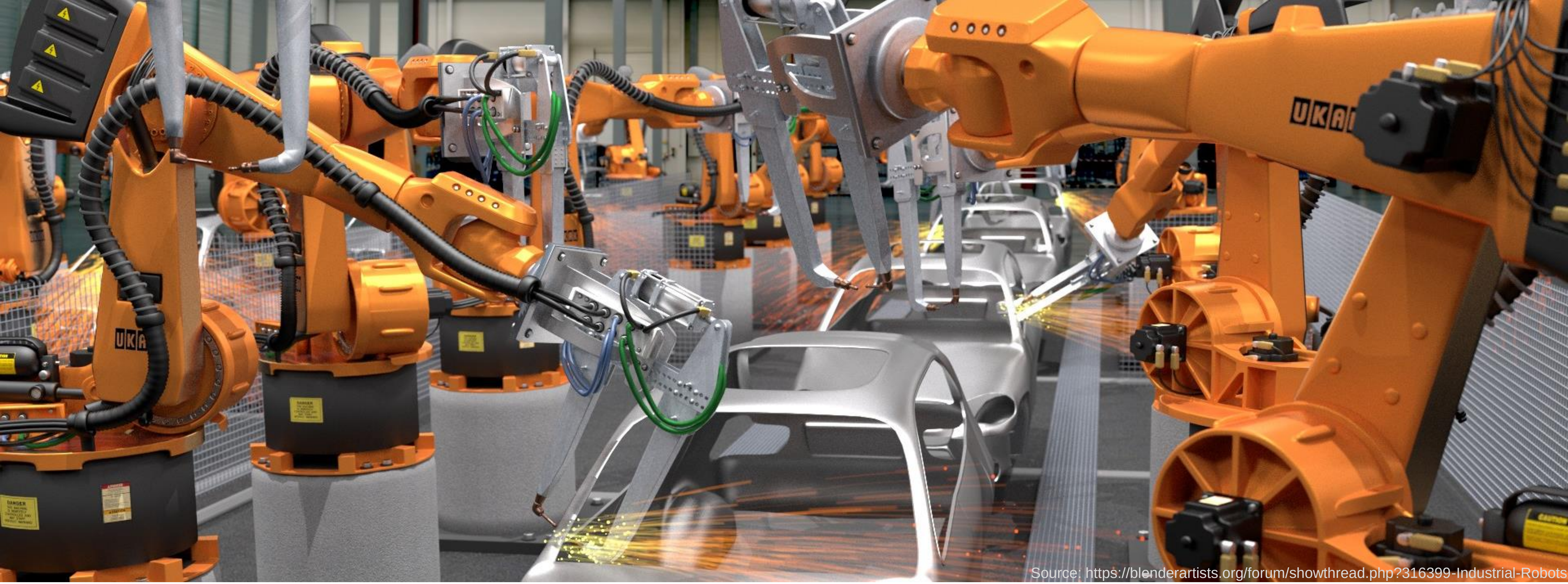


Robotics for Future Industrial Applications

Summary Week 2 – Intelligent Motion Planning

Philipp Ennen, M.Sc.

Organizational

Time	Sunday 13th August	Monday 14th August	Tuesday 15th August	Wednesday 16th August	Thursday 17th August	Friday 18th August	Saturday 19th August
9:00-9:30	Free Time for Excursions, Sight-Seeing and Self-Study	Introduction to Industrial Robots & Challenges	Fundamentals of Robot Learning and Control Theory	City Excursion to Cologne	Robot Learning with iterative Linear-Quadratic Regulator	Practical Unit: Robot Learning of an Assembly Task	Free Time for Excursions, Sight-Seeing and Self-Study
9:30-10:00							
10:00-10:30							
10:30-11:00		Motion Planning	Demonstration		Practical Unit: Robot Learning of an Assembly Task		
11:00-11:30							
11:30-12:00							
12:00-12:30		Lunch Break at “Bistro Kerres”			Lunch Break at “Bistro Kerres”		
12:30-13:00							
13:00-13:30							
13:30-14:00							
14:00-14:30		Practical Unit: Introduction to Linux and Robot Operating System	Programming using Python		Practical Unit: Robot Learning of an Assembly Task	Show: Let the robot learn!	
14:30-15:00							
15:00-15:30							
15:30-16:00			Practical Unit: Programming using Python				
16:00-16:30							
16:30-17:00							
17:00-17:30							
17:30-18:00							
18:00-18:30							
18:30-19:00							
19:00-19:30							
19:30-20:00						„Karaoke Night“ with the Summer School Team and your Buddies	
20:00-20:30			OPTIONAL International Tuesday with the INCAS Student Organisation				
20:30-21:00							
21:00-21:30							
21:30-22:00							

The central questions of robotics

Three main question:

This weeks focus!

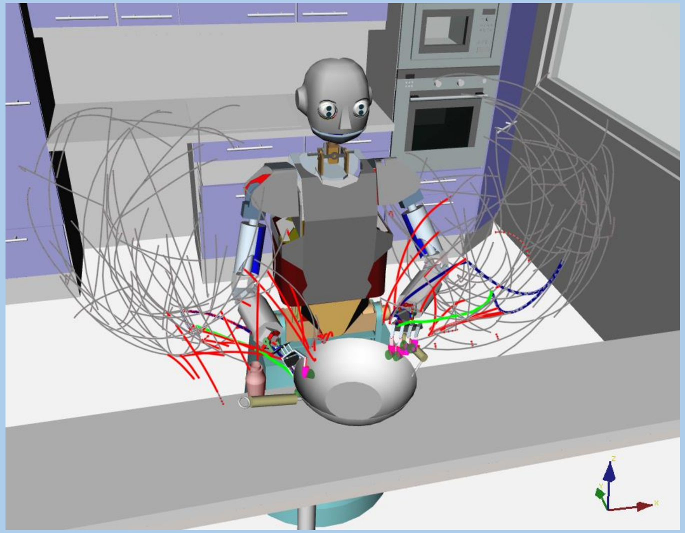
Where am I?

- Easy in Industrial Robotics
- Hard in Mobile Robotics

Where should I go?

- Often hand-engineered
- Or a Result of Task Planning

How do I go there?

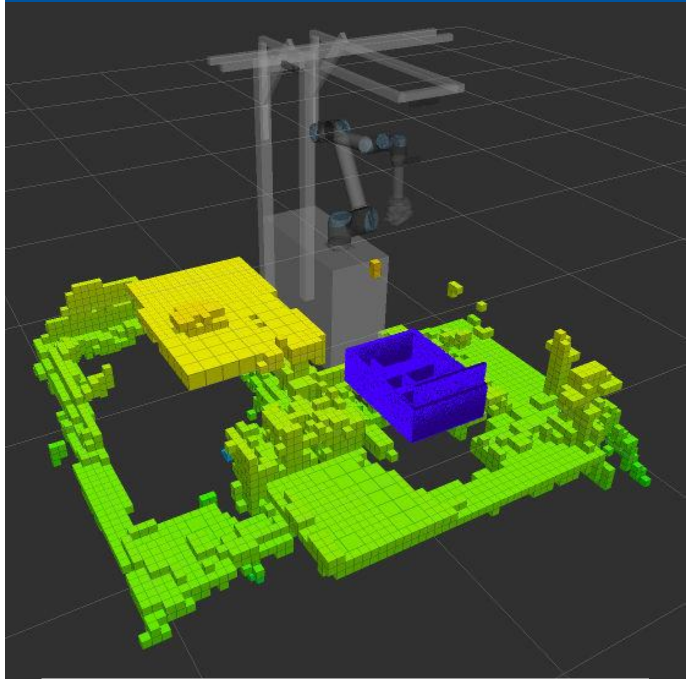


- Comparable Easy in Mobile Robotics
- Hard in Industrial Robotics

Robots as an Example for Intelligent Machines

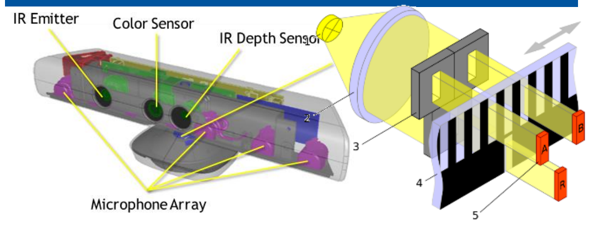
How do I (the robot) go there?

Model of the Robot and Environment



$$\begin{bmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0\end{bmatrix}\begin{bmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{bmatrix}+\begin{bmatrix}0\\ \frac{I+ml^2}{I(M+m)+Mml^2}\\ 0\\ \frac{ml}{I(M+m)+Mml^2}\end{bmatrix}u$$

Sensing



This weeks topic!

Motion Planning

Requires Goalstate:
- i.e. hand-engineered
- i.e. via a cost function

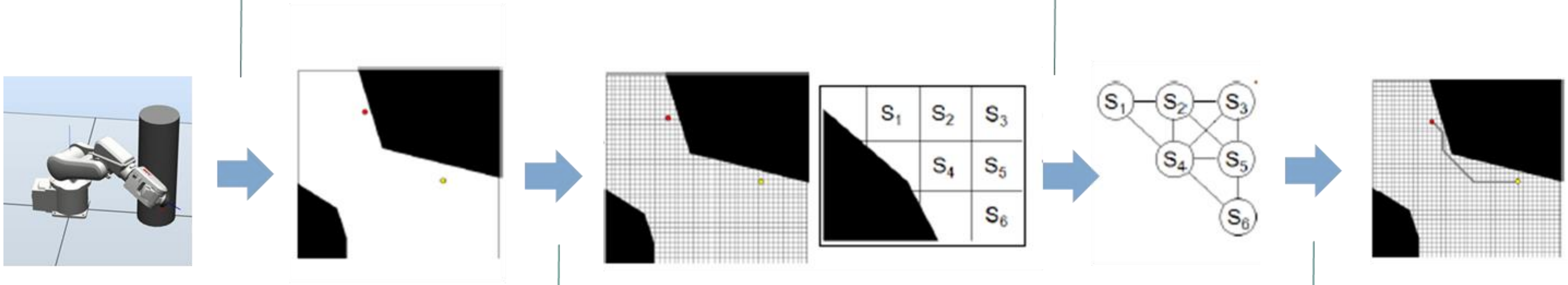
Motion Control and Execution

Motion Planning: Configuration Space

Motion Planning Pipeline within $\mathcal{C}_{space}$

transform workspace into **configuration space**

connect collision-free state transistions



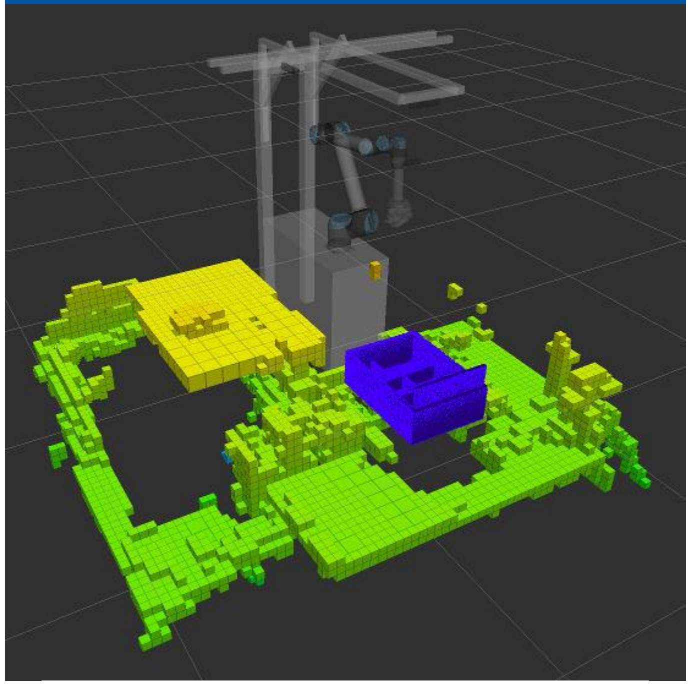
discretize configuration space

perform graph or tree **search**

Robots as an Example for Intelligent Machines

How do I (the robot) go there?

Model of the Robot and Environment



$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

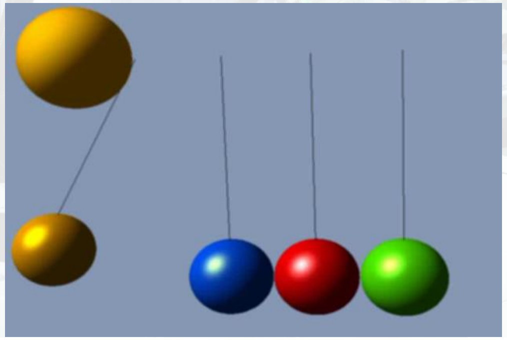
Sensing

Motion Control and Execution

What if the model of the robot and environment is hard to describe (or unknown)?

Think about flexible objects!

Think about contact-situations!

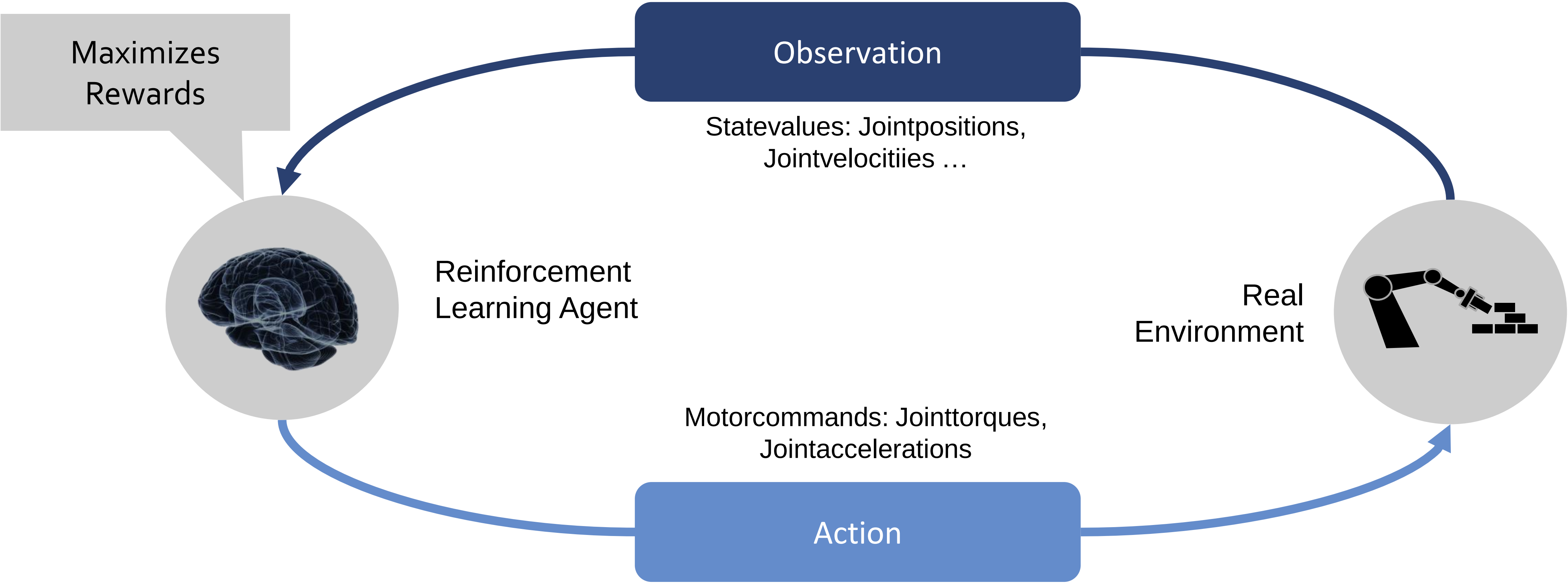


Requires Goalstate:
- i.e. hand-engineered
- i.e. via a cost function

Robots as an Example for Intelligent Machines

What if the model of the robot and environment is hard to describe (or unknown)?

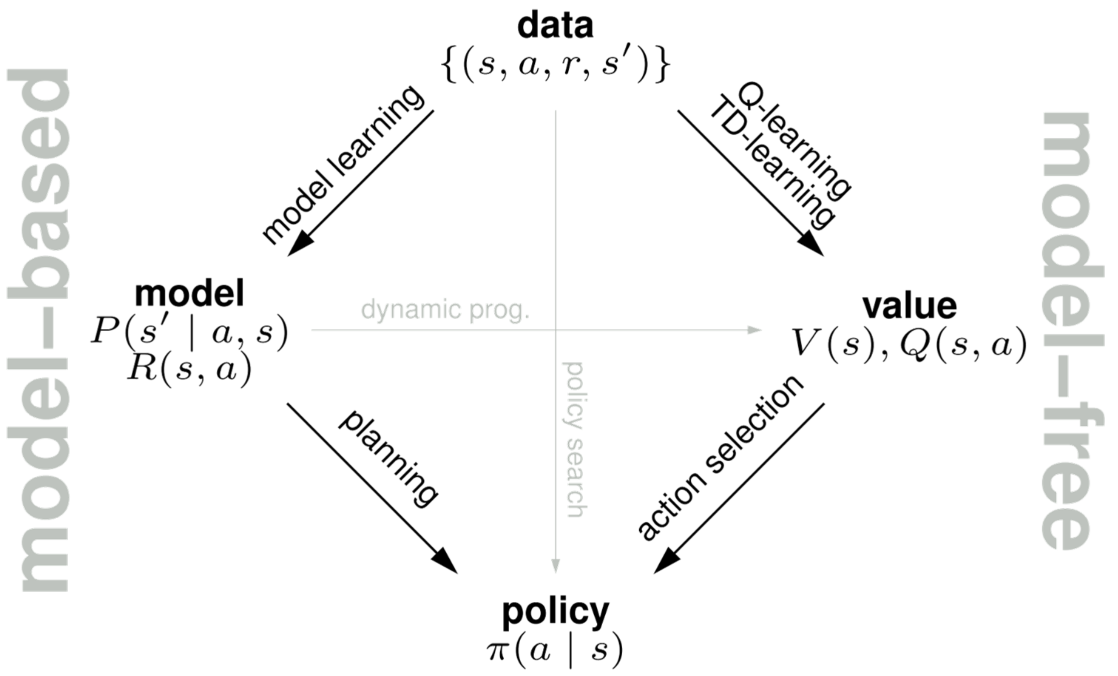
- Use an Reinforcement Learning Agent!



What if the model of the robot and environment is hard to describe (or unknown)?

This weeks topic!

Model-based RL:
- Learn to predict next state: $P(s'|s,a)$
- Learn to predict immediate reward $P(r'|s,a)$



Model-free RL:
- Learn to predict value: $V(s)$ or $Q(s,a)$

s: state
a: action
r: reward

Shooting Methode: LQR

Linear Quadratic Regulator

- Special case: Systems with
  - **Linear dynamics**
  - **Quadratic costs**
- LQR provides an exact solution

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots + c(f(f(\ldots)\ldots), \mathbf{u}_T)$$

$$\underbrace{f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t}_{\text{linear}}$$

$$\underbrace{c(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t}_{\text{quadratic}}$$

Pseudocode Algorithm

Backward recursion
(Get linear equations for $\mathbf{u}$)

for $t = T$ to 1:

$$\mathbf{Q}_t = \mathbf{C}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{F}_t$$

$$\mathbf{q}_t = \mathbf{c}_t + \mathbf{F}_t^T \mathbf{V}_{t+1} \mathbf{f}_t + \mathbf{F}_t^T \mathbf{v}_{t+1}$$

$$Q(\mathbf{x}_t, \mathbf{u}_t) = \text{const} + \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{Q}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{q}_t$$

$$\mathbf{u}_t \leftarrow \arg\min_{\mathbf{u}_t} Q(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$

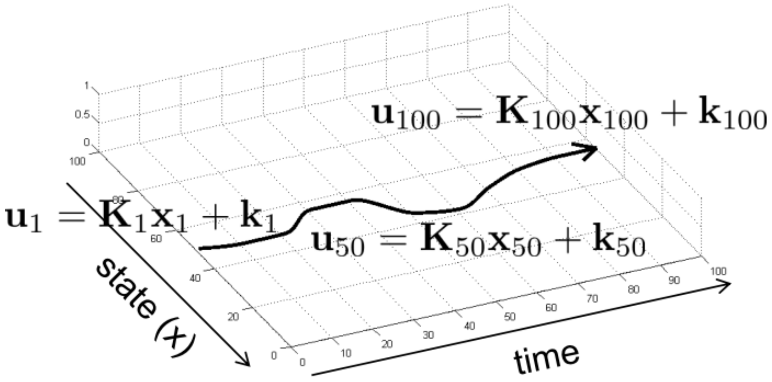
$$\mathbf{K}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{Q}_{\mathbf{u}_t, \mathbf{x}_t}$$

$$\mathbf{k}_t = -\mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t}^{-1} \mathbf{q}_{\mathbf{u}_t}$$

$$\mathbf{V}_t = \mathbf{Q}_{\mathbf{x}_t, \mathbf{x}_t} + \mathbf{Q}_{\mathbf{x}_t, \mathbf{u}_t} \mathbf{K}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{x}_t} \mathbf{x}_t + \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{K}_t$$

$$\mathbf{v}_t = \mathbf{K}_t^T \mathbf{Q}_{\mathbf{u}_t, \mathbf{u}_t} \mathbf{k}_t + \mathbf{Q}_{\mathbf{x}_t, \mathbf{u}_t} \mathbf{k}_t + \mathbf{q}_{\mathbf{x}_t} + \mathbf{K}_t^T \mathbf{q}_{\mathbf{u}_t}$$

$$V(\mathbf{x}_t) = \text{const} + \frac{1}{2} \mathbf{x}_t^T \mathbf{V}_t \mathbf{x}_t + \mathbf{x}_t^T \mathbf{v}_t$$



we know x_0

Forward recursion
(Use known initial state to get values for $\mathbf{u}$)

for $t = 1$ to T:

$$\mathbf{u}_t = \mathbf{K}_t \mathbf{x}_t + \mathbf{k}_t$$

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$$

Motivation

Why do we want to learn the dynamics?

- If $\mathbf{x}_{t+1} = f(\mathbf{x}_{t,}\mathbf{u}_t)$ is known, we can do trajectory optimization
  - In the stochastic case $\mathrm{p}(\mathbf{x}_{t+1}|\mathbf{x}_{t,}\mathbf{u}_t)$

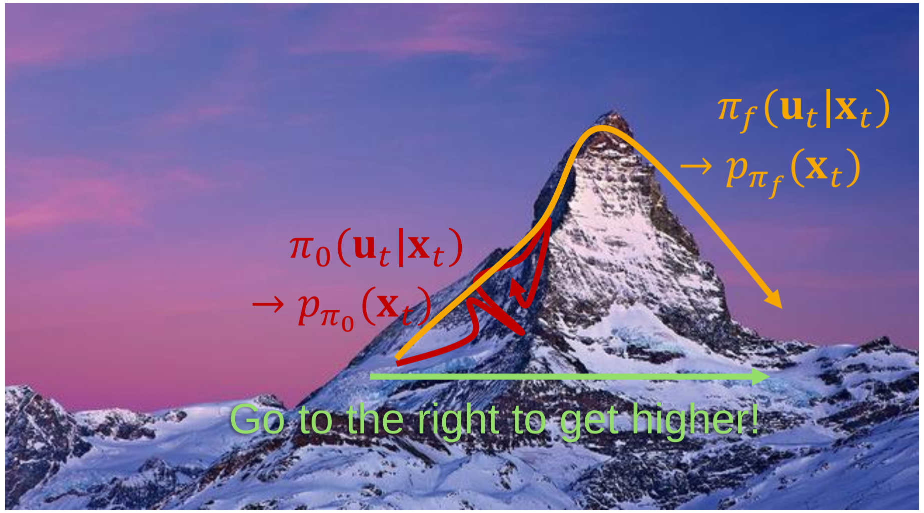
Learn $f(\mathbf{x}_{t,}\mathbf{u}_t)$ with subsequent backpropagation (i.e. iLQR)

Modelbased Reinforcement Learning Version 0.5

1. Execute initial policy $\pi_0(\mathbf{u}_t|\mathbf{x}_t)$ (i.e. a random policy) and collect data $\mathcal{D} = \{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_i\}$
2. Learn dynamics $f(\mathbf{x}_{,}\mathbf{u})$ that minimizes $\sum_i \left\| f(\mathbf{x}_{i,}\mathbf{u}_i) - \mathbf{x}_i' \right\|^2$
3. Backpropagate $f(\mathbf{x}_{,}\mathbf{u})$ and calculate sequence of actions (i.e. iLQR)

Does Version 0.5 work?

(in general) **NO!**



1. Execute initial policy $\pi_0(\mathbf{u}_t|\mathbf{x}_t)$ (i.e. a random policy) and collect data $\mathcal{D} = \{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_i\}$
2. Learn dynamics $f(\mathbf{x}, \mathbf{u})$ that minimizes $\sum_i \left\| f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_i' \right\|^2$
3. Backpropagate $f(\mathbf{x}, \mathbf{u})$ and calculate sequence of actions (i.e. iLQR) → $\pi_f(\mathrm{u}_t|\mathrm{x}_t)$

$$p_{\pi_0}(\mathbf{x}_t) \neq p_{\pi_f}(\mathbf{x}_t)$$

(Distribution Mismatch Problem)

❯ Distribution Mistmatch Problem increases if expressive classes of models are used (i.e. neural networks)

Can we do better?

Can we make $p_{\pi_0}(\mathbf{x}_t) = p_{\pi_f}(\mathbf{x}_t)$?

Need to collect data from $p_{\pi_f}(\mathbf{x}_t)$!

Modellbasiertes Reinforcement Learning Version 1.0

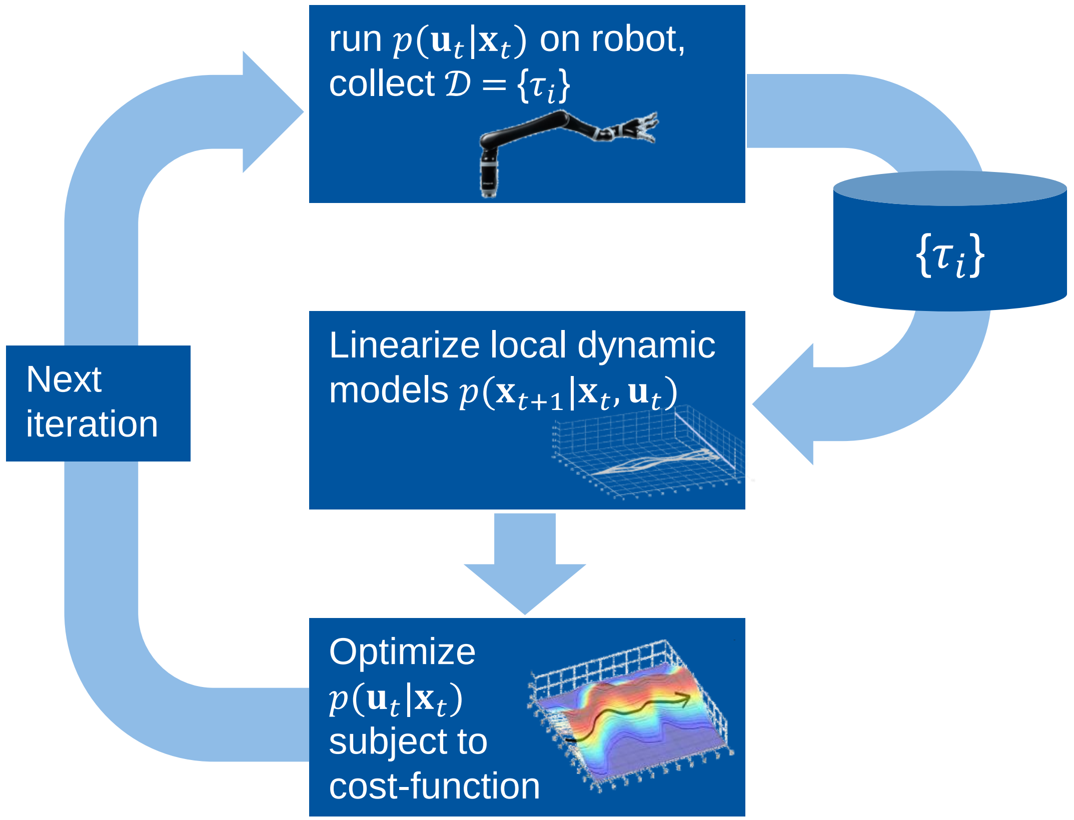
1. Execute initial policy $\pi_0(\mathbf{u}_t|\mathbf{x}_t)$ (i.e. a random policy) and collect data $\mathcal{D} = \{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_i\}$
2. Learn dynamics $f\left(\mathbf{x}_, \mathbf{u}\right)$ that minimizesLearn dynamics $f\left(\mathbf{x}_, \mathbf{u}\right)$ that minimizes $\sum_i \left\| f\left(\mathbf{x}_{i,} \mathbf{u}_i\right) - \mathbf{x}_i'\right\|^2$
3. Backpropagate $f\left(\mathbf{x}_, \mathbf{u}\right)$ and calculate sequence of actions (i.e. iLQR)
4. Execute those actions and add the resulting data $\{(\mathbf{x}, \mathbf{u}, \mathbf{x}')_i\}$ to $\mathcal{D}$

Learning a Policy

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \Sigma)$$

$$f(\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$$

$$\mathbf{A}_t = \frac{\partial f}{\partial \mathbf{x}_t} \qquad \mathbf{B}_t = \frac{\partial f}{\partial \mathbf{u}_t}$$



Local models

Linearized local dynamics

Goal: get the system dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ for each timestep t

Data: samples generated by the previous controller $\hat{p}_i(\mathbf{u}_t|\mathbf{x}_t)$ → $\{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})_i\}$

Linear Gaussian Dynamics are defined as

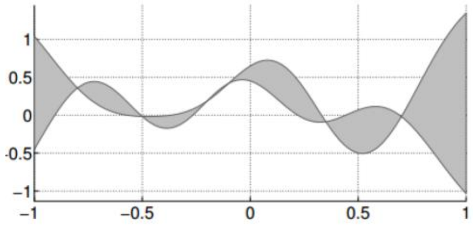
$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f_{xt}\mathbf{x}_t + f_{ut}\mathbf{u}_t + f_{ct}), \mathbf{F}_t)$$

How can we determine linear Gaussian dynamics from few samples?

What kind of models can we use?

This weeks focus!

Gaussian process



GP with input $(\mathbf{x}, \mathbf{u})$ and output $\mathbf{x}$'

Pro: very data-efficient

Con: not great with non-smooth dynamics

Con: very slow when dataset is big

Neural Network

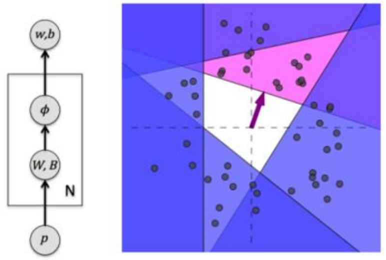


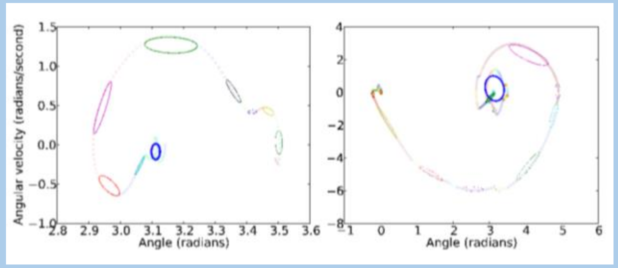
image: Punjani & Abbeel '14

Input is $(\mathbf{x}, \mathbf{u})$, output ist $\mathbf{x}$'

Pro: very expressive, can use lots of data

Con: not so great I low data regimes

Gaussian Mixture Model



GMM over $(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ tuples

Train on $(\mathbf{x}, \mathbf{u}, \mathbf{x}')$, condition to get $p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$

For i'th mixture element, $p_i(\mathbf{x}, \mathbf{u})$ gives region where the mode $p_i(\mathbf{x}'|\mathbf{x}, \mathbf{u})$ holds

Pro: very expressive, if the dynamics can be assumed as piecewise linear

Combining global and local models

$(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Global Model

- Can be wrong (distribution mismatch problem)
- Can reduce the number of required training data

$\{\tau_i\}$

$(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Local Model

- Would require lots of training data
- Uses $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ tuples of samples from previous iterations at timestep t

$p(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

$p(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Condition on given $(\mathbf{x}_t, \mathbf{u}_t)$

$p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t)$ Local linear Gaussian dynamics

Learning Local and Global Models

run $p(\mathbf{u}_t|\mathbf{x}_t)$ on robot, collect $\mathcal{D} = \{\tau_i\}$

Data contains samples from all previous iteration

$\{\tau_i\}$

Train GMM: **Global Dynamic Model**
- Uses data from nearby timesteps
- Uses data from prior iterations

Train GMM on all $\{\tau_i\} = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})_i\}$

Obtain prior parameters from GMM (mean μ_0, covariance $\boldsymbol{\Phi}$, degree of freedom n_0, m (number of datapoints))

Get normal-inverse-Wishart ($\mathcal{NIW}$) prior from GMM for a given tuple $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$

Local data $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$
From last iteration

Linearize: **Local Dynamic Model**
- Uses prior from local dynamic model
- Uses data from last iteration at timestep t
- Condition on given $(\mathbf{x}_t, \mathbf{u}_t)$

Calculate posterior covariance using $\mathcal{NIW}$-Priorparameters
Calculate posterior mean using sampled data

Condition on given $(\mathbf{x}_t, \mathbf{u}_t)$

$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ Local linear Gaussian dynamics

Thank you
for your attention!