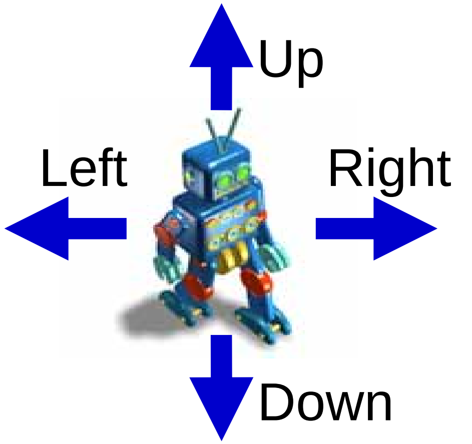
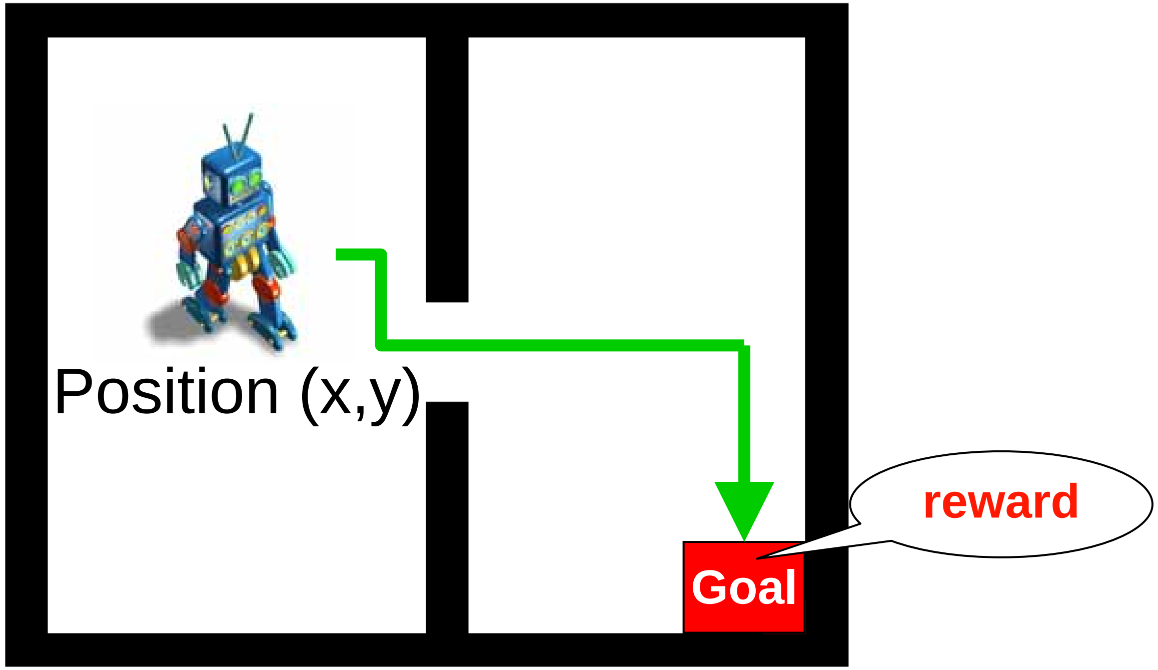
Value Function Approximation on Non-linear Manifolds for Robot Motor Control

Masashi Sugiyama[1)2)] **Hirotaka Hachiya**[1)2)]
Christopher Towell[2)] Sethu Vijayakumar[2)]

1) Computer Science, Tokyo Institute of Technology
2) School of Informatics, University of Edinburgh

Maze Problem: Guide Robot to Goal





- Robot knows its position but doesn't know which direction to go.
- We don't teach the best action to take at each position but give a reward at the goal.
- Task: make the robot select the optimal action.

Markov Decision Process (MDP)

- An MDP consists of $\{S, A, P, R\}$
  - S : set of states, $\{s_i\}$
  - A : set of actions, $\{\text{up}, \text{down}, \text{left}, \text{right}\}$
  - P : transition probability, $P(s,a,s')$
  - R : reward, $R(s,a)$
- An action a the robot takes at state s is specified by policy π .

$$a = \pi(s)$$

- Goal: make the robot learn optimal policy π^*

Definition of Optimal Policy

- Action-value function:

$$Q^{\pi}(s,a) = E\left(\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a \right)$$

  discounted sum of future rewards when taking a in s and following π thereafter

- Optimal value: $Q^*(s,a) = \arg\max_{\pi} Q^{\pi}(s,a)$

- Optimal policy: $\pi^*(s,a) = \arg\max_{a} Q^*(s,a)$

- π^* is computed if Q^* is given.
- Question: How to compute Q^* ?

Policy Iteration

(Sutton & Barto, 1998)

- Starting from some initial policy π iterate Steps 1 and 2 until convergence.

  Step 1. Compute $Q^{\pi}(s,a)$ for current π

  Step 2. Update π by

$$\pi(s) = \arg\max_{a} Q^{\pi}(s,a)$$

- Policy iteration always converges to π^* if $Q^{\pi}(s,a)$ in step 1 can be computed.
- Question: How to compute $Q^{\pi}(s,a)$?

$$Q^{\pi}(s,a) = E\left(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right)$$

Bellman Equation

- $Q^{\pi}(s,a)$ can be recursively expressed by

$\forall s, \forall a$

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s'} P(s,a,s') Q^{\pi}(s', \pi(s'))$$

- $Q^{\pi}(s,a)$ can be computed by solving Bellman equation
- Drawback: dimensionality of Bellman equation becomes huge in large state and action spaces

$$|S| \times |A|$$

→ high computational cost

Least-Squares Policy Iteration

(Lagoudakis and Parr, 2003)

- Linear architecture:

$$\hat{Q}^{\pi}(s,a) = \sum_{i=1}^{K} w_i \phi_i(s,a)$$

$\phi_i(s,a)$: fixed basis functions

w_i : parameters

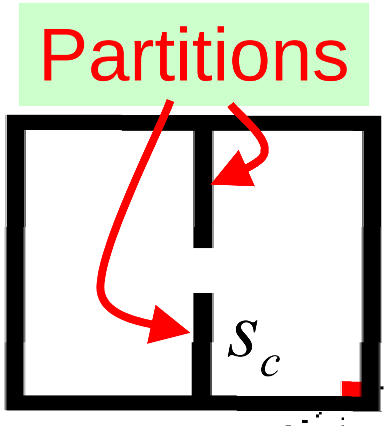
K : # of basis functions

- $\{w_i\}_{i=1}^{K}$ is learned so as to optimally approximate Bellman equation in the least-squares sense
- # of parameters is only K :

$$K << |S| \times |A|$$

- LSPI works well if we choose appropriate $\{\phi_i\}_{i=1}^{K}$
- Question: How to choose $\{\phi_i\}_{i=1}^{K}$?

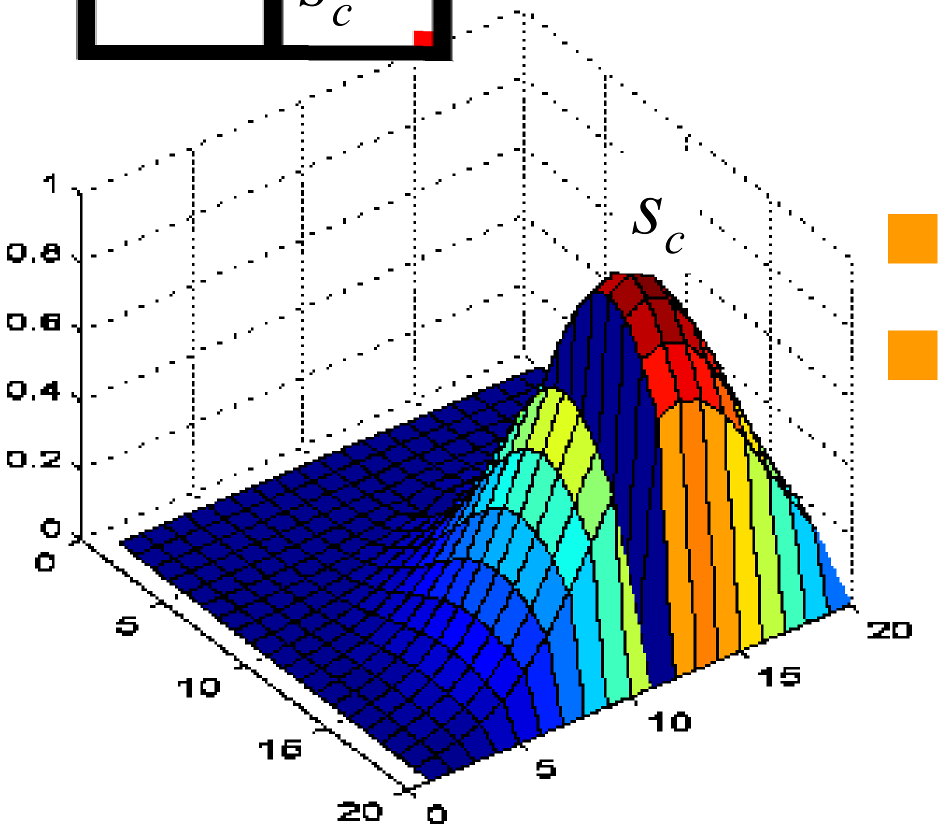
Popular Choice: Gaussian Kernel (GK)



$$k(s) = \exp\left(-\frac{ED(s_c, s)^2}{2\sigma^2} \right)$$

ED : Euclidean distance

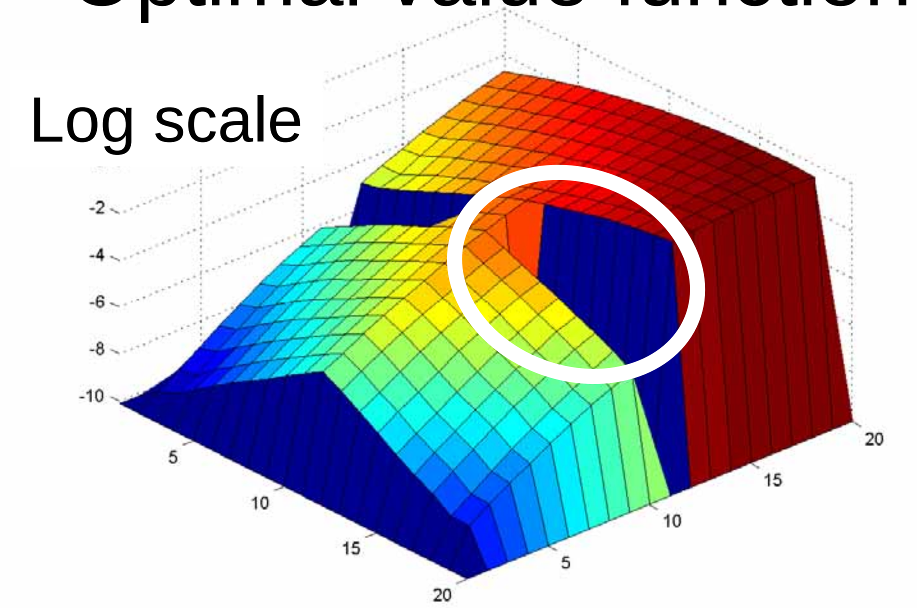
s_c : Centre state



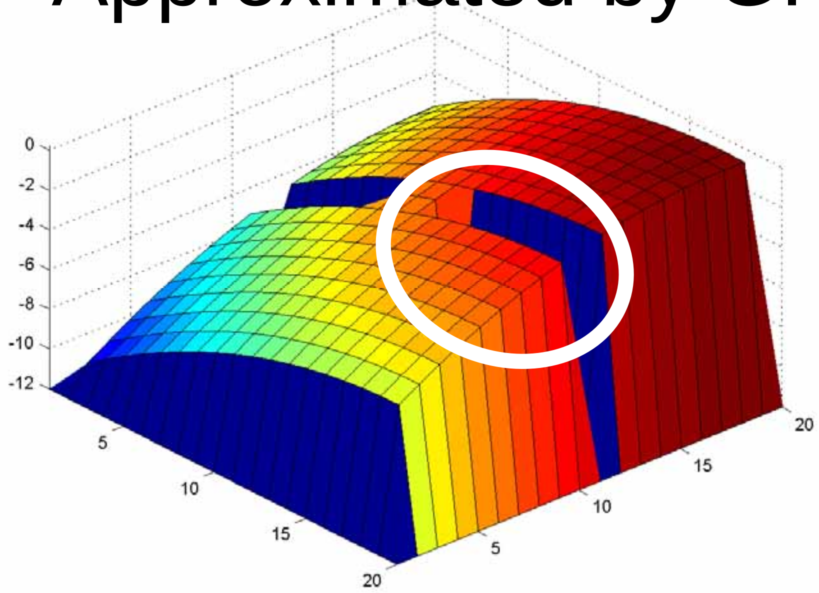
- Smooth
- Gaussian tail goes over partitions

Approximated Value Function by GK

Optimal value function



Approximated by GK

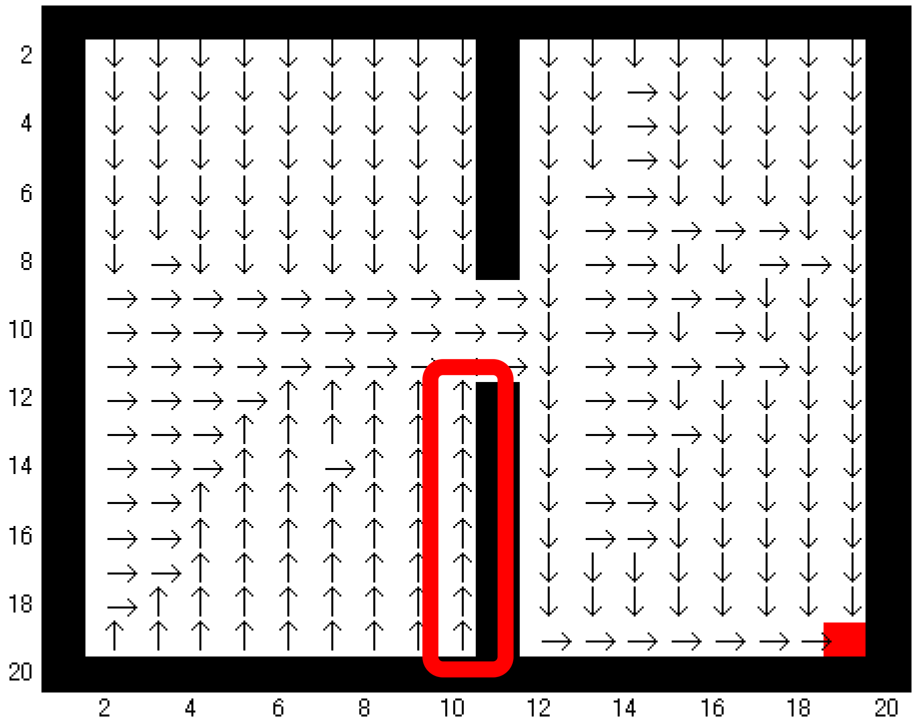


20 randomly located Gaussians

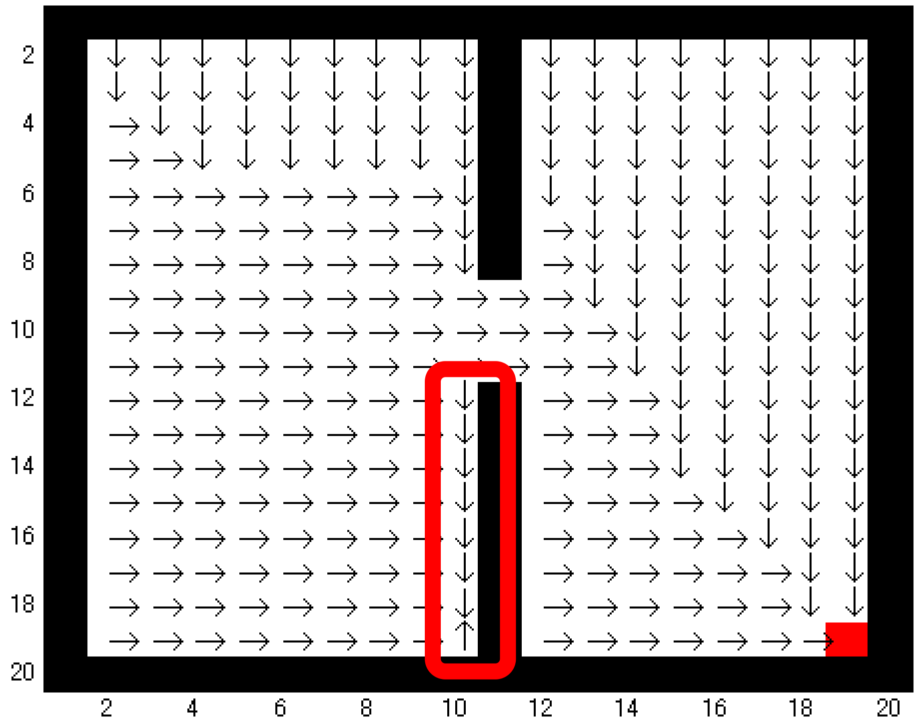
- Values around the partitions are not approximated well.

Policy Obtained by GK

Optimal policy



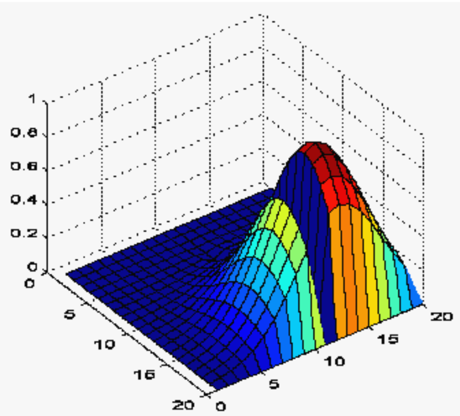
GK-based policy

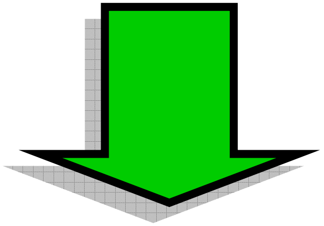


- GK provides an undesired policy around the partition.

Aim of This Research

- Gaussian tails go over the partition.
- Not suited for approximating discontinuous value functions.





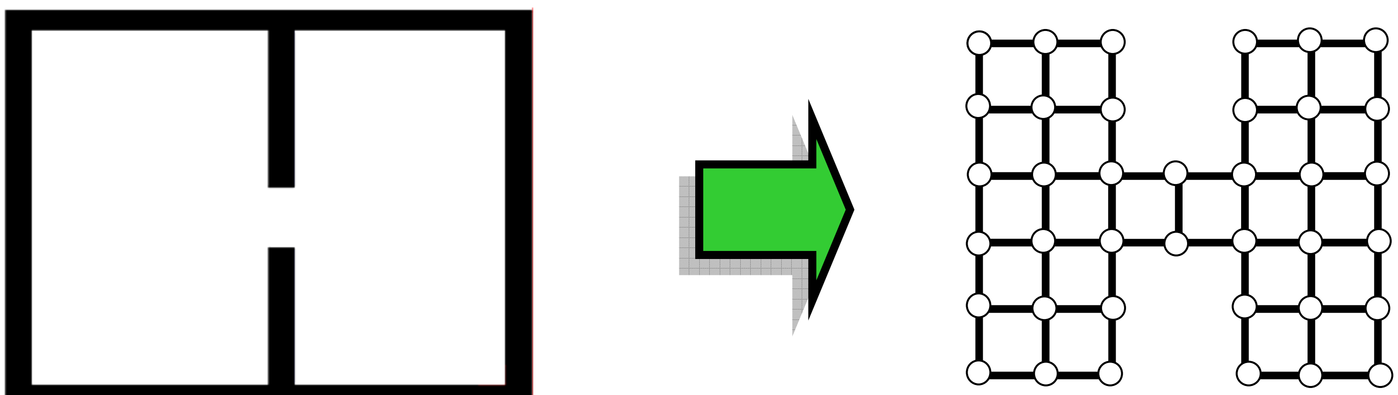
We propose new Gaussian kernel to overcome this problem.

State Space as a Graph

- Ordinary Gaussian uses Euclidean distance.

$$k(s) = \exp\left(-\frac{ED(s_c, s)^2}{2\sigma^2} \right)$$

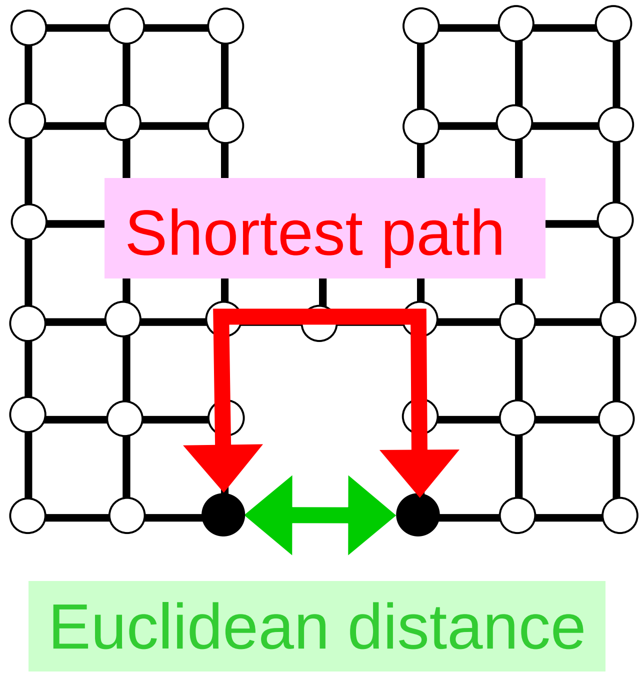
- Euclidean distance does not incorporate state space structure, so tail problems occur.
- We represent state space structure by a graph, and use it for defining Gaussian kernels.



(Mahadevan, ICML 2005)

Geodesic Gaussian Kernels

- Natural distance on graph is shortest path.

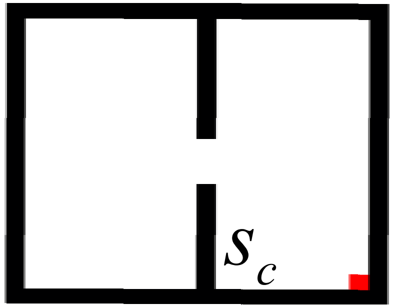


- We use shortest path in Gaussian function.

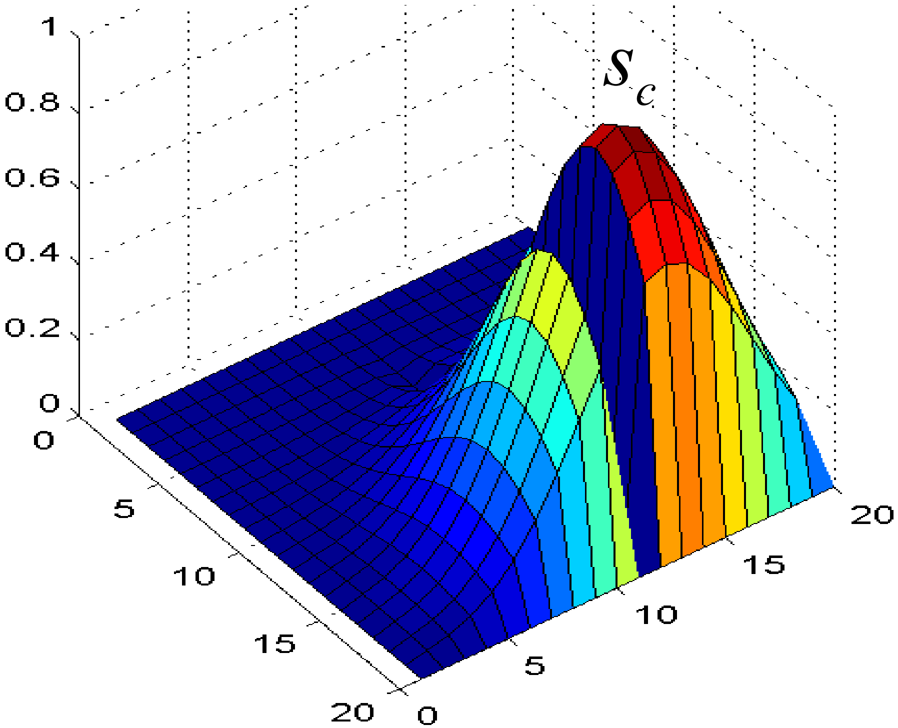
$$k(s) = \exp\left(-\frac{SP(s_c, s)^2}{2\sigma^2} \right)$$

- We call this kernel geodesic Gaussian.
- SP can be efficiently computed by Dijkstra.

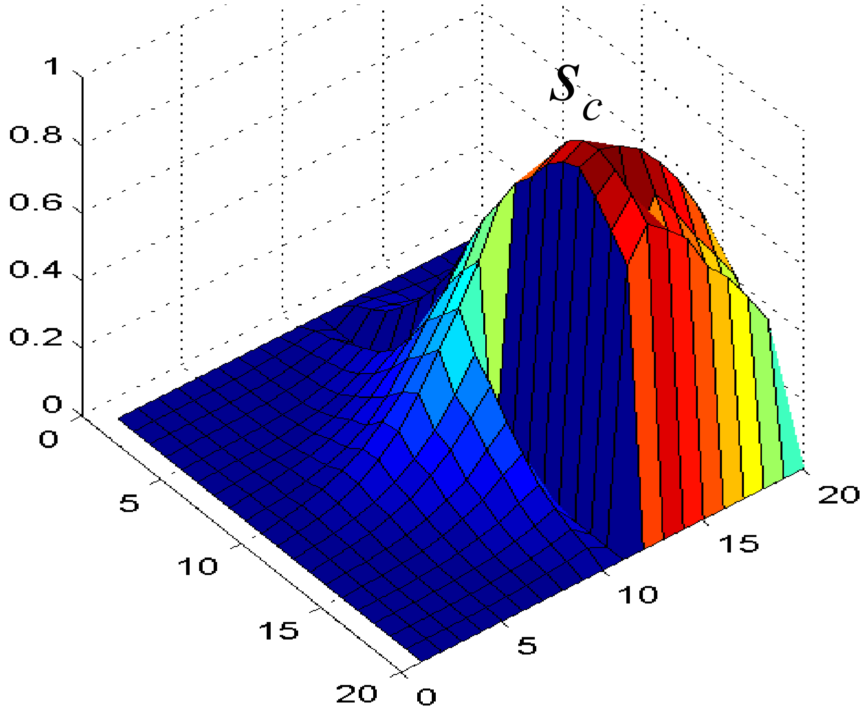
Example of Kernels



Ordinary Gaussian

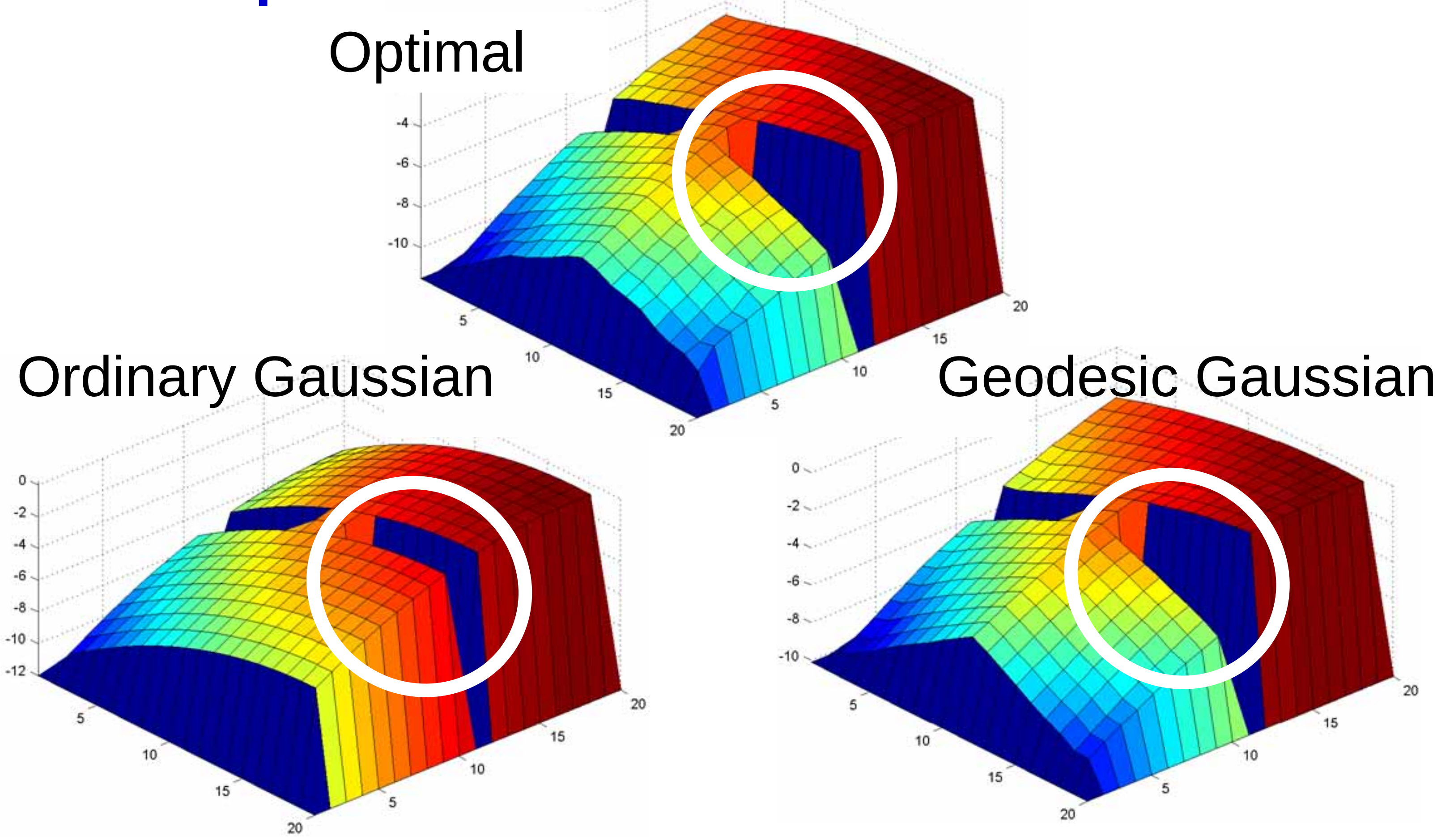


Geodesic Gaussian



- Tails do not go across the partition.
- Values smoothly decrease along the maze.

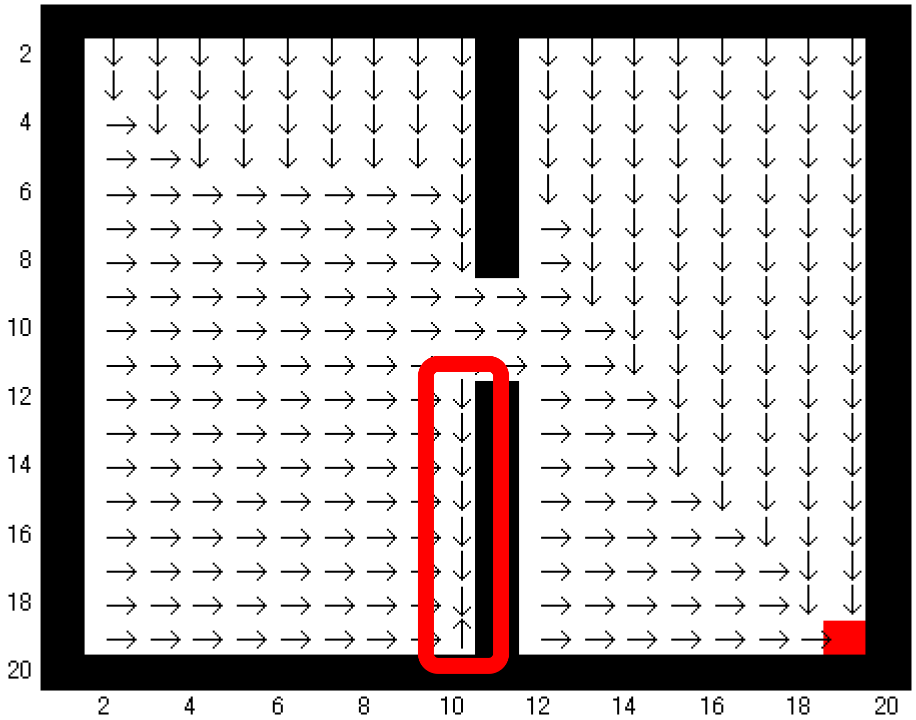
Comparison of Value Functions



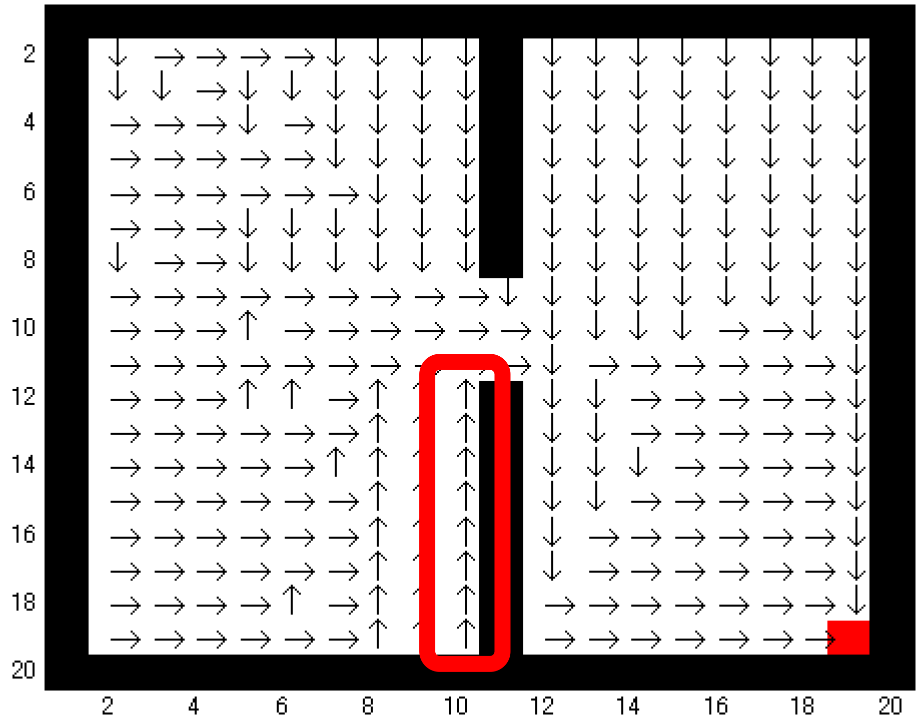
- Values near the partition are well approximated.
- Discontinuity across the partition is preserved.

Comparison of Policies

Ordinary Gaussian

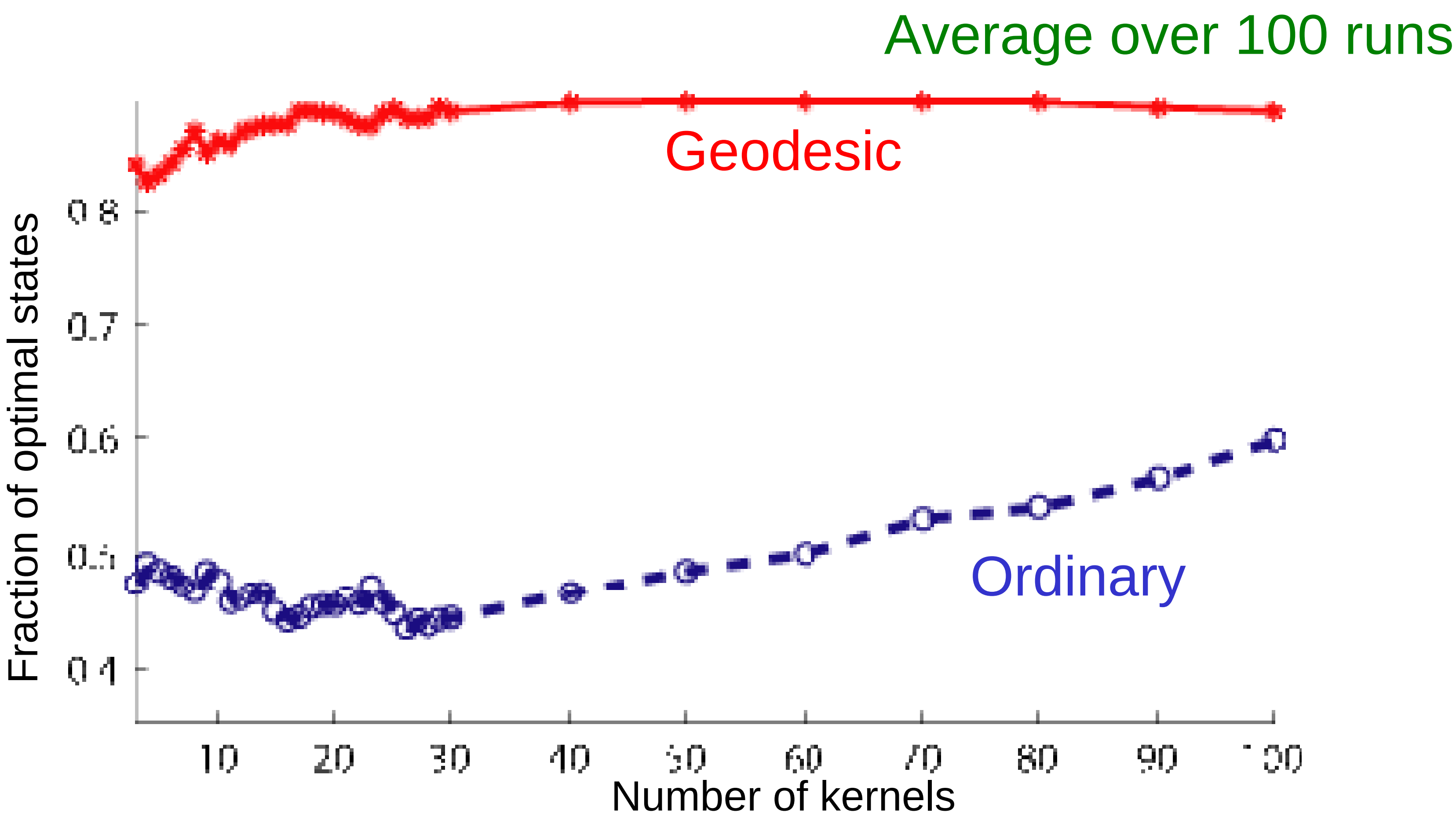


Geodesic Gaussian



- GGKs provide good policies near the partition.

Experimental Result

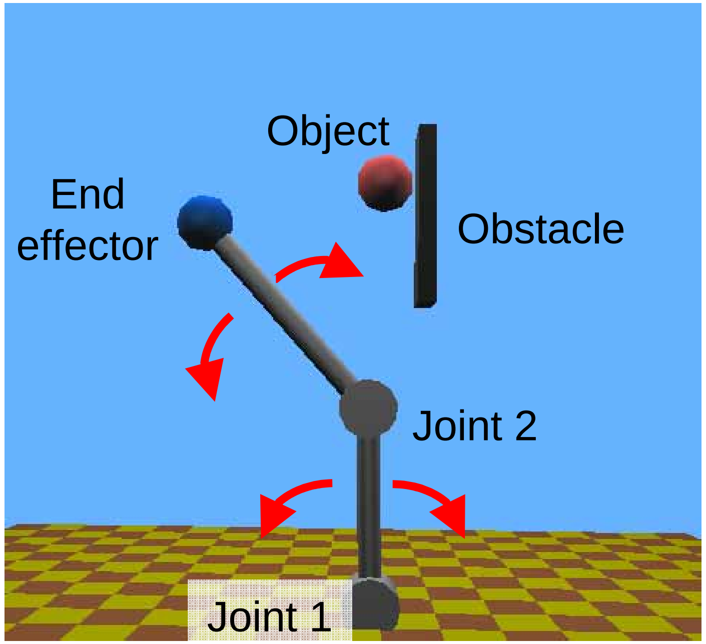


- Ordinary Gaussian: tail problem
- Geodesic Gaussian: no tail problem

Robot Arm Reaching

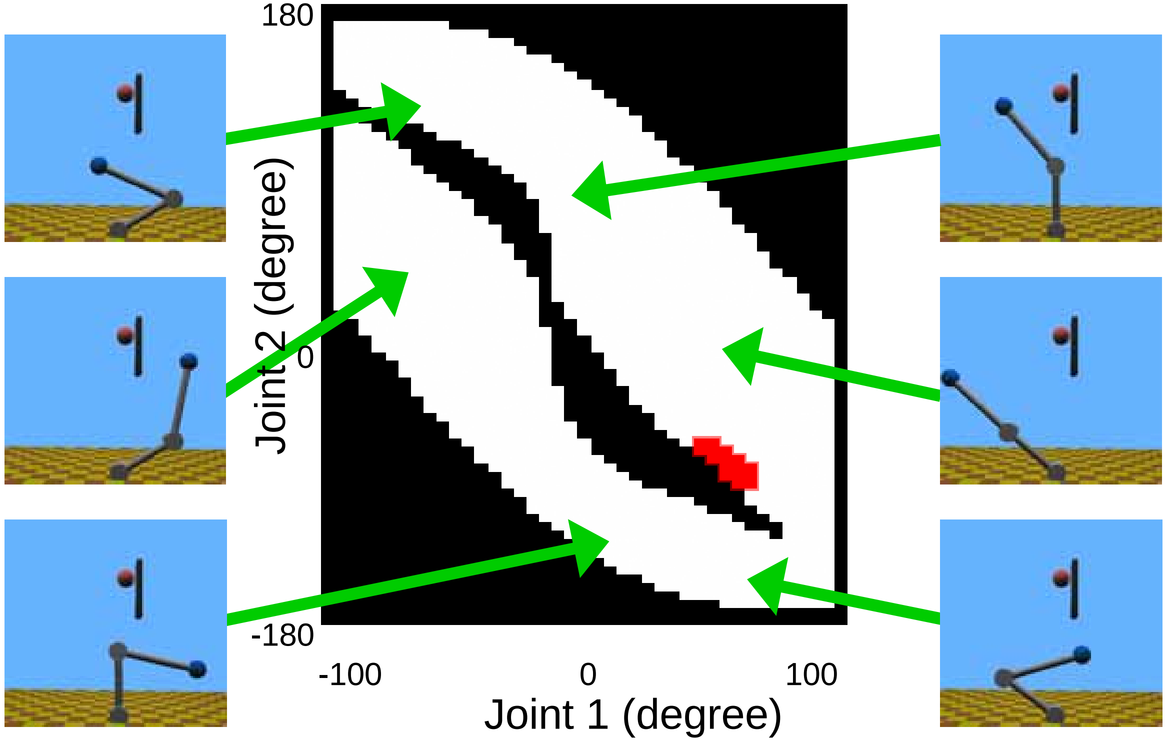
- Task: move the end effector to reach the object

2-DOF robot arm

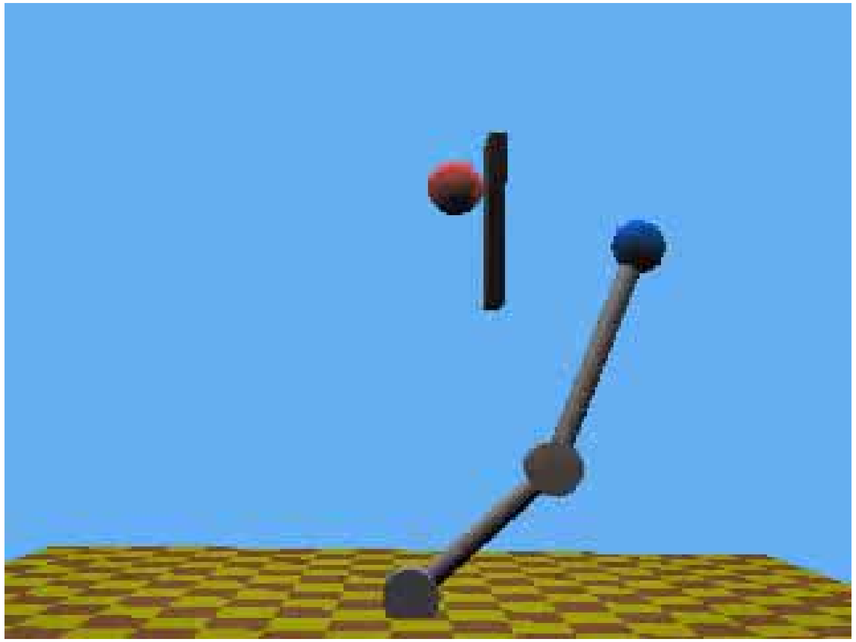
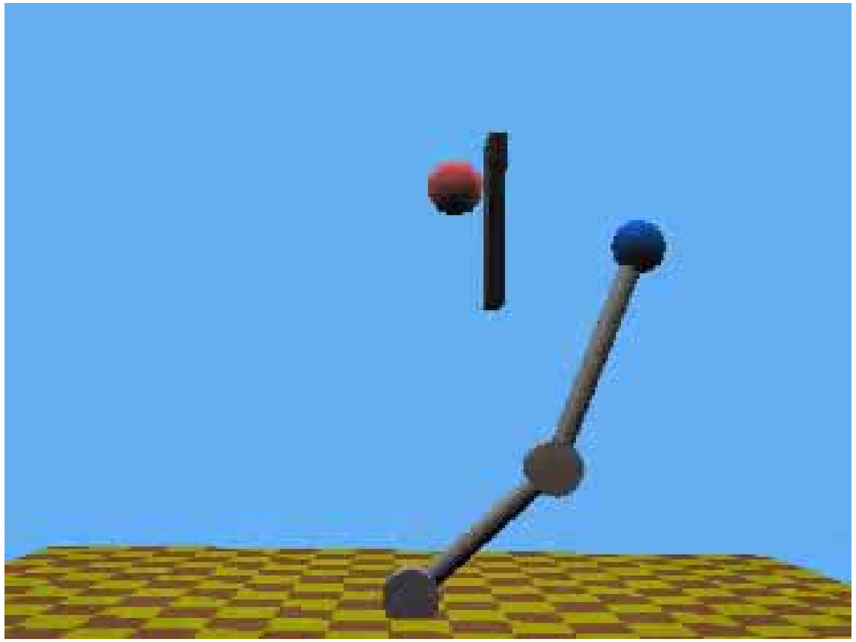


Reward:
+1 reach the object
0 otherwise

State space

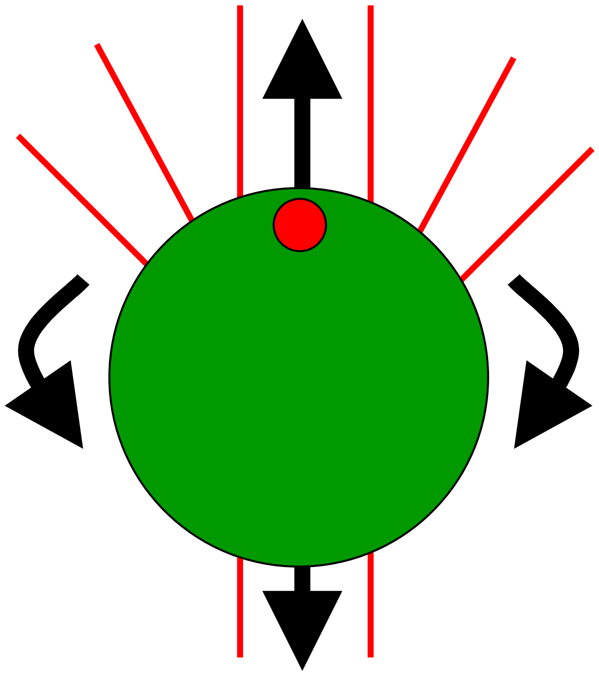


Robot Arm Reaching

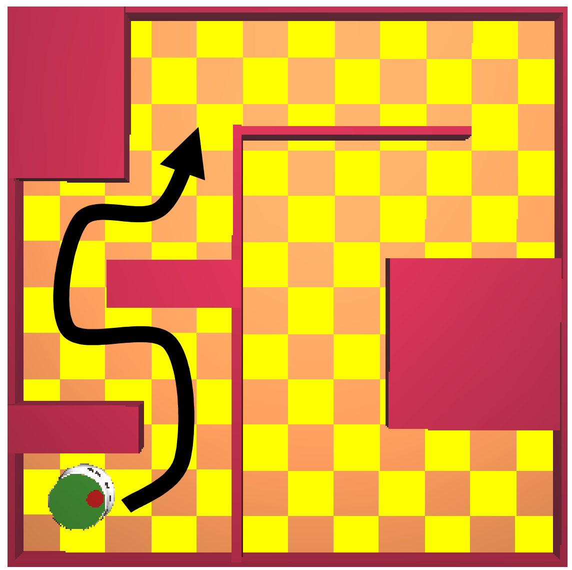
Ordinary Gaussian	Geodesic Gaussian
	
Moves directly towards the object without avoiding the obstacle.	Successfully avoids the obstacle and can reach the object.

Khepera Robot Navigation

- Khepera has 8 IR sensors measuring the distance to obstacles.
- Task: explore unknown maze without collision



Sensor value: 0 - 1030

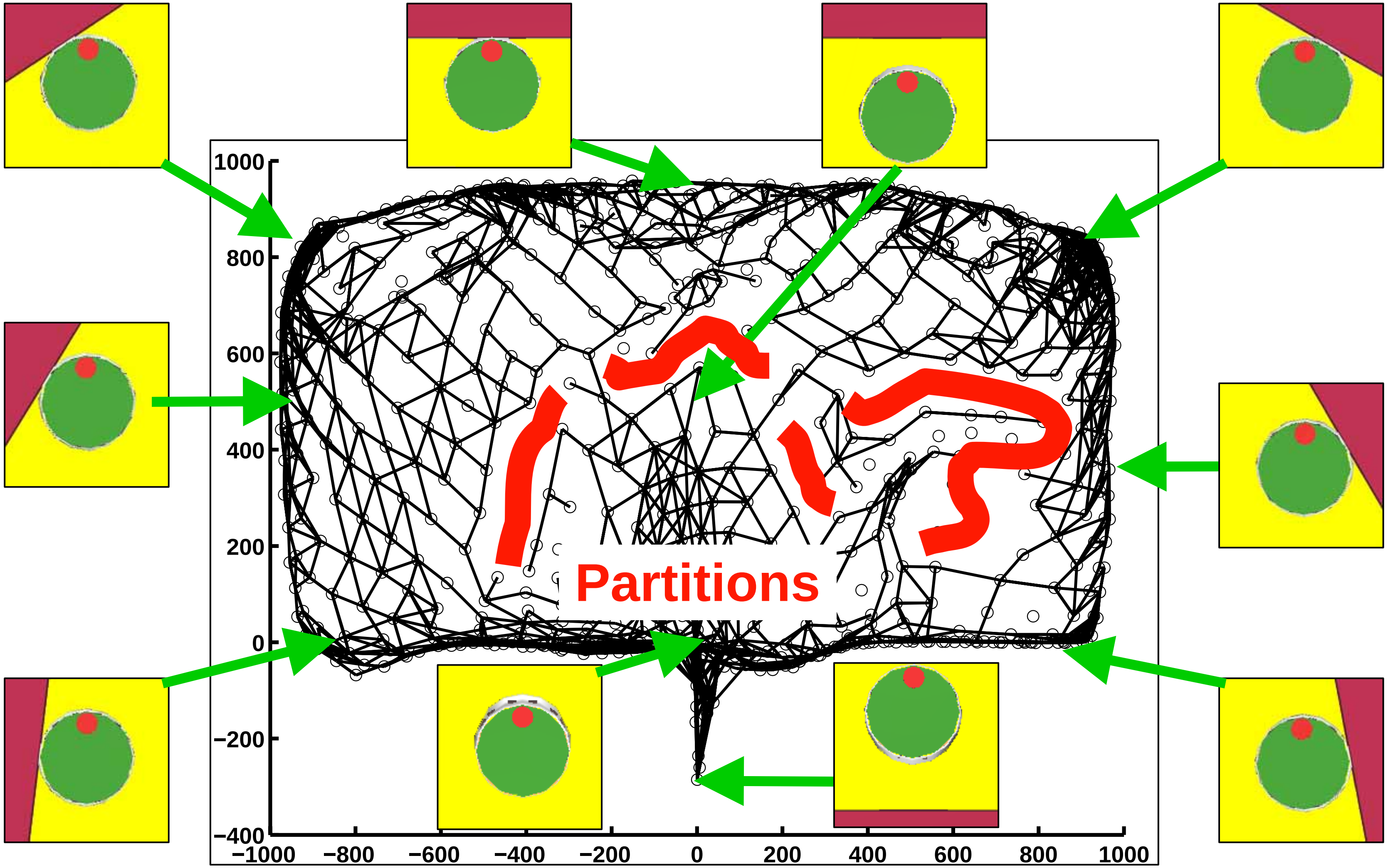


Reward:
+1 (forward)
-2 (collision)
0 (others)

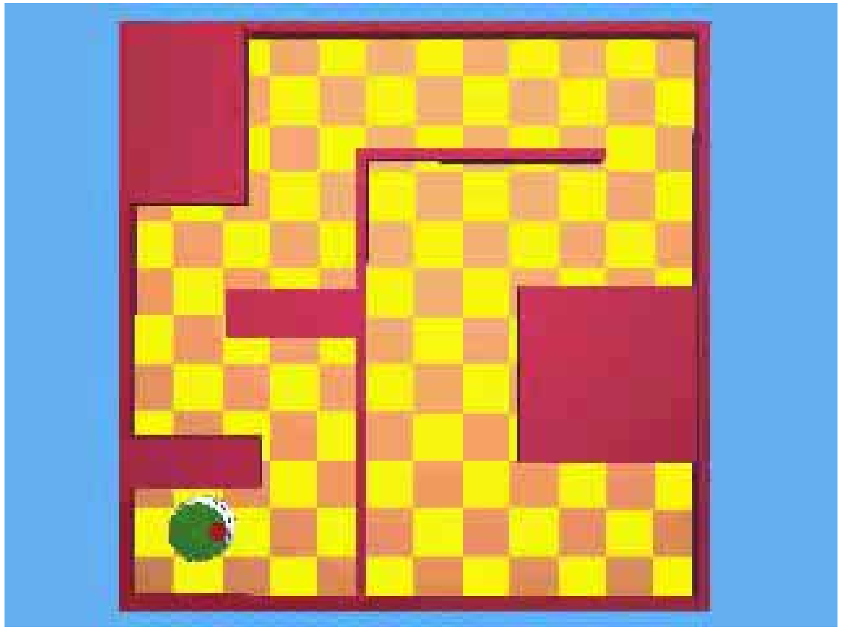
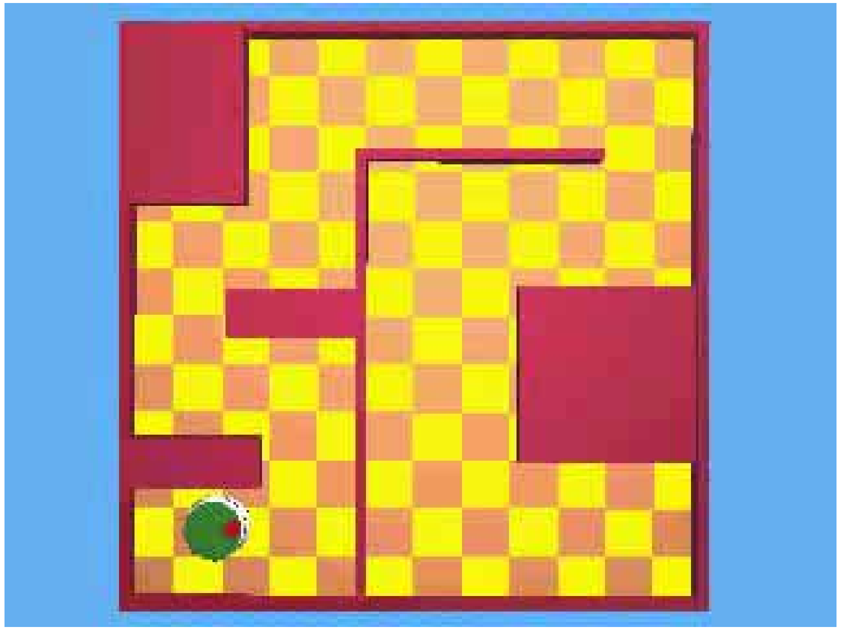
State Space and Graph

Discretize 8D state space by self-organizing map.

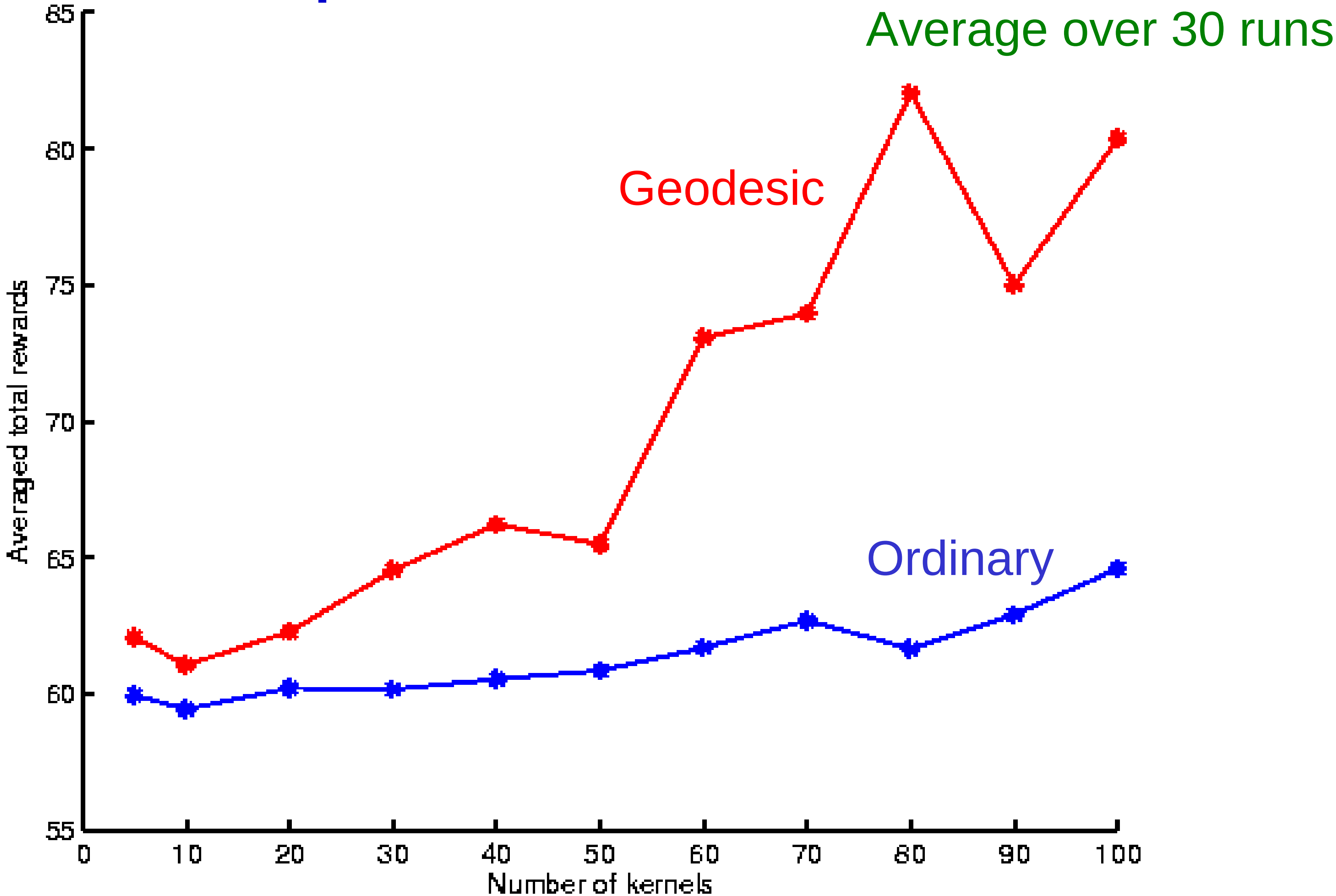
2D visualization



Khepera Robot Navigation

Ordinary Gaussian	Geodesic Gaussian
	
When facing obstacle, goes backward (and goes forward again).	When facing obstacle, makes a turn (and go forward).

Experimental Results



- Geodesic outperforms ordinary Gaussian.

Conclusion

- Value function approximation:
  - ⇒ good basis function needed
- Ordinary Gaussian kernel:
  - ⇒ tail goes over discontinuities
- Geodesic Gaussian kernel:
  - ⇒ smooth along the state space
- Through the experiments, we showed geodesic Gaussian is promising in high-dimensional continuous problems!