CSE-571
Robotics

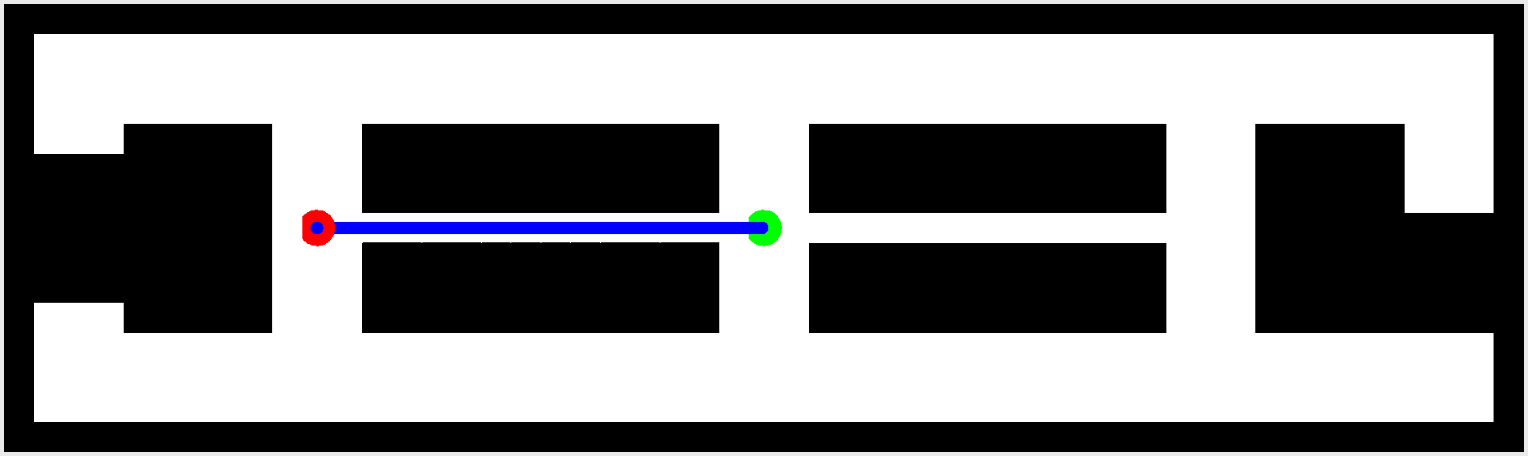
Planning and Control:

Markov Decision Processes

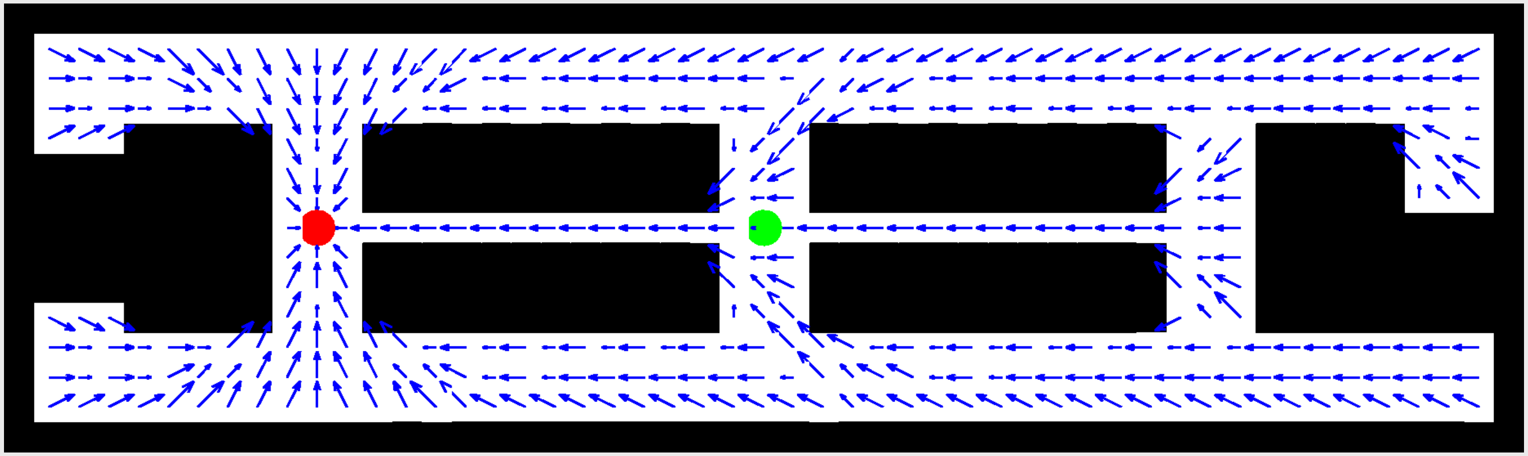
Problem Classes

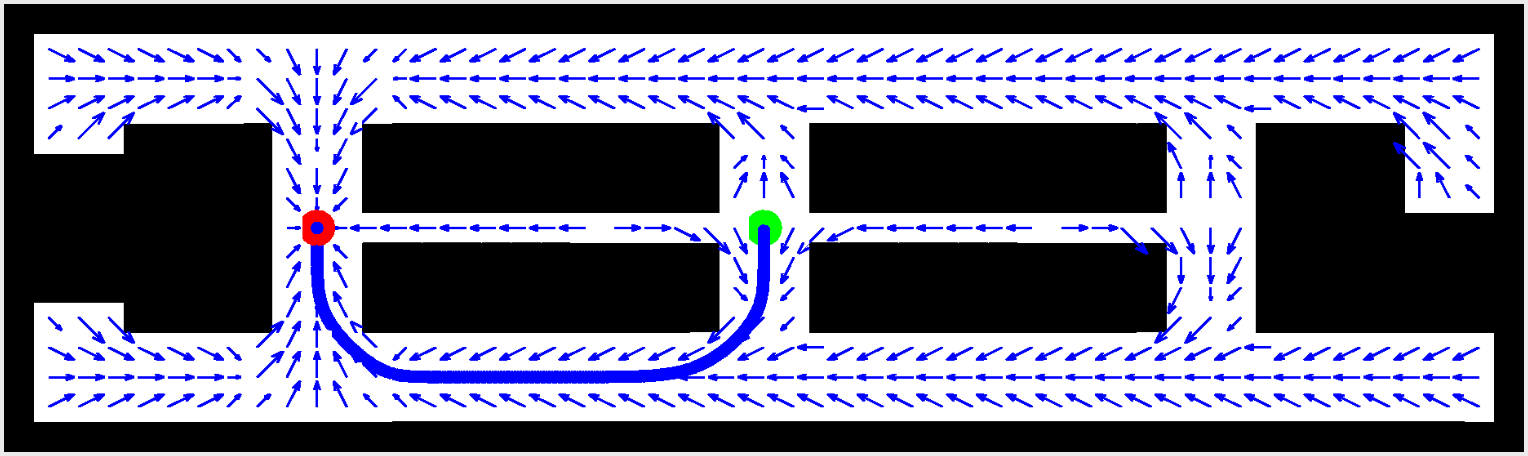
- Deterministic vs. stochastic actions
- Full vs. partial observability

Deterministic, fully observable

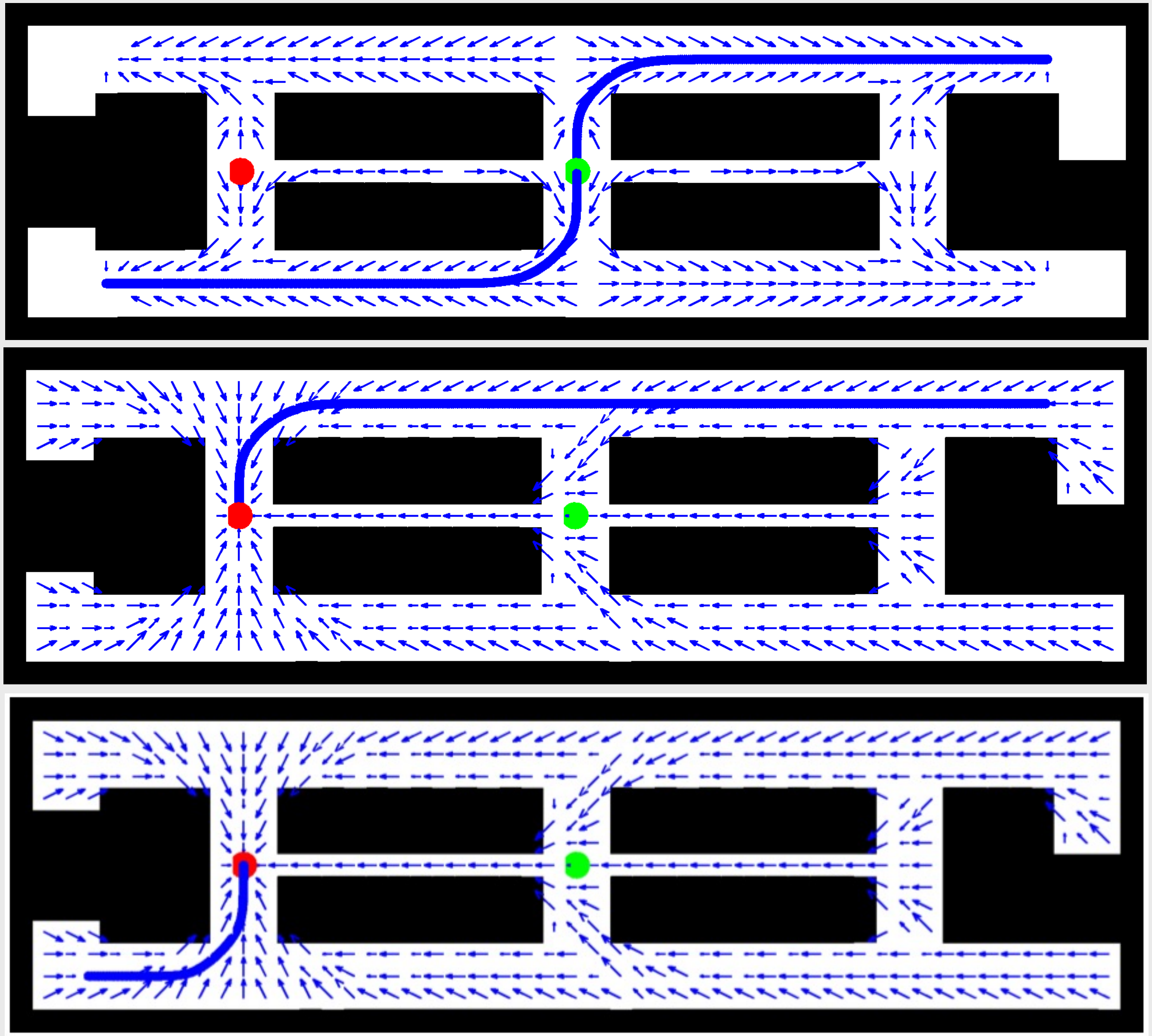


Stochastic, Fully Observable

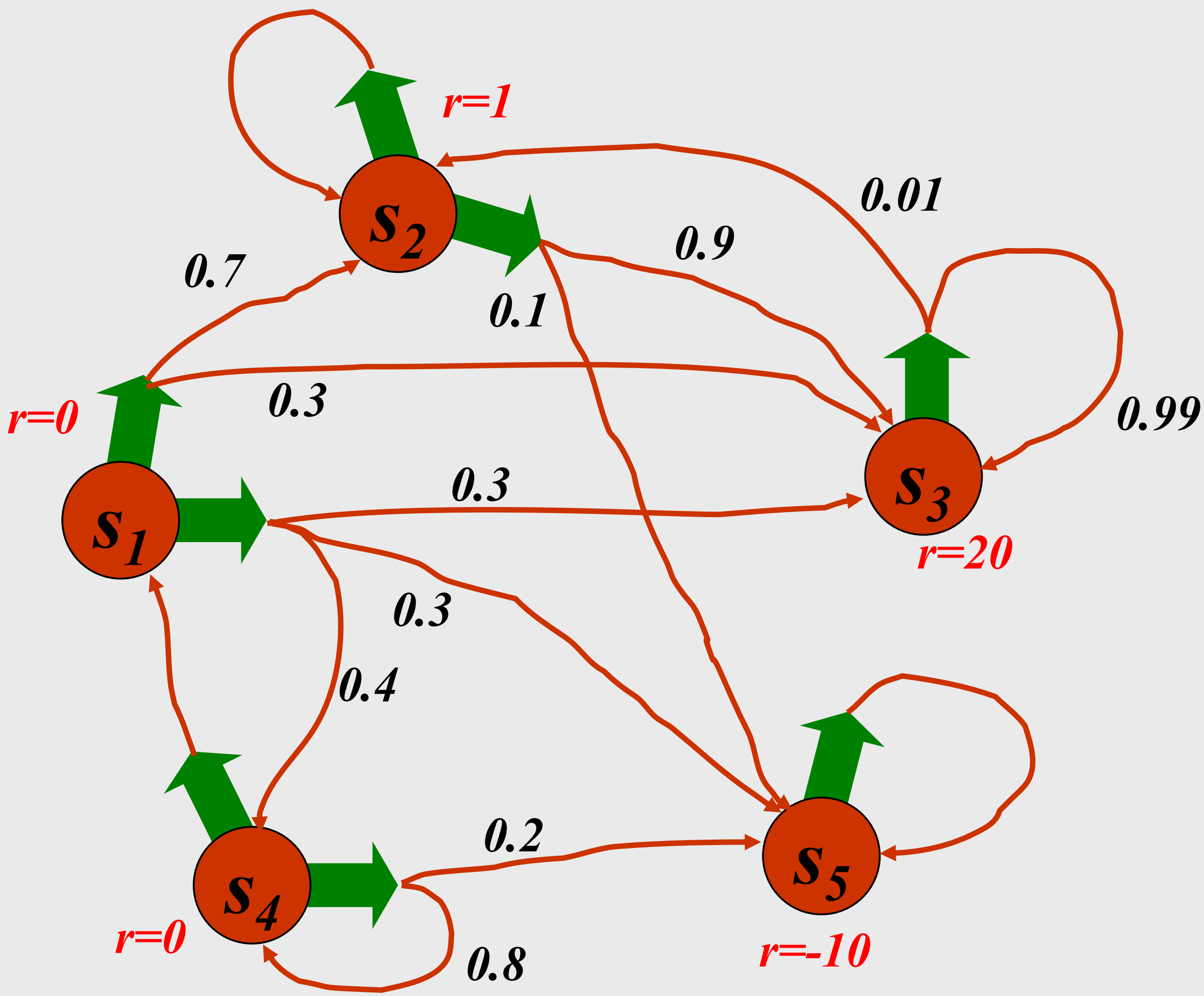




Stochastic, Partially Observable



Markov Decision Process (MDP)



Markov Decision Process (MDP)

- **Given:**
- States x
- Actions u
- Transition probabilities $p(x'|u,x)$
- Reward / payoff function $r(x,u)$

- **Wanted:**
- Policy $\pi(x)$ that maximizes the future expected reward

Rewards and Policies

- Policy (general case):

$$\pi: \quad z_{1:t-1}, u_{1:t-1} \rightarrow \; u_t$$

- Policy (fully observable case):

$$\pi: \quad x_t \rightarrow \; u_t$$

- Expected cumulative payoff:

$$R_T = E \left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{t+\tau} \right]$$

  - T=1: greedy policy
  - T>1: finite horizon case, typically no discount
  - T=infty: infinite-horizon case, finite reward if discount < 1

Policies contd.

- Expected cumulative payoff of policy:

$$R_T^\pi(x_t) = E\left[\sum_{\tau=1}^{T} \gamma^\tau r_{t+\tau} \mid u_{t+\tau} = \pi\left(z_{1:t+\tau-1} u_{1:t+\tau-1}\right)\right]$$

- Optimal policy:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \quad R_T^\pi(x_t)$$

- 1-step optimal policy:

$$\pi_1(x) = \underset{u}{\operatorname{argmax}} \quad r(x,u)$$

- Value function of 1-step optimal policy:

$$V_1(x) = \gamma \max_u \; r(x,u)$$

2-step Policies

- Optimal policy:

$$\pi_2(x) = \operatorname*{argmax}_u \left[r(x,u) + \int V_1(x')p(x' \mid u,x)dx' \right]$$

- Value function:

$$V_2(x) = \gamma \max_u \left[r(x,u) + \int V_1(x')p(x' \mid u,x)dx' \right]$$

T-step Policies

- Optimal policy:

$$\pi_T(x) = \underset{u}{\operatorname{argmax}} \left[r(x,u) + \int V_{T-1}(x') p(x' \mid u, x) dx' \right]$$

- Value function:

$$V_T(x) = \gamma \max_u \left[r(x,u) + \int V_{T-1}(x') p(x' \mid u, x) dx' \right]$$

Infinite Horizon

- Optimal policy:

$$V_\infty(x) = \gamma \max_u \left[r(x,u) + \int V_\infty(x') p(x' \mid u,x) dx' \right]$$

- Bellman equation
- Fix point is optimal policy
- Necessary and sufficient condition

Value Iteration

- for all *x* do

$$\hat{V}(x) \leftarrow r_{\min}$$

- endfor

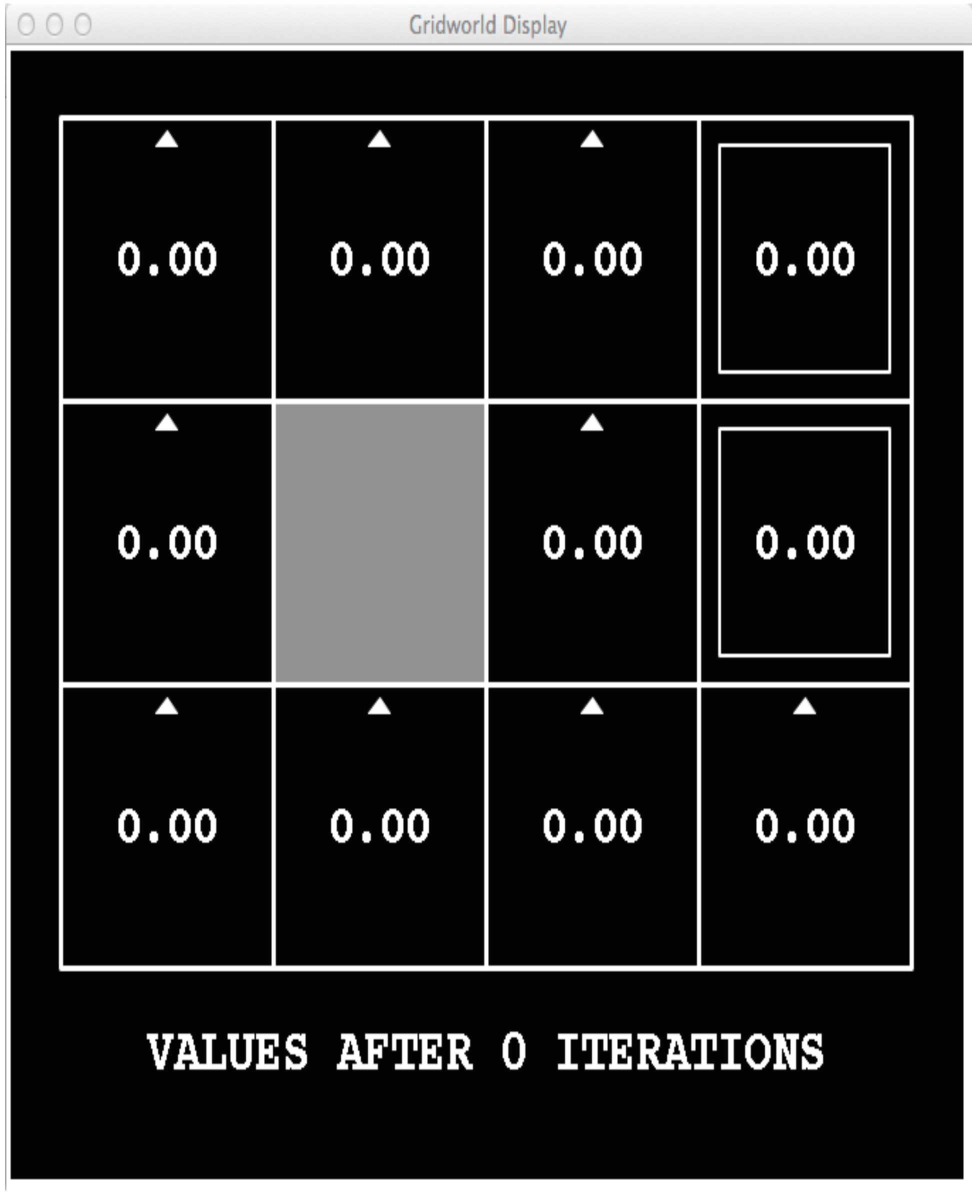
- repeat until convergence
  - for all *x* do

$$\hat{V}(x) \leftarrow \gamma \max_u \left[r(x,u) + \int \hat{V}(x') p(x' \mid u,x) dx' \right]$$

  - endfor
- endrepeat

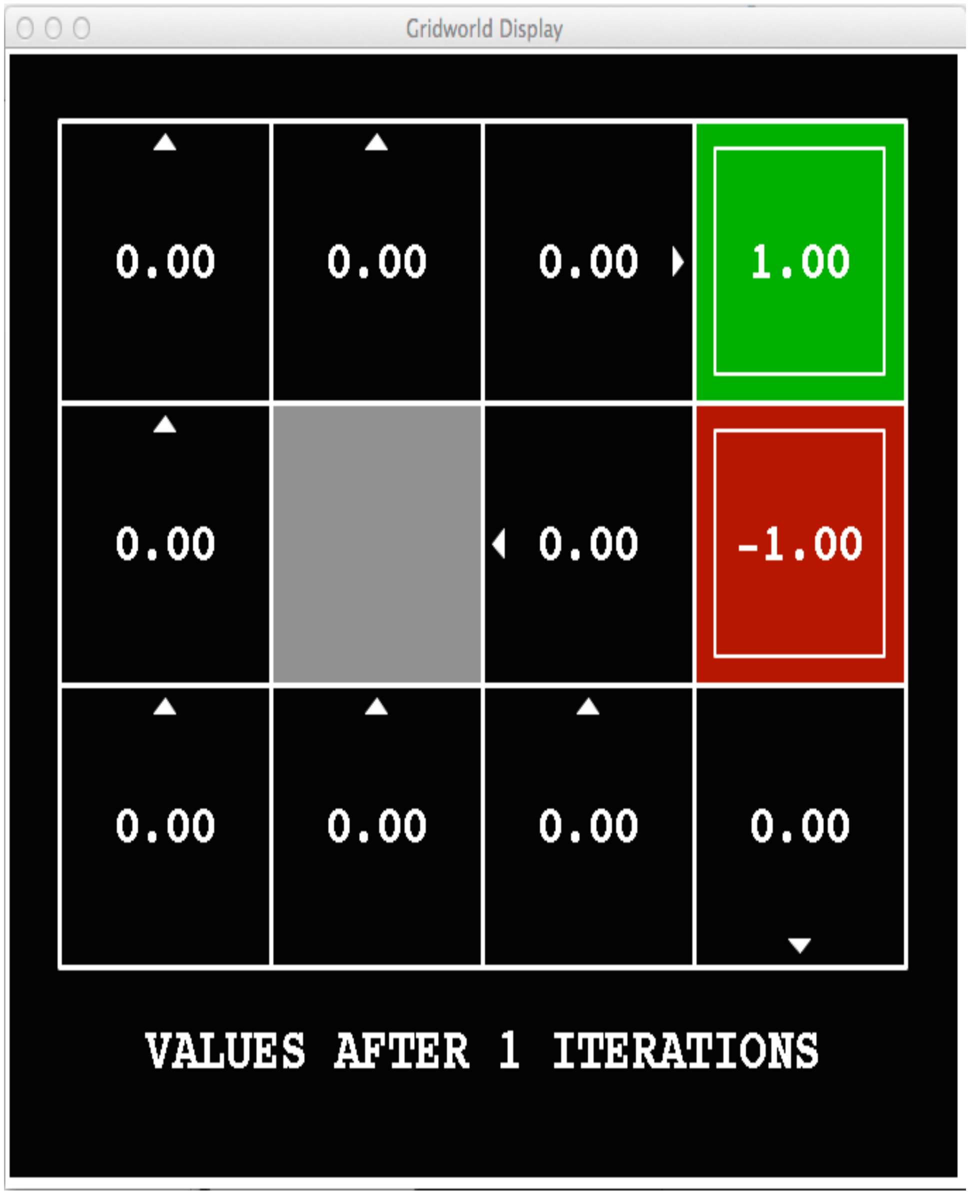
$$\pi(x) = \operatorname*{argmax}_u \left[r(x,u) + \int \hat{V}(x') p(x' \mid u,x) dx' \right]$$

k=0



Noise = 0.2
Discount = 0.9
Living reward = 0

k=1

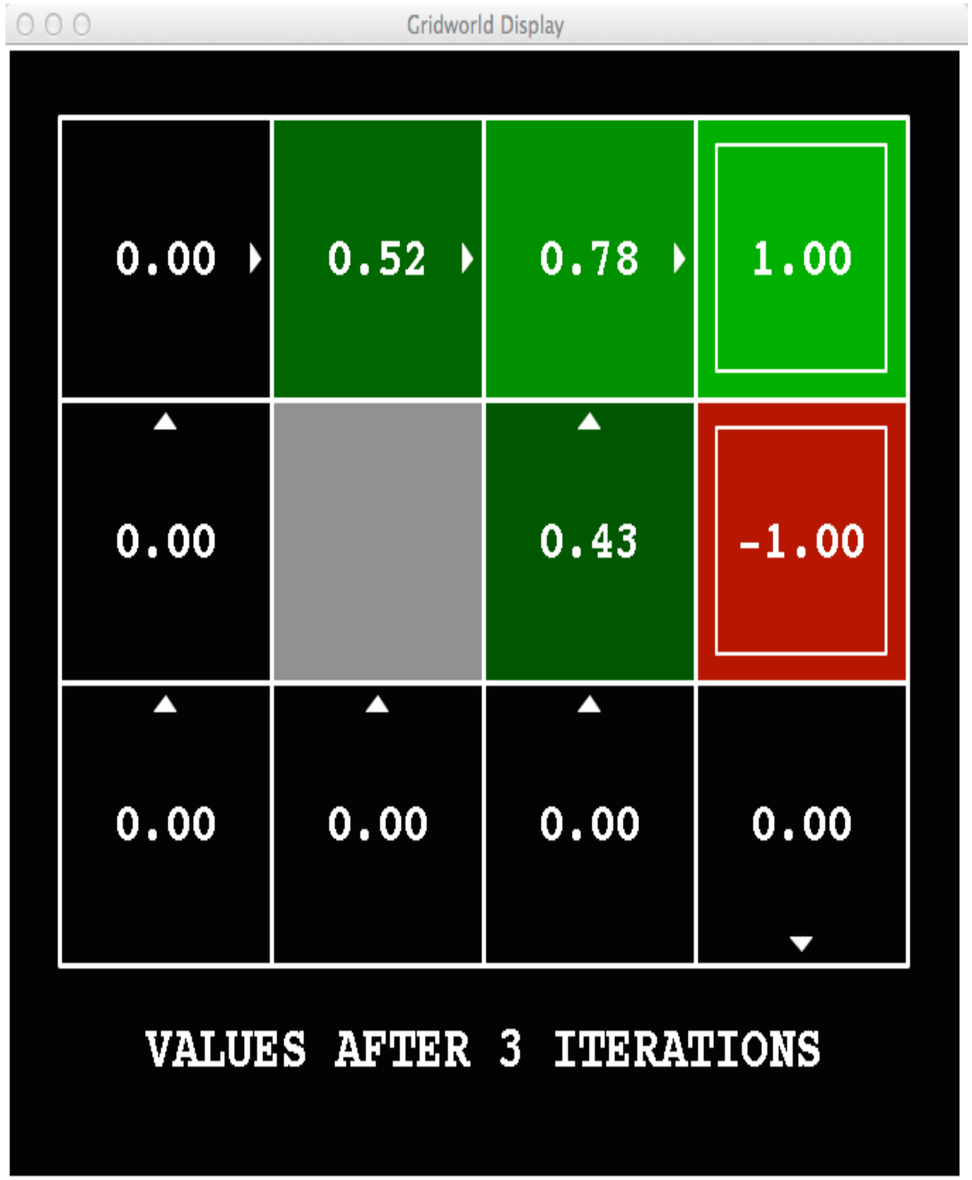


Noise = 0.2
Discount = 0.9
Living reward = 0

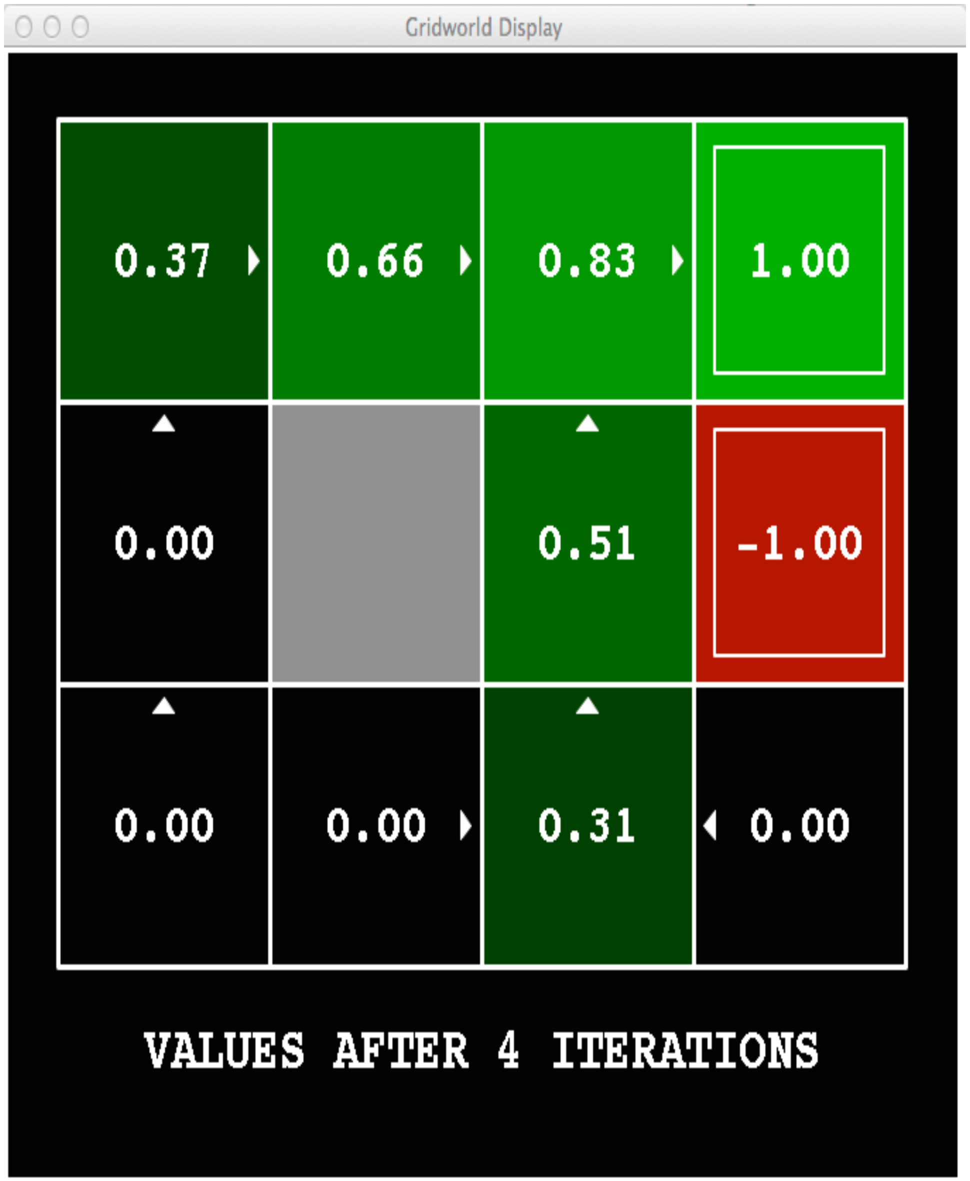
k=2


Gridworld Display
0.00
0.00
0.72
1.00
0.00
0.00
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 2 ITERATIONS

k=3


Gridworld Display
0.00
0.52
0.78
1.00
0.00
0.43
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 3 ITERATIONS

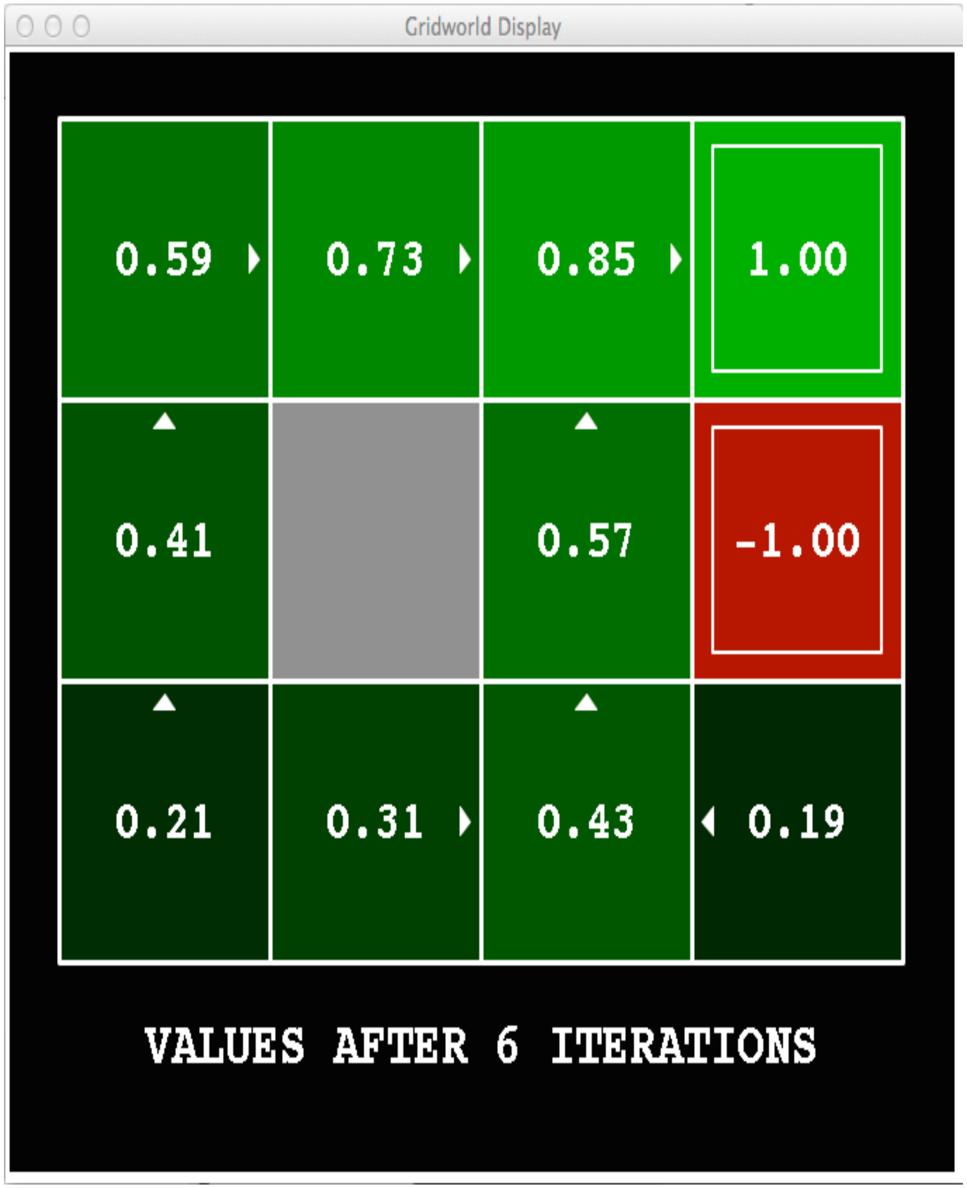
k=4


Gridworld Display
0.37
0.66
0.83
1.00
0.00
0.51
-1.00
0.00
0.00
0.31
0.00
VALUES AFTER 4 ITERATIONS

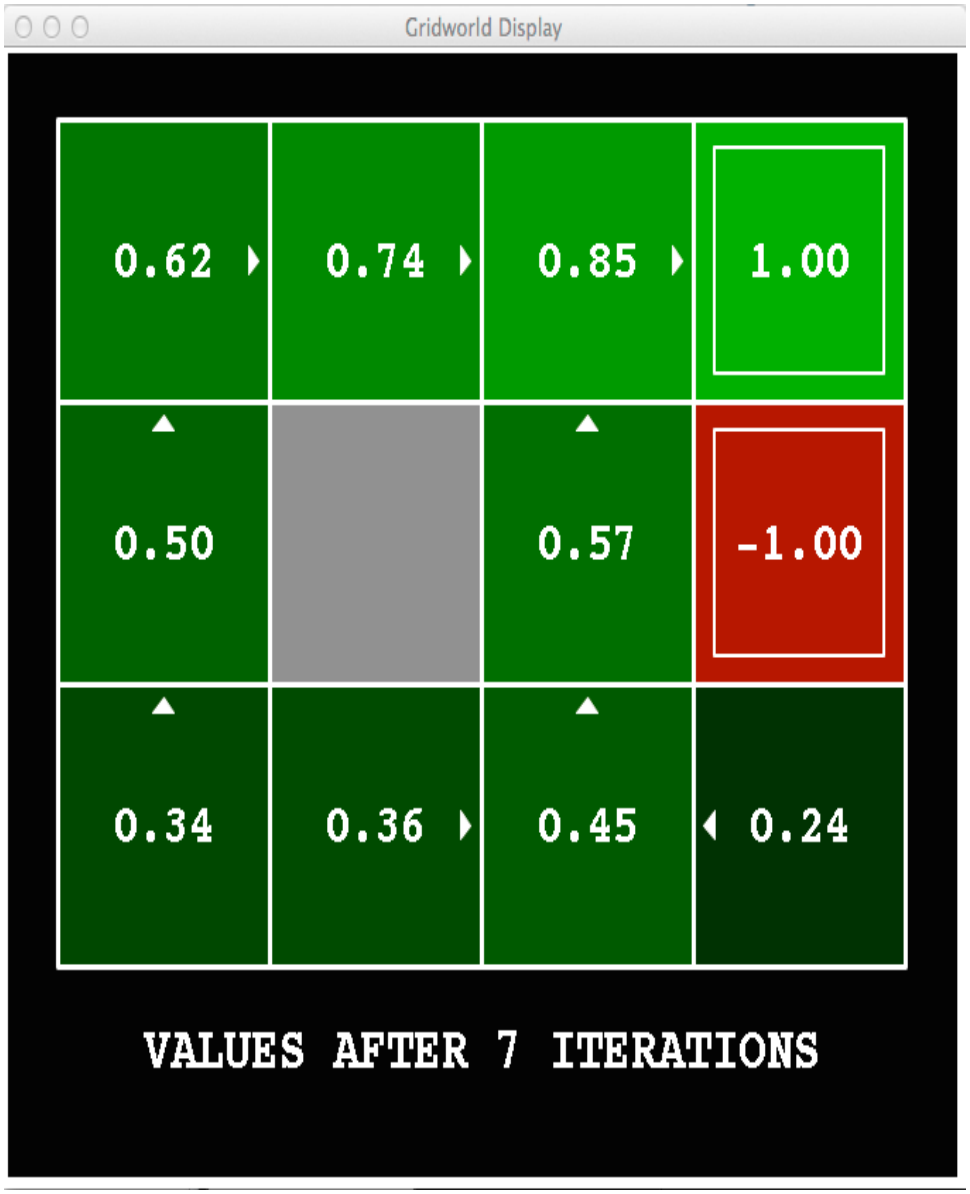
k=5

Gridworld Display
0.51
0.72
0.84
1.00
0.27
0.55
-1.00
0.00
0.22
0.37
0.13
VALUES AFTER 5 ITERATIONS

k=6


Gridworld Display
0.59
0.73
0.85
1.00
0.41
0.57
-1.00
0.21
0.31
0.43
0.19
VALUES AFTER 6 ITERATIONS

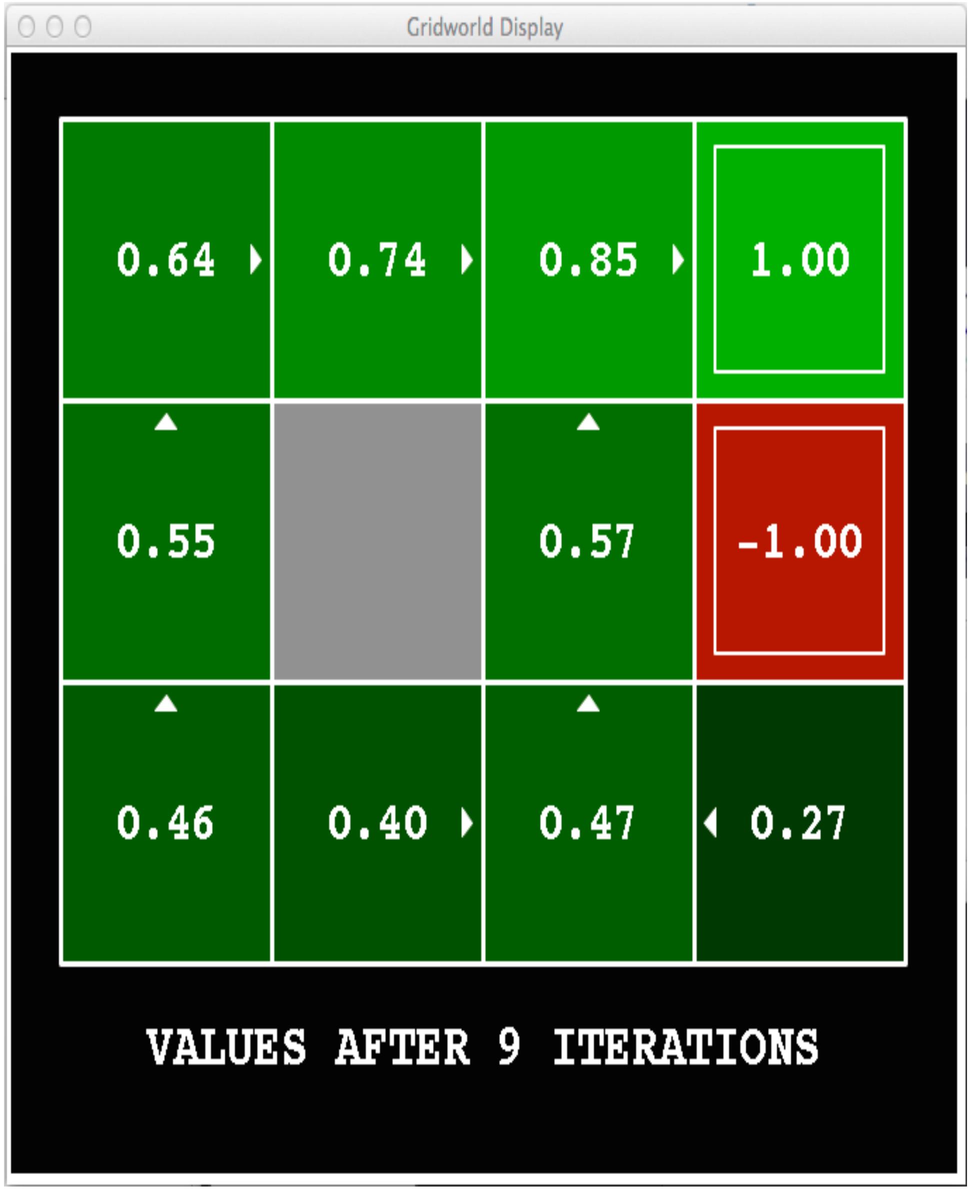
k=7


Gridworld Display
0.62
0.74
0.85
1.00
0.50
0.57
-1.00
0.34
0.36
0.45
0.24
VALUES AFTER 7 ITERATIONS

k=8


Gridworld Display
0.63
0.74
0.85
1.00
0.53
0.57
-1.00
0.42
0.39
0.46
0.26
VALUES AFTER 8 ITERATIONS

k=9


Gridworld Display
0.64
0.74
0.85
1.00
0.55
0.57
-1.00
0.46
0.40
0.47
0.27
VALUES AFTER 9 ITERATIONS

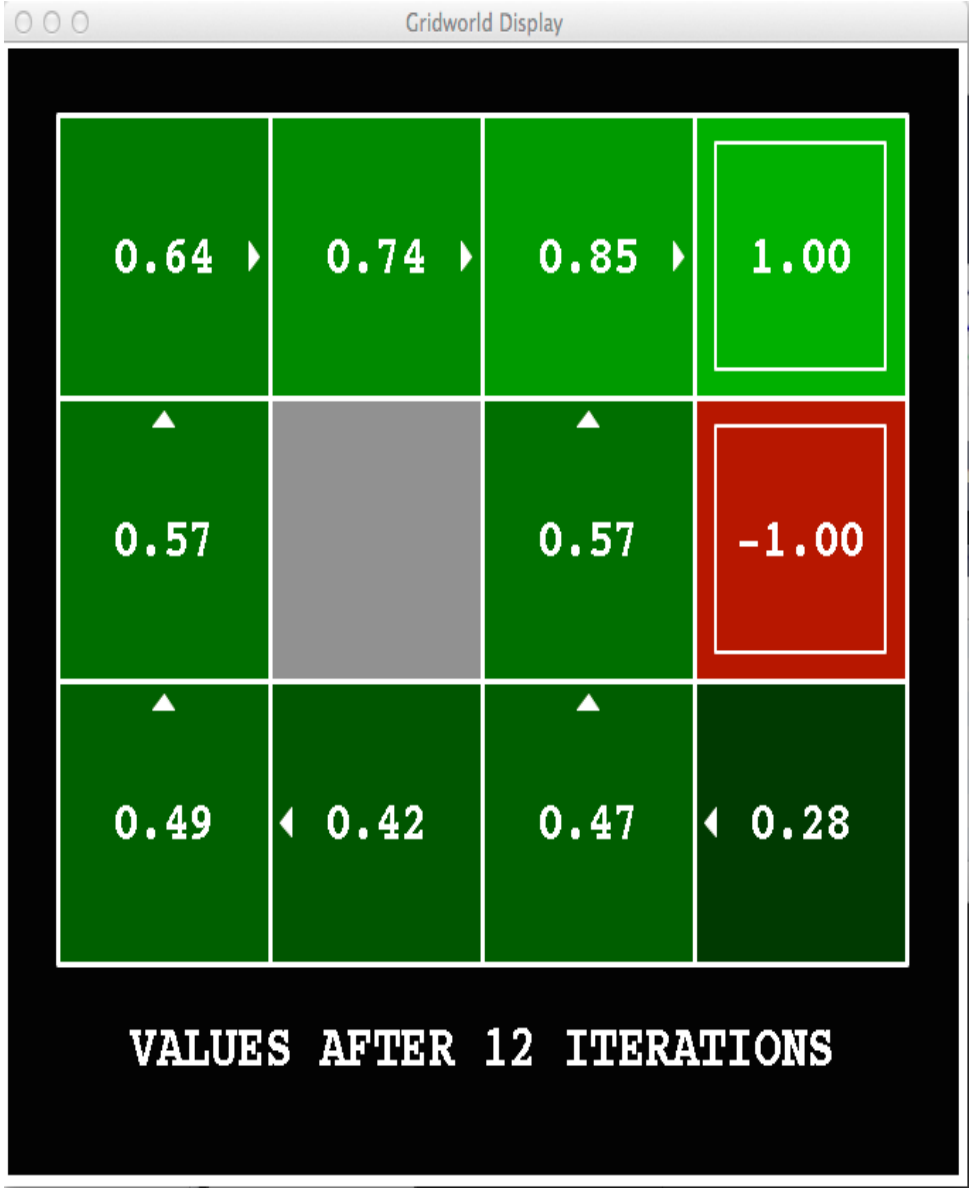
k=10

Gridworld Display
0.64
0.74
0.85
1.00
0.56
0.57
-1.00
0.48
0.41
0.47
0.27
VALUES AFTER 10 ITERATIONS

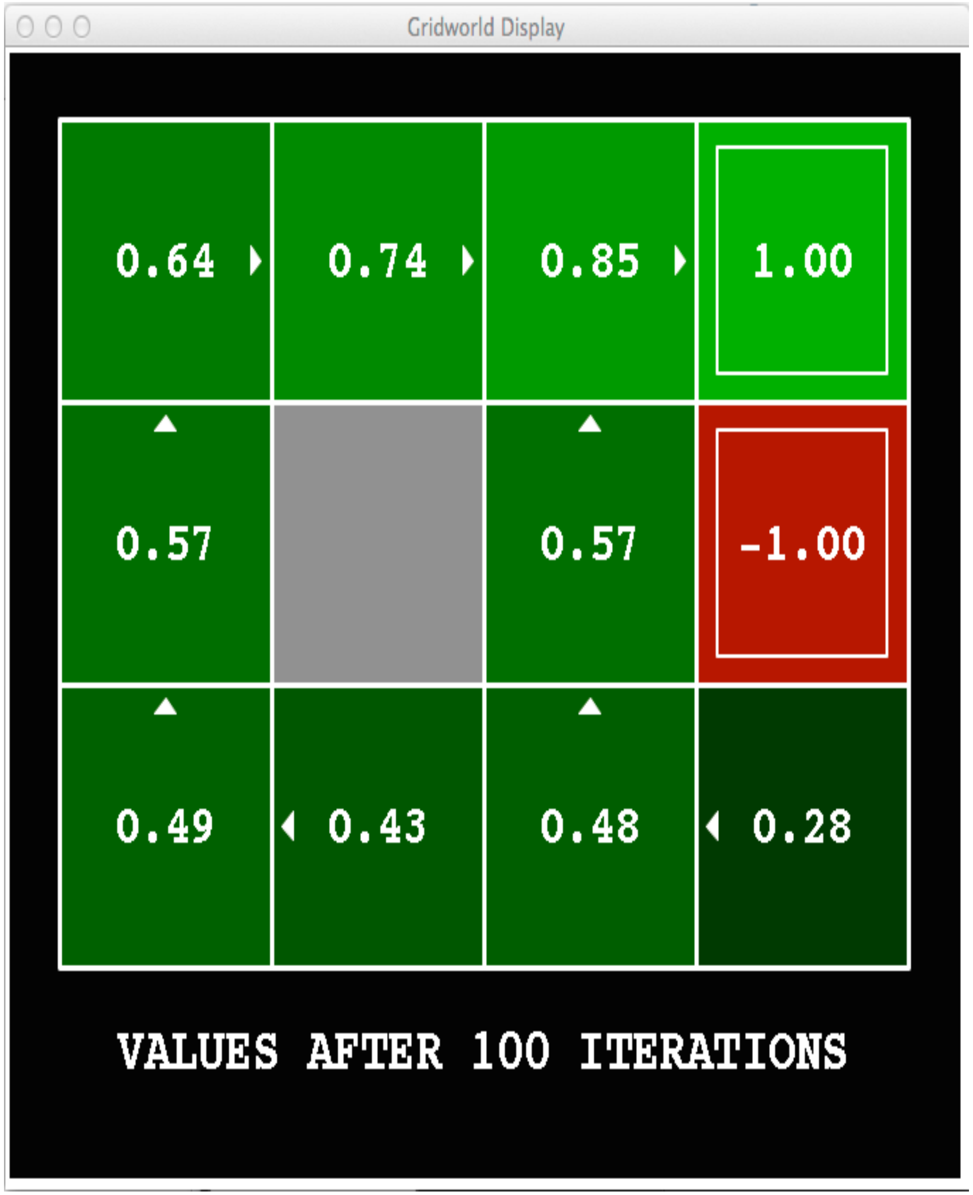
k=11

Gridworld Display
0.64
0.74
0.85
1.00
0.56
0.57
-1.00
0.48
0.42
0.47
0.27
VALUES AFTER 11 ITERATIONS

k=12

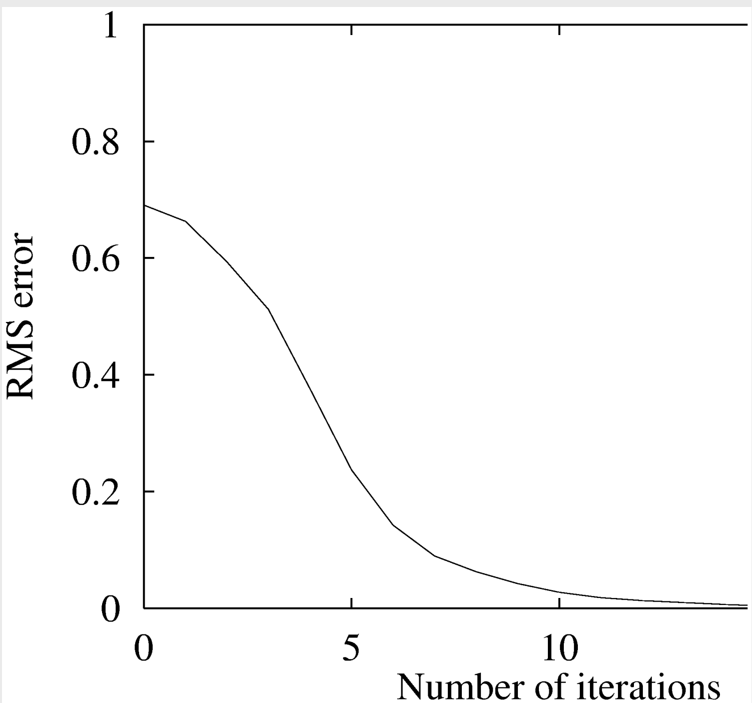

Gridworld Display
0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.42
0.47
0.28
VALUES AFTER 12 ITERATIONS

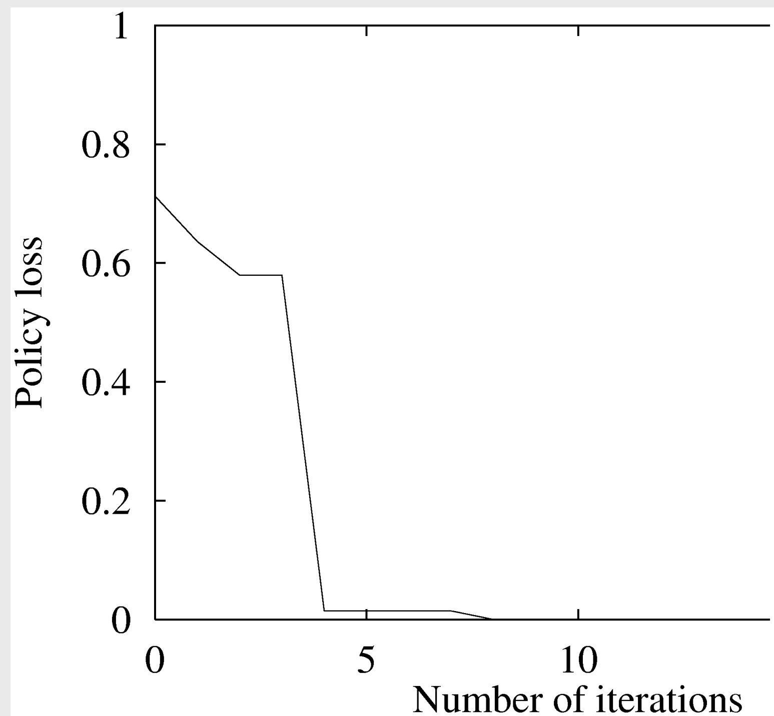
k=100


Gridworld Display
0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

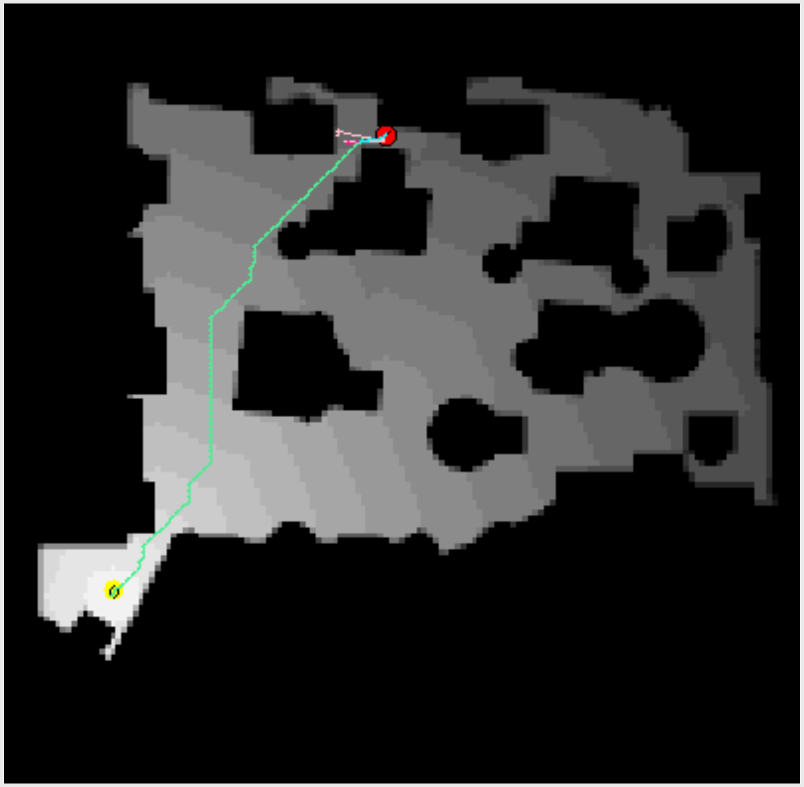
Value Function and Policy

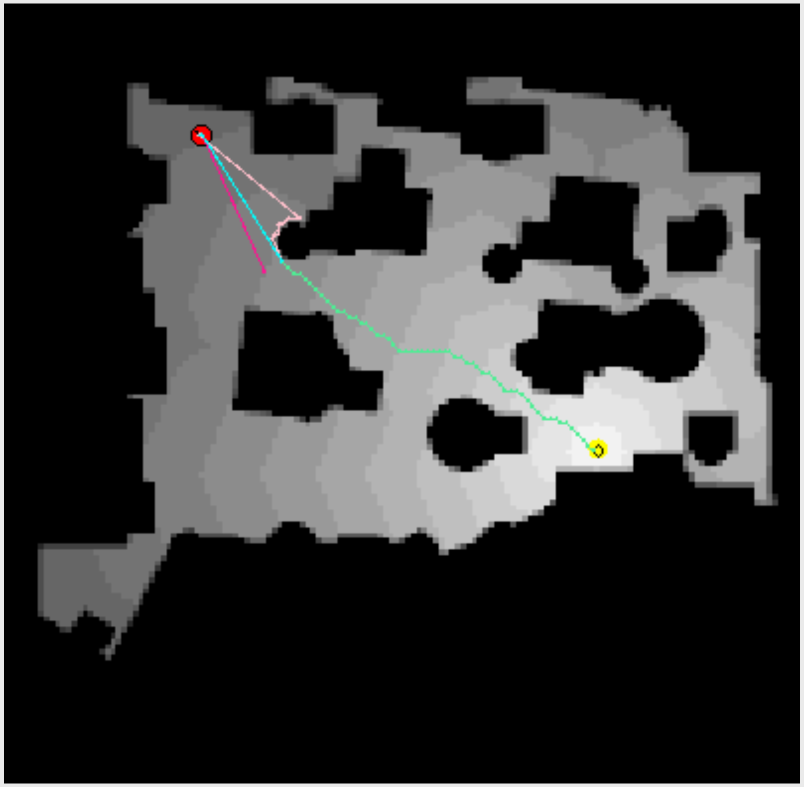
- Each step takes O(|A| |S| |S|) time.
- Number of iterations required is polynomial in |S|, |A|, 1/(1-gamma)





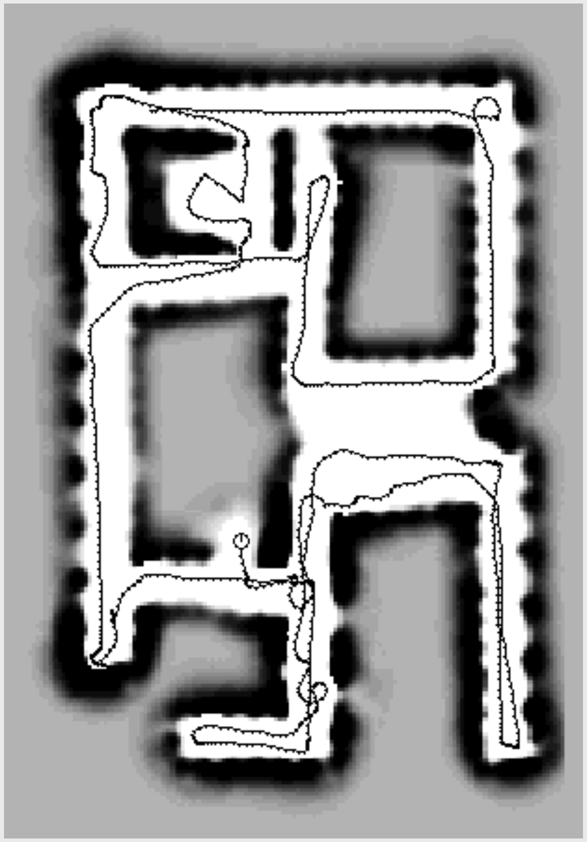
Value Iteration for Motion Planning
(assumes knowledge of robot’s location)

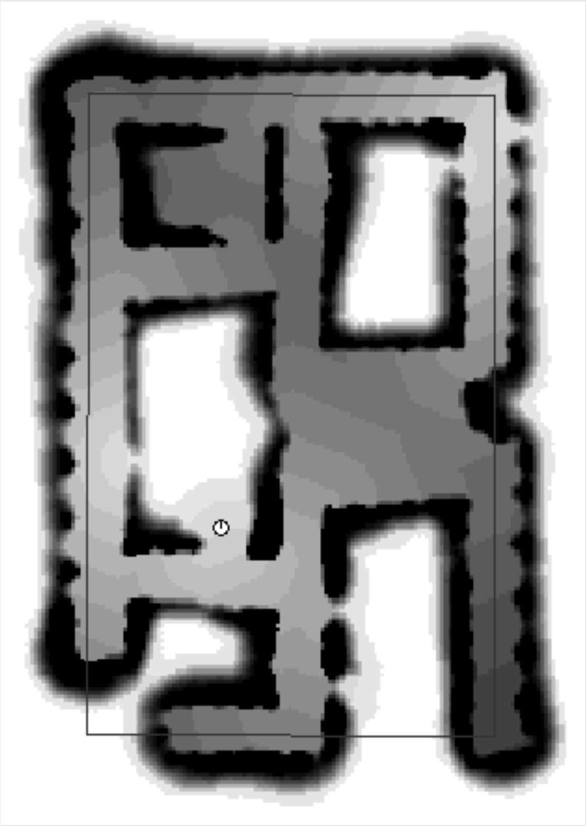




Frontier-based Exploration

- Every unknown location is a target point.





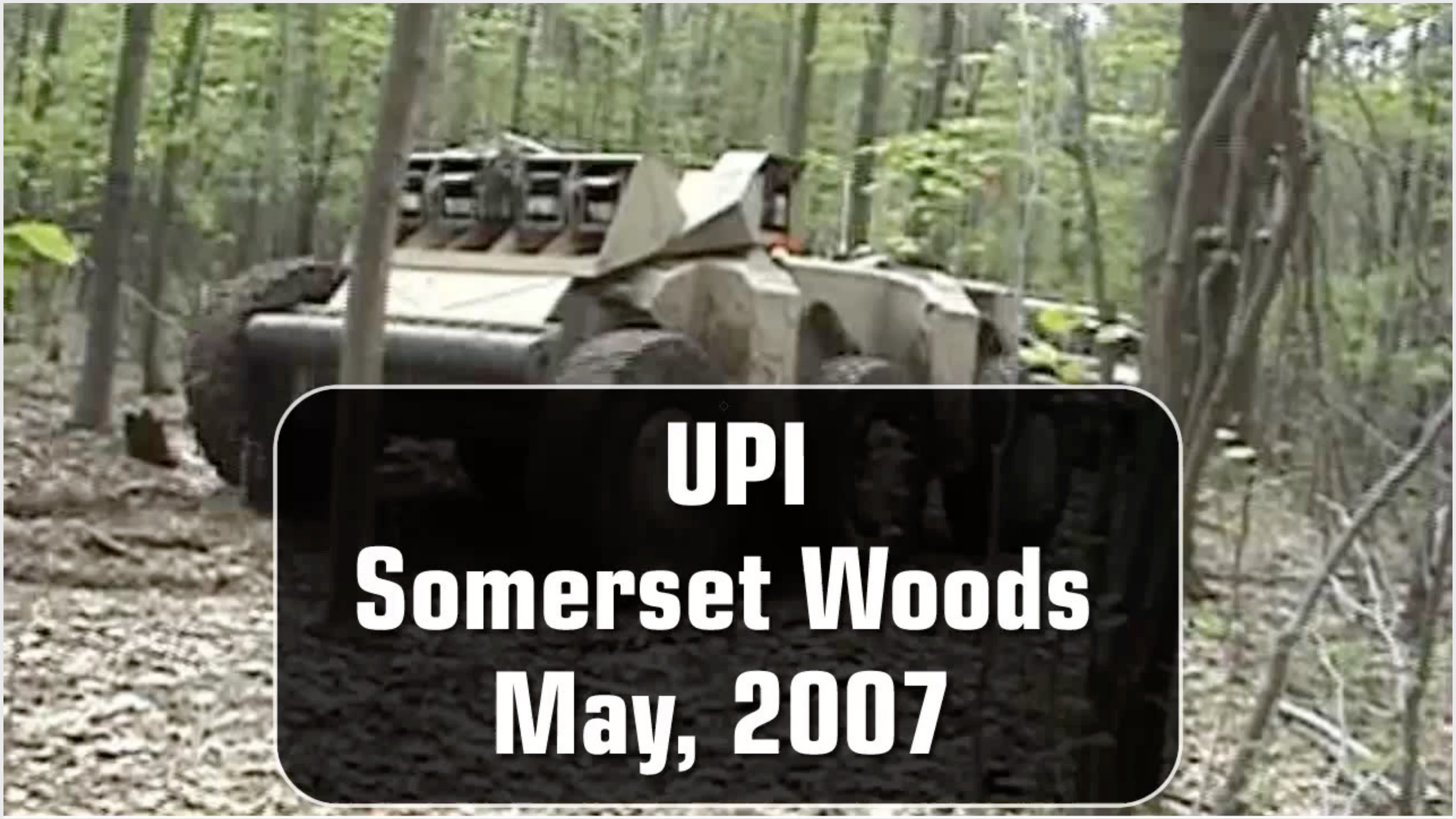
POMDPs

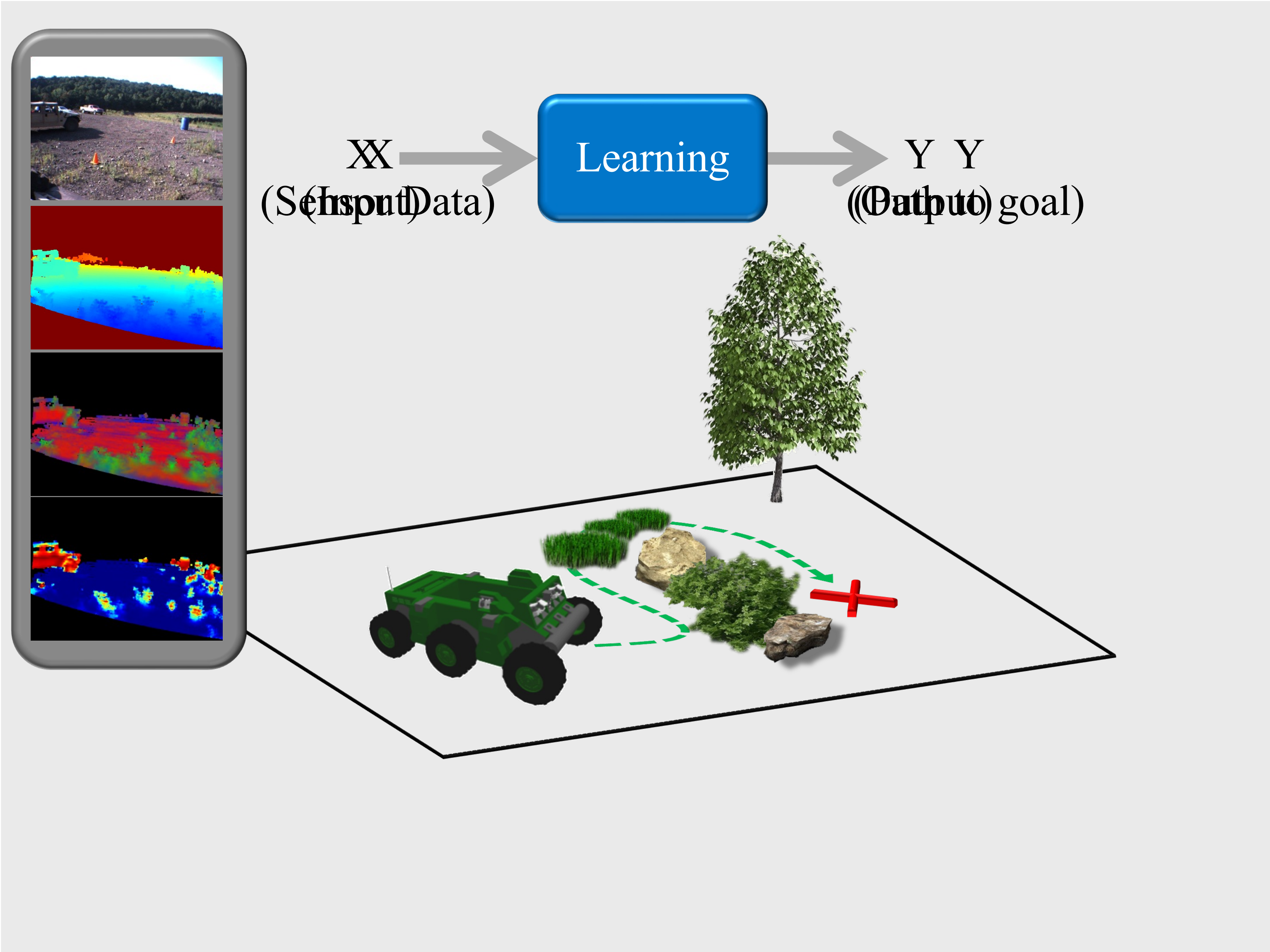
- In POMDPs we apply the very same idea as in MDPs.
- Since the **state is not observable**, the agent has to **make its decisions based on the belief state** which is a posterior distribution over states.
- For finite horizon problems, the resulting value functions are piecewise linear and convex.
- In each iteration the **number of linear constraints grows exponentially**.
- Full fledged POMDPs have only been applied to very small state spaces with small numbers of possible observations and actions.
- **Approximate solutions are becoming more and more capable**.

CSE 571
Inverse Optimal Control
(Inverse Reinforcement Learning)

Many slides by Drew Bagnell
Carnegie Mellon University

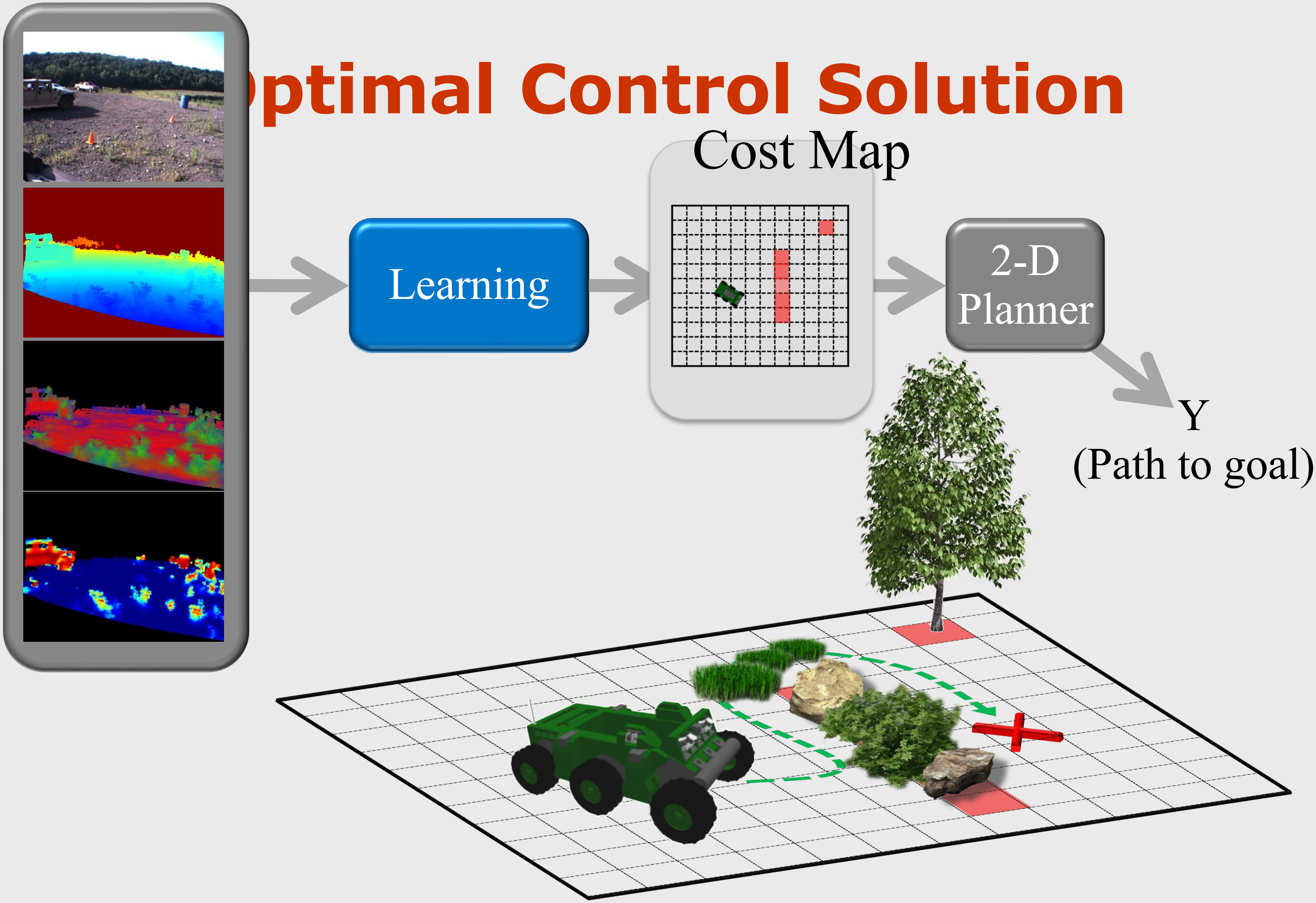
Autonomous Navigation





X
(Sensor Data)
(Input)
Learning
Y
(Path to goal)
(Output)

Optimal Control Solution



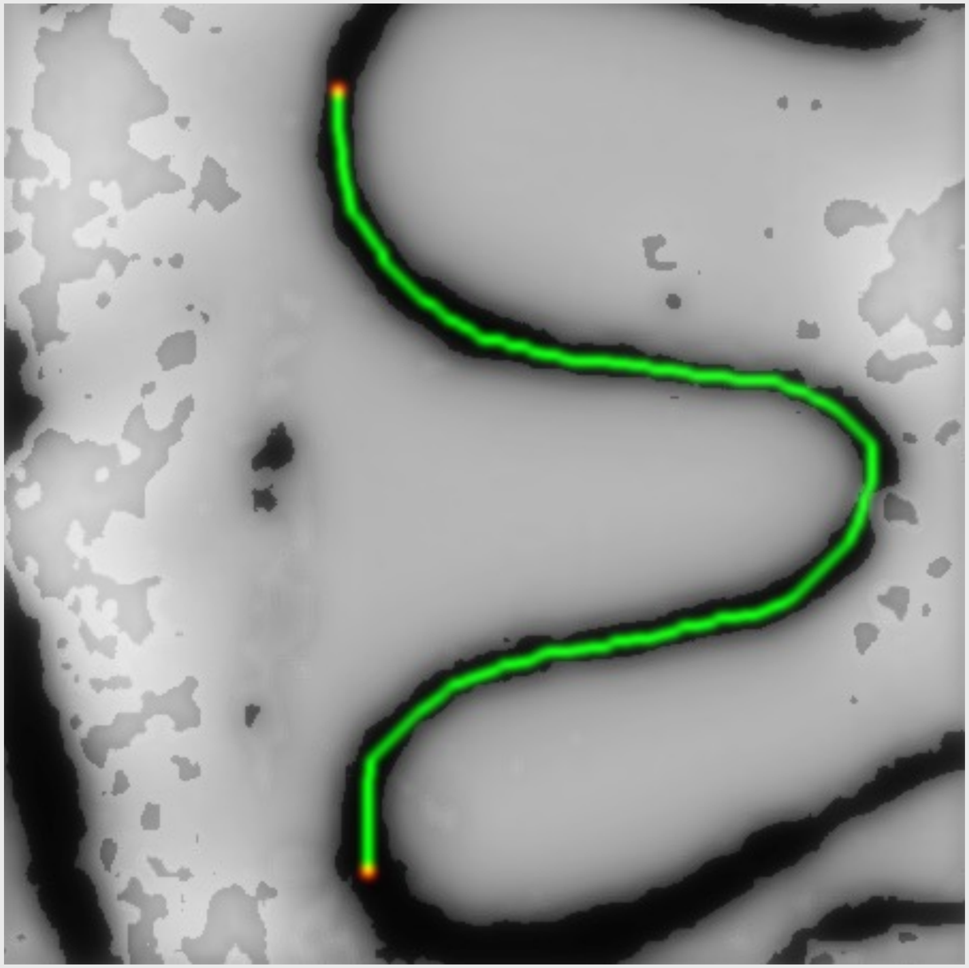
Mode 1: Training example

Mode 1: Training example

Mode 1: Learned behavior

Mode 1: Learned behavior

Mode 1: Learned cost map



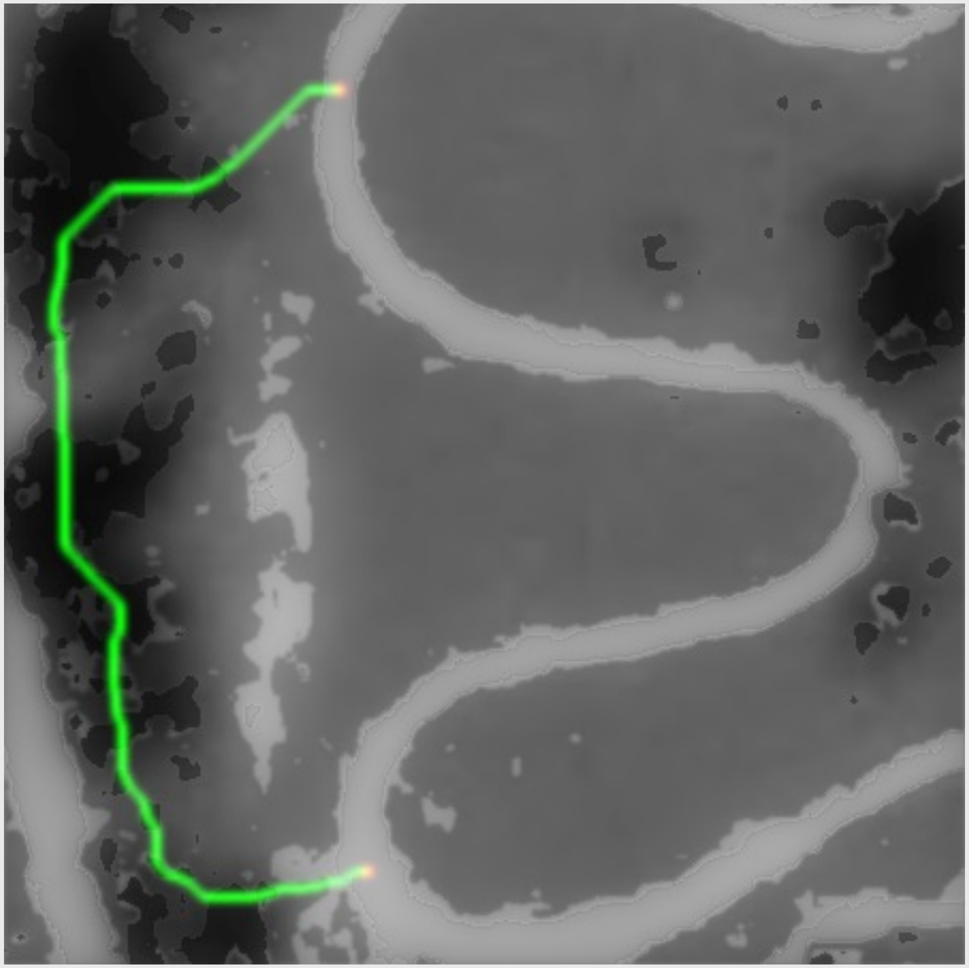
Mode 2: Training example

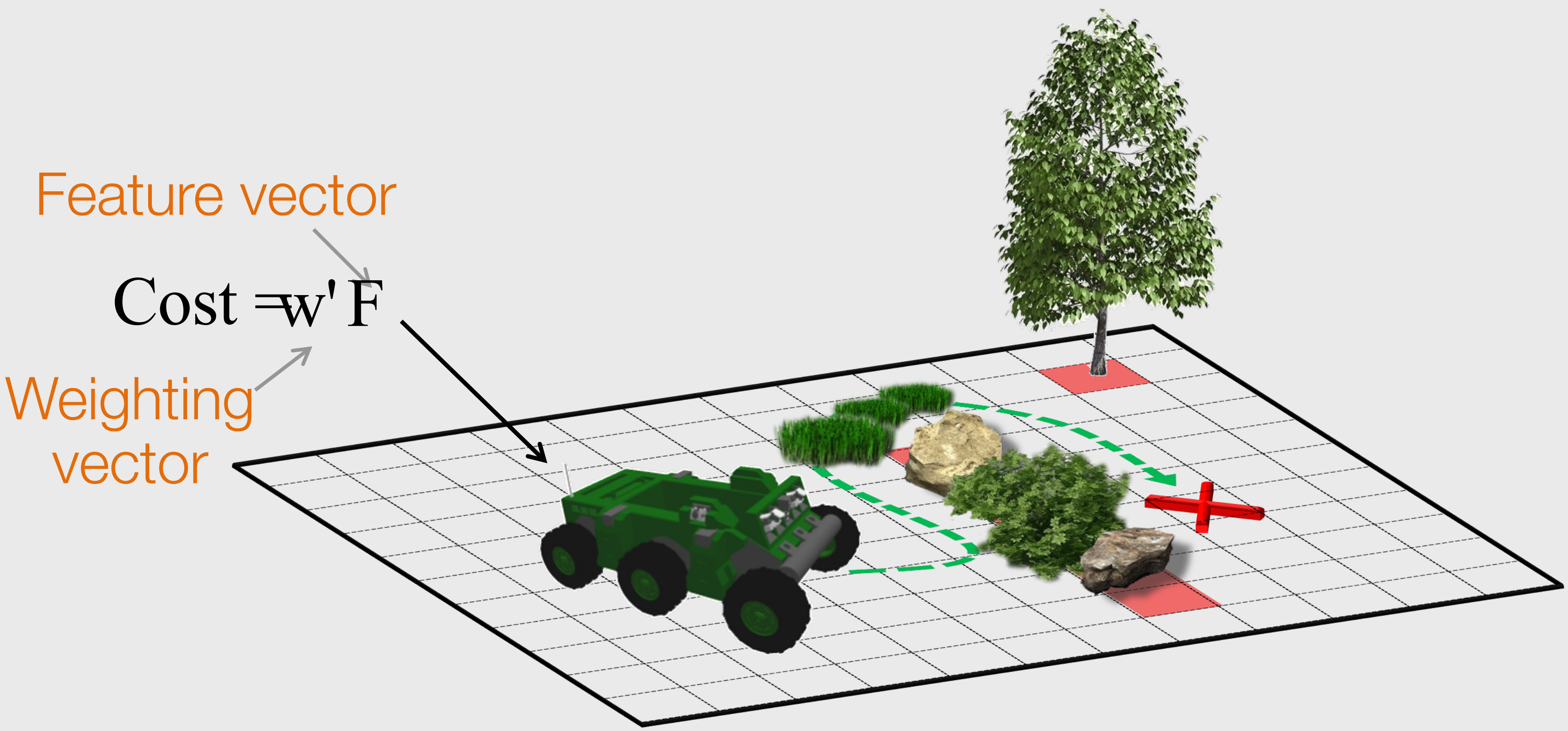
Mode 2: Training example

Mode 2: Learned behavior

Mode 2: Learned behavior

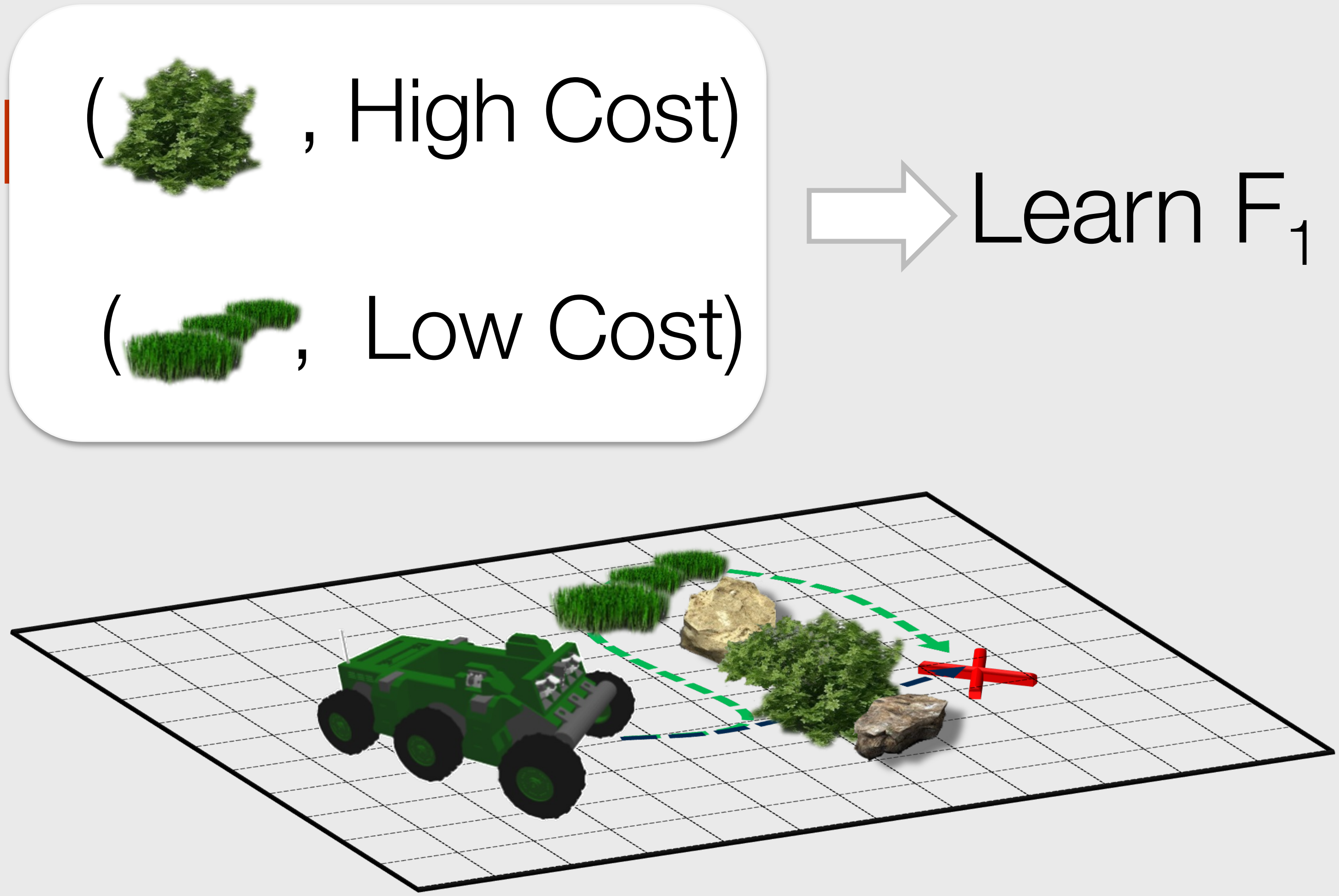
Mode 2: Learned cost map





Ratliff, Bagnell, Zinkevich 2005
Ratliff, Bradley, Bagnell, Chestnutt, NIPS 2006
Silver, Bagnell, Stentz, RSS 2008

w=



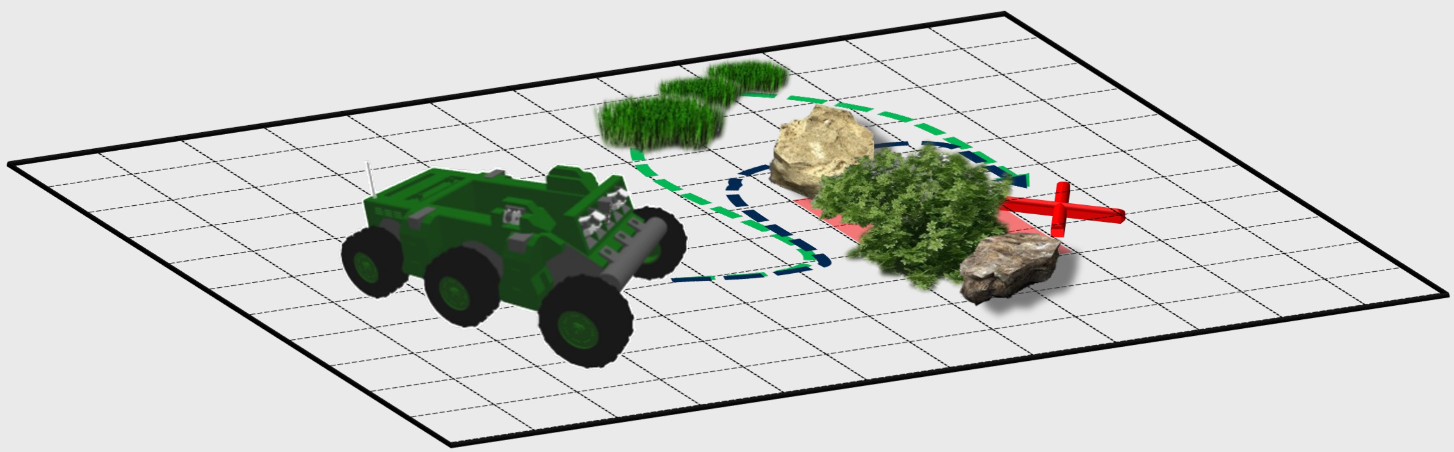
Ratlif, Bagnell, Zinkevich, ICML 2006
Ratliff, Bradley, Bagnell, Chestnutt, NIPS 2006
Silver, Bagnell, Stentz, RSS 2008

w=[

( , High Cost)

( , Low Cost)

Learn F_2

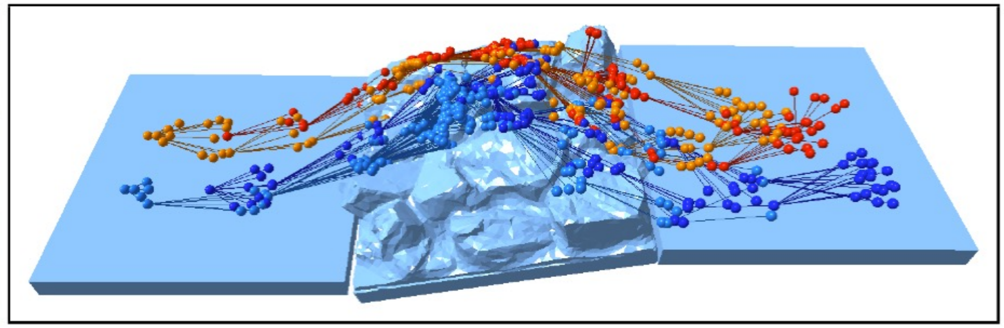


Ratliff, Bagnell, Zinkevich, ICML 2006

Ratliff, Bradley, Bagnell, Chestnutt, NIPS 2006

Silver, Bagnell, Stentz, RSS 2008

example path

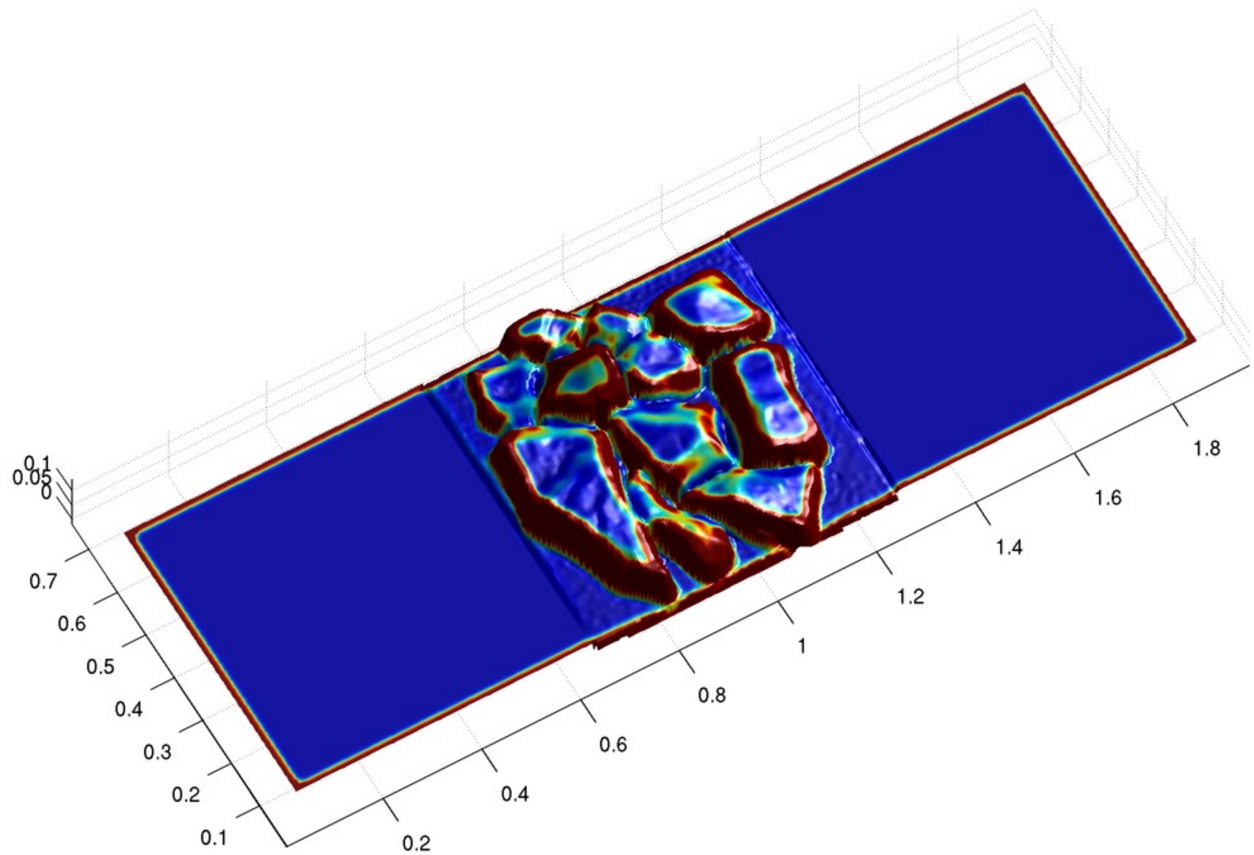


liff, Bradley, Che

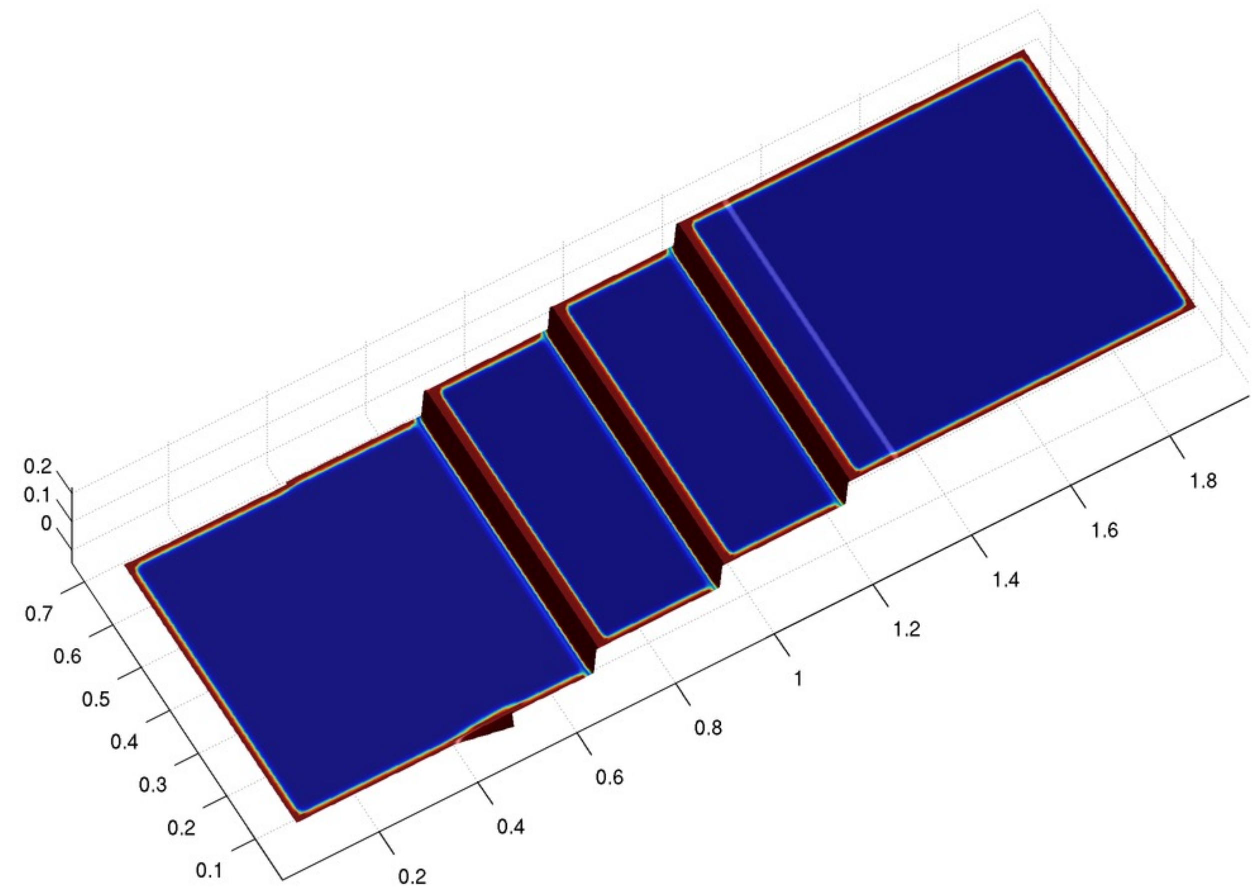
Bagnell 06

Zucker,
Ratliff, Stolle,
Chesnutt,
Bagnell,
Atkeson,

Learned Cost Function Examples



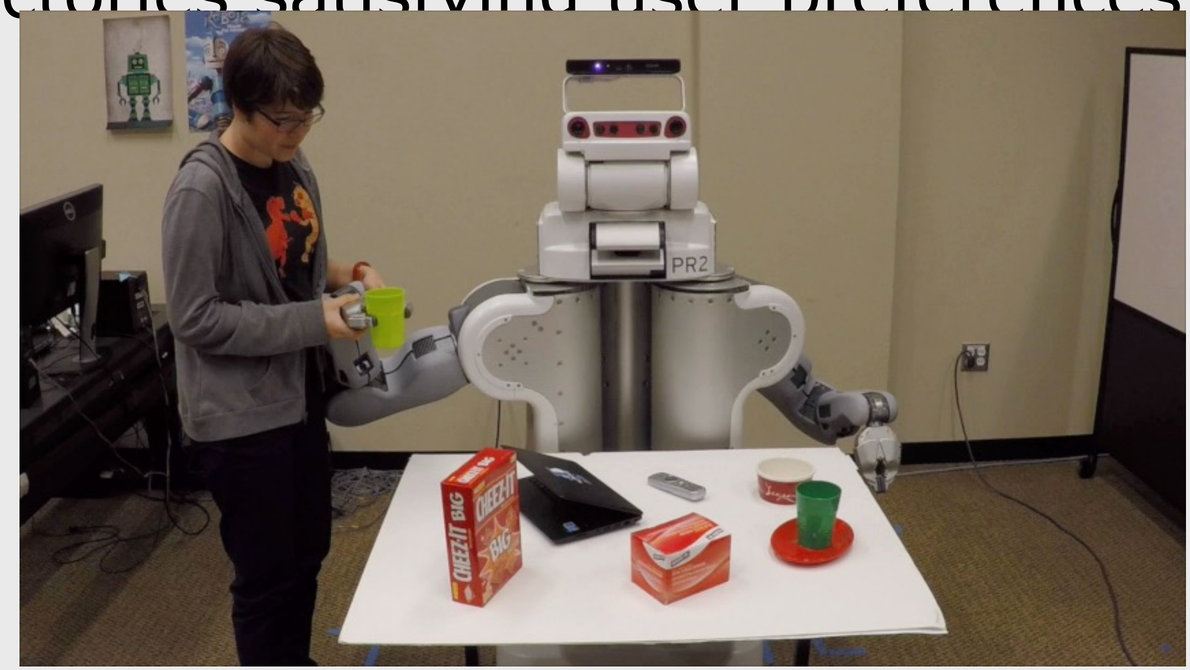
Learned Cost Function Examples

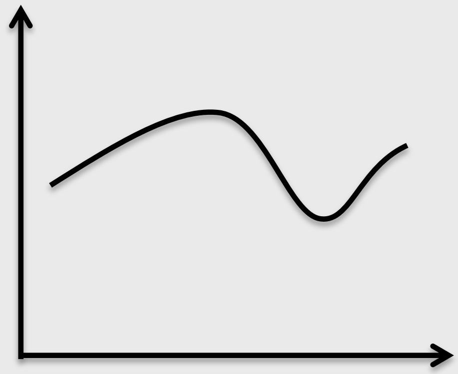


Learning Manipulation Preferences

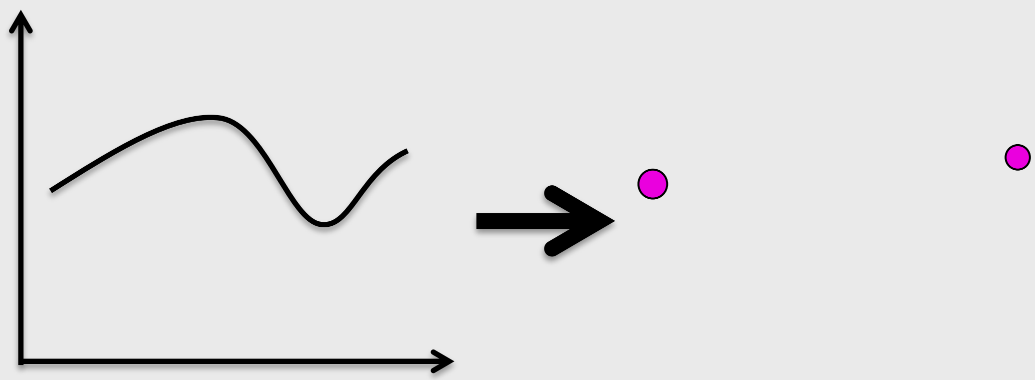
- ***Input:*** Human demonstrations of preferred behavior (e.g., moving a cup of water upright without spilling)

- ***Output:*** Learned cost function that results in trajectories satisfying user preferences

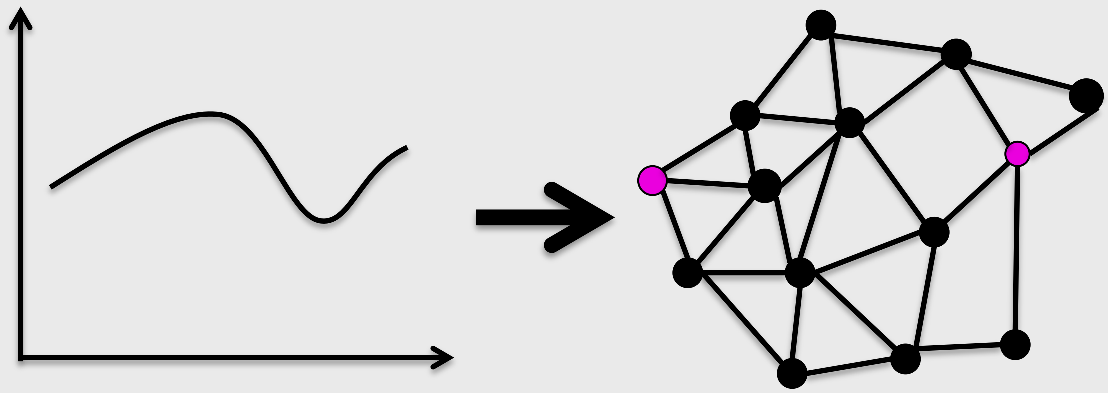




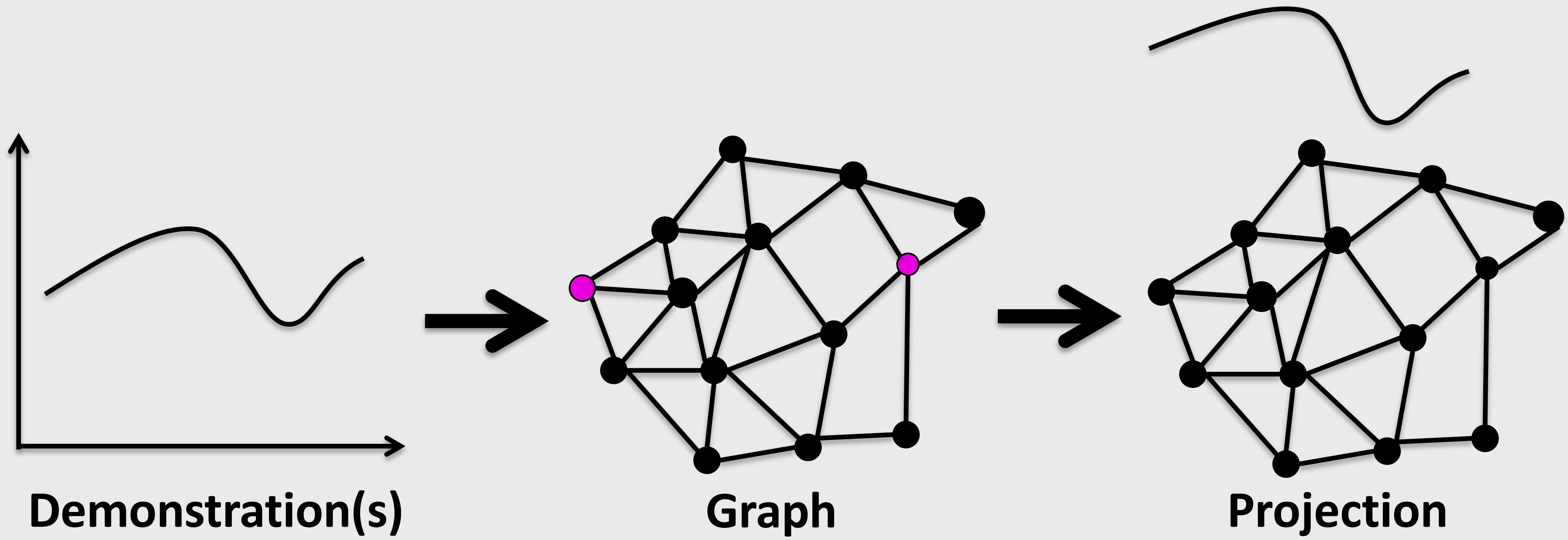
Demonstration(s)

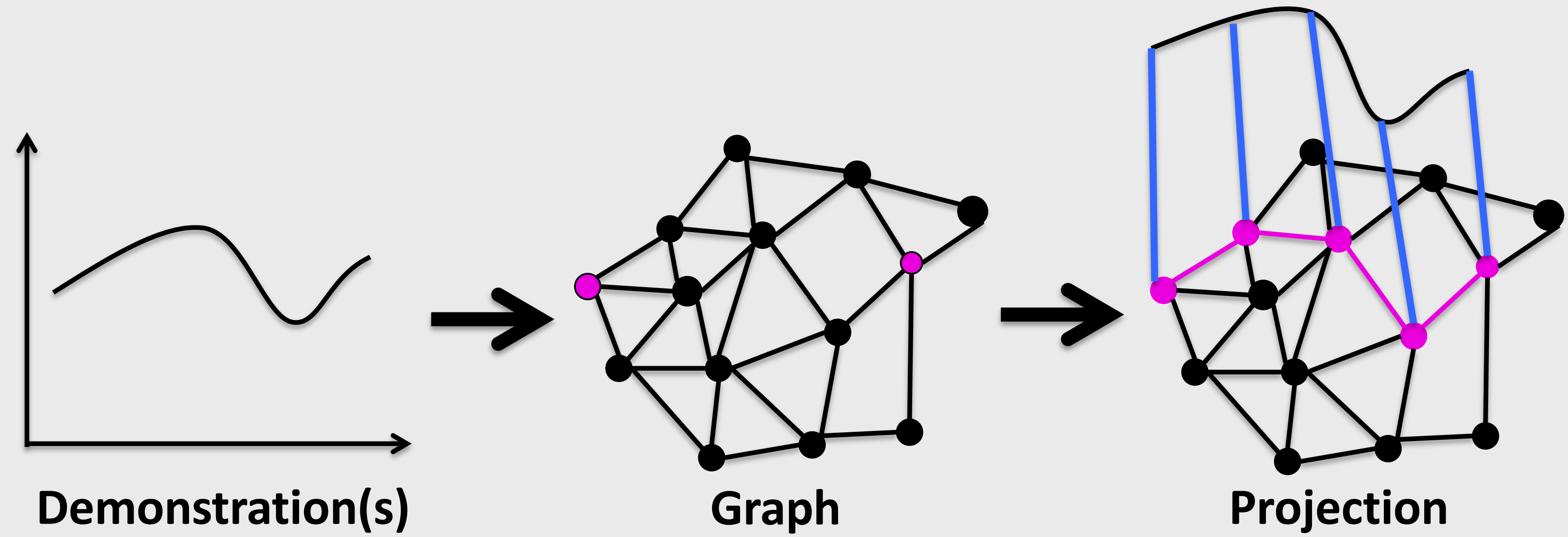


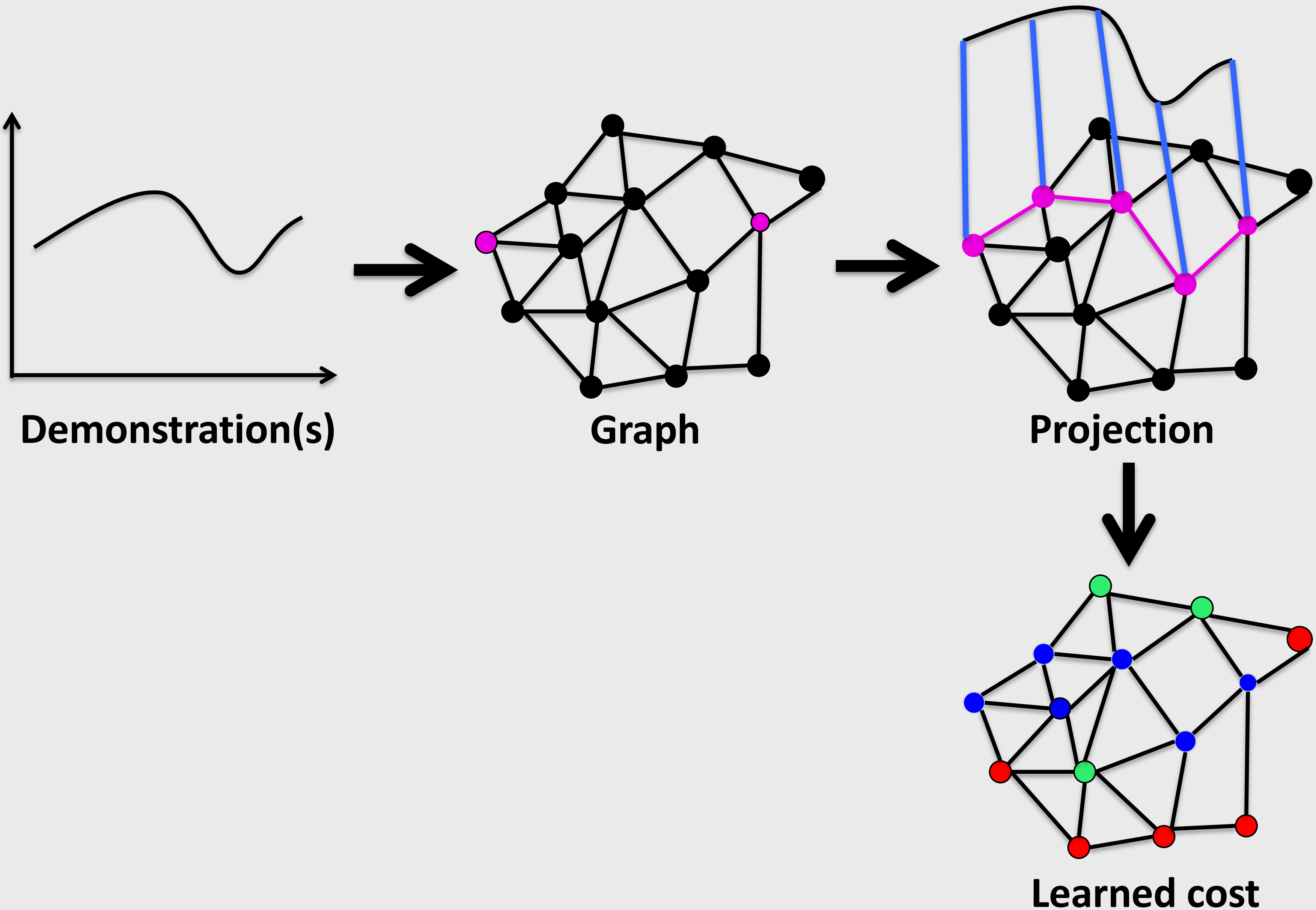
Demonstration(s) **Graph**

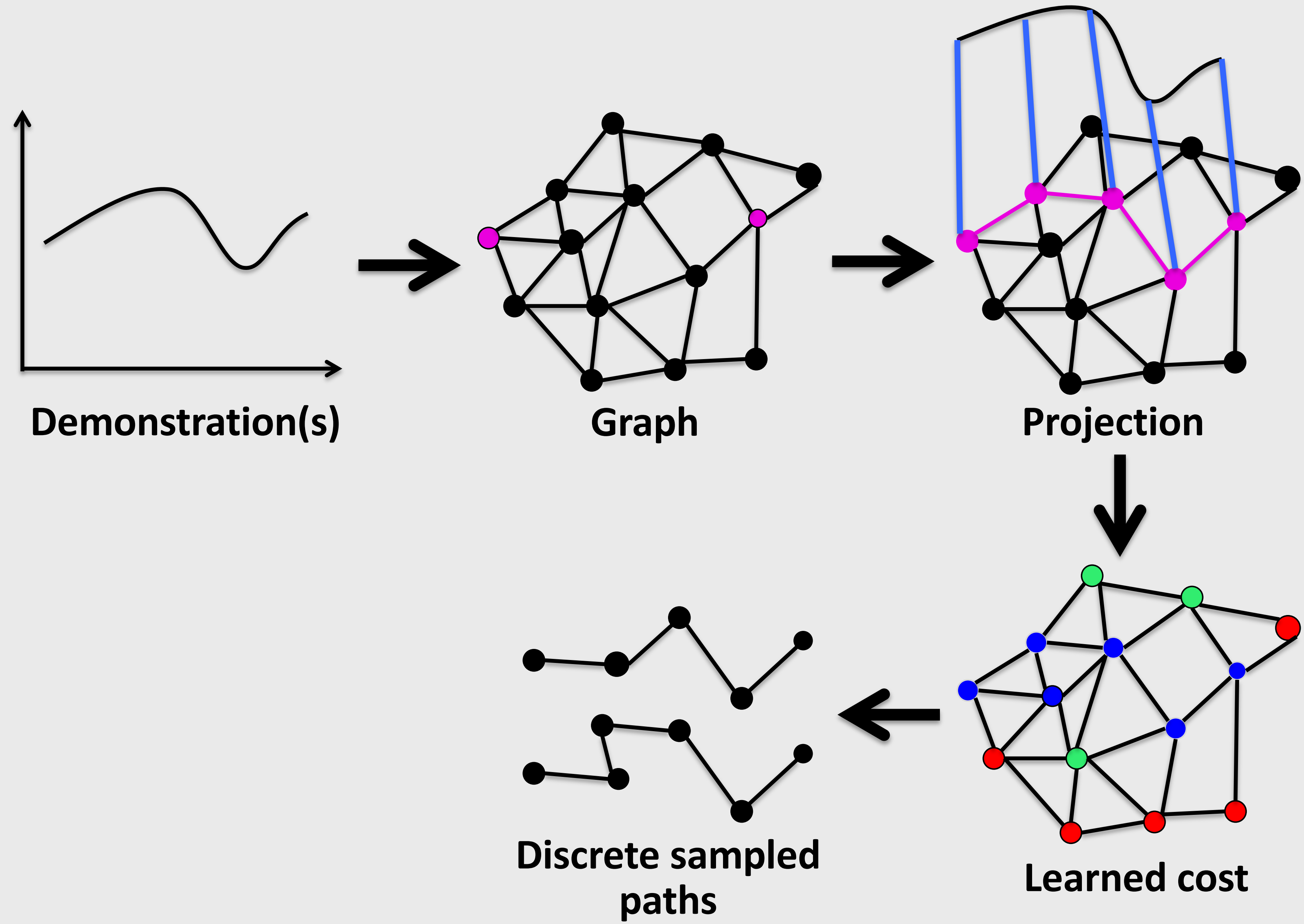


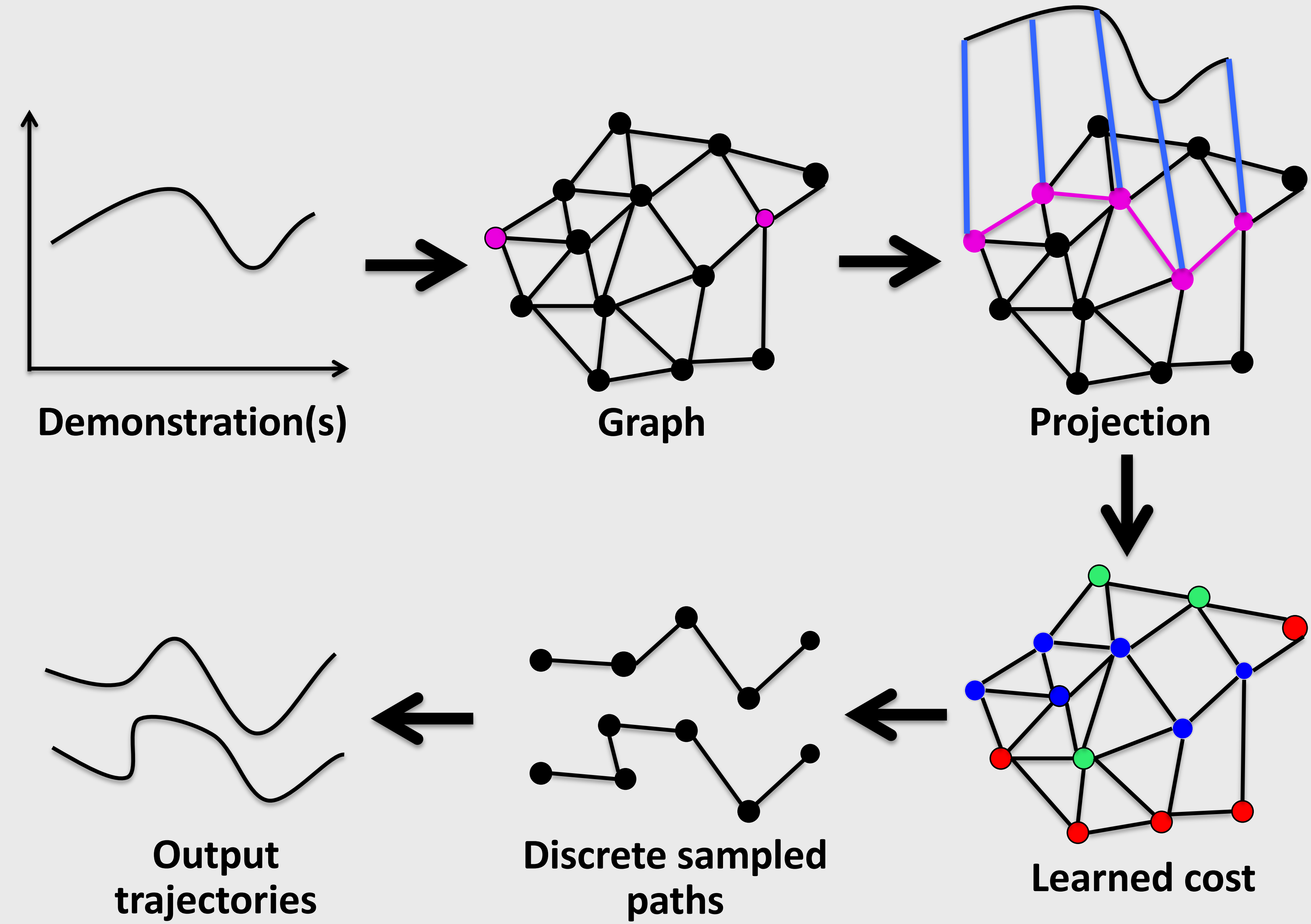
Demonstration(s) **Graph**


Demonstration(s)
Graph
Projection

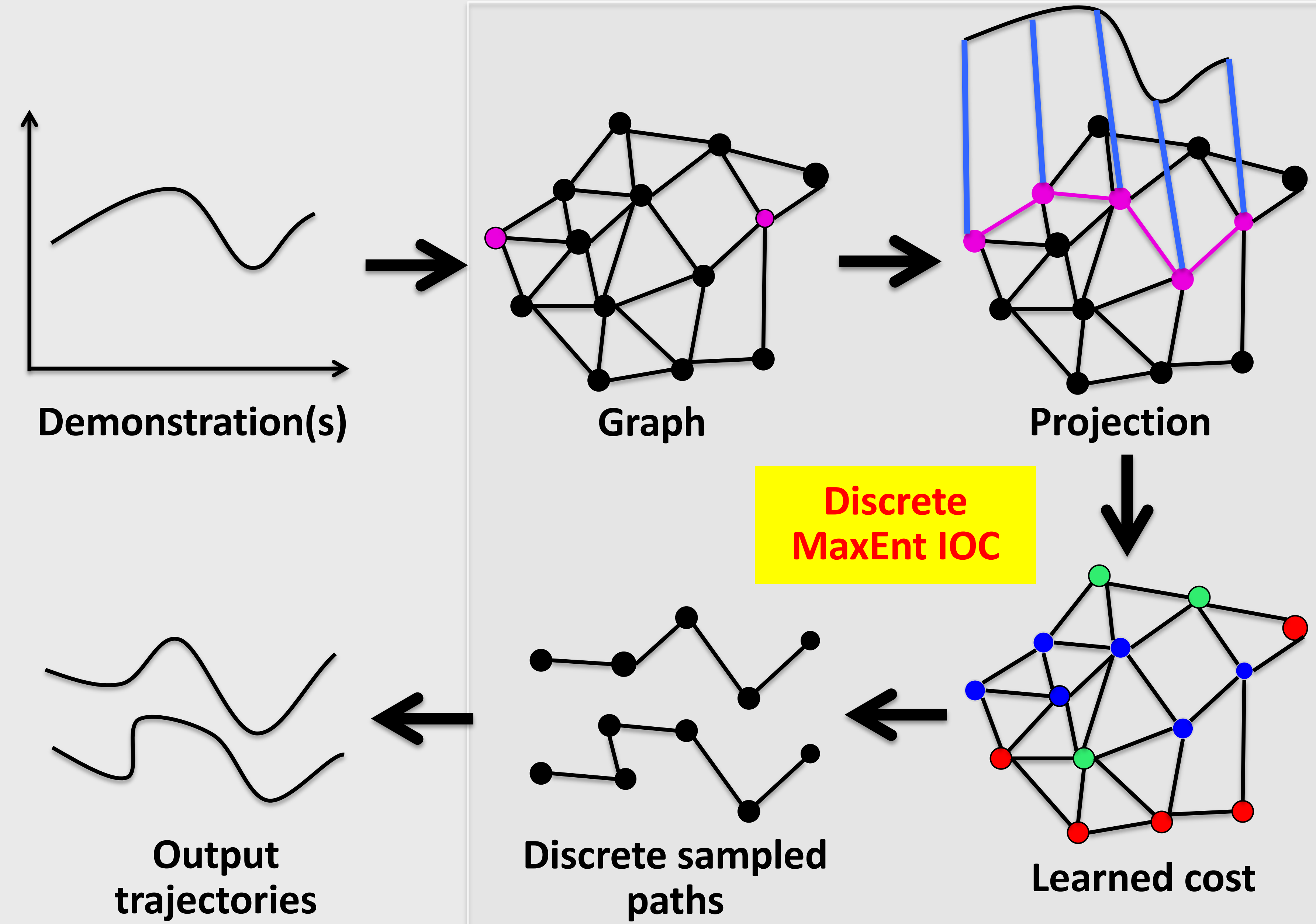

Demonstration(s)
Graph
Projection

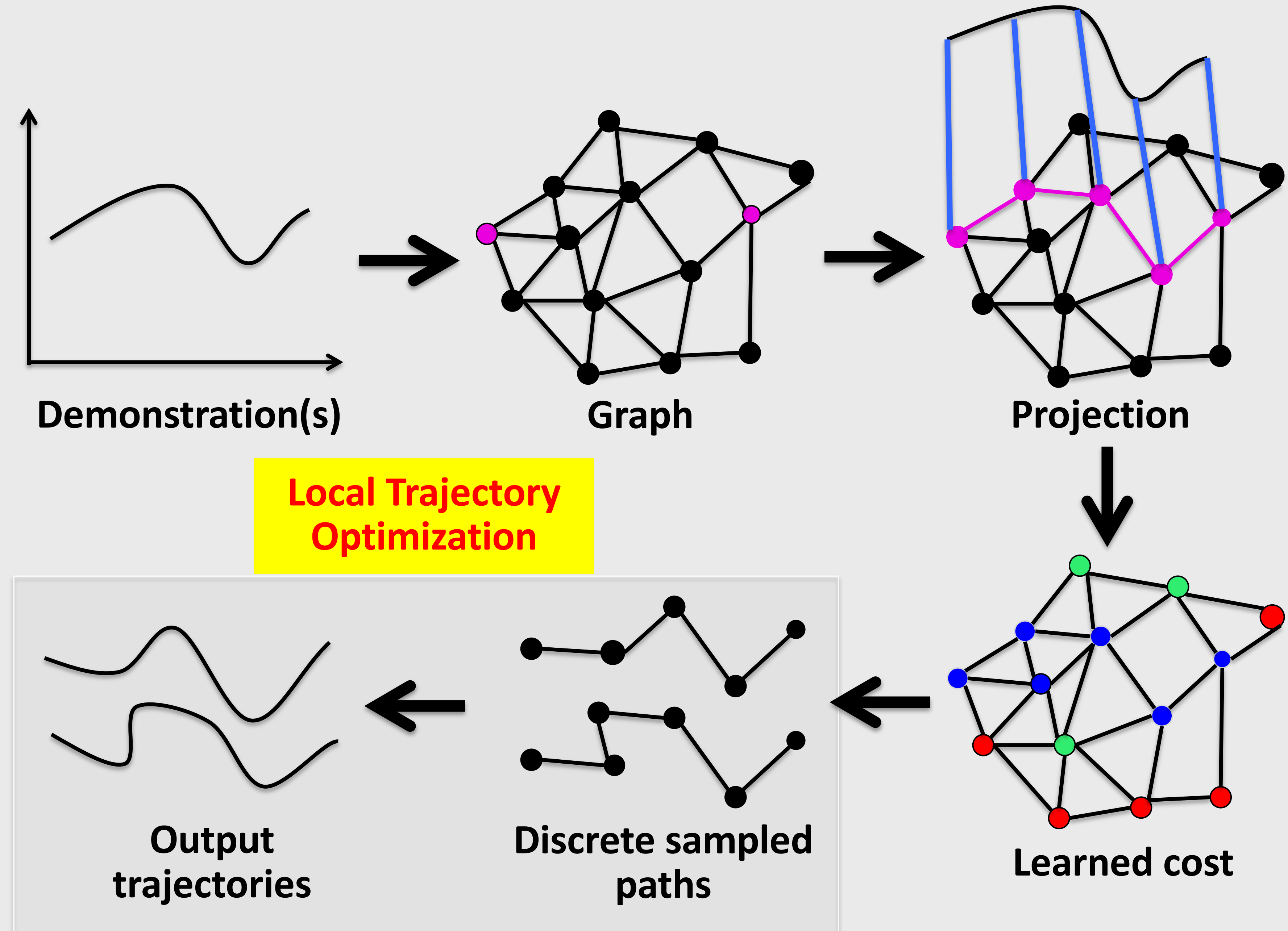

Demonstration(s)
Graph
Projection
Learned cost


Demonstration(s)
Graph
Projection
Learned cost
Discrete sampled paths



Demonstration(s)
Graph
Projection
Learned cost
Discrete sampled paths
Output trajectories


Demonstration(s)
Graph
Projection
Discrete
MaxEnt IOC
Learned cost
Discrete sampled
paths
Output
trajectories



Demonstration(s)
Graph
Projection
Learned cost
Discrete sampled paths
Output trajectories
Local Trajectory Optimization

Setup

- ***Binary*** state-dependent features (~95)
  - Histograms of distances to objects
  - Histograms of end-effector orientation
  - Object specific features (electronic vs non-electronic)
  - Approach direction w.r.t goal
- **Task**
  - Hold cup upright while not moving above electronics

Laptop task: Demonstration
(Not part of training set)

Laptop task: LTO + Smooth random path

Readings

- Max-Ent IRL (Ziebart, Bagnell): http://www.cs.cmu.edu/~bziebart/
- CIOC (Levine) http://graphics.stanford.edu/projects/cioc/cioc.pdf
- Manipulation (Byravan/Fox): https://rse-lab.cs.washington.edu/papers/graph-based-IOC-ijcai-2015.pdf
- Imitation learning (Ermon): https://cs.stanford.edu/~ermon/
- Human/manipulation (Dragan): https://people.eecs.berkeley.edu/~anca/research.html