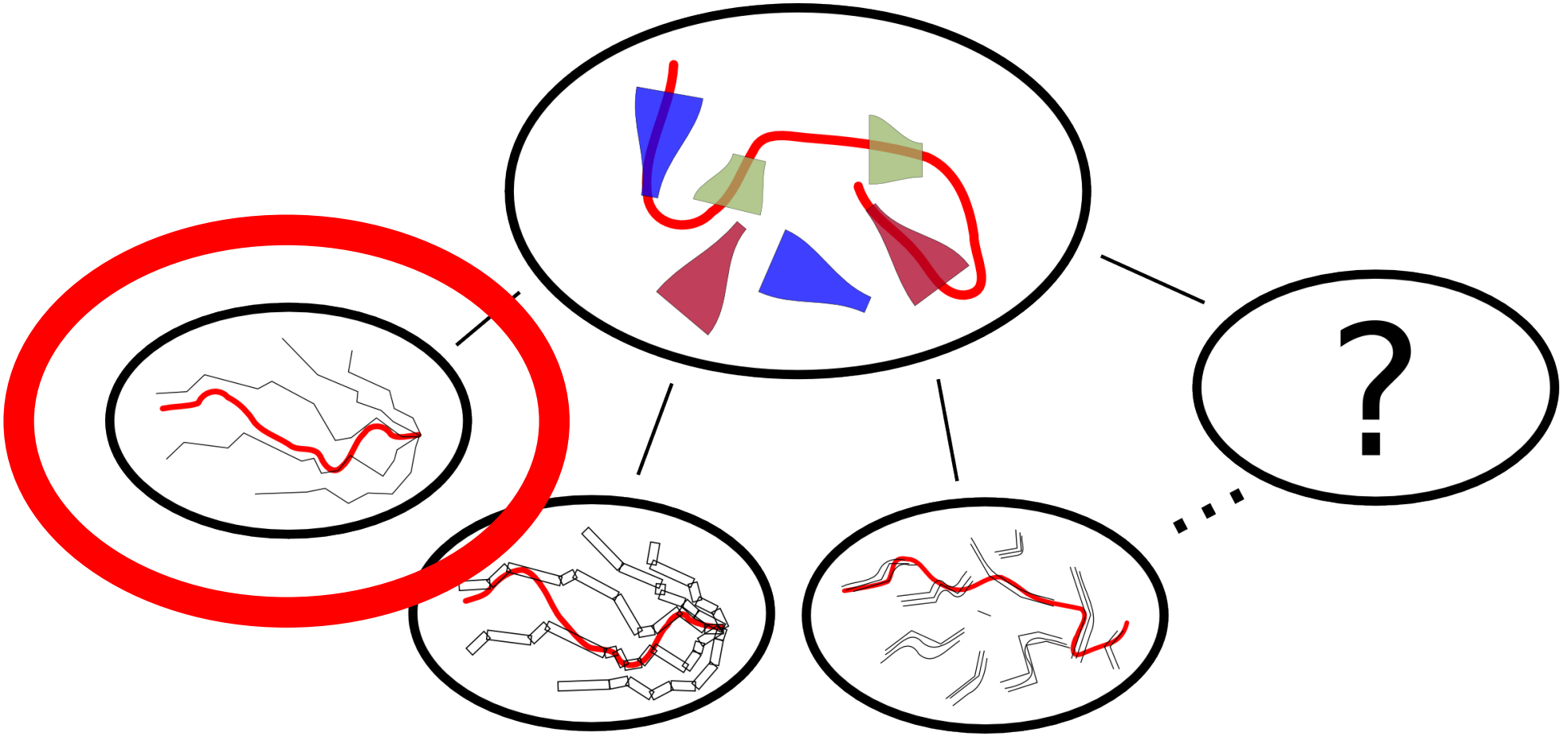


Marble Maze Representation

- State representation:
  <x, y, dx, dy>
- Action representation:
  <fx, fy>
- Force similar to tilt



Kinds of Libraries

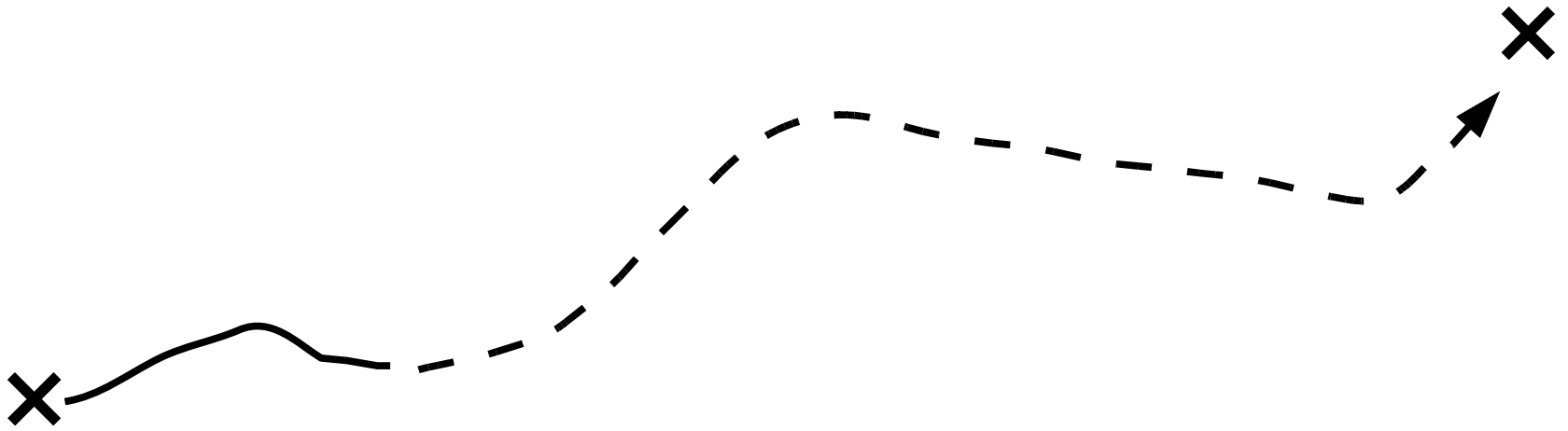


Key Issues

- how to create trajectories
- when and where to add new trajectories
- how to “fine tune” library

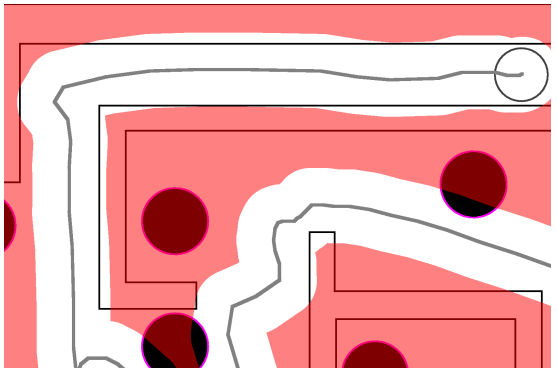
How?

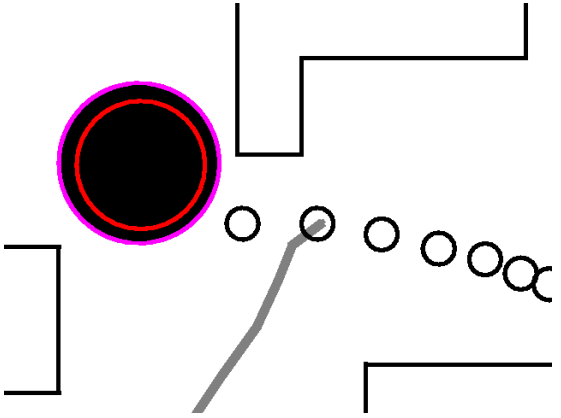
- different planning algorithms
  - examples: A*, RRT, domain specific ...
- interchangeable

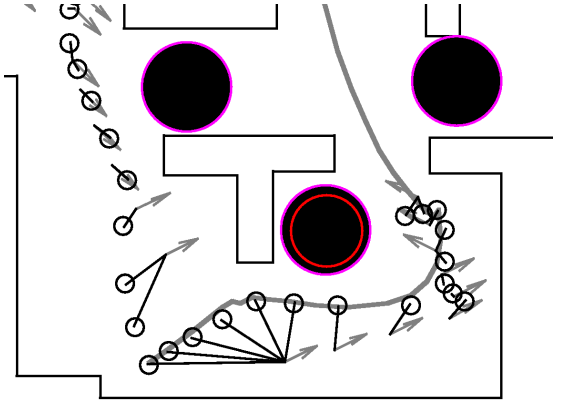


When and Where Add Trajectories?

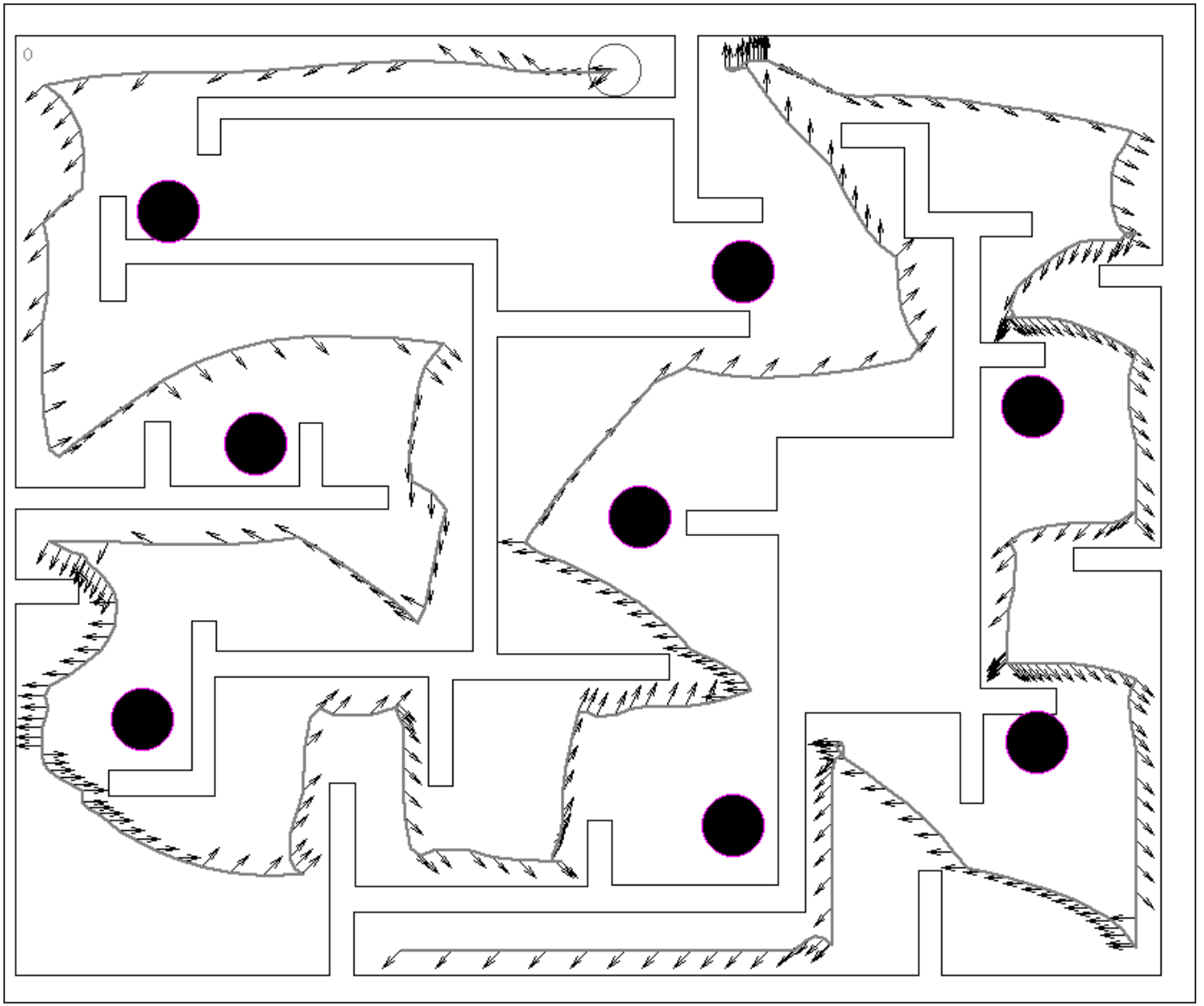
- From start state
- From vicinity of original trajectory
- From last states before failure
- From states with largest distance to library
- From states when stuck



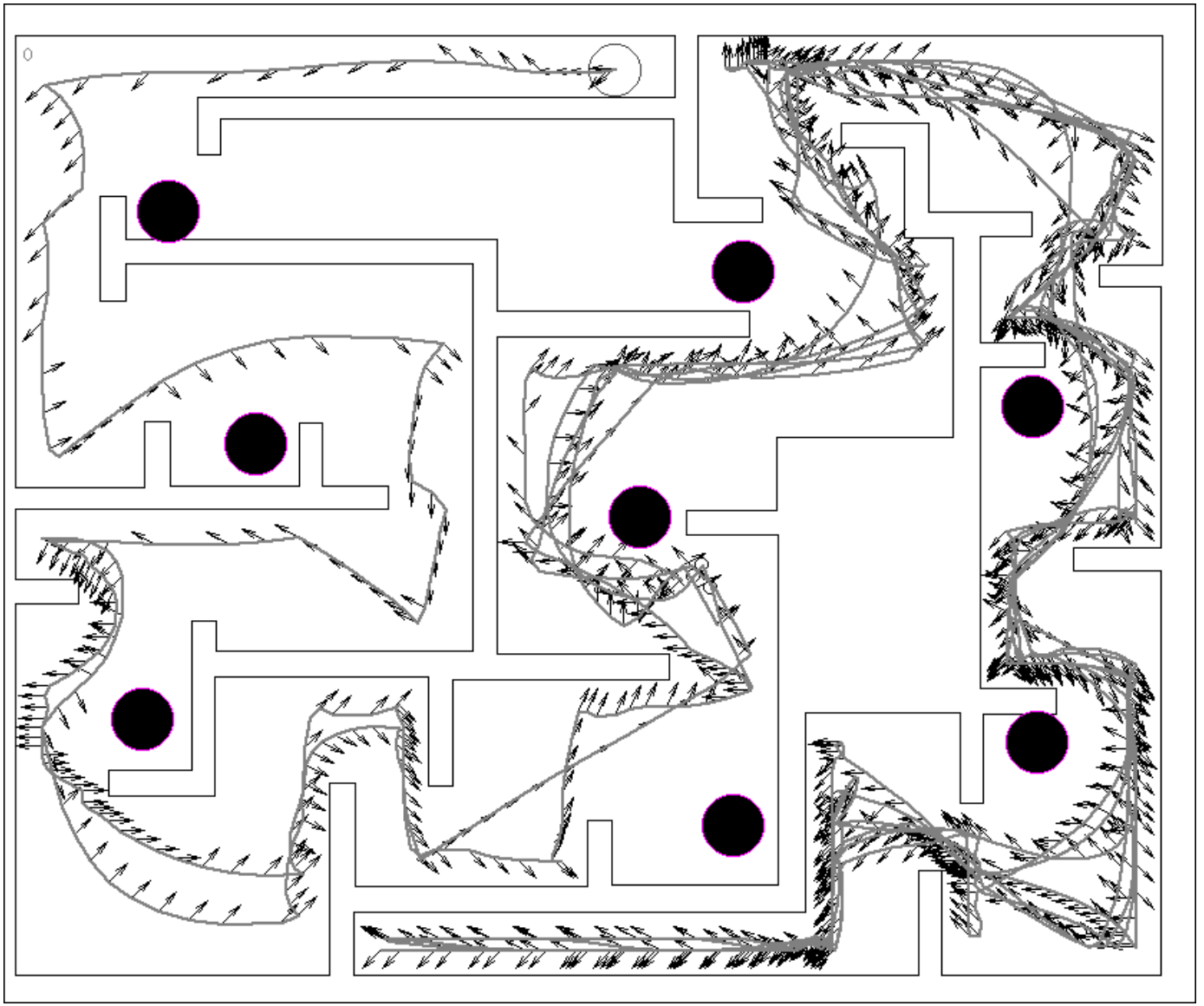




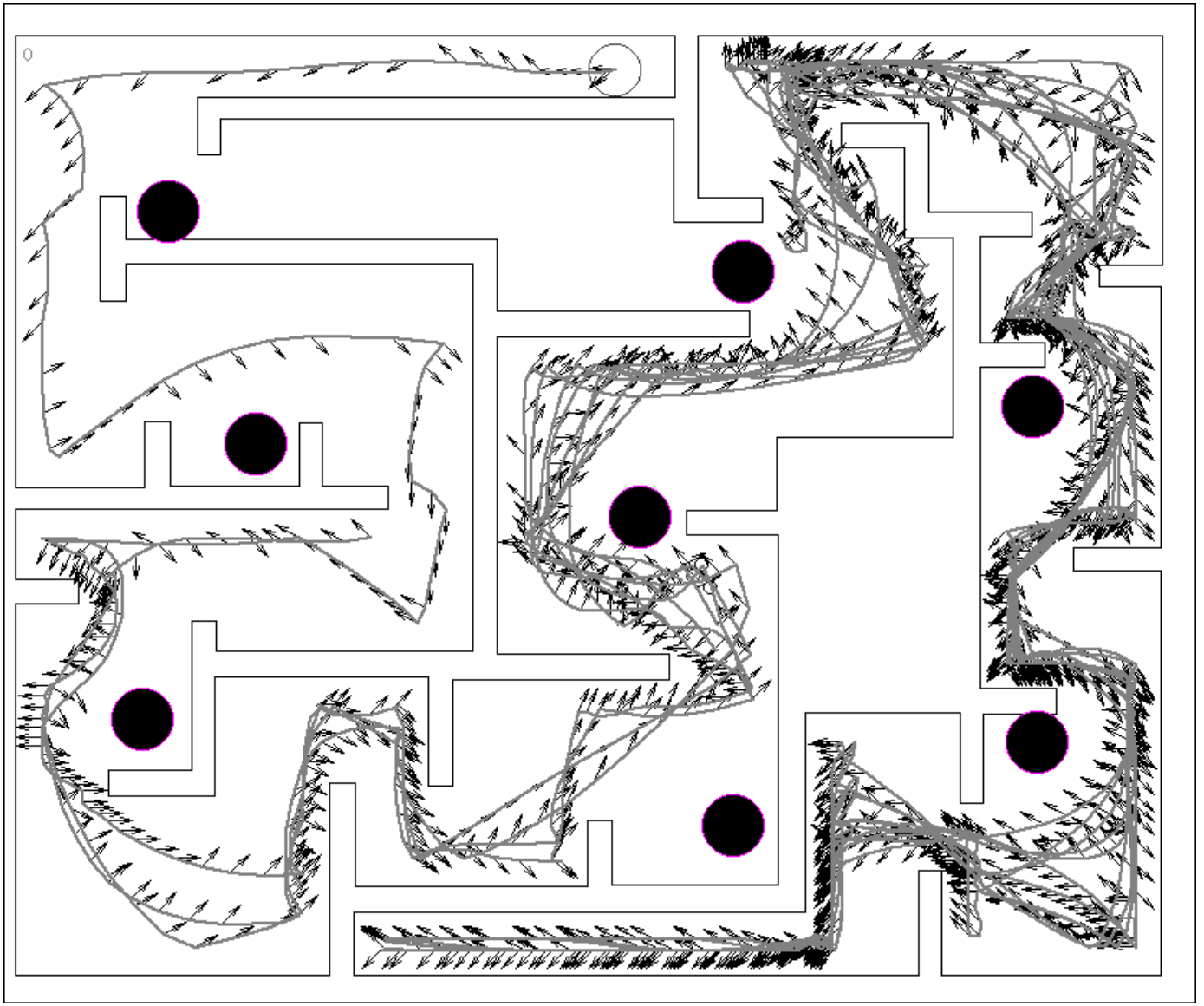
Library Evolution



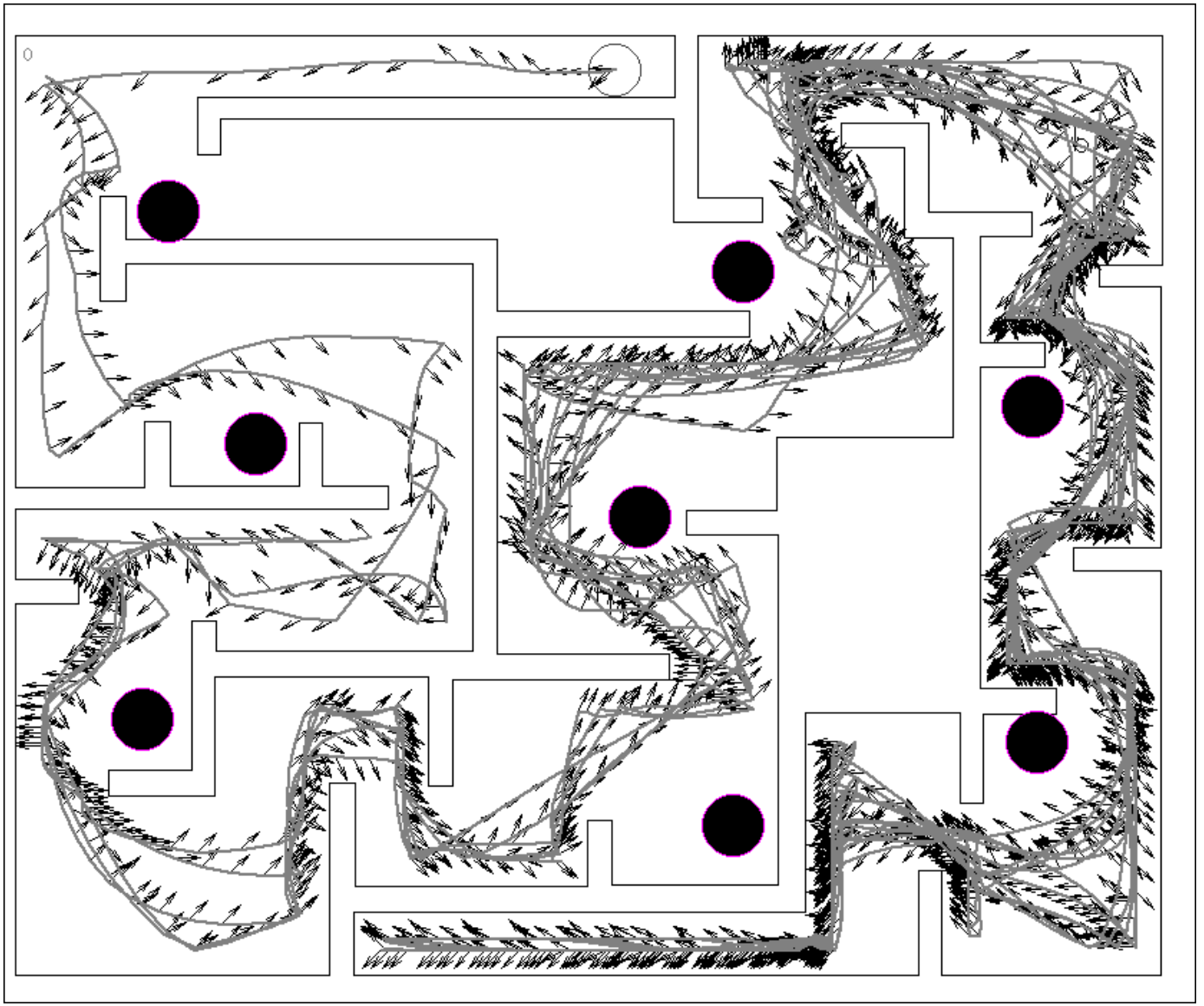
Library Evolution



Library Evolution



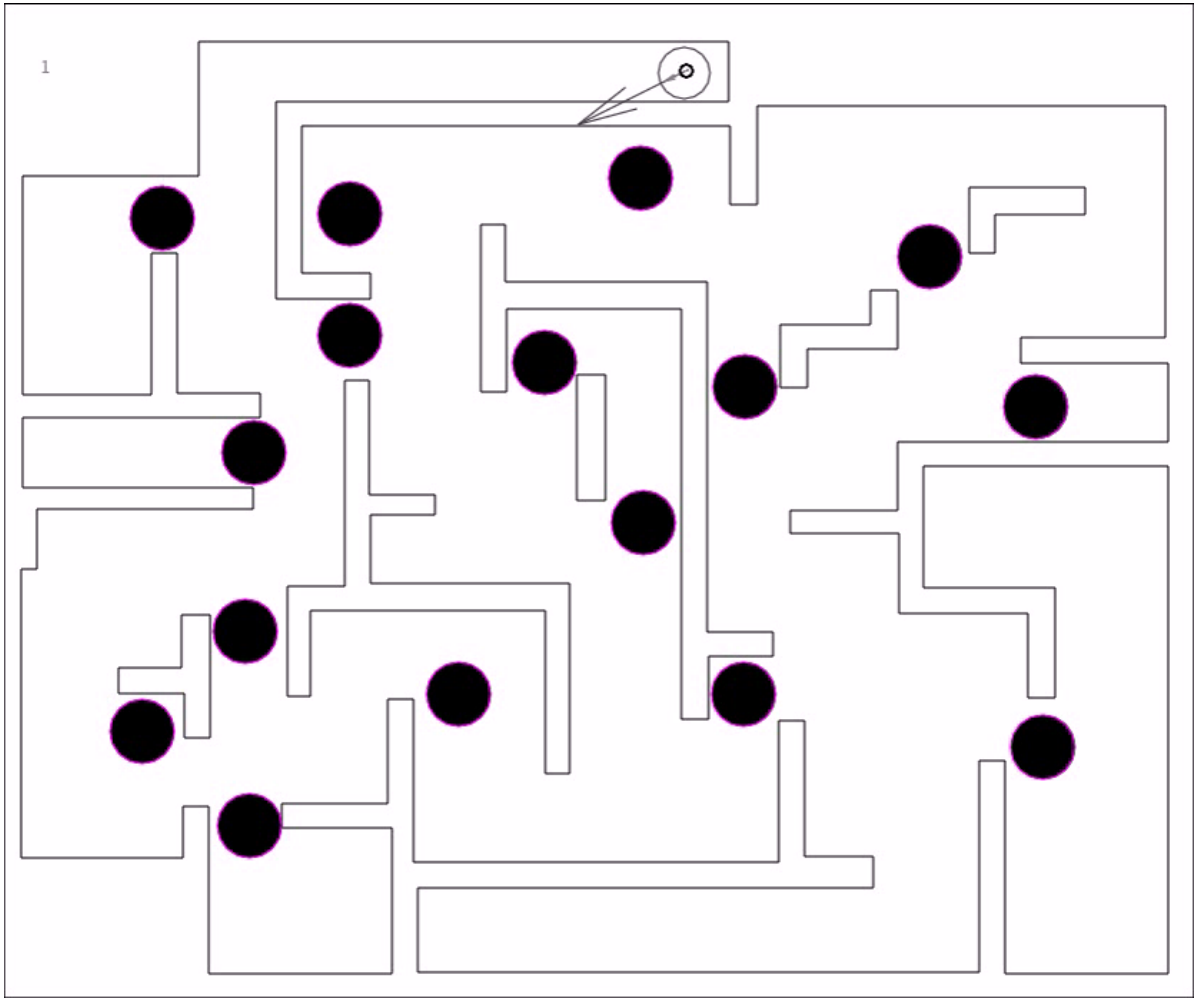
Library Evolution



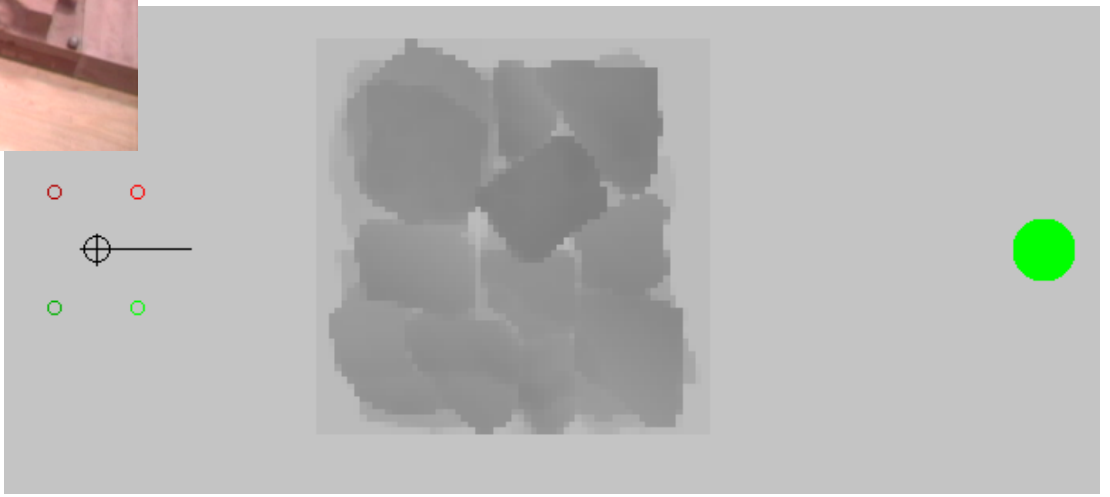
Fine Tune

- penalize states in library that lead to failure

Video



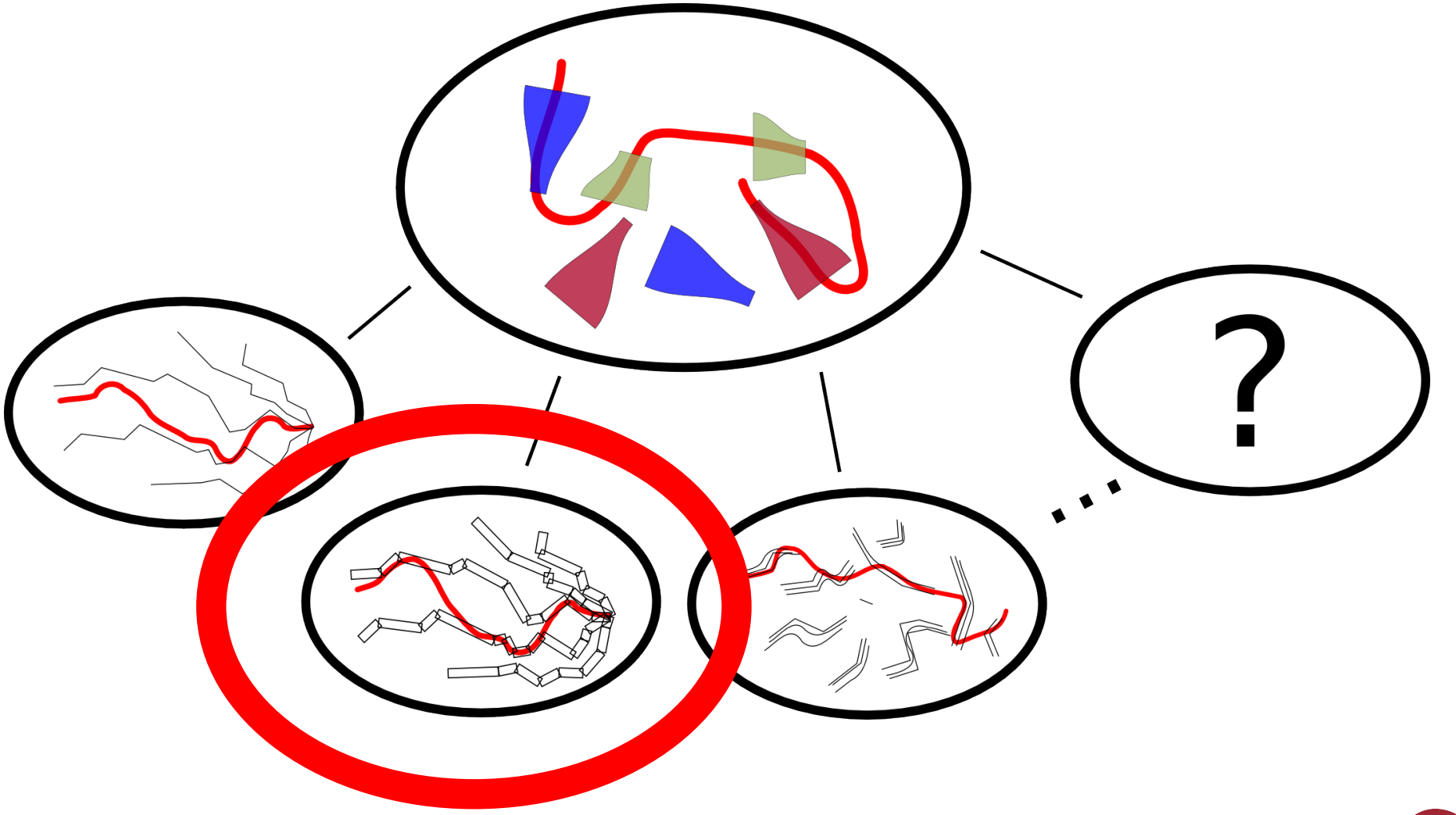
Little Dog



Little Dog Representation

- Used for early experiments
- State representation:
  <fl_x, fl_y, ... hl_y>
- Action representation:
  <{fl, fr, ... hl}, new_x, new_y>

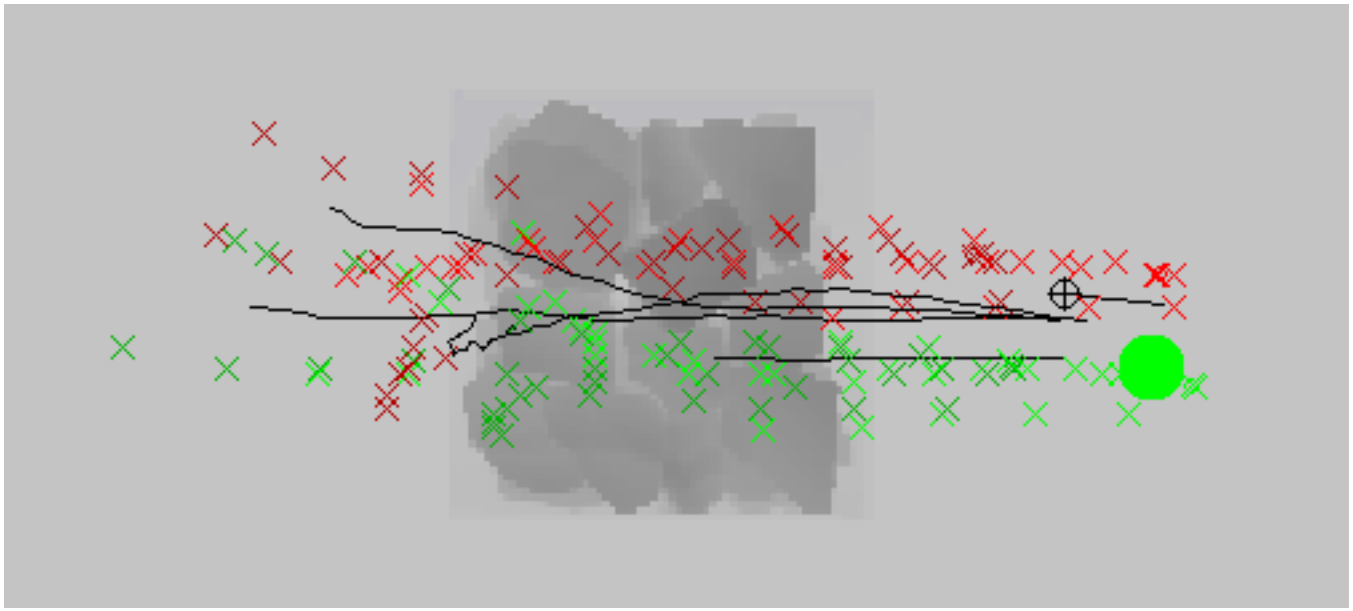
Kinds of Libraries



Key Issues

- how to create trajectories
  - foot step planner (Joel Chestnutt)
- when and where to add new trajectories
  - from start
  - unsafe step
  - excessive repeats
- how to “fine tune” library
  - slow down step

Example Library



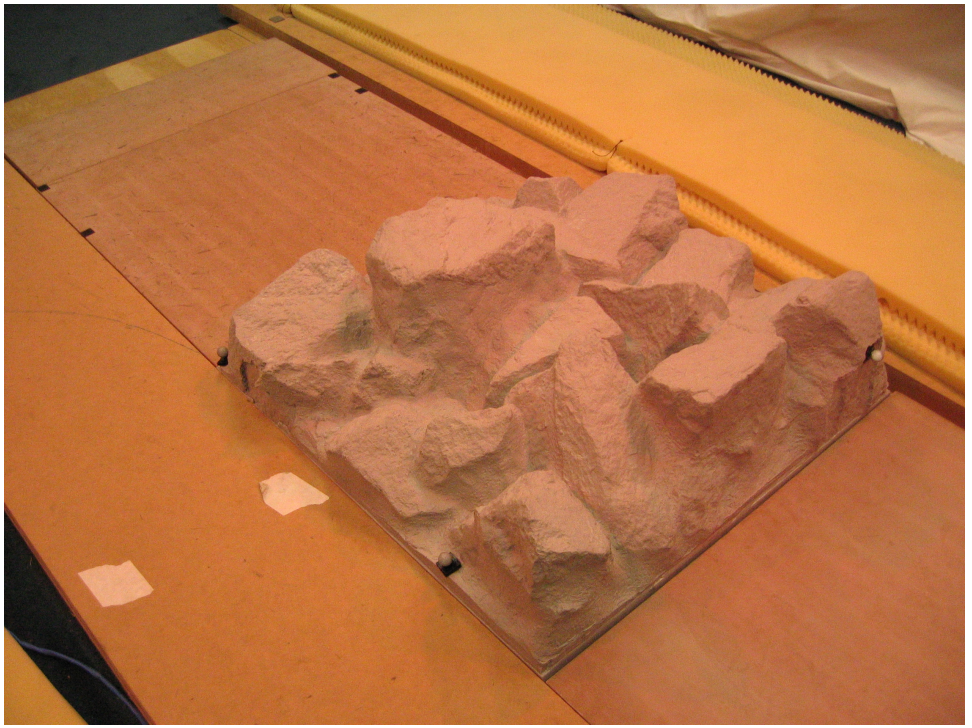
Video

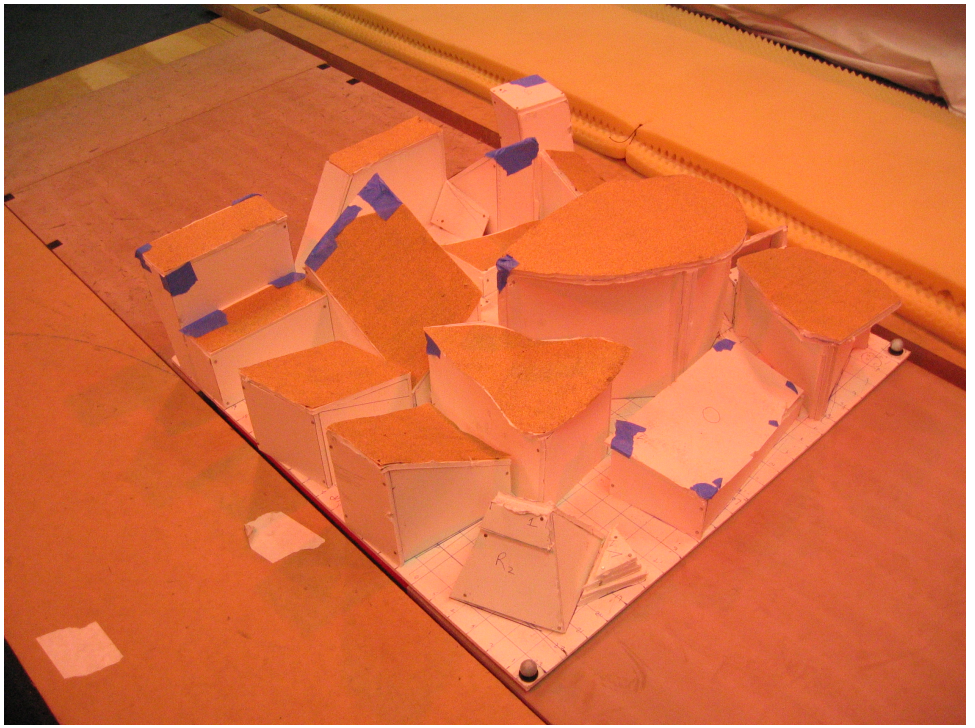
Proposed Work

- Create libraries from demonstration
- Use local state representations to transfer libraries
  - within task
  - across tasks

Learning from Demonstration

- Dynamic maneuvers
- Difficult terrains





Learning from Demonstration

- using keyboard/joystick to command legs of Little Dog
- segment into “steps”
- add controller during playback to adapt to differences

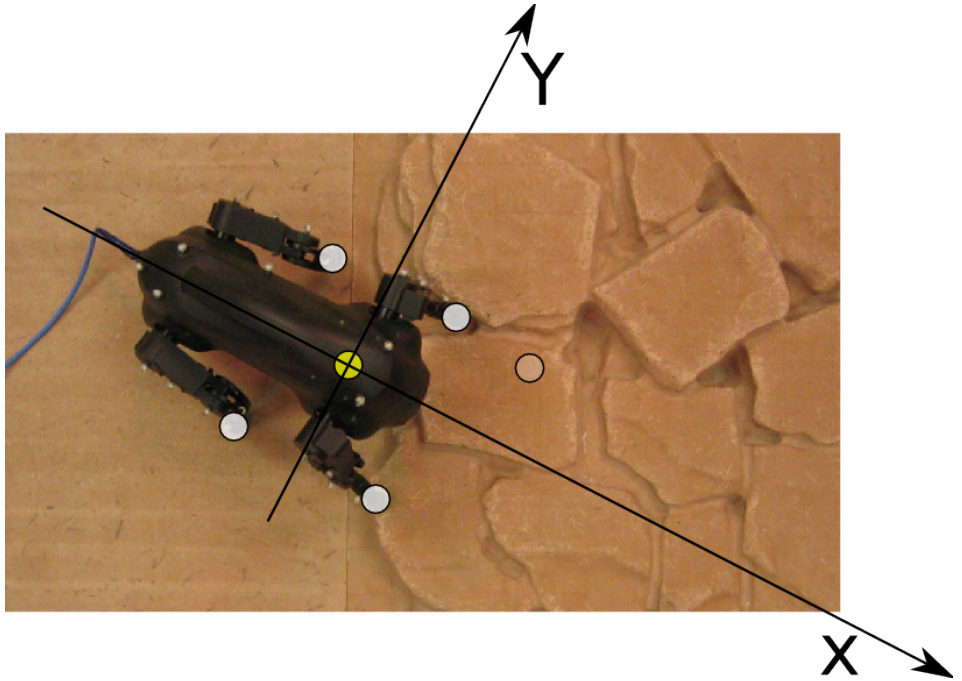
- instead of joystick: learn from spying

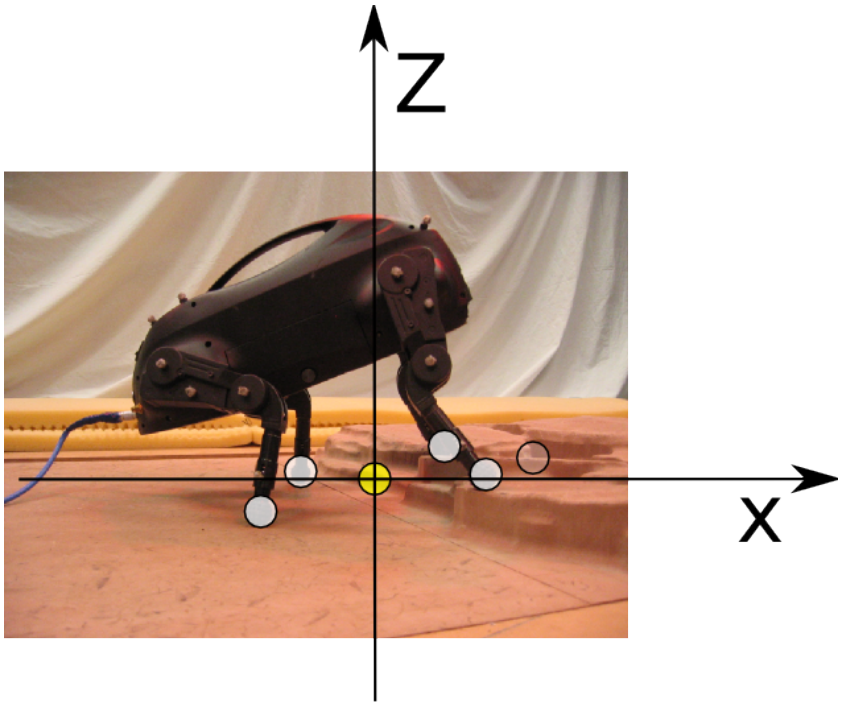
Local State

- a local feature-based representation of the state
  - Little Dog: footprint/terrain
  - Marble Maze: local walls, holes
- Planning or No Planning?
  - without planning: greedy, requires goal feature
  - with planning: no goal feature, more flexible

Local State
Little Dog

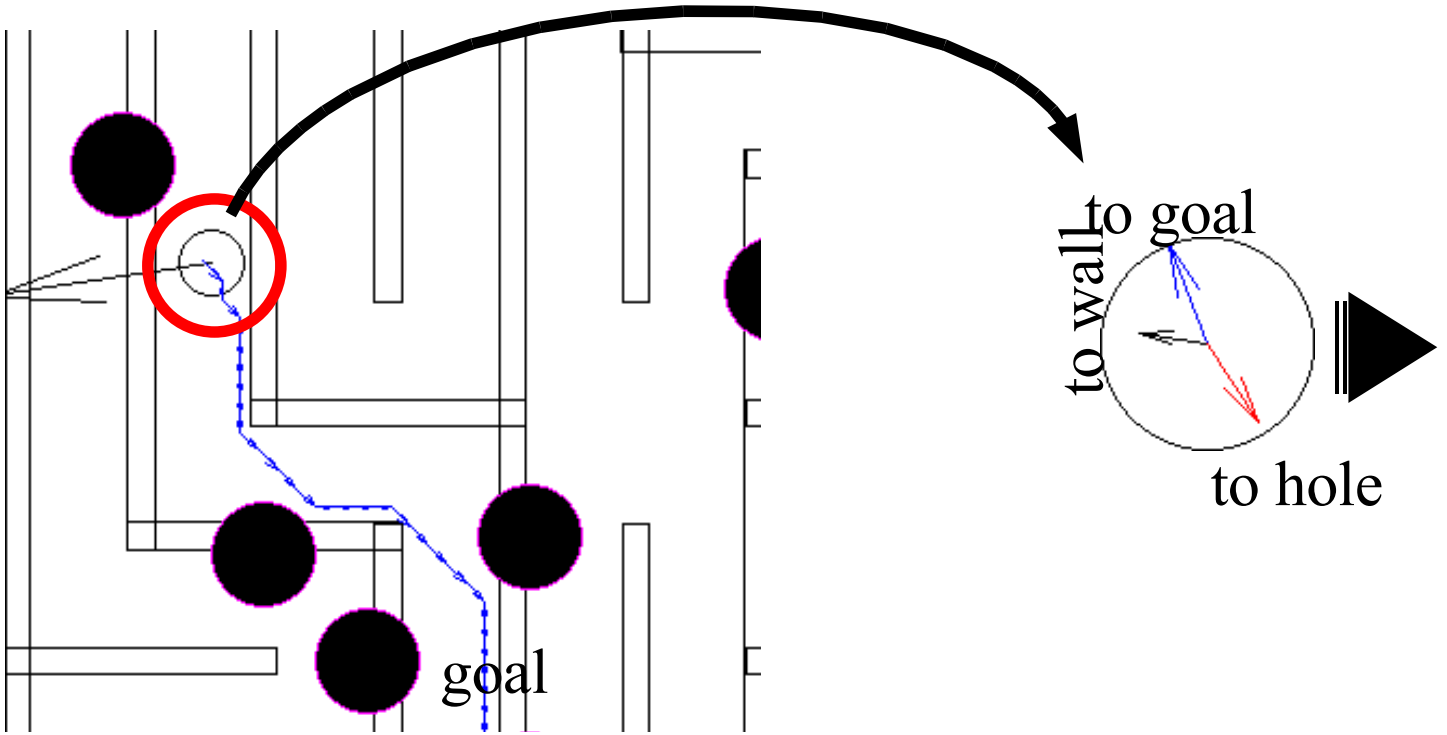
- footprint around robot
- can exploit symmetries





Local State
Marble Maze

- frame attached to marble, aligned with velocity
- vectors to closest hole, wall and toward goal

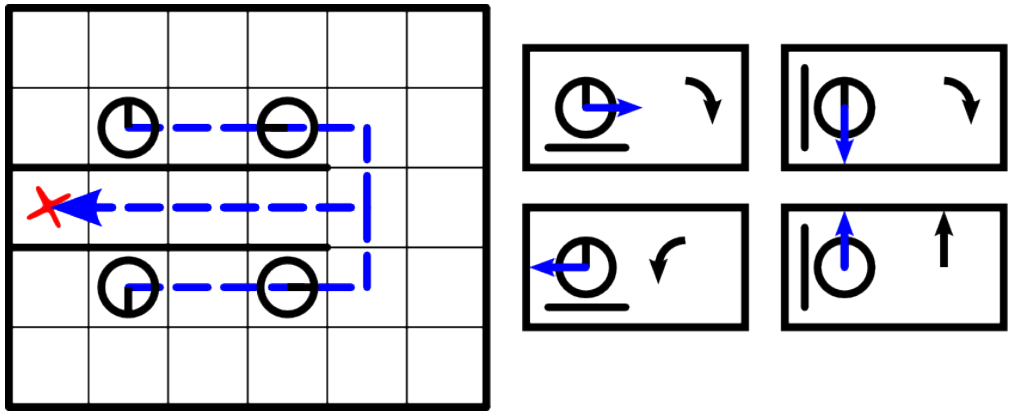


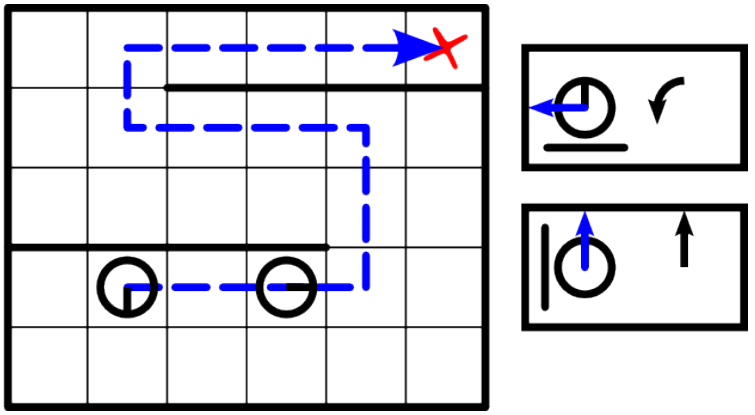
Local State: Explicit goal feature

- goal-based feature as part of state
- associate action/sub policy with given state

Local State: Explicit goal feature

- a simple example:



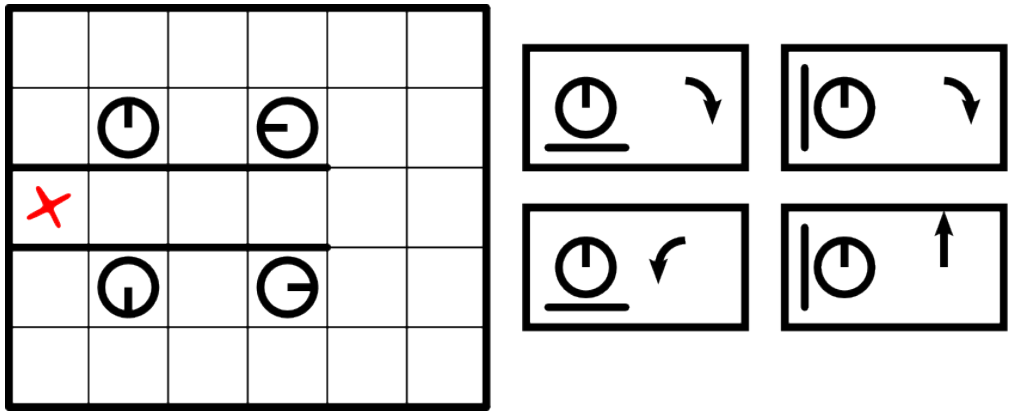


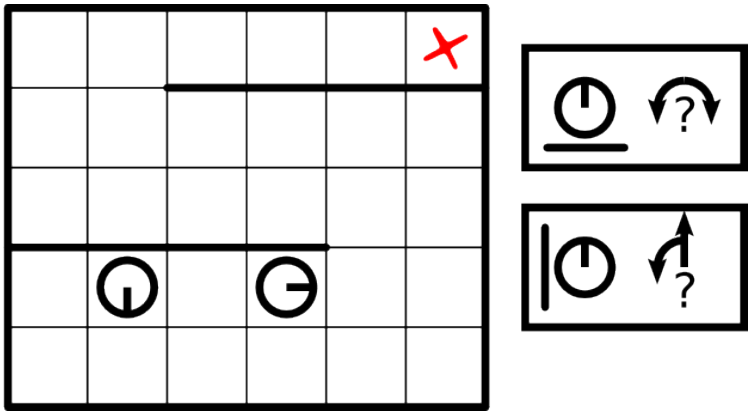
Local State: Goal-less features

- a local feature-based representation of the state
- no notion of "goal"
- allows for greater re-use
- requires search to disambiguate and find path reaching goal
  - how to model actions (subpolicies?)

Local State: Goal-less features

- a simple example:



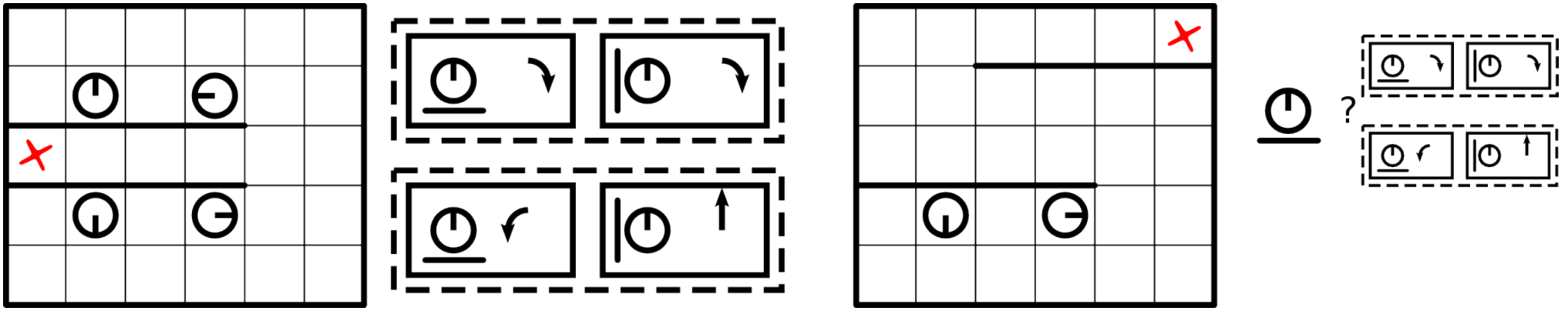


Local State: Extended Action Generation

- a local feature-based representation of the state
- no notion of "goal"
- cluster similar actions in similar states
- mostly useful for low-level actions (Marble Maze)

Local State: Extended Action Generation

- a simple example:



Research Topics

- how to create trajectories
  - planning ☑
  - learning from demonstration [LDog] ☐
  - learning from spying [LDog] ☐
- how to organize
  - local vs global state, transfer [MMaze,LDog] (☑)☐
  - goal vs no goal+search [LDog] ☐
  - action vs policies (abstraction) ☑

Research Topics

- how to improve
  - reward/penalize [MMaze] ☑
  - modify trajectories (DDP, DIRCOL) ☒
  - modify subpolicies [LDog] (☑)
  - clustering [MMaze] ☐
  - library improves planner ☒
- Graduation: Winter 2007

Conclusion

- Presented trajectory libraries to solve high-dimensional control problems
- Applied to Marble Maze and Little Dog using global state
- Proposed learning from demonstration
- Proposed additional flavors of trajectory libraries using local state for transfer