Robotics

Miao Li

Fall 2023, Wuhan University

WeChat: 15527576906

Email: limiao712@gmail.com

2023-10-9

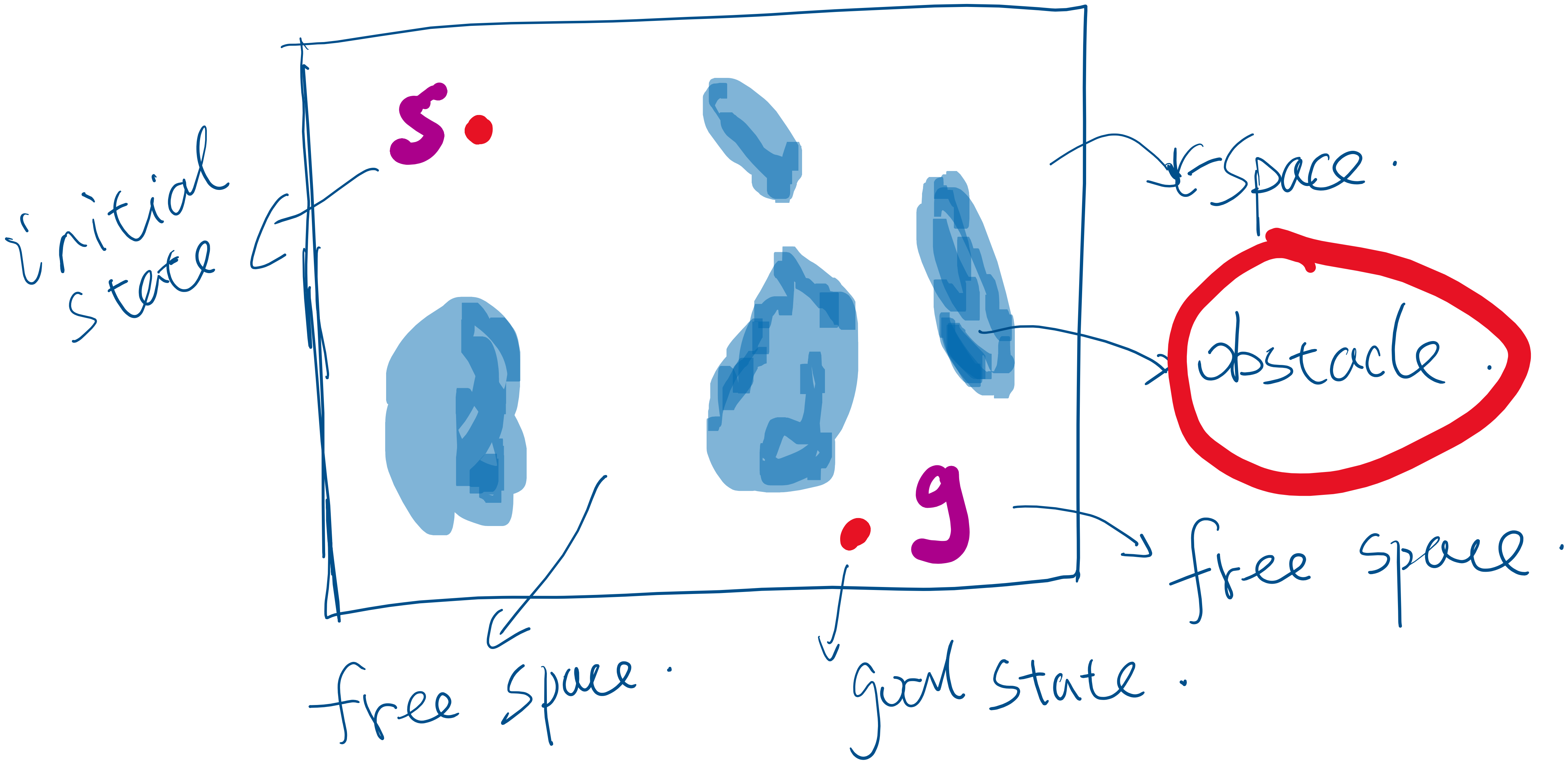
Goal for this course

- **Design： soft hand design x1**
- **Perception: vision, point cloud, tactile, force/torque x1**
- **Planning: sampling-based, optimization-based, learning-based x3**
- **Control: feedback, multi-modal x2**
- **Learning: imitation learning, RL x2**
- **Simulation tool (pybullet, matlab, OpenRAVE, Issac Nvidia, Gazebo)**
- **How to get a robot moving!**

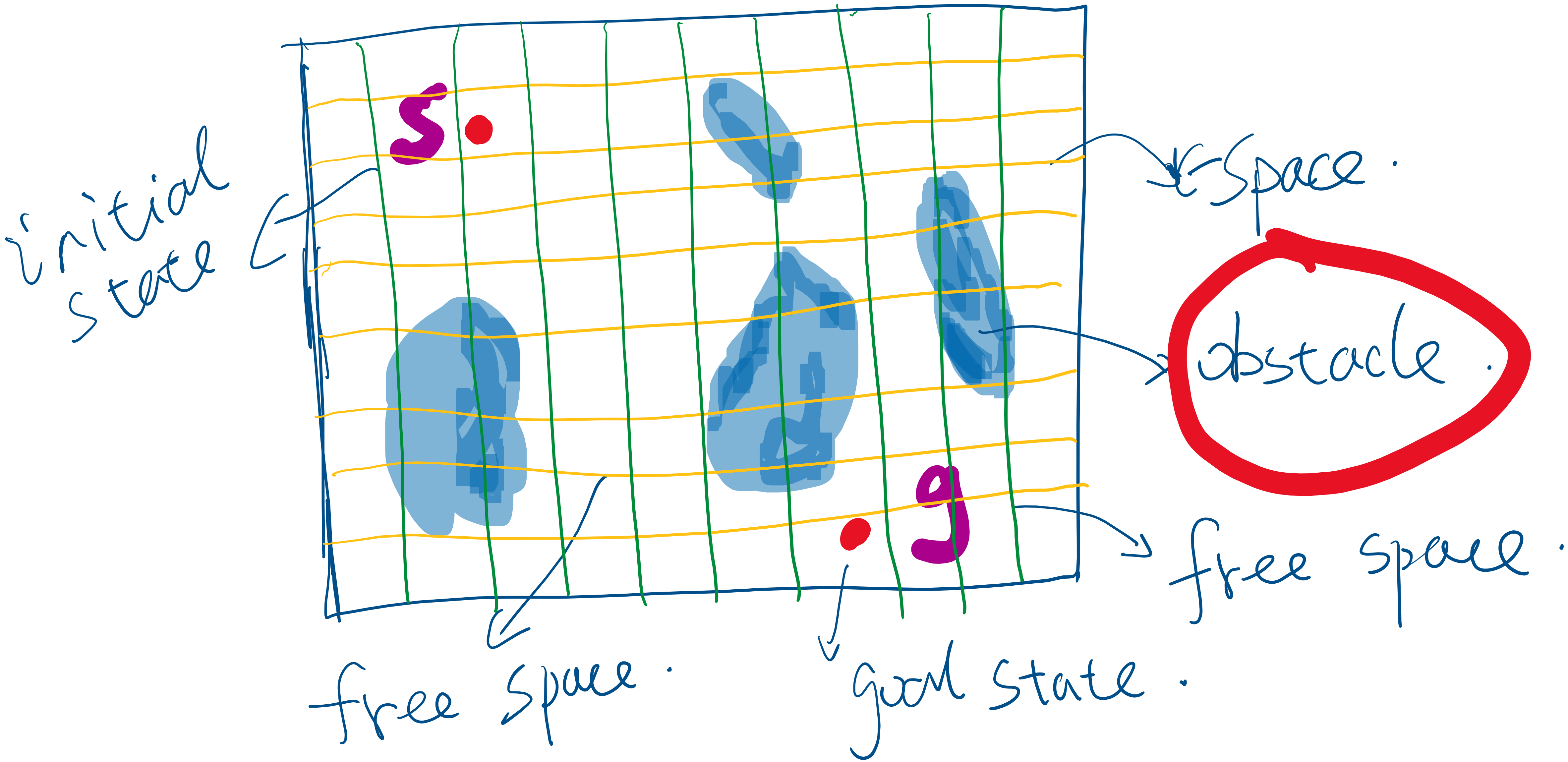
Today's Agenda

- **Recap of sampling-based approach (~10)**
- **Recap of optimization-based approach (~20)**
- **Drawback of sampling and optimization (~5)**
- **Recap of perception-action loop (~2)**
- **Learning-based motion planning (~5)**
- **Imitation learning (~20)**
- **Reinforcement learning (~10)**

Motion Planning in 2D



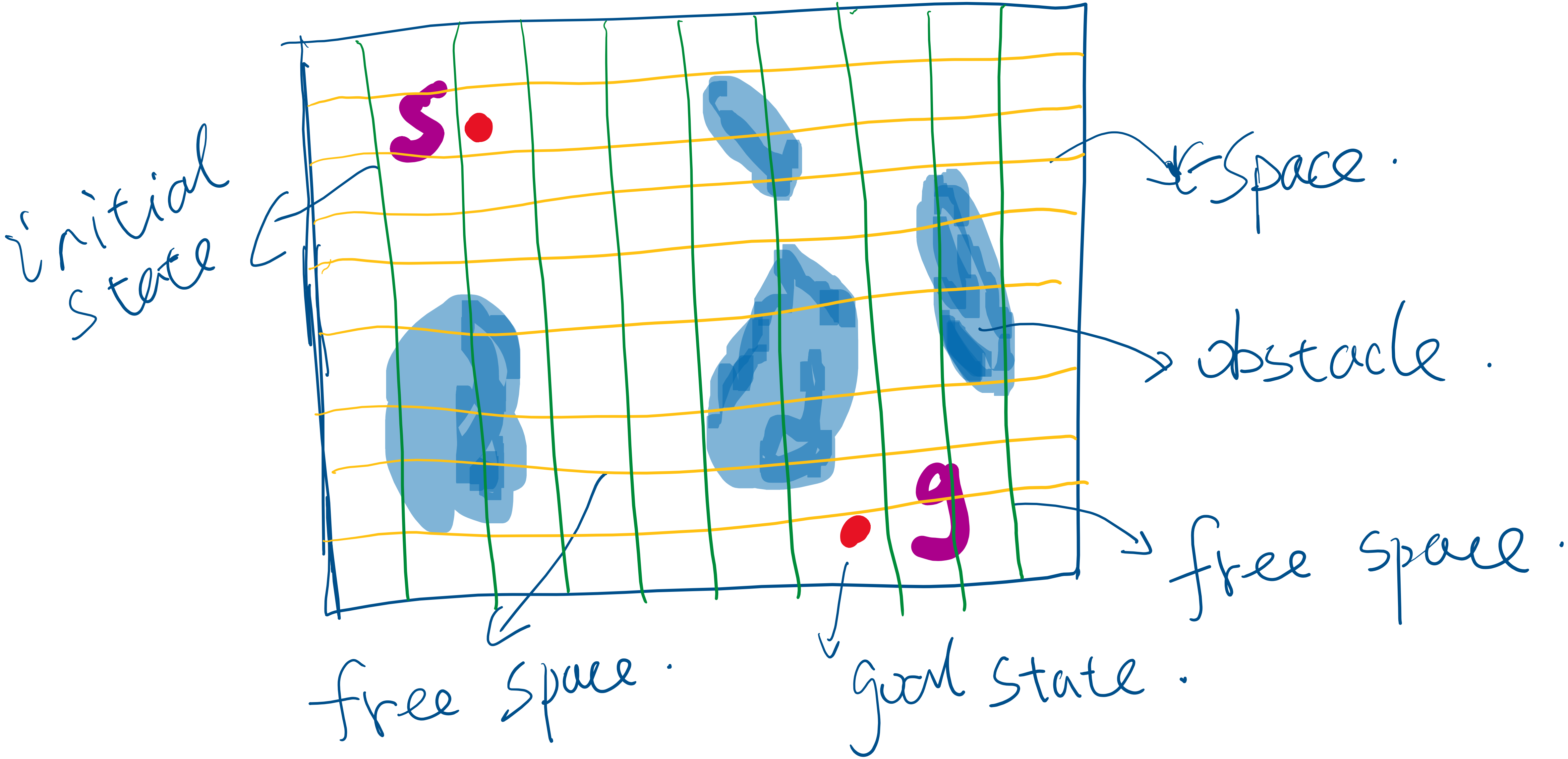
Motion Planning in Grid World



Recap of sampling-based approach

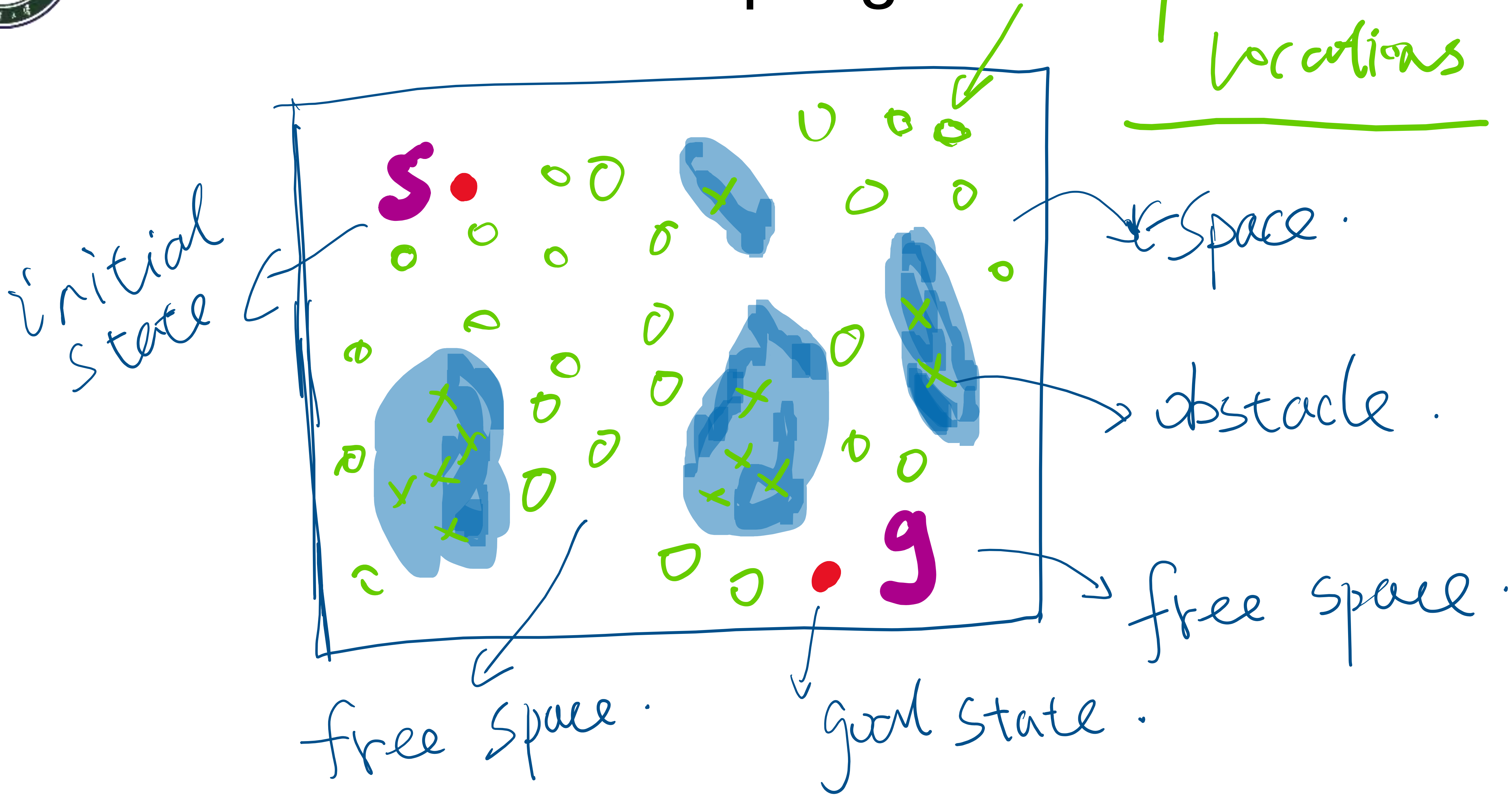
- **Completely describing and optimally exploring is too hard in high dimension space**
- **It is not necessary**
- **Limit ourselves to finding a “good” sampling**

Sampling

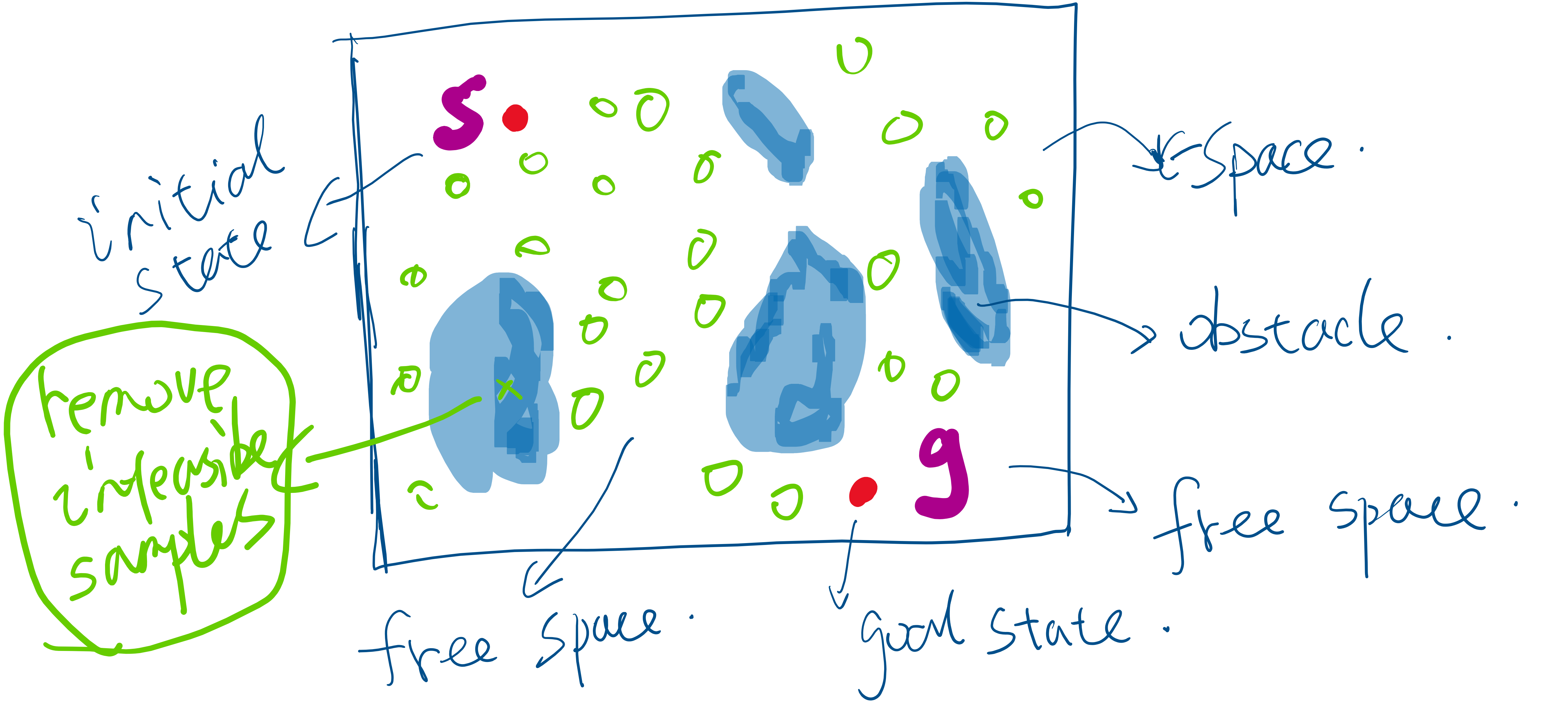


initial state
s
X-space
obstacle
free space
g
free space
goal state

Sampling

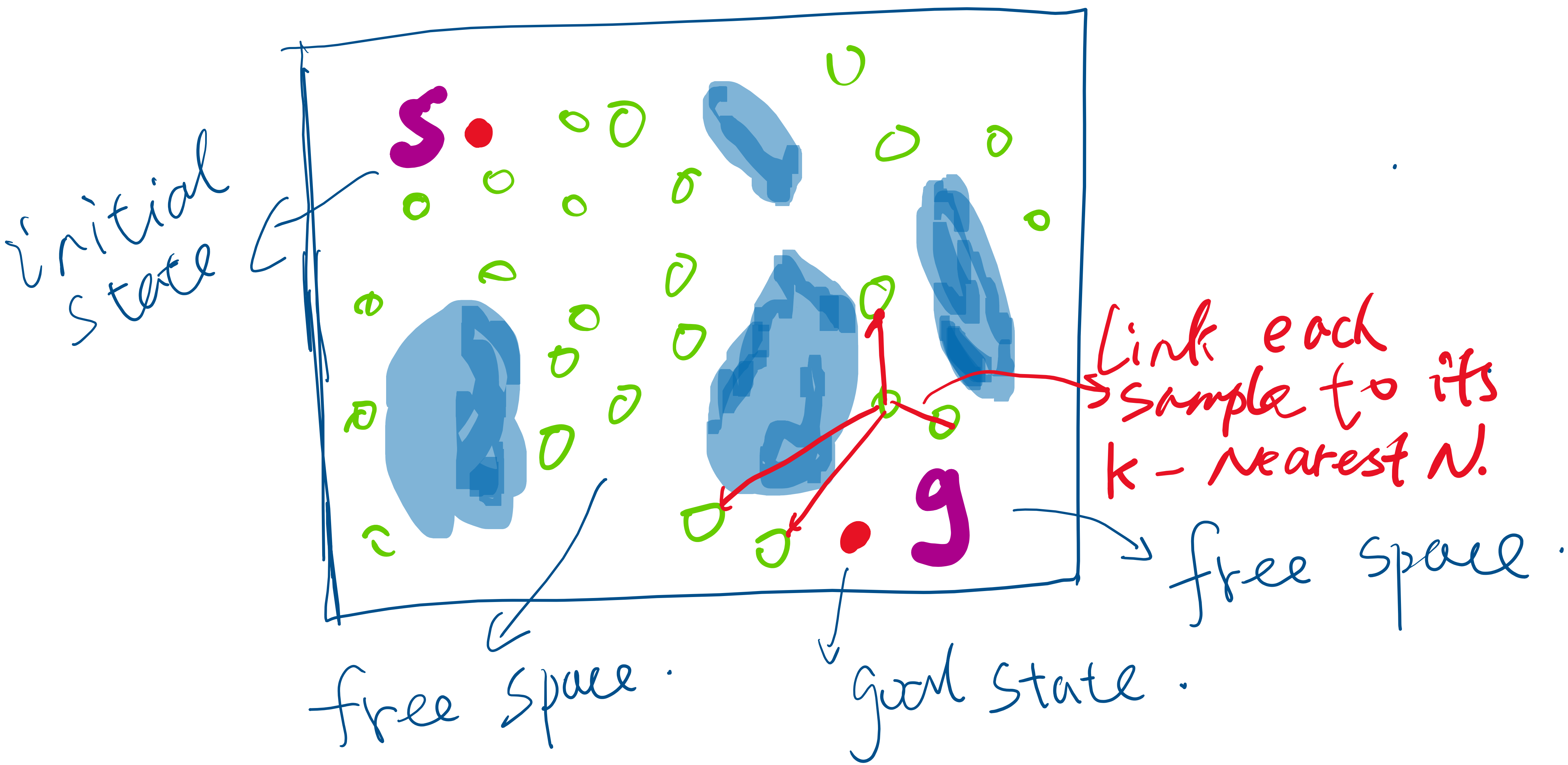

sample random locations
initial state
S
g
space
obstacle
free space
free space
goal state

Sampling



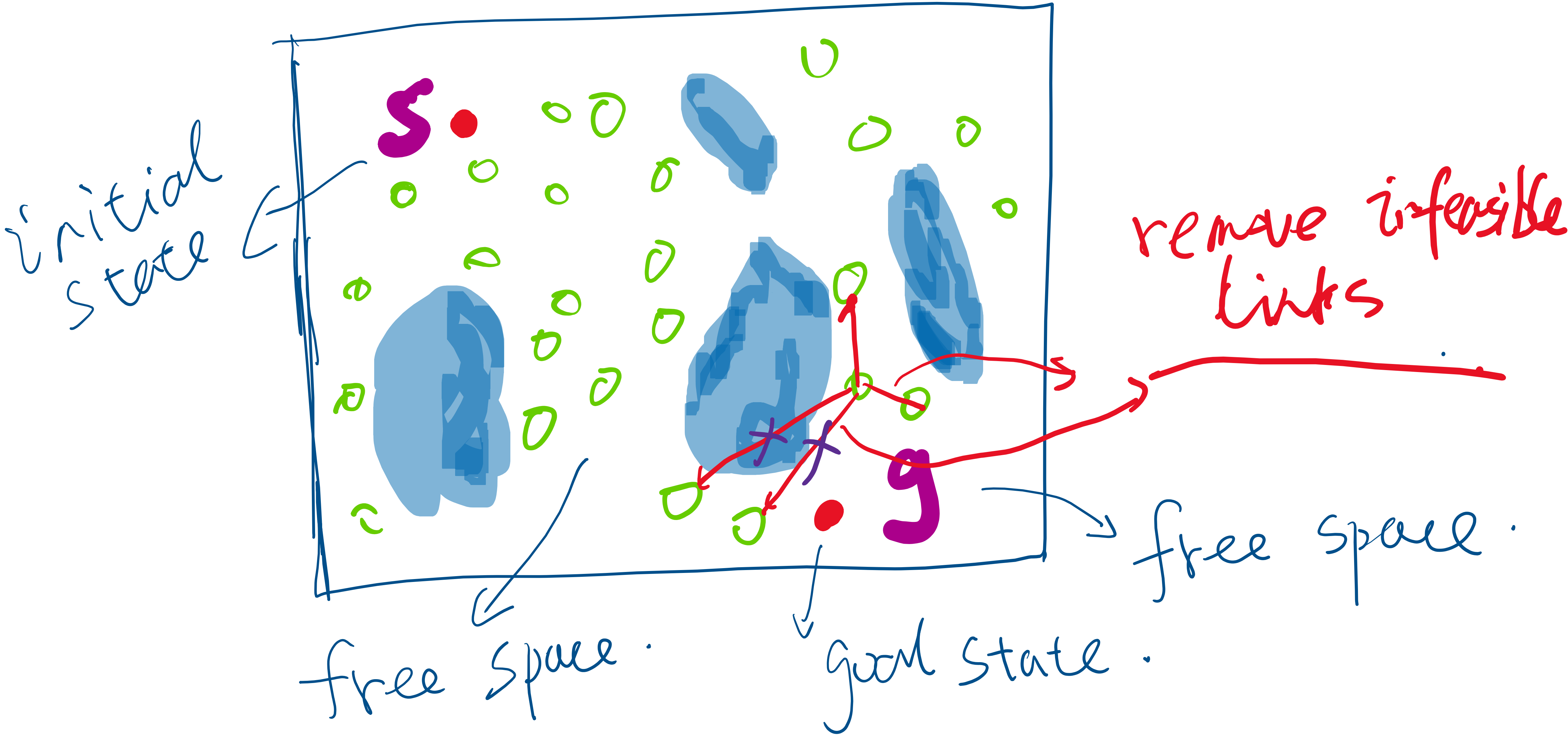
initial state
remove infeasible samples
s
g
C-space.
obstacle.
free space.
free space.
goal state.

Sampling



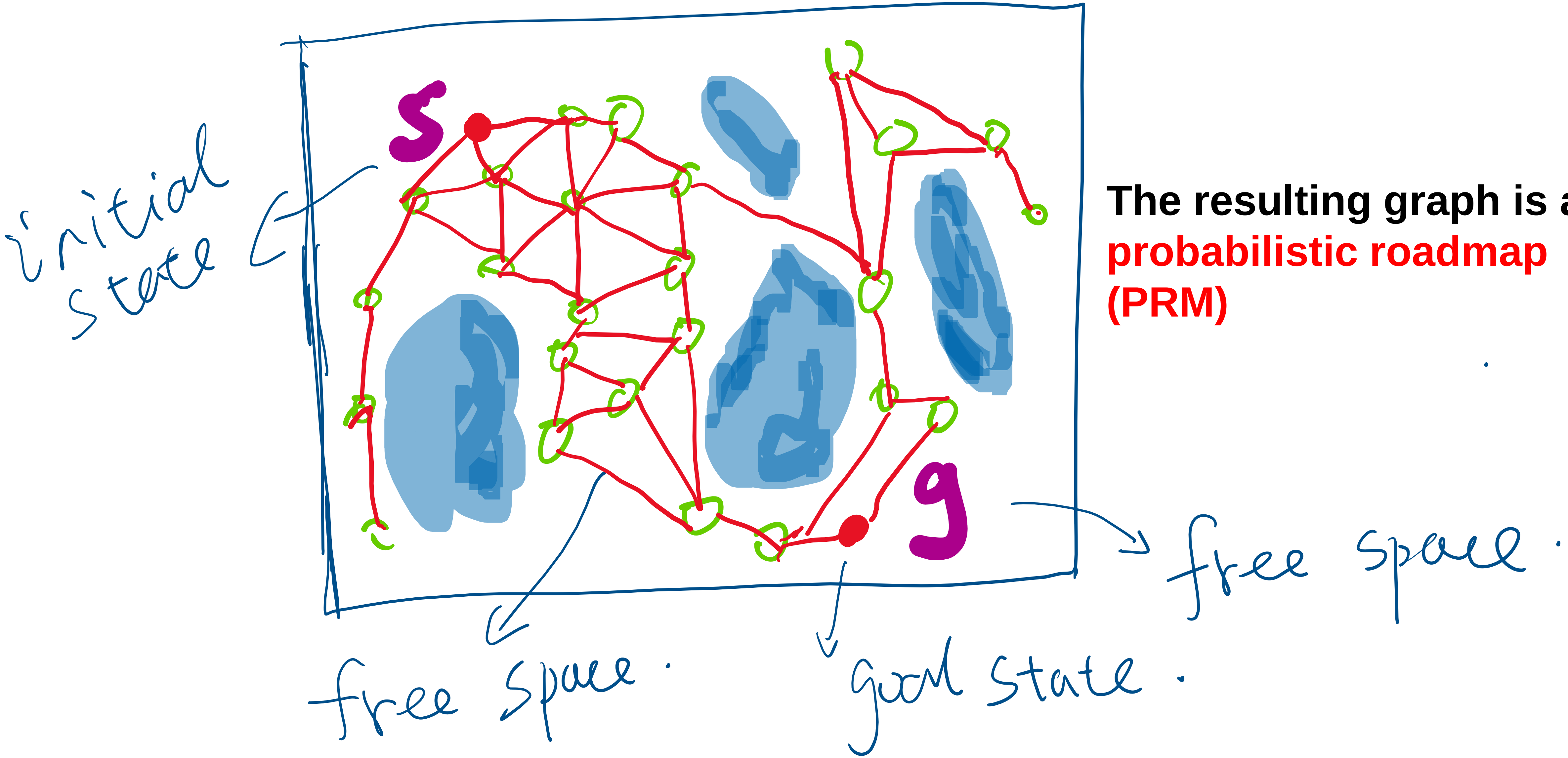
initial state
s
g
Link each sample to its k-Nearest N.
free space.
free space.
goal state.

Sampling



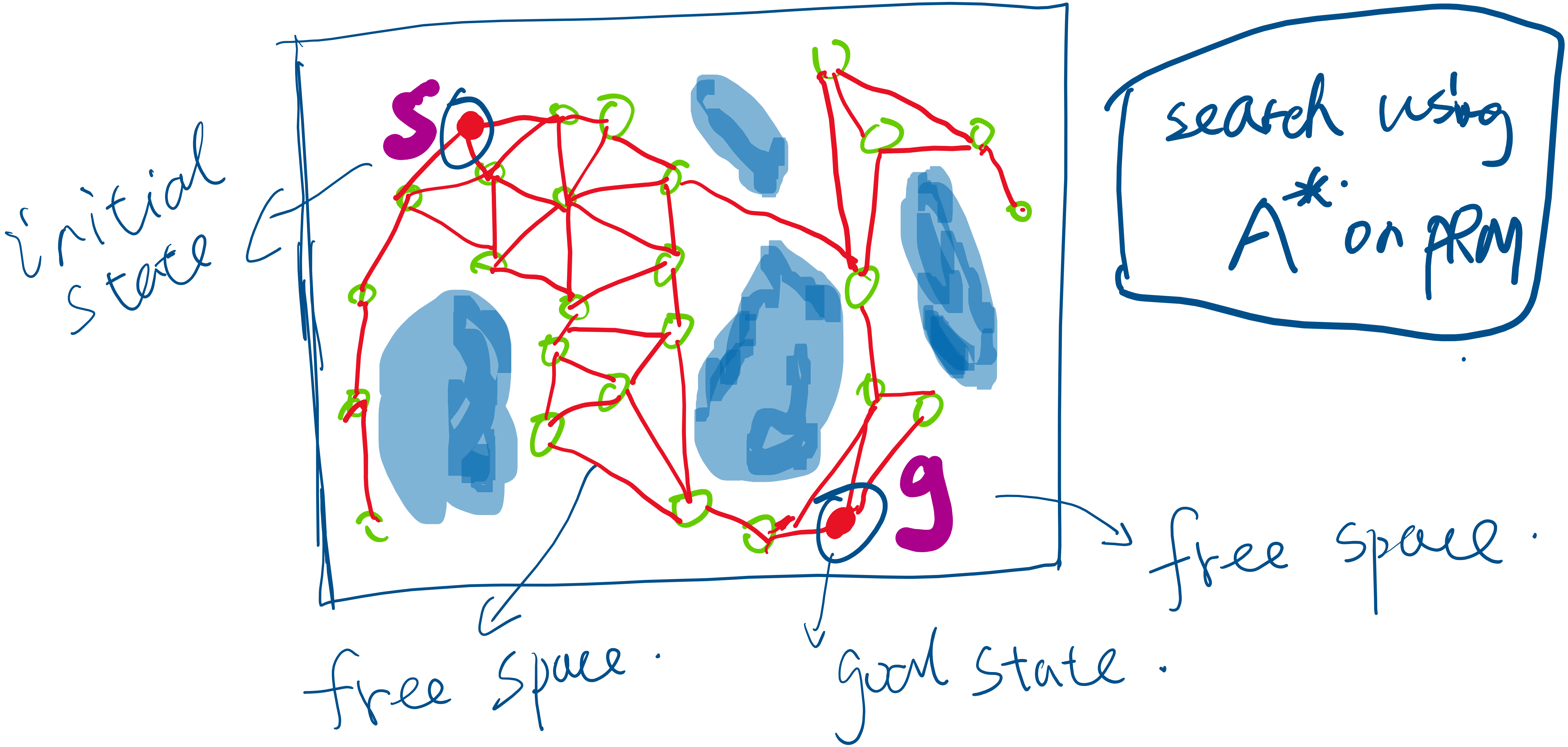
initial state
s
g
remove infeasible links
free space.
free space.
goal state.

PRM
(probabilistic roadmap)

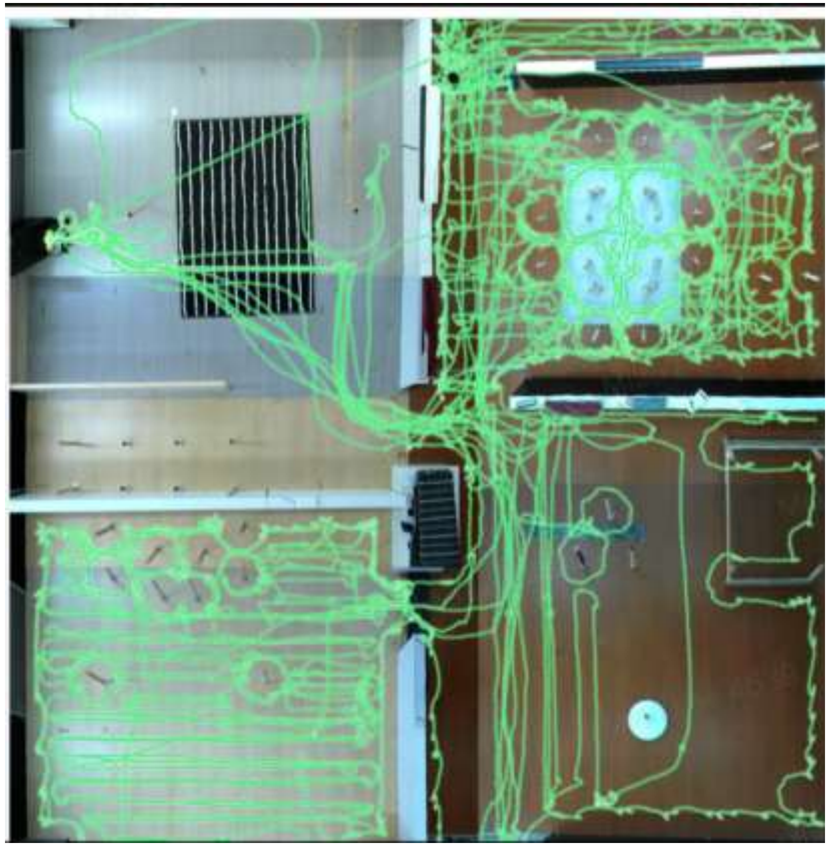


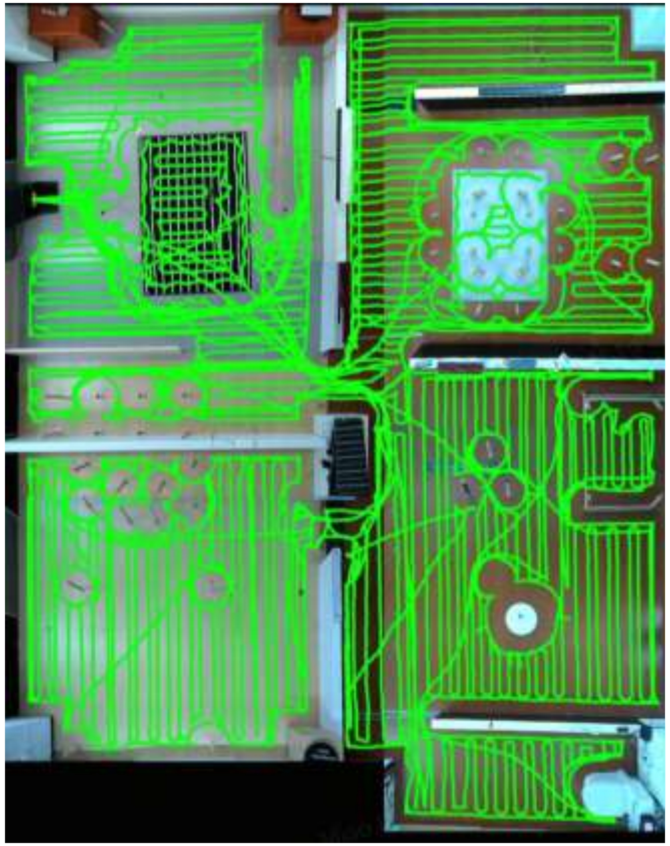
The resulting graph is a probabilistic roadmap (PRM)

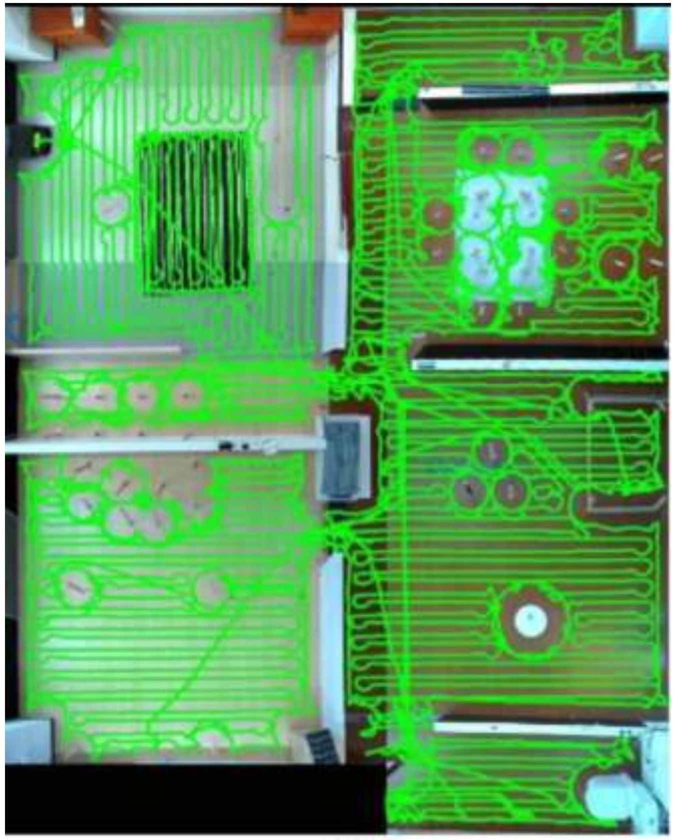
PRM
(probabilistic roadmap)



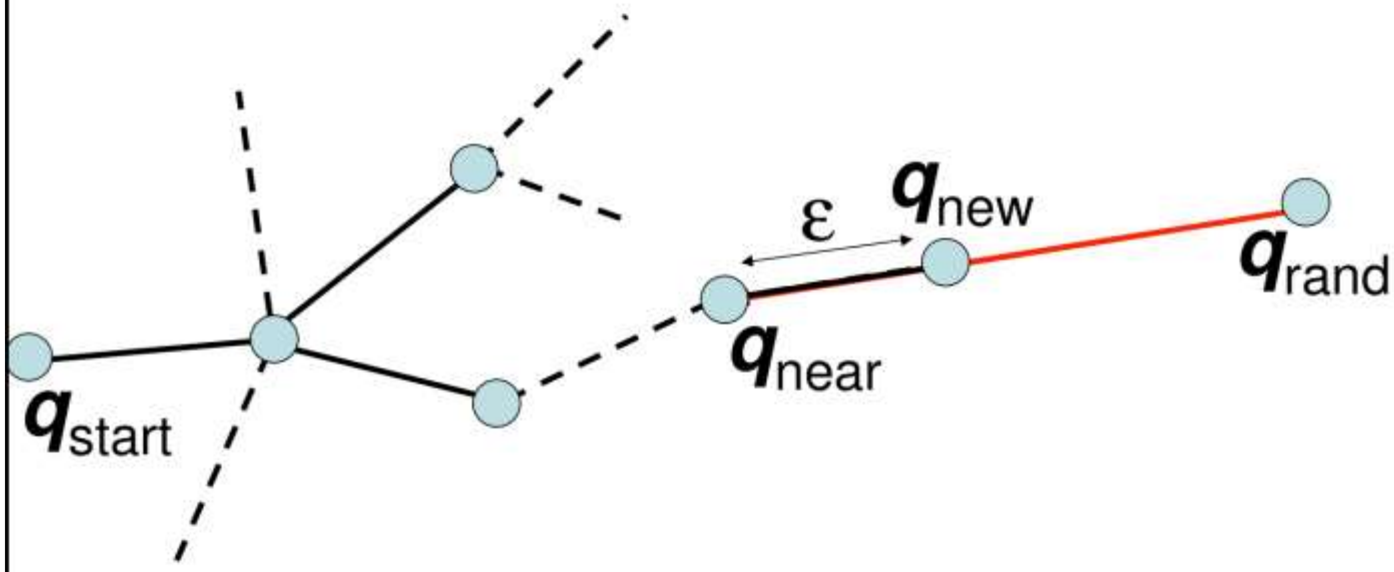
Example







RRT **Rapidly Exploring Random Trees**

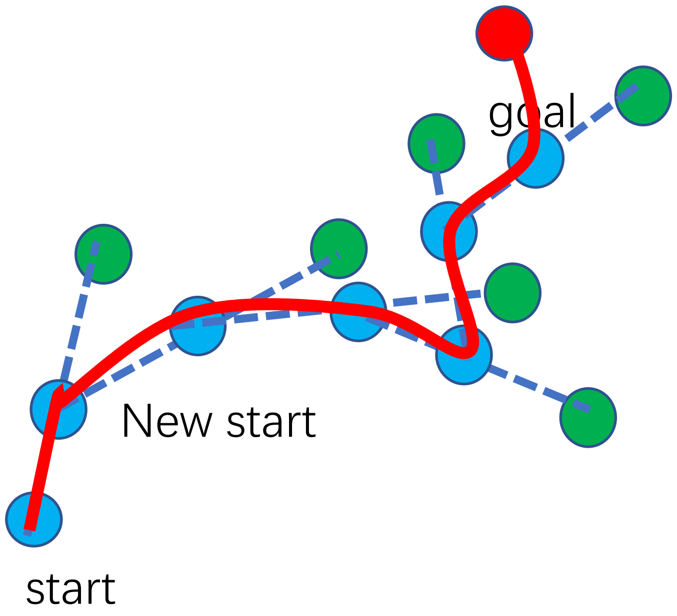


Remarkably, we can find a solution by using *relatively few randomly* sampled points.

RRT

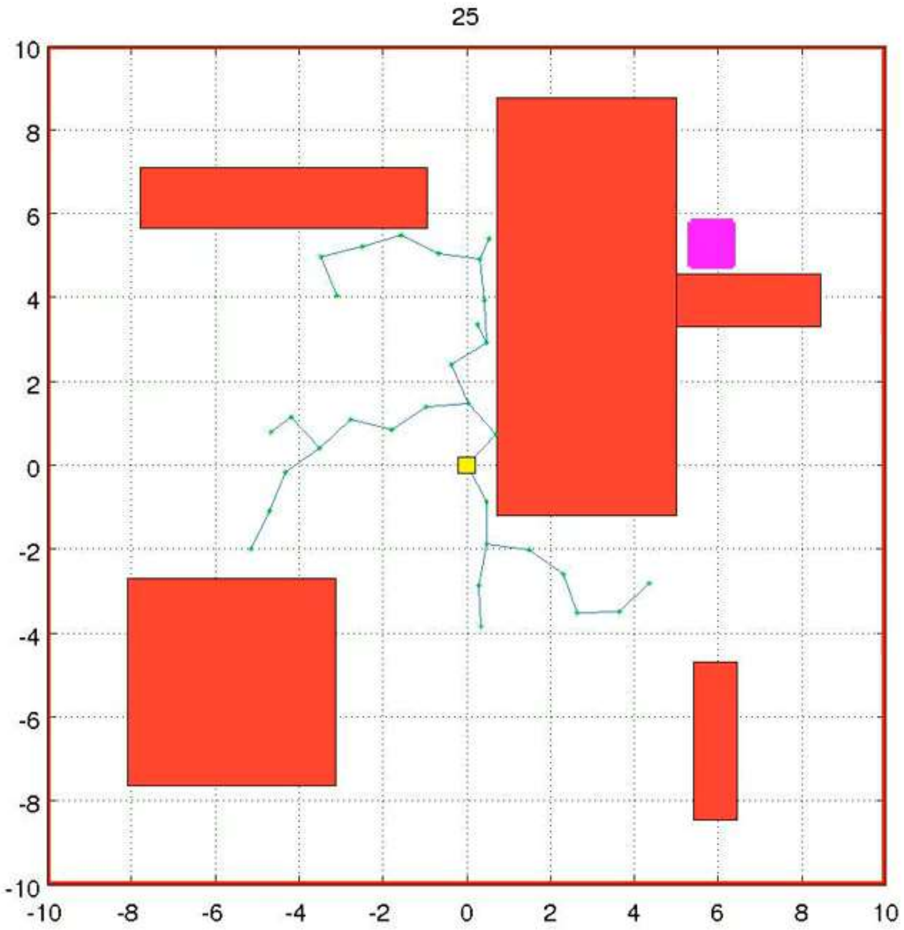
RRT Algorithm (x_{start}, x_{goal}, step, n)

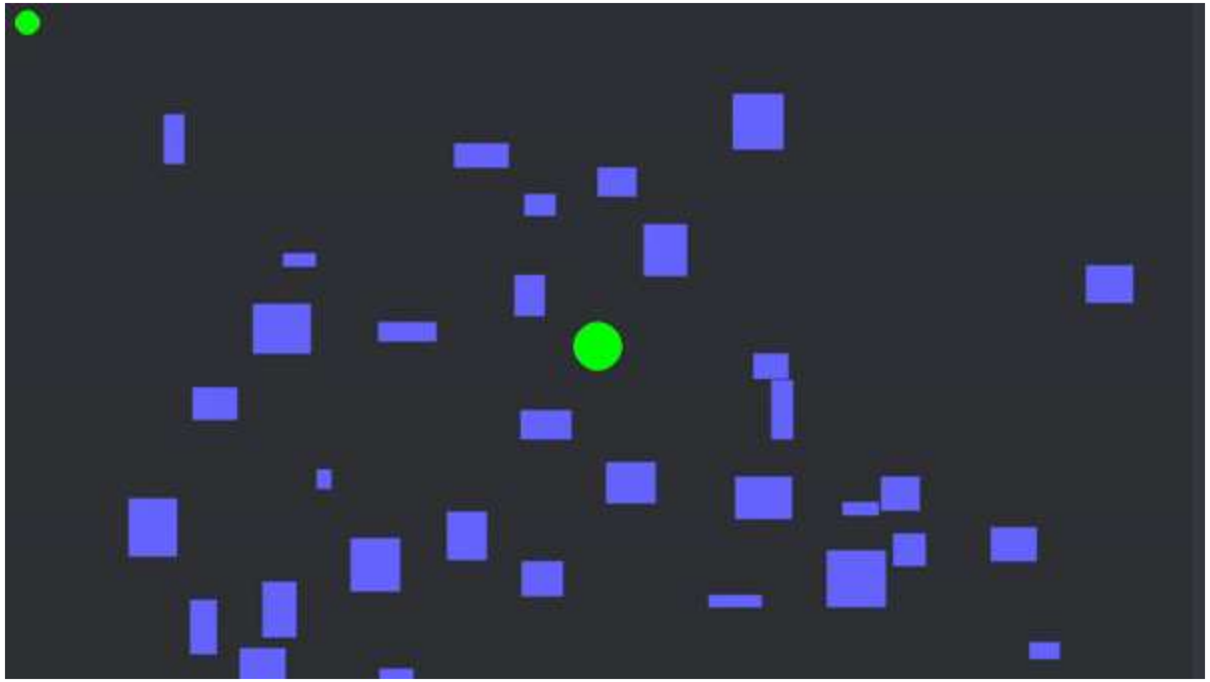
```
G.initialize(x_start)
for i = 1 to n do
    x_rand = Sample()
    x_near = near(x_rand, G)
    x_new = steer(x_rand, x_near, step_size)
    G.add_node(x_new)
    G.add_edge(x_new, x_near)
    if  x_new = x_goal
        success()
```



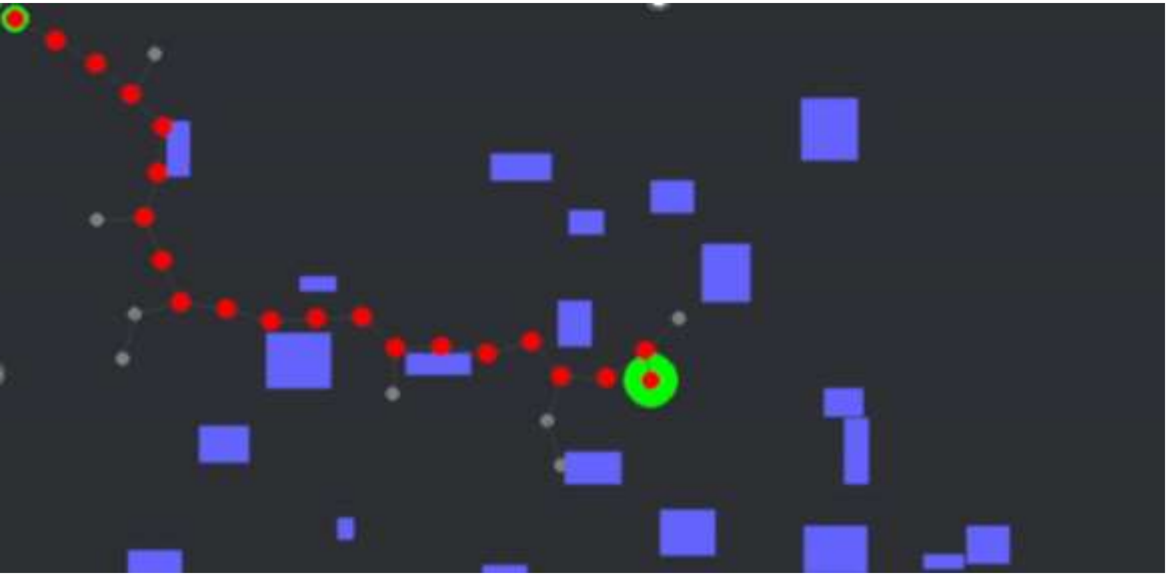
- J-C. Latombe. Robot Motion Planning. Kluwer. 1991.
- S. Lavalle. Planning Algorithms. 2006. http://msl.cs.uiuc.edu/planning/
- H. Choset et al., Principles of Robot Motion: Theory, Algorithms, and Implementations. 2006.

RRT





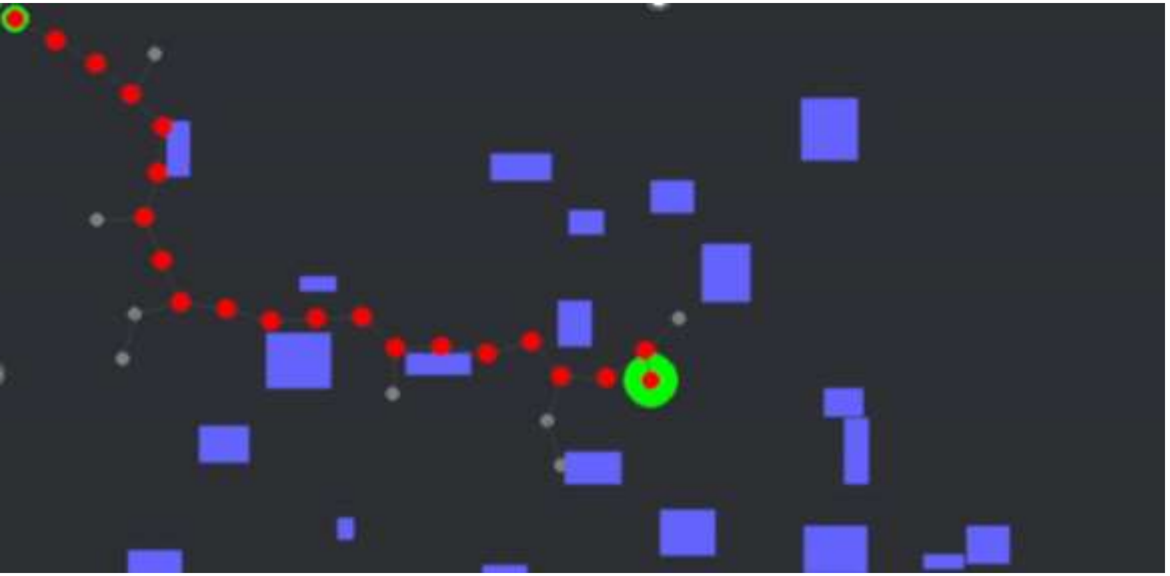
RRT revisit



- **Few control params of the solution**
- **Near to collisions**
- **Ignore trivial solution**
- **Path quality can be bad**
- **Quite different with different seeds**
- **Additional steps for collision checking**

What is the problem with this approach?

RRT revisit



RRT is not optimal

What is the problem with this approach?

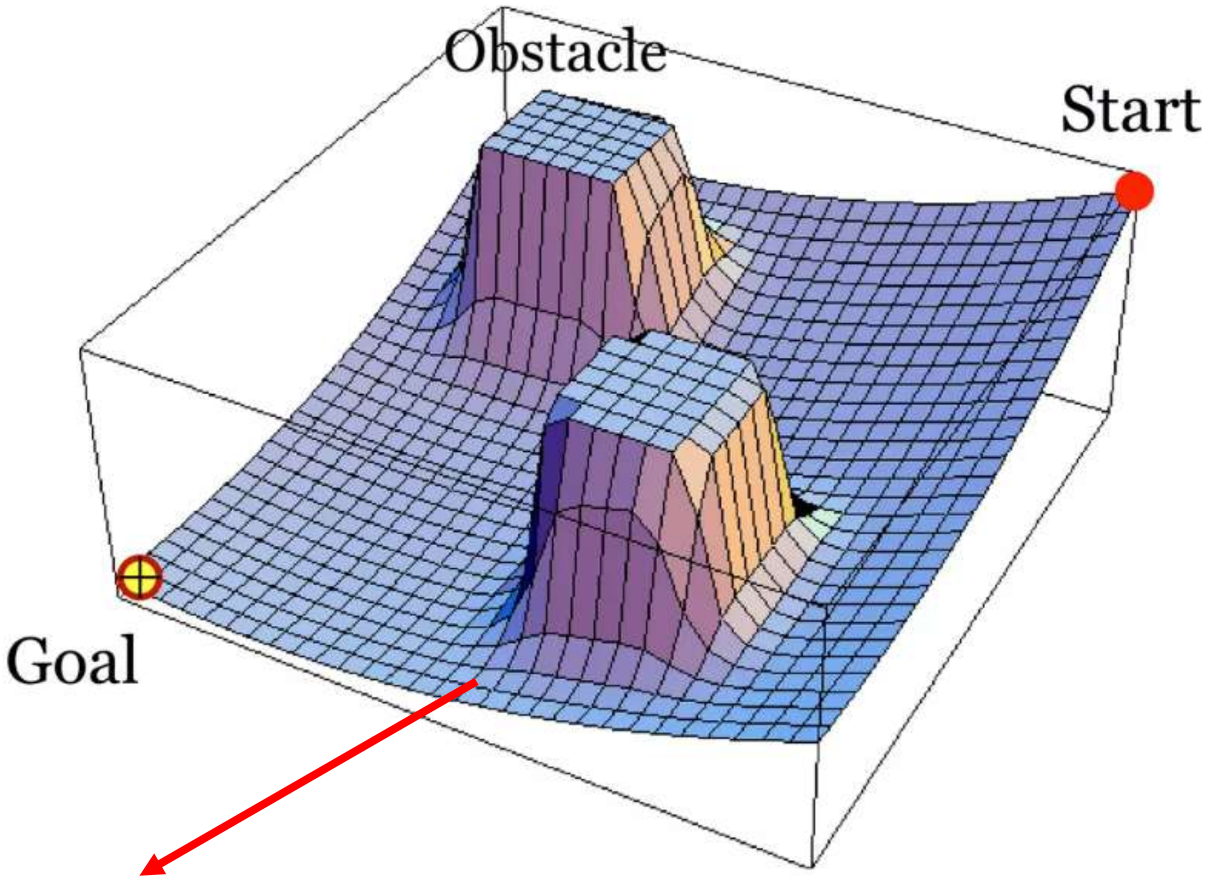
Today's Agenda

- **Recap of sampling-based approach (~10)**
- **Recap of optimization-based approach (~20)**
- **Drawback of sampling and optimization (~5)**
- **Recap of perception-action loop (~2)**
- **Learning-based motion planning (~5)**
- **Imitation learning (~20)**
- **Reinforcement learning (~10)**

Recap of optimization-based approach

Can we develop a motion planner that relies on cost function instead?

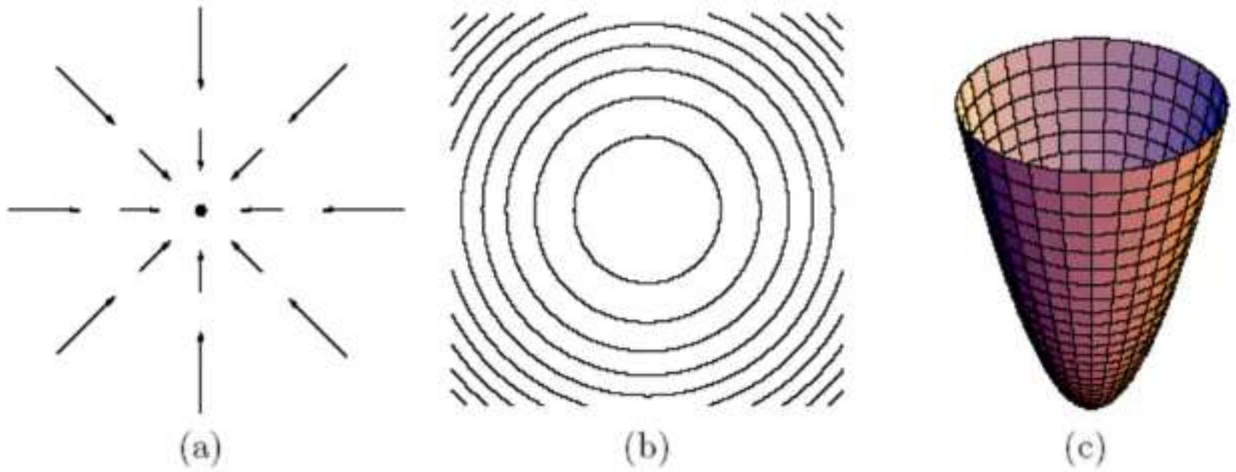
Potential field method



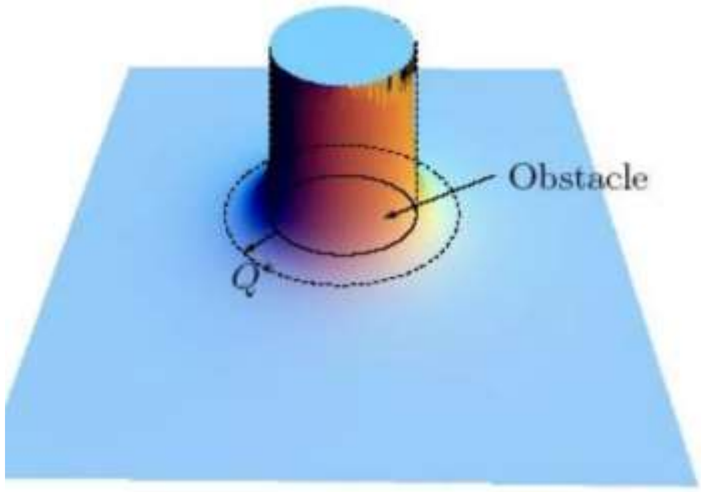
Can we create such a cost function?

Potential field method

Attraction



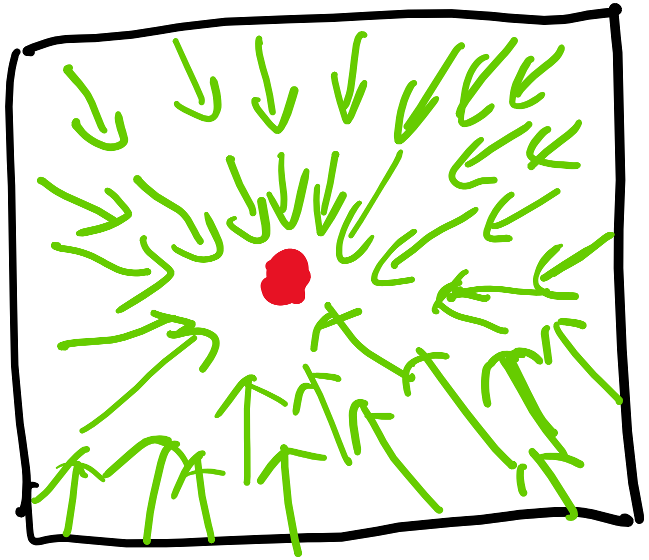
Repulsion



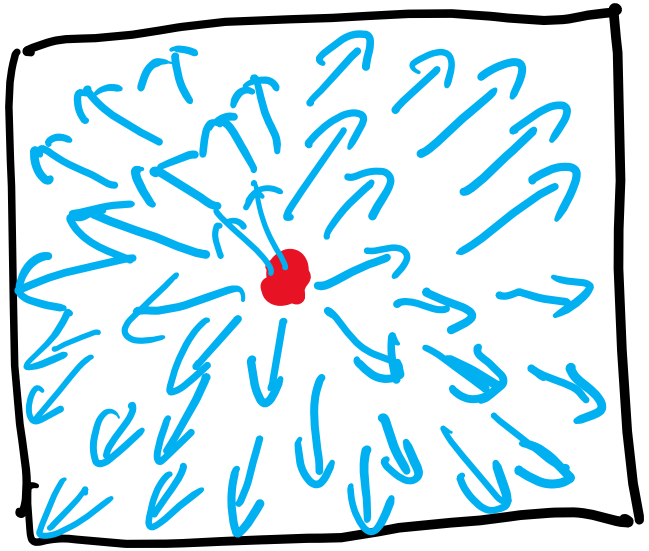
Minimize the cost function

Potential field method

Attraction

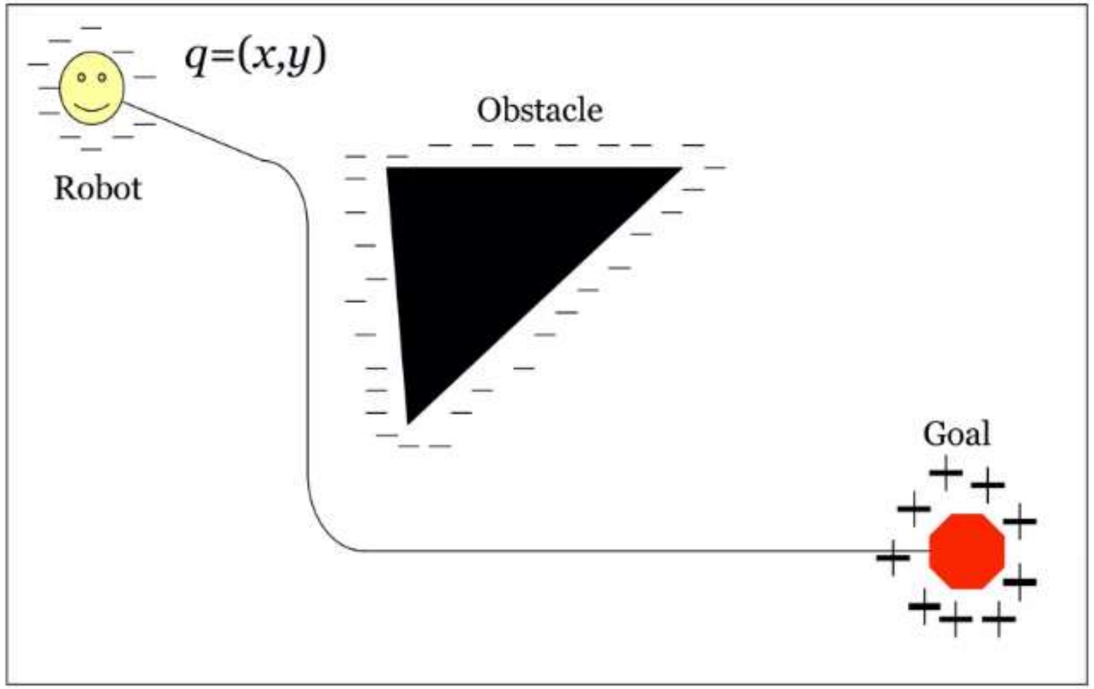


Repulsion



Gradient

Cost function as potential



differential potential:

$U(q)$

artifial force:

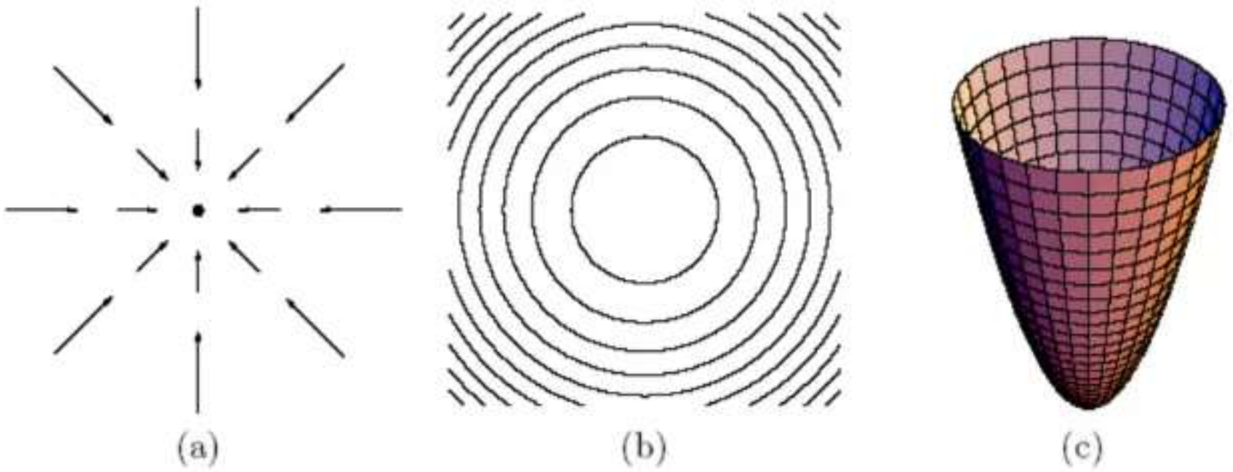
$F(q) = -\nabla U(q)$

gradient

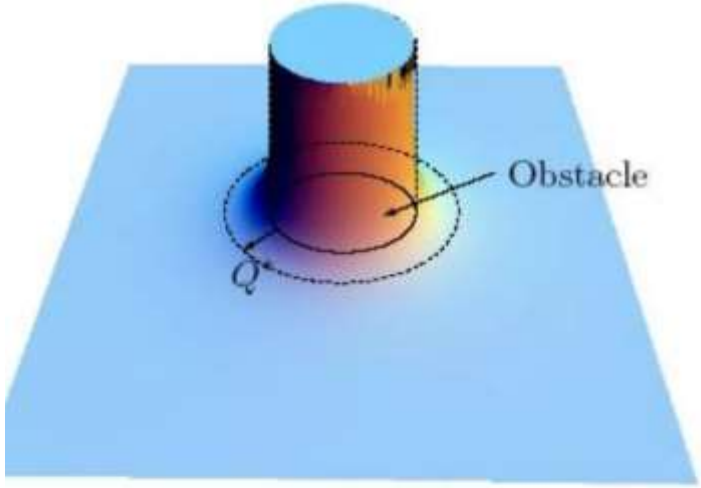
$$\nabla U(q) = \begin{bmatrix} \frac{\partial U(q)}{\partial x} \\ \frac{\partial U(q)}{\partial y} \end{bmatrix}$$

Potential field method

Attraction



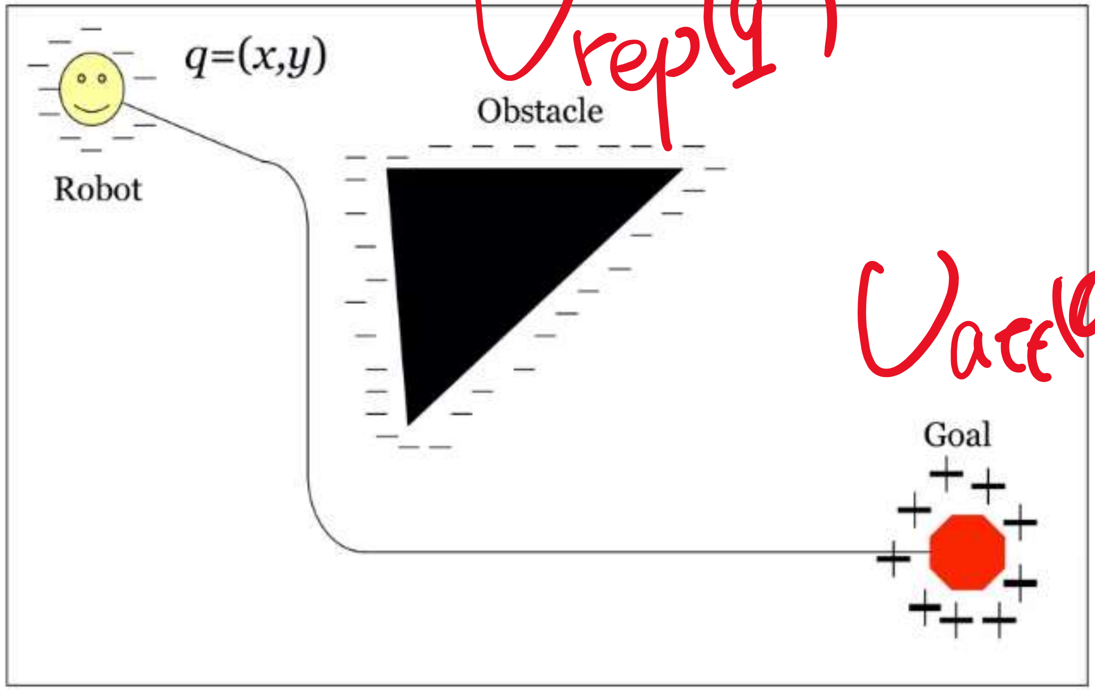
Repulsion



$U_{att}(q)$

$U_{rep}(q)$

Potential field method



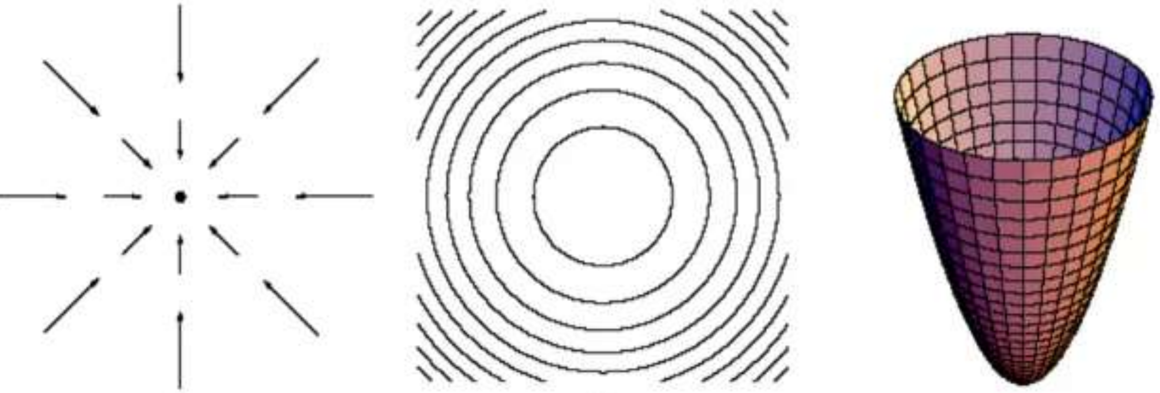
$$U(q) = U_{att}(q) + U_{rep}(q)$$

$U_{att}(q) \to$ move to the goal

$U_{rep}(q) \to$ avoid obstacles.

Potential field method

attractive potential.


(a) (b) (c)

examples:

quadratic potential:

$$U_{att}(q) = \frac{1}{2} K_{att} d^2_{goal}(q)$$

K_{att}: R^+, positive scaling param

$$d_{goal} = ||q - q_{goal}||$$

Potential field method

attractive potential

★ $\boxed{\text{differentiable}}$

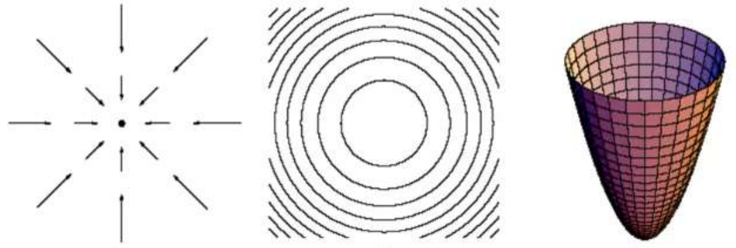
$$U_{att}(q) = \frac{1}{2} K_{att}\, d_{goal}^2(q)$$

force

$$★\ F_{att}(q) = -\nabla U_{att}(q) = -K_{att}\, d_{goal} \nabla d_{goal} = -K_{att}(q - q_{goal})$$

converge linear towards the goal

$$d_{goal} = \| q - q_{goal} \|$$



(a) (b) (c)

Potential field method

repulsive potential.

key idea: generate a force away from all known obstacles

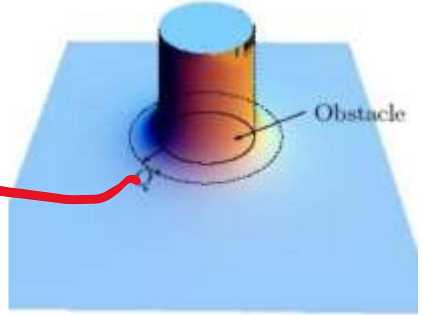
① very strong: close

② zero: far away.

$$U_{rep}(q)=\begin{cases}\frac{1}{2}k_{rep}\cdot\left(\frac{1}{d_{obj}(q)}-\frac{1}{Q^*}\right)^2 & d_{obj}(q)\le Q^*\\ 0 & d_{obj}(q)>Q^*\end{cases}$$

where

- k_{rep} is again a scaling factor,
- d_{obj} is the minimal distance from q to the object and
- Q^* is the distance of influence of the object.



Potential field method

repulsive potential.

$$U_{rep}(q)=\begin{cases}\frac{1}{2}k_{rep}\cdot\left(\frac{1}{d_{obj}(q)}-\frac{1}{Q^*}\right)^2 & d_{obj}(q)\le Q^* \\ 0 & d_{obj}(q)>Q^*\end{cases}$$

where

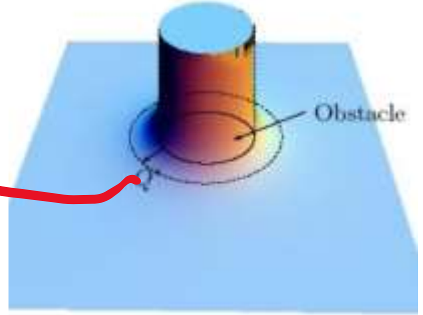
- ◇ k_{rep} is again a scaling factor,
- ◇ d_{obj} is the minimal distance from q to the object and
- ◇ Q^* is the distance of influence of the object.

$d_{obj}(q)\to 0$, $U_{rep}\to +\infty$

?

$d_{obj}=Q^*$, $U_{rep}\to 0$

$d_{obj}>Q^*$ $U_{rep}=0$



Potential field method

repulsive force.

$$U_{rep}(q)=\begin{cases}\frac{1}{2}k_{rep}\cdot\left(\frac{1}{d_{obj}(q)}-\frac{1}{Q^*}\right)^2 & d_{obj}(q)\le Q^*\\ 0 & d_{obj}(q)>Q^*\end{cases} \Rightarrow F_{rep}(q)=-\nabla U_{rep}(q)=\begin{cases}k_{rep}\cdot\left(\frac{1}{d_{obj}(q)}-\frac{1}{Q^*}\right)\cdot\frac{1}{d_{obj}^2}.\nabla d_{obj} & d_{obj}(q)\le Q^*\\ 0 & d_{obj}(q)>Q^*\end{cases}$$

$$\nabla d_{obj}=\begin{bmatrix}\frac{\partial d_{obj}}{\partial x}\\ \frac{\partial d_{obj}}{\partial y}\end{bmatrix}$$

$F_{rep}(q) \nearrow$: when $d_{obj} \downarrow$

where

- ◇ k_{rep} is again a scaling factor,
- ◇ d_{obj} is the minimal distance from q to the object and
- ◇ Q^* is the distance of influence of the object.

Potential field method

repulsive force.

$$U_{rep}(q)=\begin{cases}\frac{1}{2}k_{rep}\cdot\left(\frac{1}{d_{obj}(q)}-\frac{1}{Q^*}\right)^2 & d_{obj}(q)\le Q^* \\ 0 & d_{obj}(q)>Q^*\end{cases}$$

$\Rightarrow$

$$F_{rep}(q)=-\nabla U_{rep}(q)=\begin{cases}k_{rep}\cdot\left(\frac{1}{d_{obj}(q)}-\frac{1}{Q^*}\right)\cdot\frac{1}{d_{obj}^2}.\nabla d_{obj} & d_{obj}(q)\le Q^* \\ 0 & d_{obj}(q)>Q^*\end{cases}$$

where

- ◇ k_{rep} is again a scaling factor,
- ◇ d_{obj} is the minimal distance from q to the object and
- ◇ Q^* is the distance of influence of the object.

$\Downarrow$

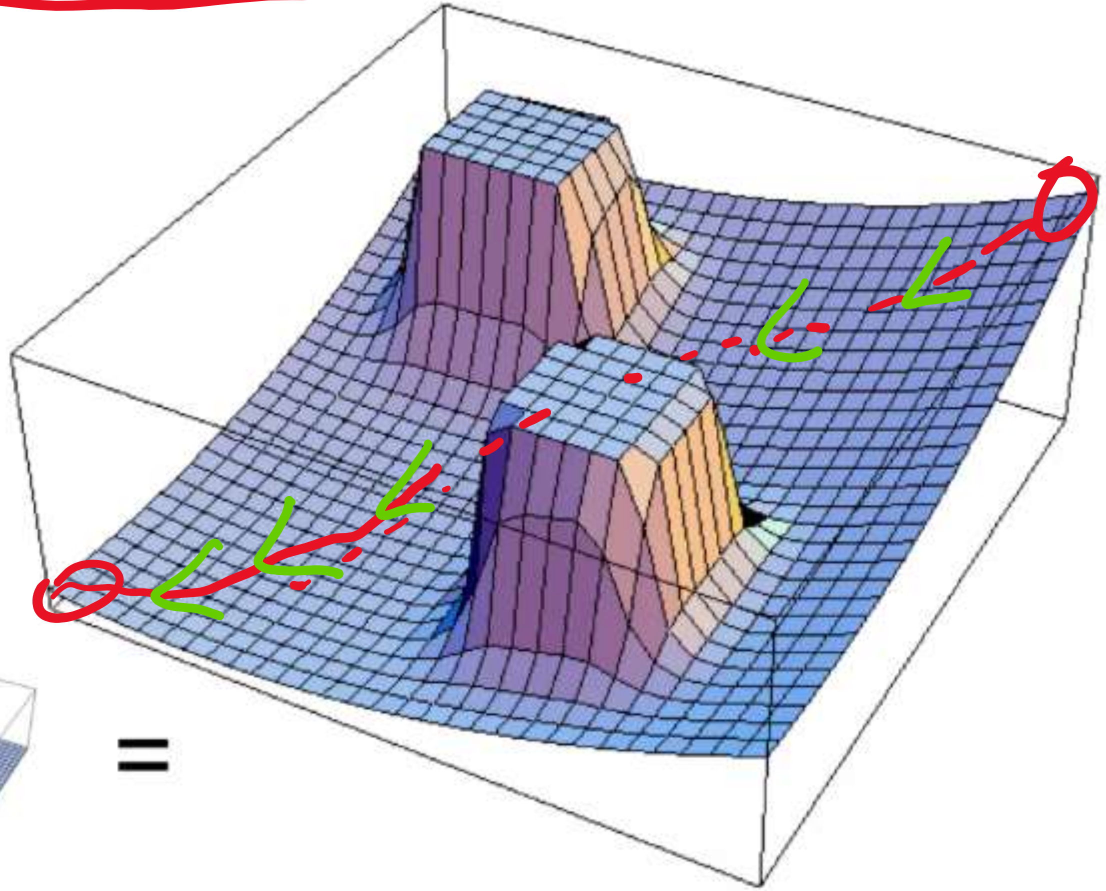
How to compute d_{obj}?

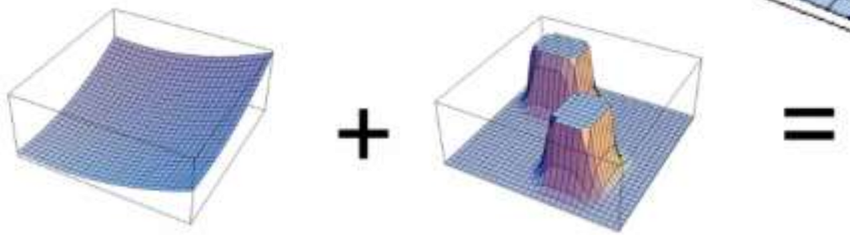
geometry

Potential field method

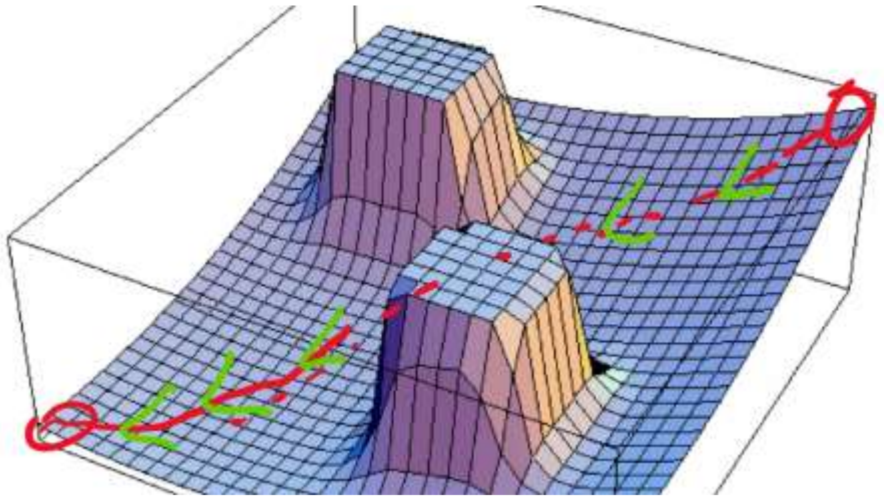
$$F(q) = F_{att}(q) + F_{rep}(q) = -\nabla U(q)$$

A first-order optimization algorithm such as **gradient descent** (also known as **steepest descent**) can be used to minimize this function by taking steps proportional to the negative of the gradient.





Potential field method

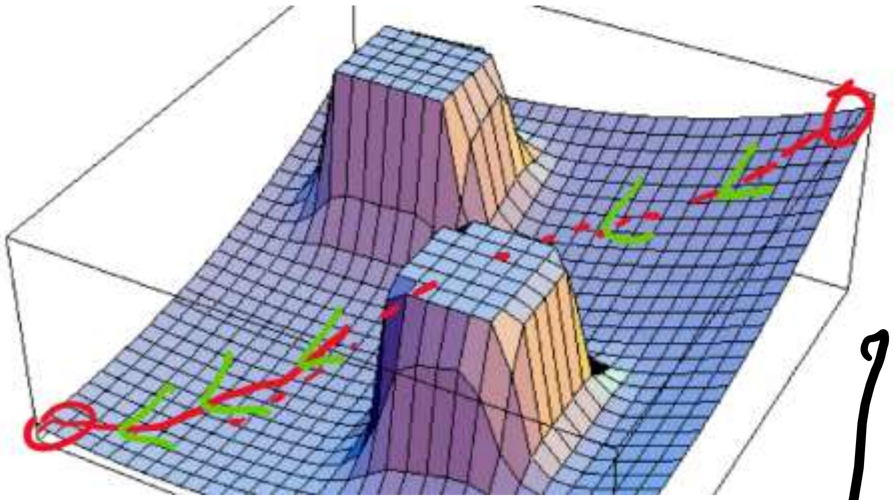


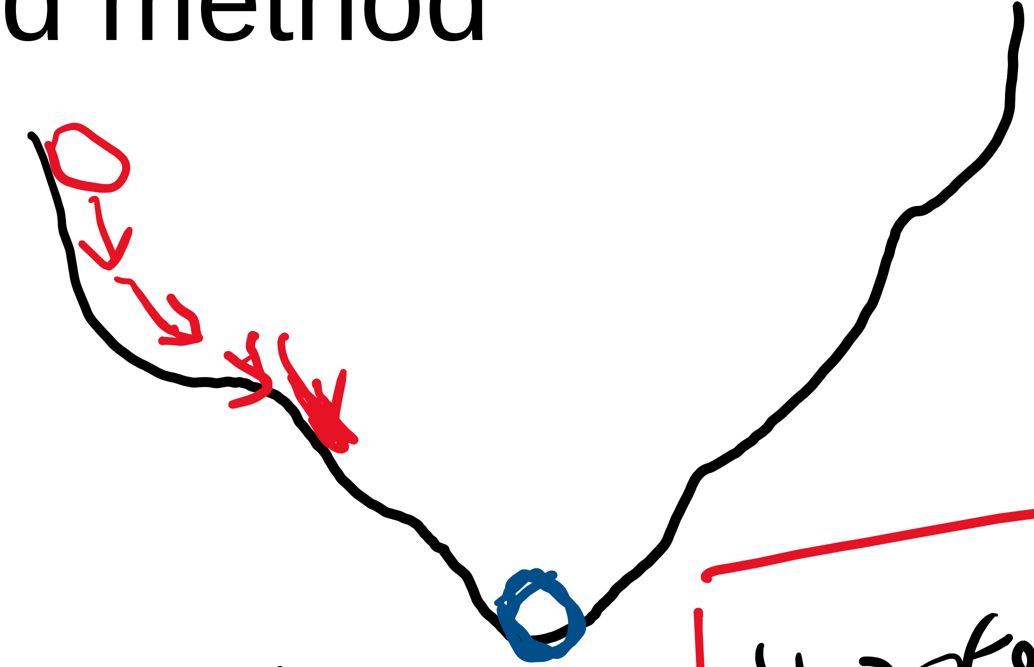
problems:

- **Local minima**
- **Hand crafted potential function**
- **Hard to compute distance**
- **Minimal distance may not be continuous**
- **No passage between closely spaces obstacles**
- **Oscillation**

Potential field method

Rective control:

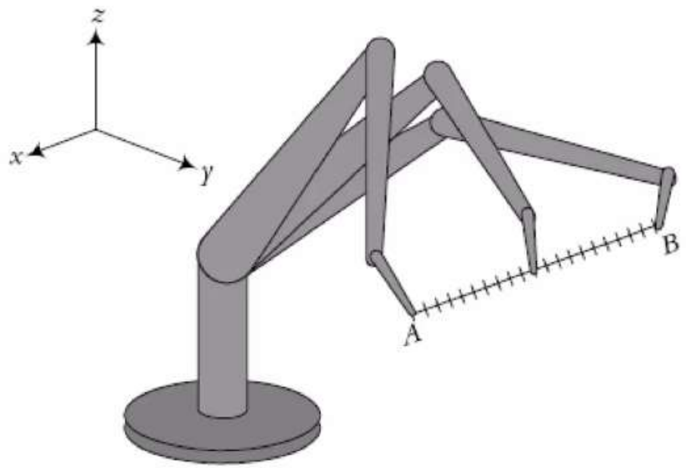




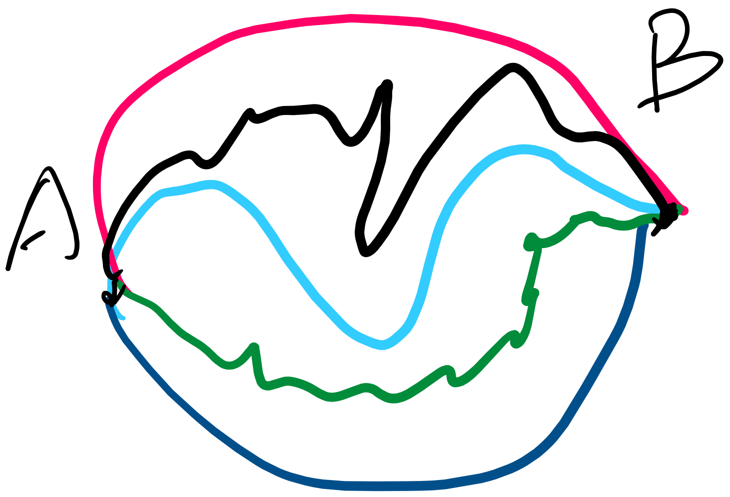
Open question:

Can we design a "potential function" that can globally converge to a desired point?

Trajectory planning (Cartesian space)

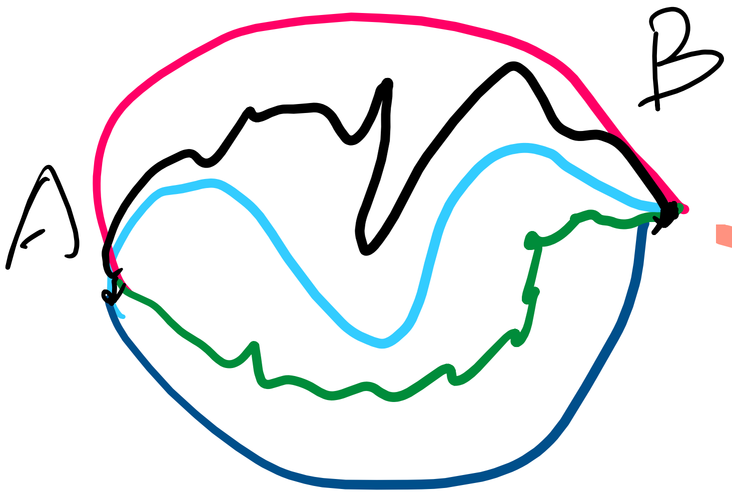


Sequential motions of a robot to follow a straight line

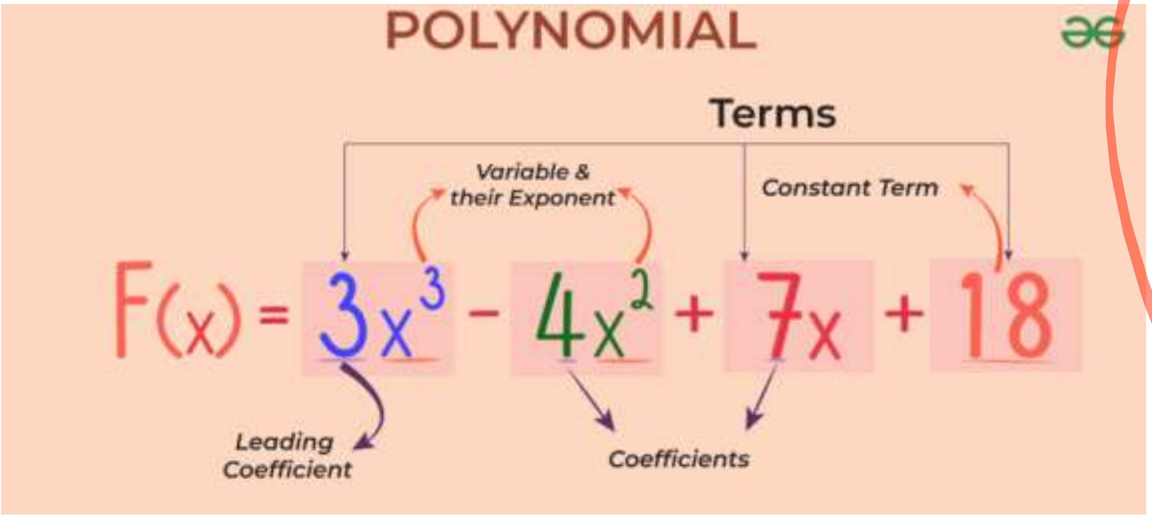


- **Cartesian space trajectories are very to visualize**
- **Computationally expensive: IK at each intermediate point**

Trajectory planning (Cartesian space)



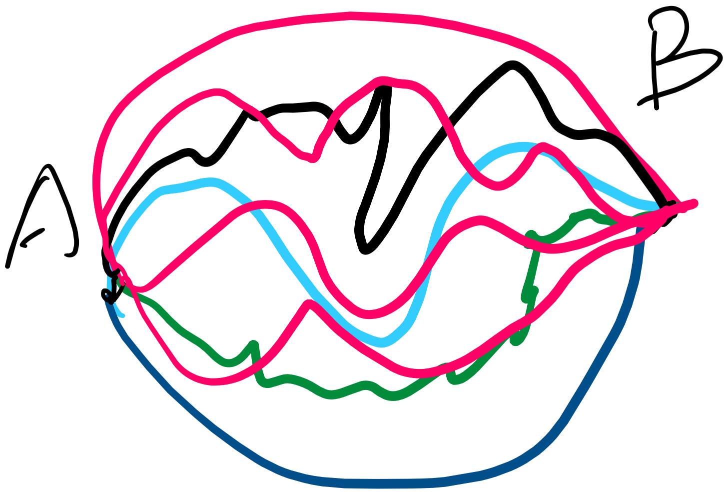
polynomial



any other form of trajectory? ✓

Trajectory planning

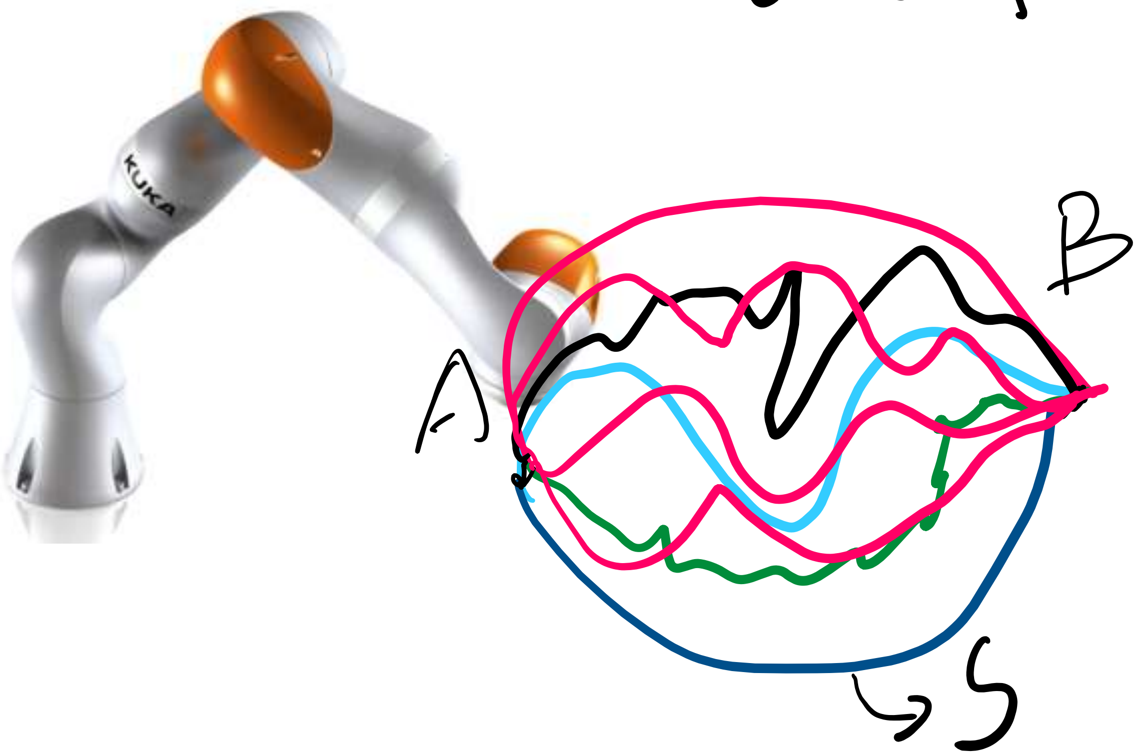
- **The key idea of trajectory planning is to use some form of trj representation to choose the proper trj profile (polynomial…)**
- **This process can be applied in both joint space and Cartesian space**
- **Have more flexibility than sampling-base methods**



special form

Trajectory optimization

Cost function: $V: S \to R^+$



- path length
- efficiency
- obstacle avoidance
- uncertainty reduction
- predictability
- legibility / intent expression
- human comfort
- naturalness

difficult to represent.

Trajectory optimization

cost function: $U: S \rightarrow R^+$

trj optimization:

$$S^* = \arg\min_{S \in E} U[S]$$

$$\text{s.t.} \begin{cases} S(0) = q_s \\ S(T) = q_g \\ \text{other constraints} \end{cases}$$

- path length
- efficiency
- obstacle avoidance
- uncertainty reduction
- predictability
- legibility / intent expression
- human comfort
- naturalness

difficult to represent.

Trajectory optimization

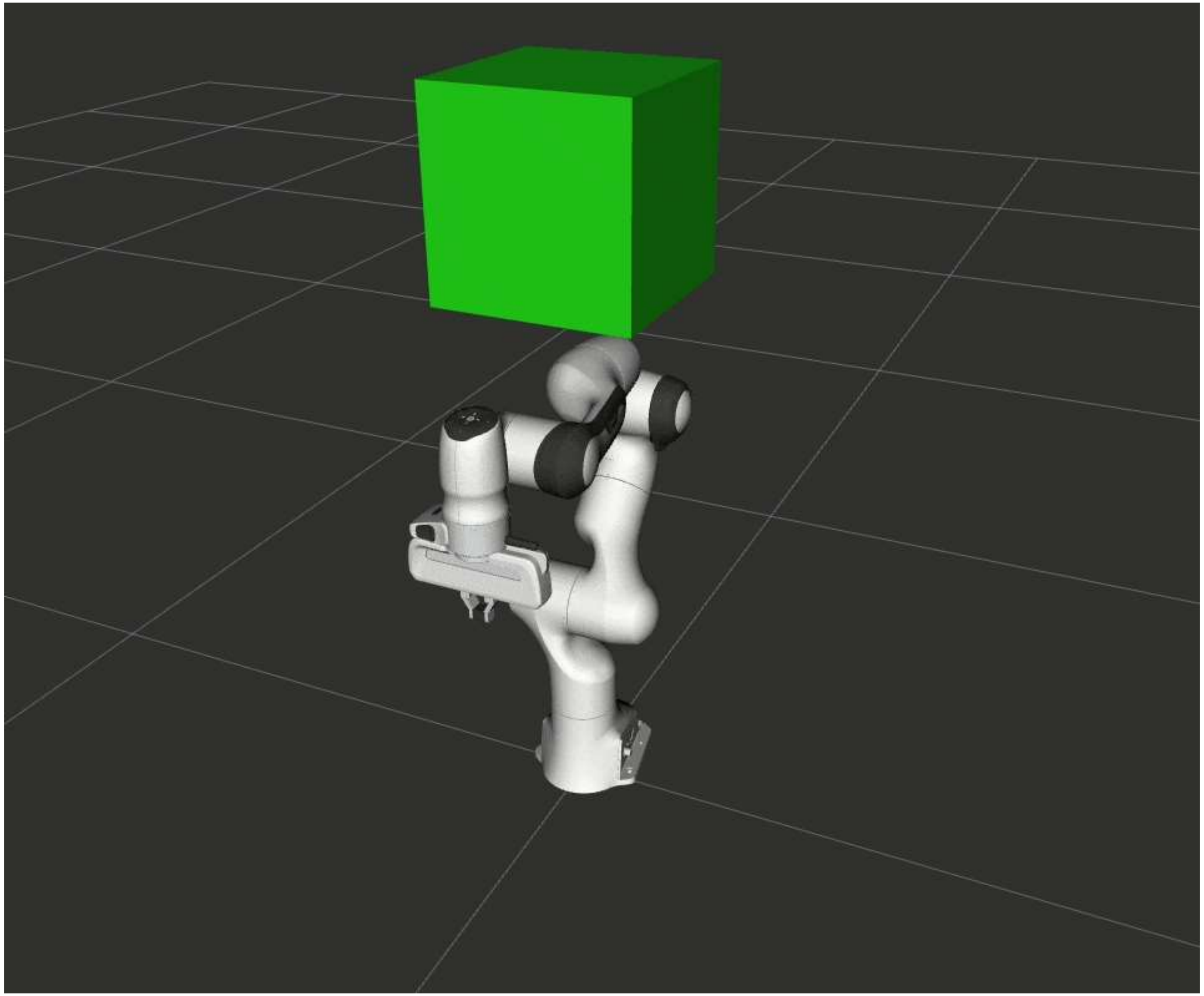
- **Optimization-based motion planning approaches, such as Nonlinear Programming (NLP) and Mixed-Integer Programming (MIP), solve optimization problems, and find solutions using gradient descent while satisfying constraints.**

- **For instance, CHOMP optimizes a cost functional using covariant gradient descent while TrajOpt solves a sequential convex optimization and performs convex collision checking.**

- **Various tasks including navigation, grasping, manipulation, collision-avoidance, running, cooking, and flying under various conditions.**

- **Local optimal (a general problem for nonlinear optimization)**

Trajectory optimization



Trajectory Optimization

$$\min_{\theta_{1:T}} \quad \sum_t \|\theta_{t+1} - \theta_t\|^2 + \text{other costs}$$

$$\text{subject to} \quad \theta_0 = \text{start state}, \quad \theta_T \text{ in goal set}$$

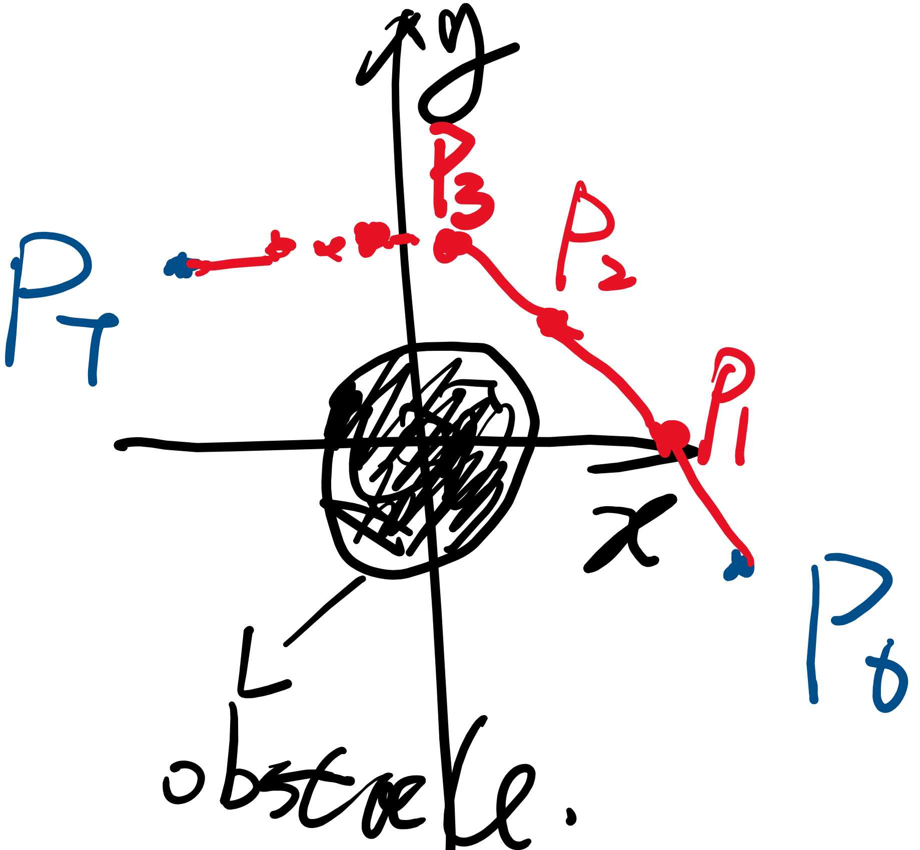
joint limits

for all robot parts, for all obstacles:
no collision → *non-convex*

Solution method: sequential convex optimization

https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/slides/Lec10-motion-planning.pdf

Trajectory optimization (Example)



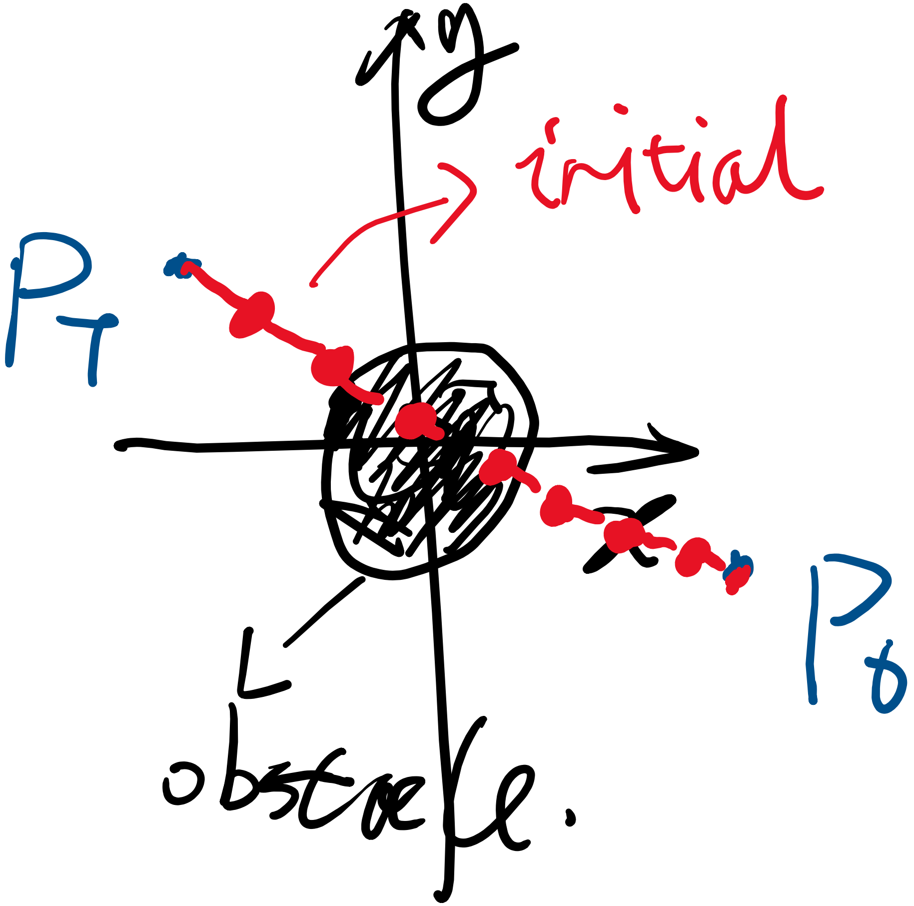
$x^2+y^2 \leq 1$

$$\min_{P_{0 \ldots T}} : \sum_n \|P_n - P_{n-1}\|^2 + \text{other cost}$$

$$\text{s.t.}: \quad \|P_n\| \geq 1 + \delta_{safe}.$$

try to play with this example

Trajectory optimization (Example)

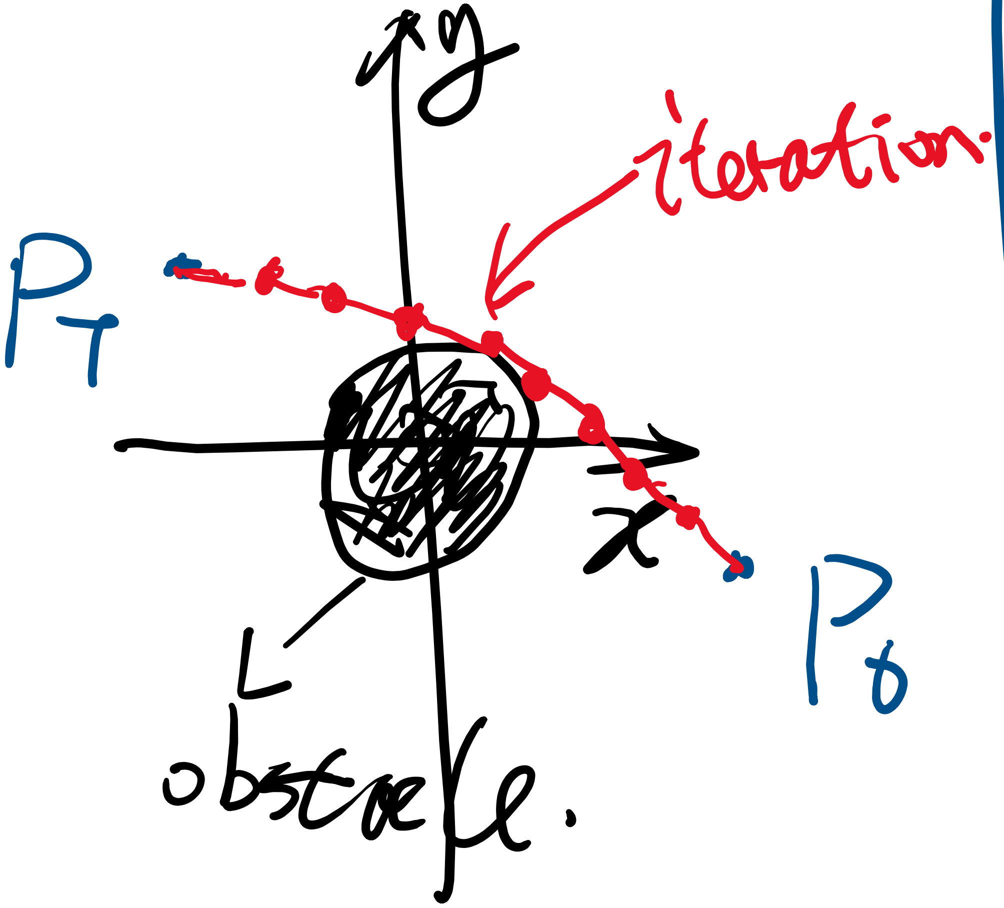


$$x^2+y^2\leq 1$$

$$\min_{P_{0\dots T}}: \sum_n \|P_n - P_{n-1}\|^2 + \text{other cost}$$

$$\text{s.t.}: \|P_n\| \geq 1 + \delta_{safe}.$$

Trajectory optimization (Example)



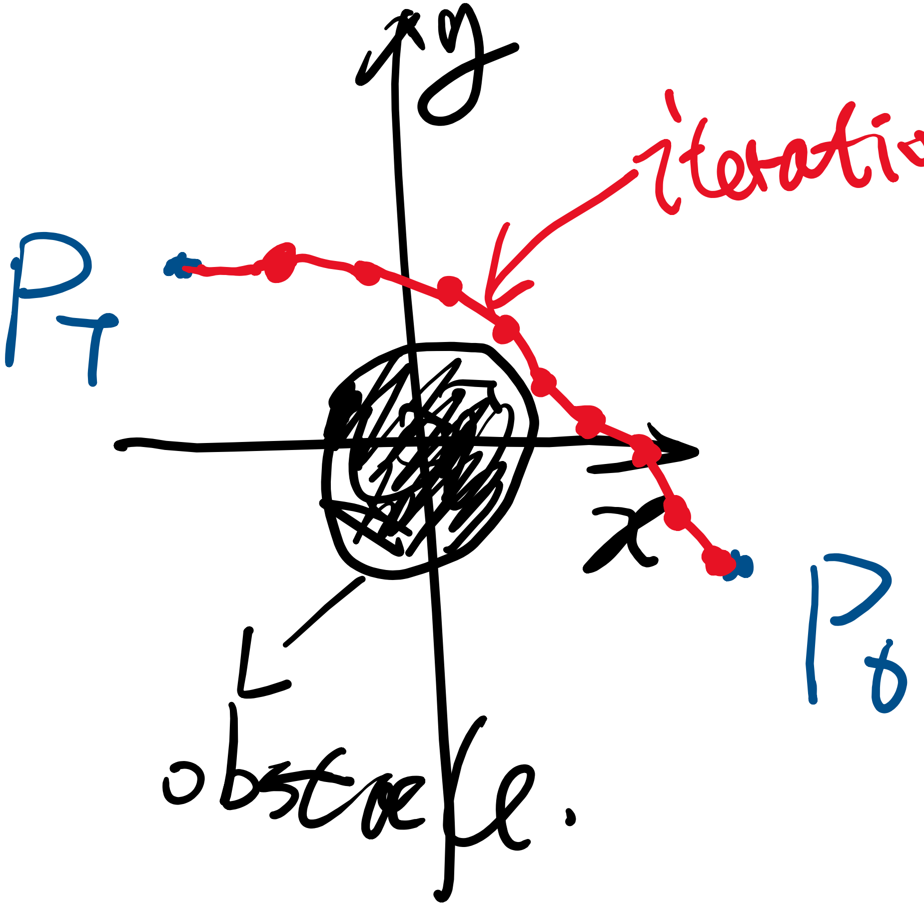
$x^2+y^2 \leq 1$

$$\min_{P_{0 \dots T}} : \sum_n \|P_n - P_{n-1}\|^2 + \text{other cost}$$

s.t.:

$$\|P_n\| \geq 1 + \delta_{safe}.$$

Trajectory optimization (Example)

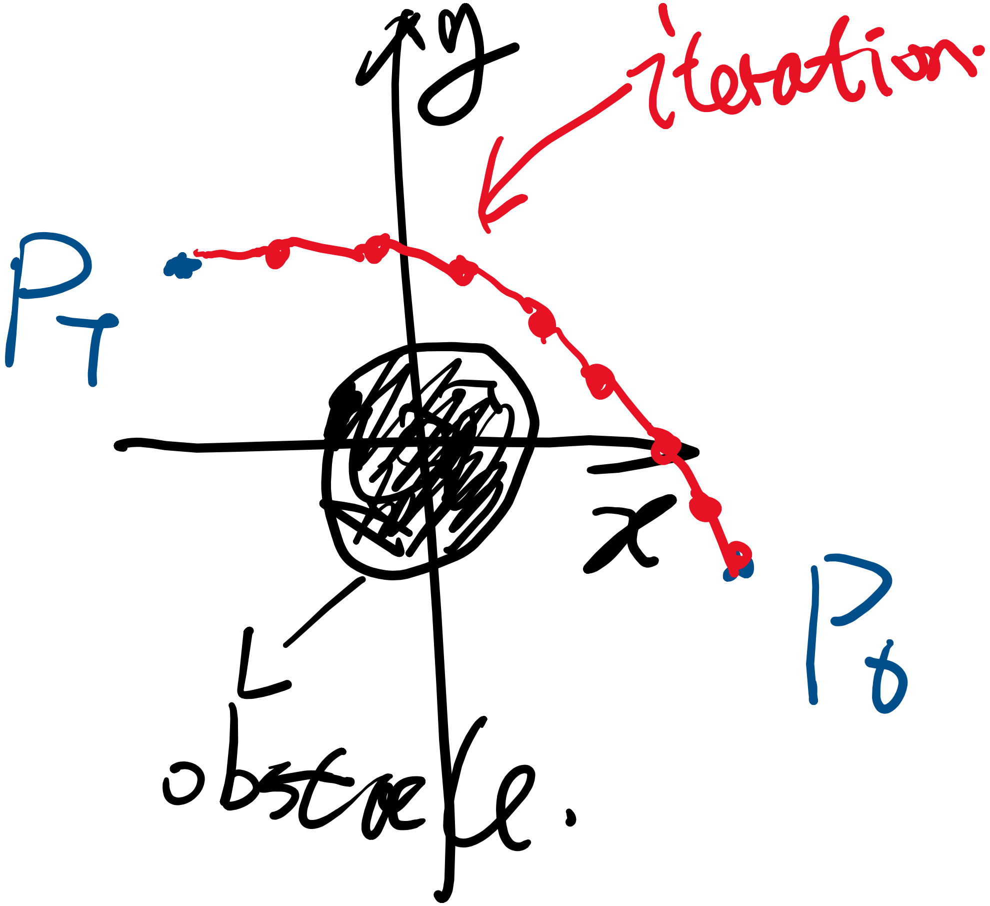


$$x^2+y^2\le 1$$

$$\min_{P_{0\ldots T}} : \sum_n \|P_n - P_{n-1}\|^2 + \text{other cost}$$

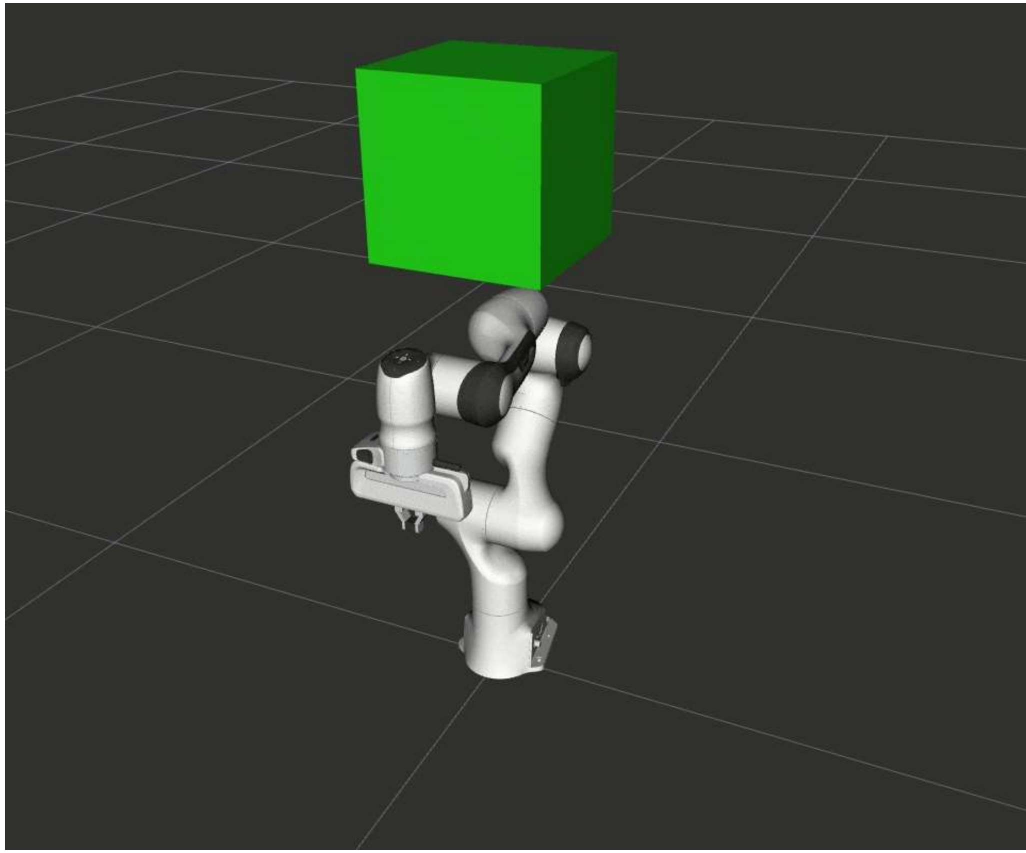
$$\text{s.t.}: \quad \|P_n\| \ge 1 + \delta_{safe}.$$

Trajectory optimization (Example)



$$x^2+y^2 \le 1$$

$\Rightarrow$

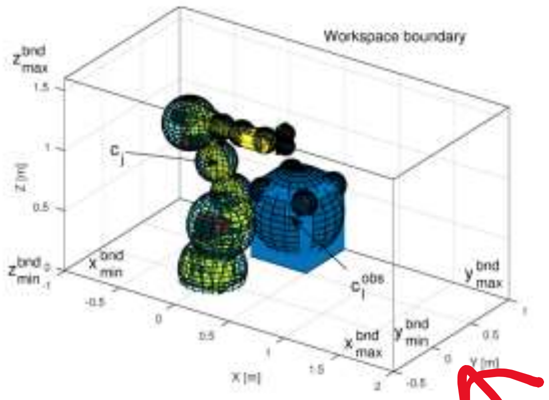


more complex constraints

Trajectory optimization

Efficient Trajectory Optimization for Robot Motion Planning

Yu Zhao, Hsien-Chung Lin, and Masayoshi Tomizuka



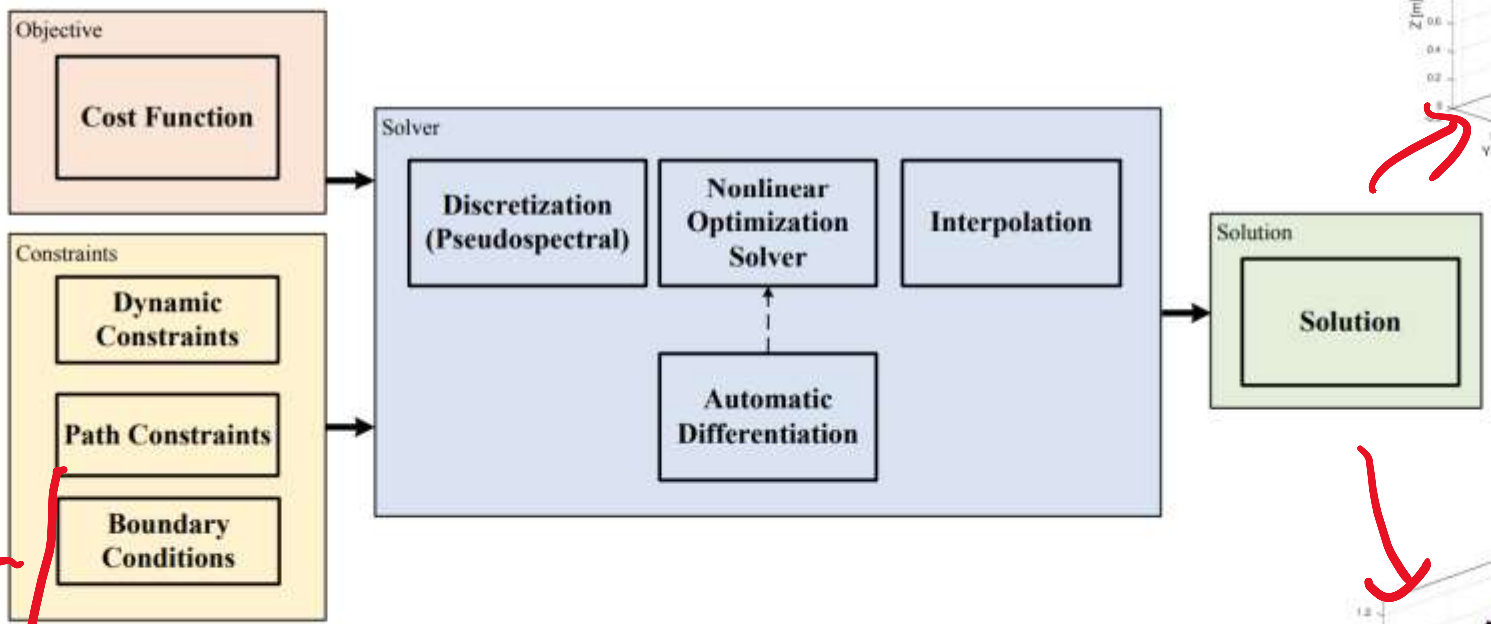
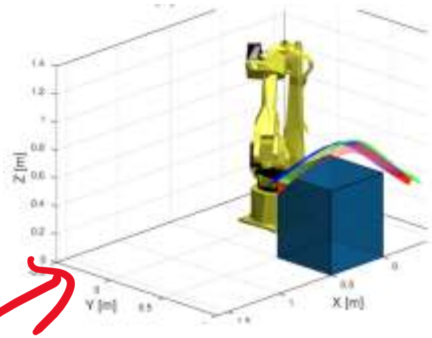
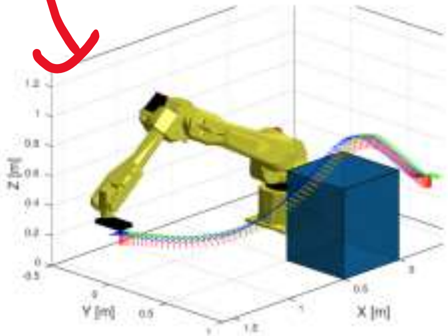


Fig. 2: Efficient numerical method for trajectory optimization



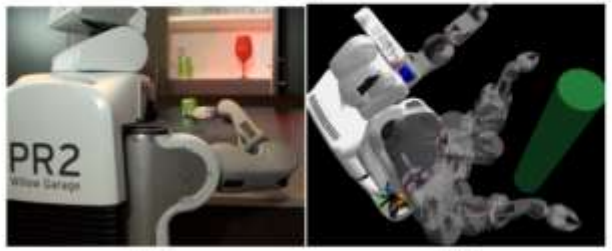


Position bounds:	$\boldsymbol{q}_{min} \le \boldsymbol{q}(t) \le \boldsymbol{q}_{max}$
Velocity bounds:	$\dot{\boldsymbol{q}}_{min} \le \dot{\boldsymbol{q}}(t) \le \dot{\boldsymbol{q}}_{max}$
Torque bounds:	$\boldsymbol{\tau}_{\min} \le \boldsymbol{\tau}(t) \le \boldsymbol{\tau}_{\max}$
Torque rate bounds:	$\dot{\boldsymbol{\tau}}_{\min} \le \dot{\boldsymbol{\tau}}(t) \le \dot{\boldsymbol{\tau}}_{\max}$

Trajectory optimization

STOMP: Stochastic Trajectory Optimization for Motion Planning

Mrinal Kalakrishnan[1] Sachin Chitta[2] Evangelos Theodorou[1] Peter Pastor[1] Stefan Schaal[1]



(a) (b)

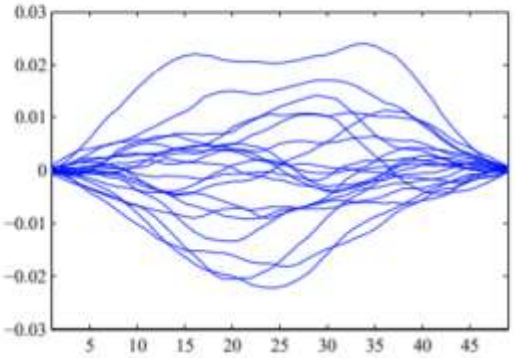
Fig. 1. (a) The Willow Garage PR2 robot manipulating objects in a household environment. (b) Simulation of the PR2 robot avoiding a pole in a torque-optimal fashion.

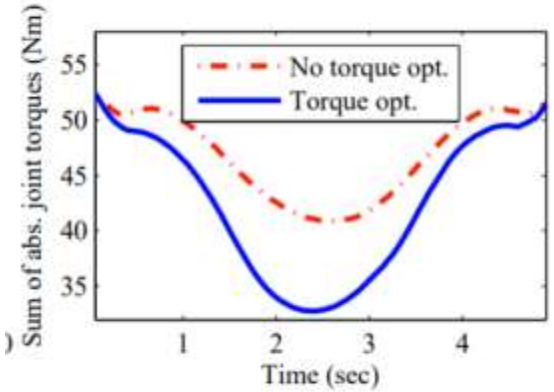
$$\min_{\tilde{\boldsymbol{\theta}}} \mathbb{E}\left[\sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i) + \frac{1}{2}\tilde{\boldsymbol{\theta}}^{\mathrm{T}}\mathbf{R}\tilde{\boldsymbol{\theta}}\right]$$

STOMP is an algorithm that performs local optimization, i.e. it finds a locally optimum trajectory rather than a global one. Hence, performance will vary depending on the initial

sample locally

optimize. locally





Today’s Agenda

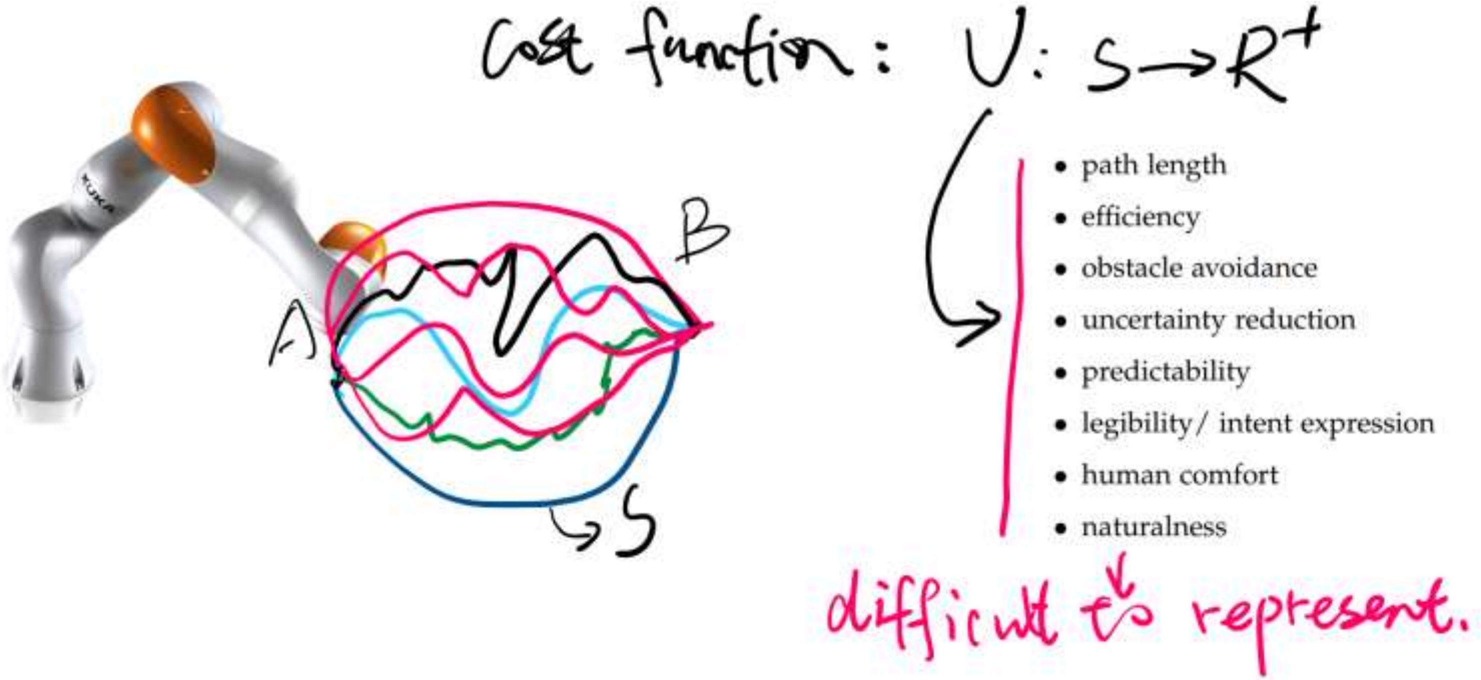
- **Recap of sampling-based approach (~10)**
- **Recap of optimization-based approach (~20)**
- **Drawback of sampling and optimization (~5)**
- **Recap of perception-action loop (~2)**
- **Learning-based motion planning (~5)**
- **Imitation learning (~20)**
- **Reinforcement learning (~10)**

Drawback of sampling and optimization-based approaches

- Flexibility
- Human-like
- Reactive
- Sensory feedback

Drawback of sampling and optimization-based approaches

- **Flexibility**
- Human-like
- Reactive
- Sensory fee



Drawback of sampling and optimization-based approaches

- Flexibility
- **Human-like**
- Reactive
- Sensory feedb

https://www.therobotreport.com/researchers-develop-human-aware-motion-planning-algorithm/

Drawback of sampling and optimization-based approaches

- Flexibility
- **Human-like**
- Reactive
- Sensory feedba

https://www.youtube.com/watch?v=-9JrDMBg2HE&t=38s&ab_channel=MITCSAIL

Drawback of sampling and optimization-based approaches

- Flexibility
- Human-like
- **Reactive**
- Sensory feed

Reactive Human-to-Robot Handovers of Arbitrary Objects

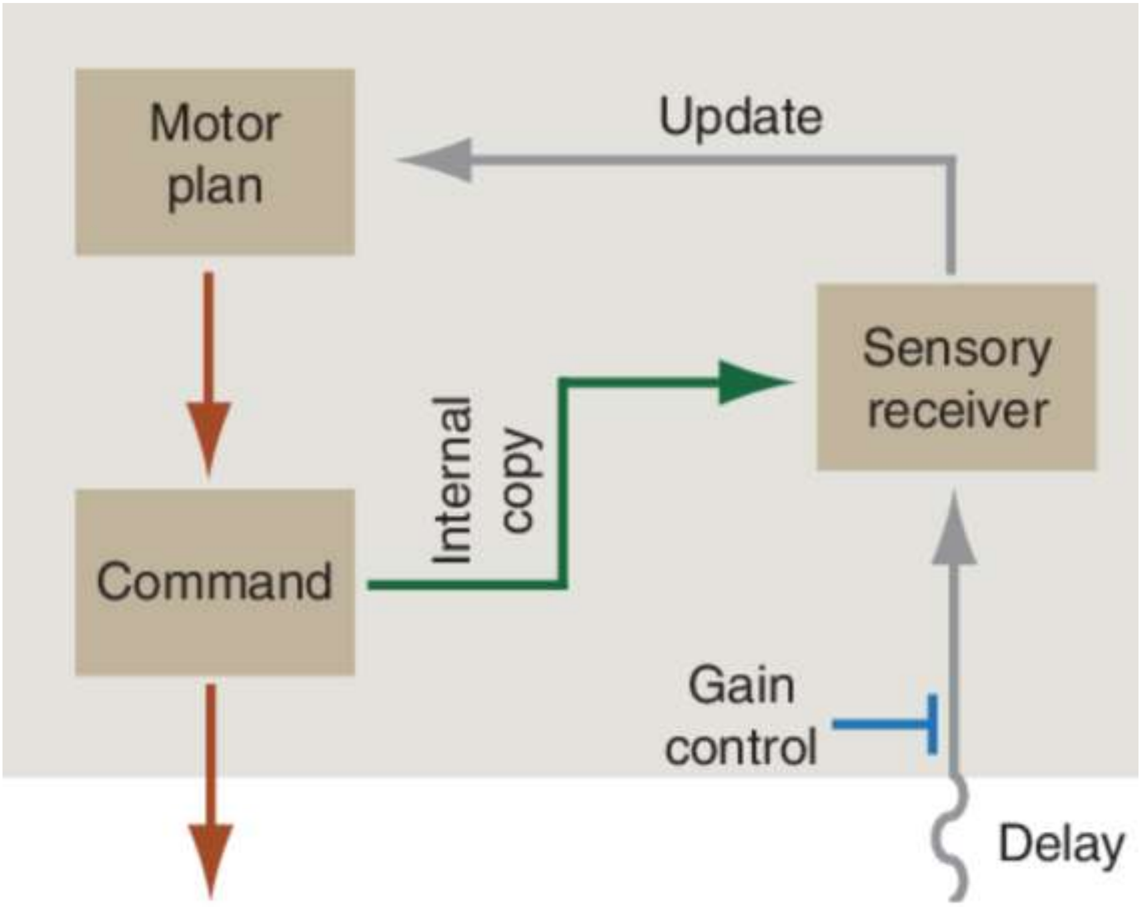
Drawback of sampling and optimization-based approaches

- Flexibility
- Human-like
- **Reactive**
- Sensory feed

https://research.nvidia.com/publication/2021-03_reactive-human-robot-handovers-arbitrary-objects

Drawback of sampling and optimization-based approaches

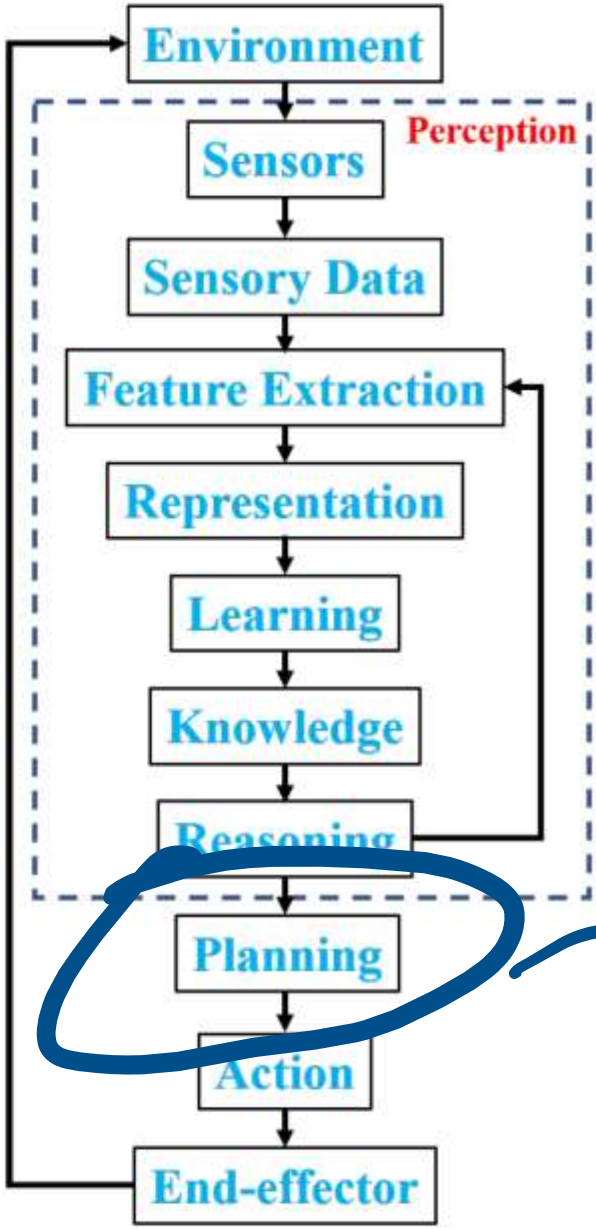
- Flexibility
- Human-like
- Reactive
- **Sensory feedback**



Today’s Agenda

- **Recap of sampling-based approach (~10)**
- **Recap of optimization-based approach (~20)**
- **Drawback of sampling and optimization (~5)**
- **Recap of perception-action loop (~2)**
- **Learning-based motion planning (~5)**
- **Imitation learning (~20)**
- **Reinforcement learning (~10)**

Planning in Robotics



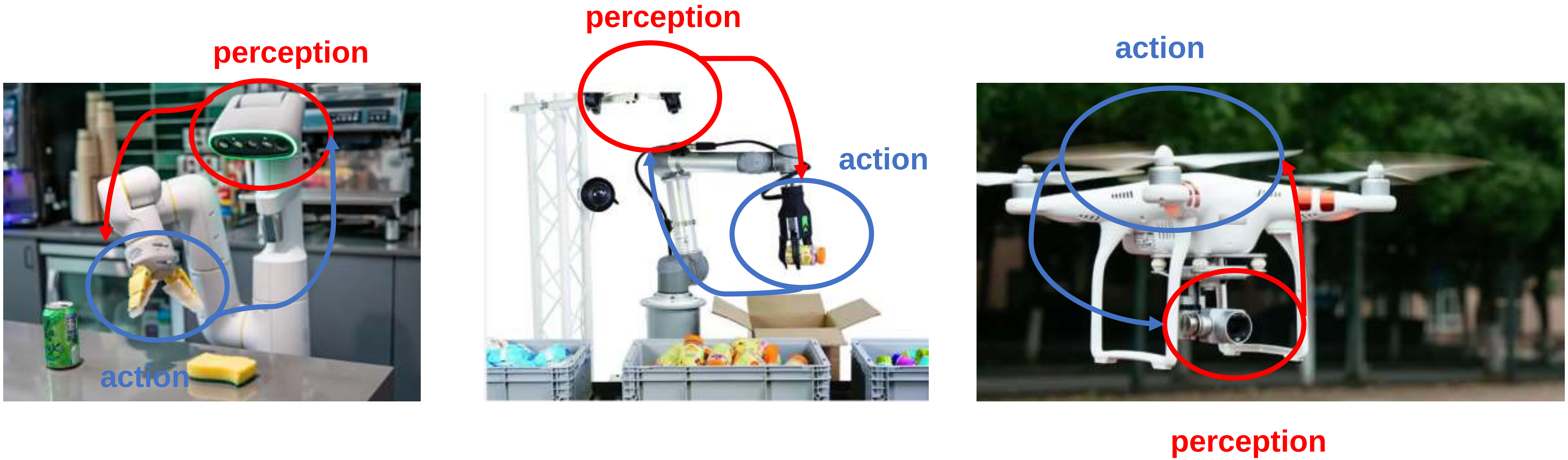
Robotics – Learn the mapping from perception to action

real-time.

fast sampling

fast optimization

Motion Planning in Robotics



Robotics – Learn the mapping from perception to action

↓ not just a $y=f(x)$

Today’s Agenda

- **Recap of sampling-based approach (~10)**
- **Recap of optimization-based approach (~20)**
- **Drawback of sampling and optimization (~5)**
- **Recap of perception-action loop (~2)**
- **Learning-based motion planning (~5)**
- **Imitation learning (~20)**
- **Reinforcement learning (~10)**

Learning

Learning Modes

Explicit Learning:
Reinforcement
Verbal instructions

Implicit Learning:
Observational learning
Imitation learning

Imitation learning

Learning seems to be a negative force in evolution.

How can learning have evolved?

Learning serves as a pacemaker for evolution, when exploratory behavior leads to a breakthrough for the survival of the species, the capacity for that kind of exploratory behavior and the imitation of this act is favored by natural selection.

E. Wilson, *Sociobiology*, Belknap Harvard, 2000

Imitation learning

Biological Inspiration

Imitation Capabilities in Animals

Which species may exhibit imitation is still a main area of discussion and debate

One differentiate "true" imitation from copying (flocking, schooling, following), stimulus enhancement, contagion or emulation

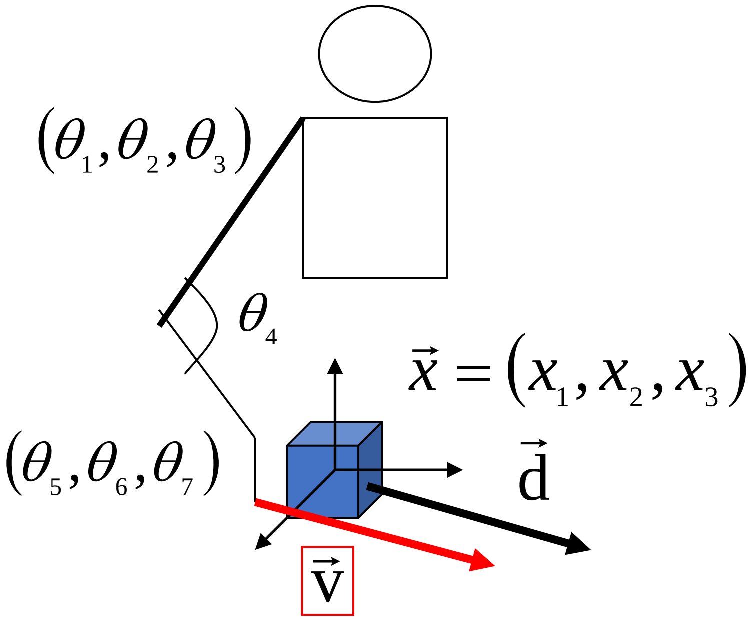
Imitation learning

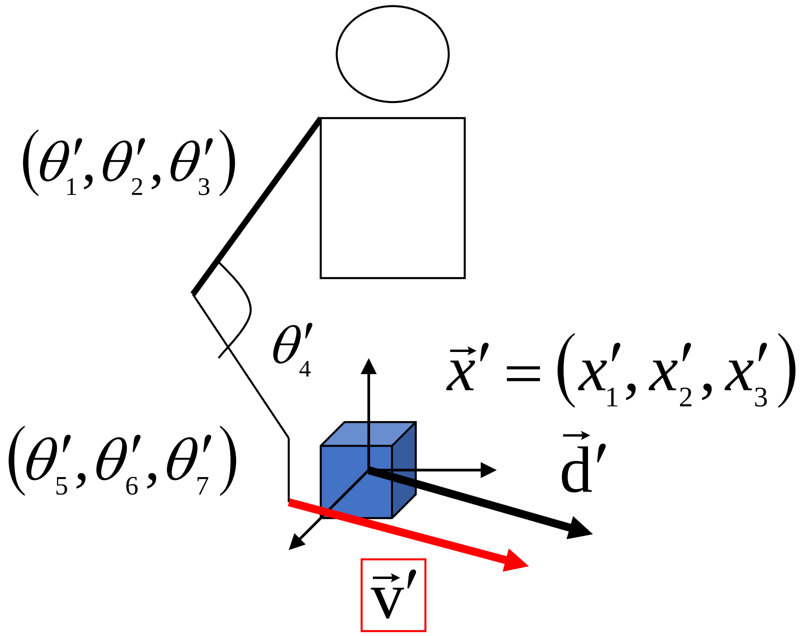
$\vec{x} = \vec{x}'$ Same Object, same target location

$\vec{\mathrm{d}} = \vec{\mathrm{d}}'$ Same direction of motion

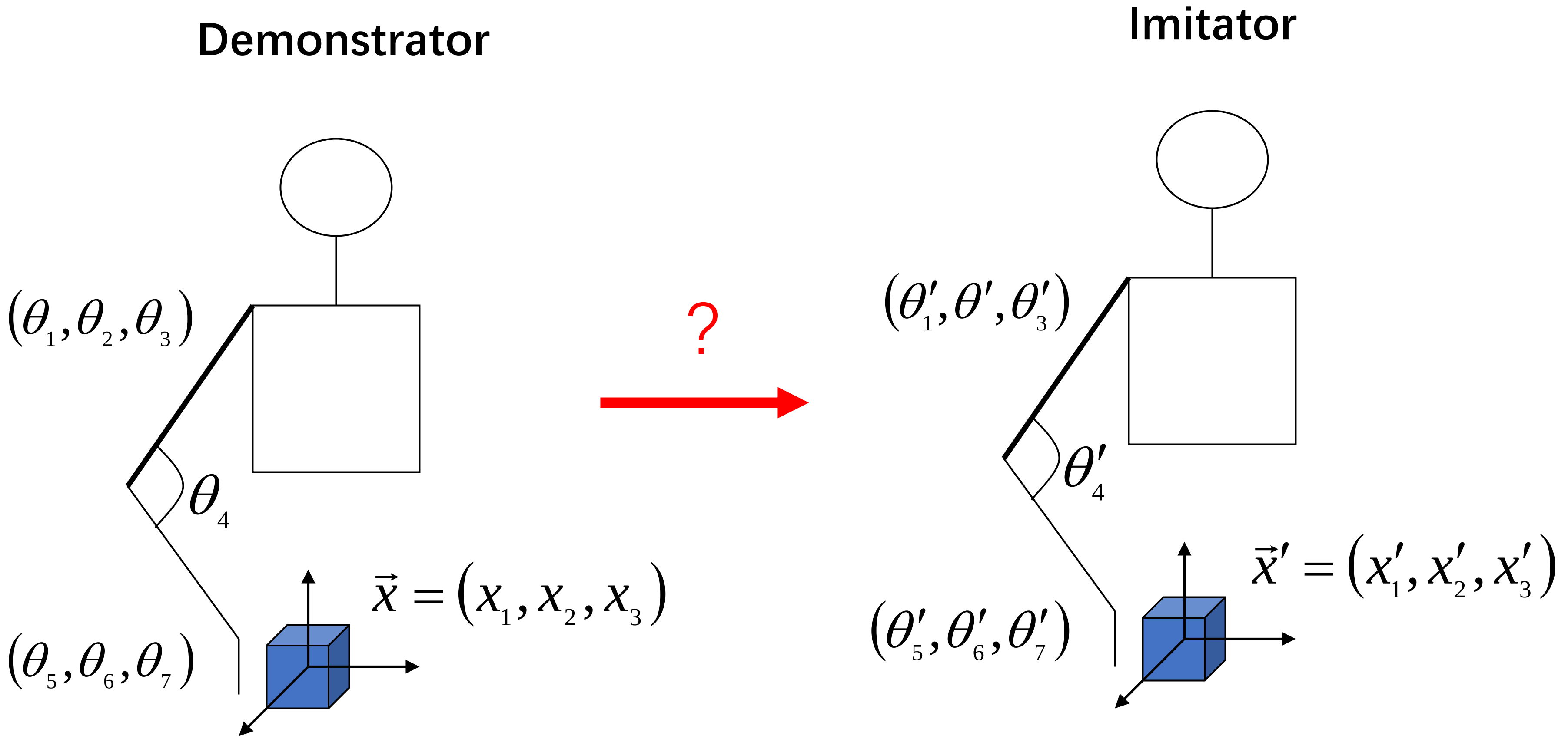
$\vec{\mathrm{v}} = \vec{\mathrm{v}}'$ Same speed, same force

$\vec{\theta} = \vec{\theta}'$ Same posture



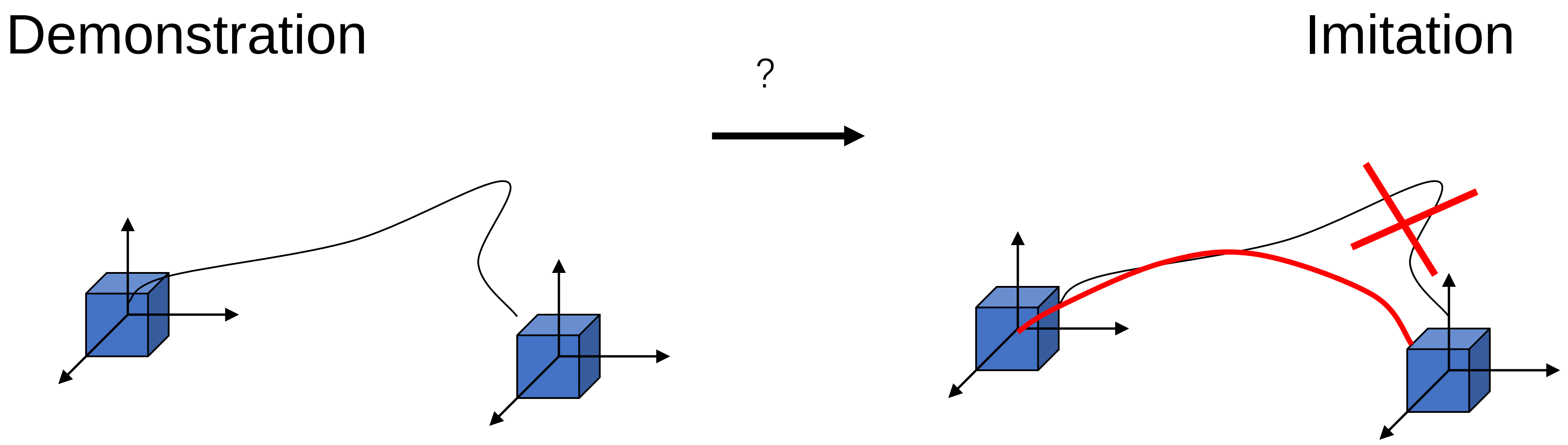


Imitation learning



The Transfer problem

Imitation learning



No solutions (smaller range of motion)

→ Find the closest solution according to a metric

How to Imitate?

The correspondence problem

Imitation learning

Learning What to imitate

Imitation learning – Programming by Demonstration:

- A way to speed up learning, to reduce the search space
- A way to share with robots the same vocabulary of motor skills

Imitation learning

Imitation Learning in Robots
Prof. Aude Billard, lasa.epfl.ch

Granularity

Cognition

How to imitate?

Level of granularity

Level 3: Learning primitives of motion

Level 2: Exact reproduction of trajectories

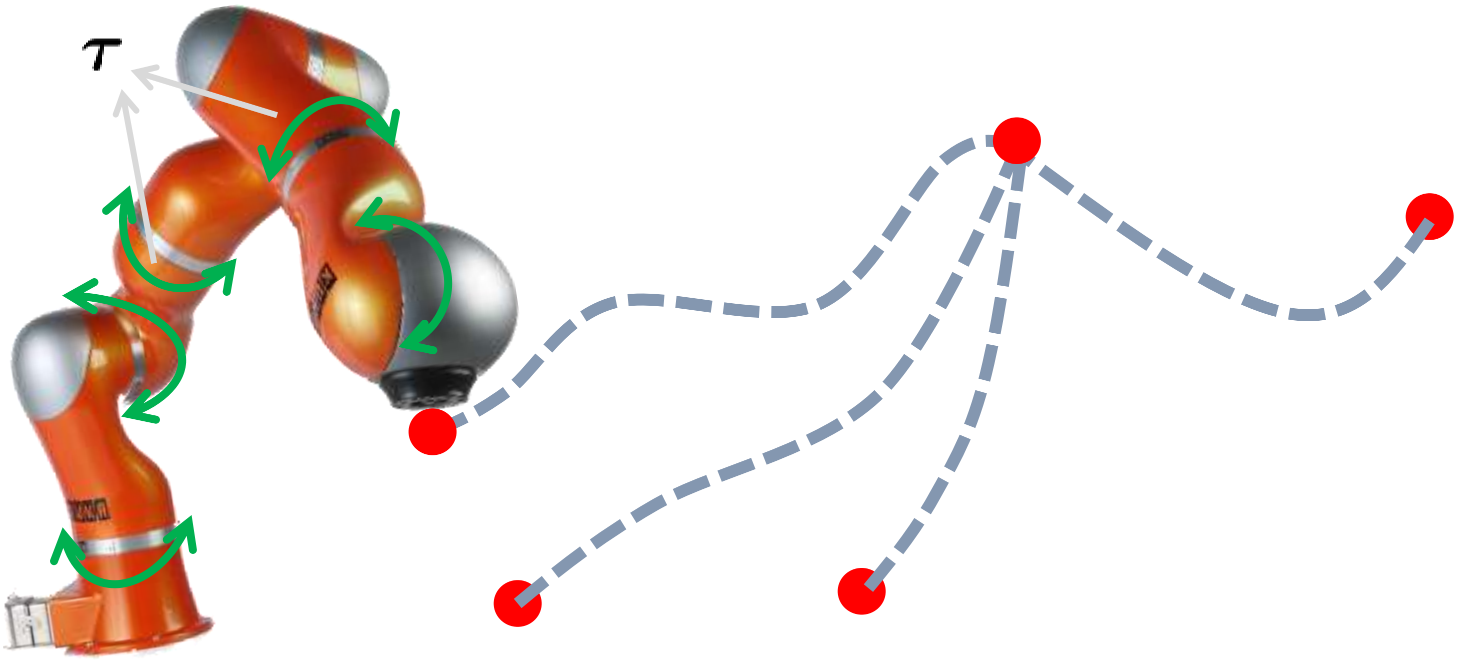
Level 1: One-shot learning

Level 0: Following – an implicit imitation mechanism

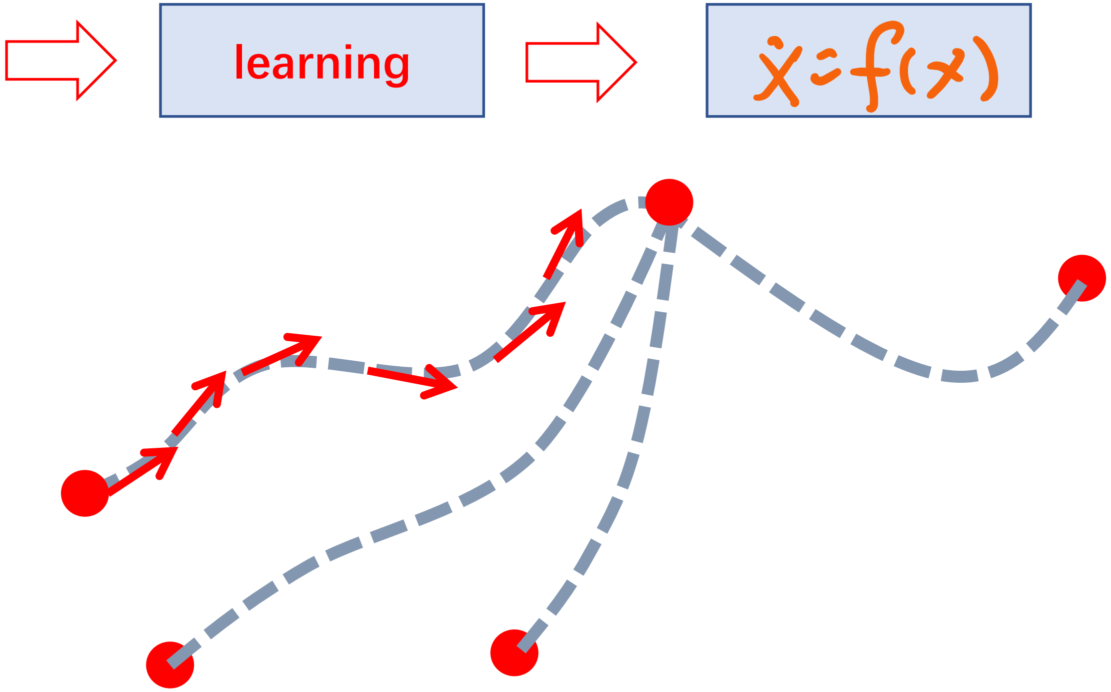
Level of cognition

Imitation learning

$$\mathbf{M}_h(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}_r + \mathbf{C}_h(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{g}_h(\boldsymbol{\theta}) = \boldsymbol{\tau}$$



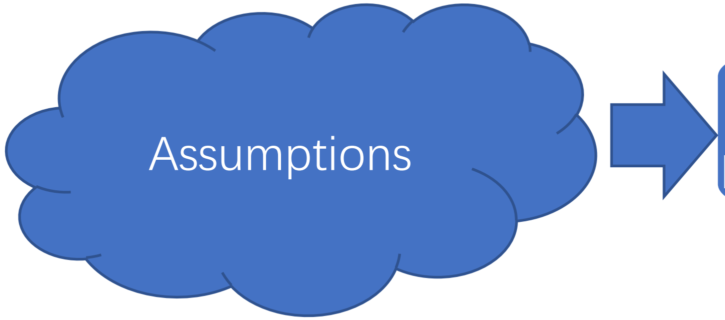
Imitation learning



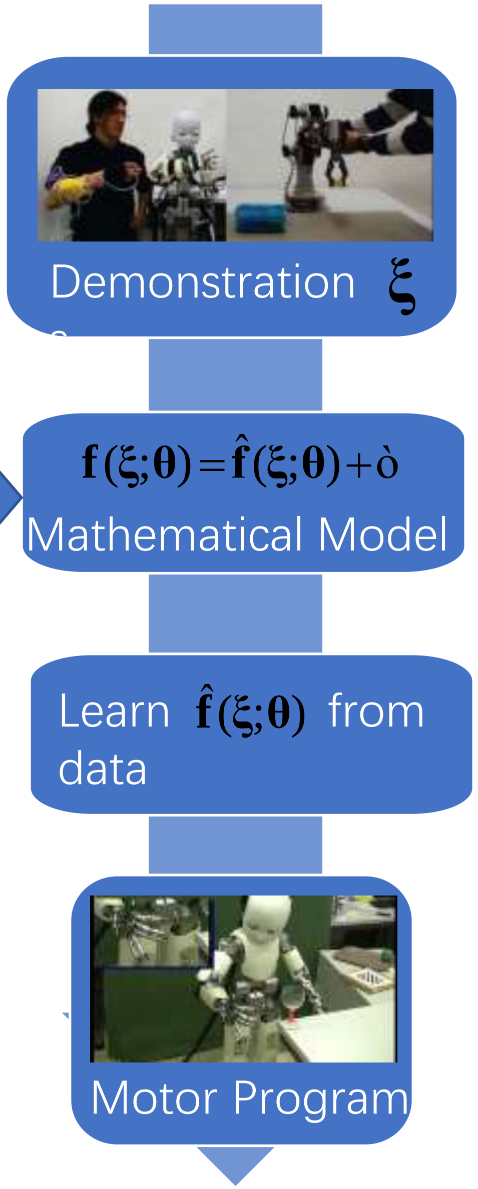
Imitation learning

Programming by Demonstration (Imitation Learning)

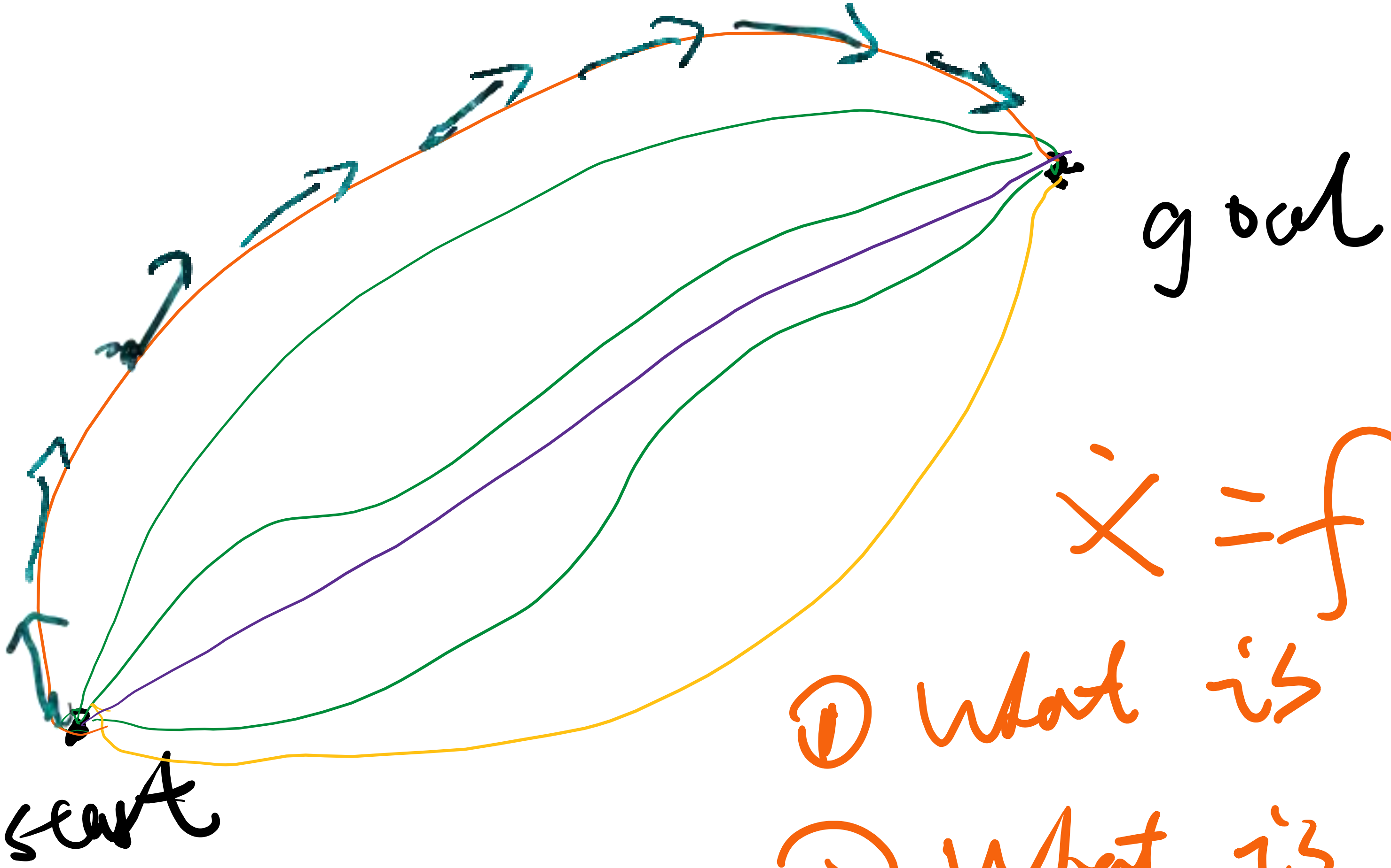
- A task is characterized by an underlying deterministic relationship between the **relevant** variables



- Demonstration: reproducing the underlying relationships corrupted by **white** noise.



Imitation learning



$\dot{x} = f(x)$

① what is x?

② what is f?

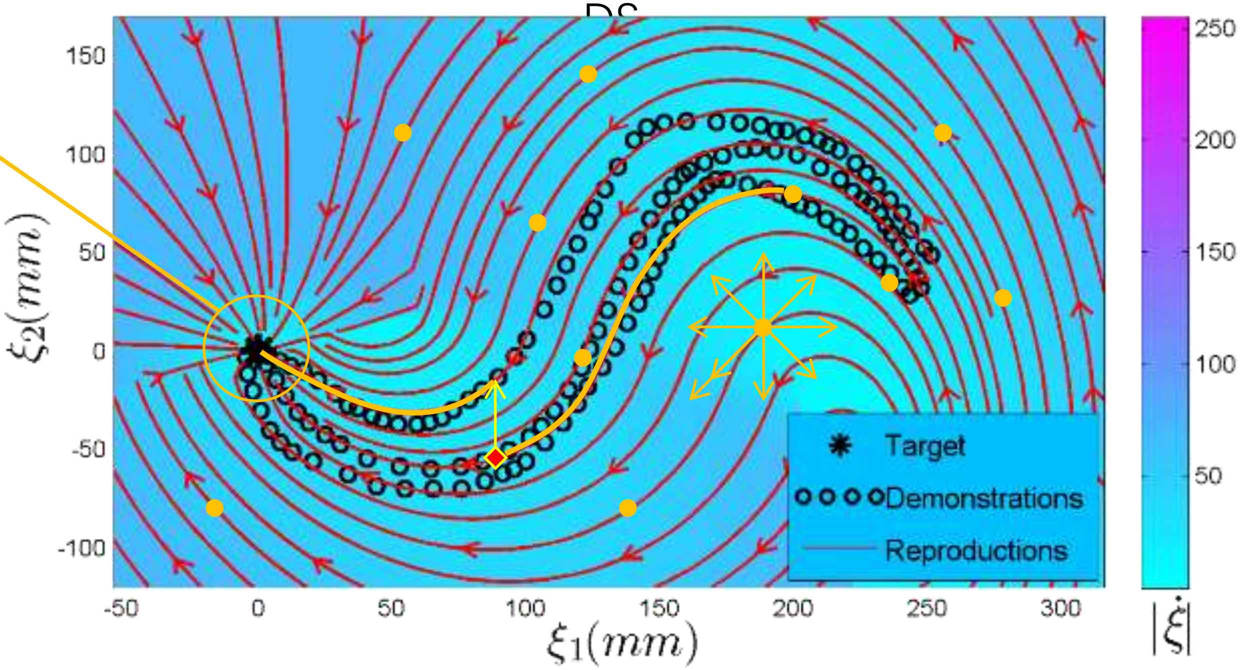
Question: collision?

Imitation learning

$$\dot{\xi} = f(\xi)$$

Streamlines of a *globally asymptotically* stable DS

Target



Given: Some demonstrations of a point-to-point motion.

Learned: Globally asymptotically stable map from states to velocities stable at the sole target.

Imitation learning

exp design:

hardware
sensor
protocol.
intention
interface

data collection:

joint angles
pos/ori
force
tactile
vision
⋮

learning Alg

GMM
GP
SVM
⋮
Deep learning
LLM
RT-2.

Imitation learning

exp design:

hardware

sens

protocol

intention

interface

data collection:

joint angles

tactile

vision

⋮

learning Alg

GMM

M

⋮

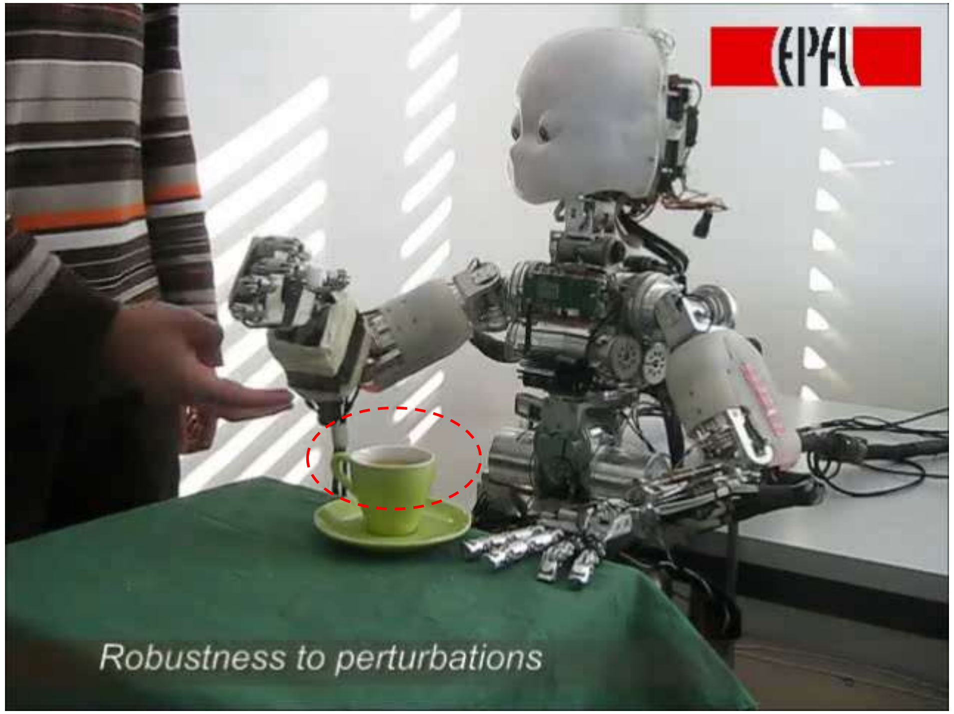
Deep learning

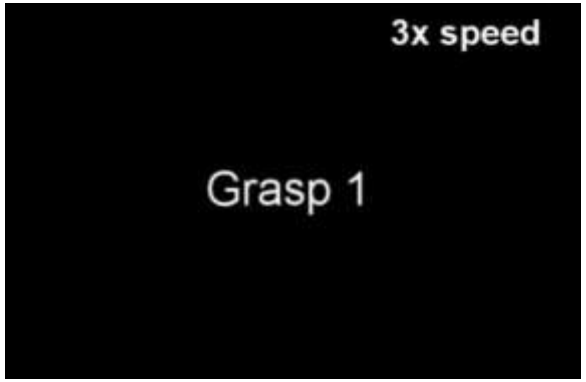
LLM

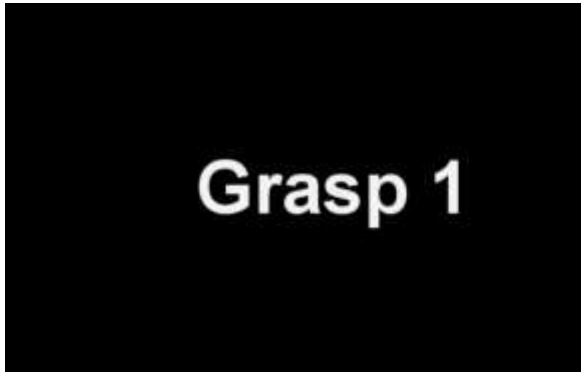
RT-2.

More details will be introduced in imitation learning course!

Imitation learning







Imitation learning

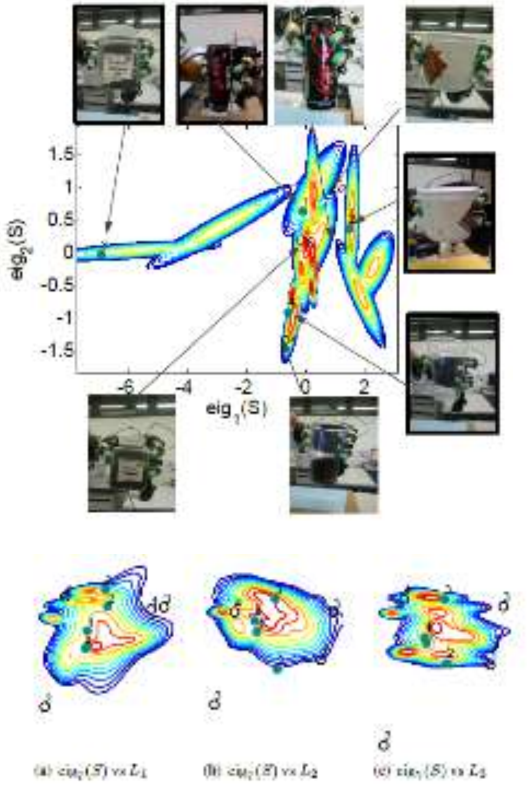
Google Deep Learning for Grasping

Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection

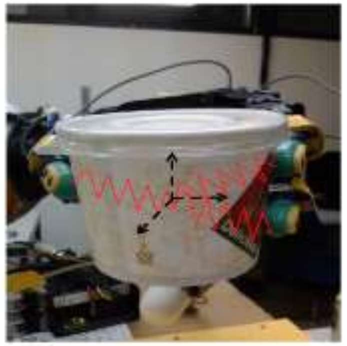
Sergey Levine SLEVINE@GOOGLE.COM
Peter Pastor PETERPASTOR@GOOGLE.COM
Alex Krizhevsky AKRIZHEVSKY@GOOGLE.COM

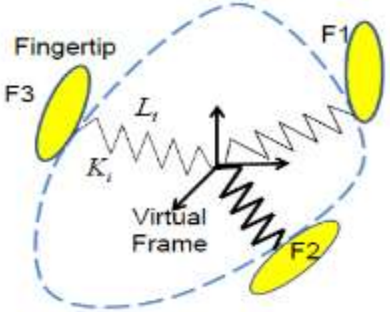
Imitation learning

Stability = sensory information
+ motor action



Object-level Impedance Controller





Imitation learning

Student project intro

1. ultrasound Robot
2. GraspAda.

Today's Agenda

- **Recap of sampling-based approach (~10)**
- **Recap of optimization-based approach (~20)**
- **Drawback of sampling and optimization (~5)**
- **Recap of perception-action loop (~2)**
- **Learning-based motion planning (~5)**
- **Imitation learning (~20)**
- **Reinforcement learning (~10)**

Goal for this course

- **Design：soft hand design x1**
- **Perception: vision, point cloud, tactile, force/torque x1**
- **Planning: sampling-based, optimization-based, learning-based x3**
- **Control: feedback, multi-modal x2**
- **Learning: imitation learning, RL x2**
- **Simulation tool (pybullet, matlab, OpenRAVE, Issac Nvidia, Gazebo)**
- **How to get a robot moving!**