Robotics for Future Industrial Applications

Summary Week 2 – Intelligent Motion Planning

Philipp Ennen, M.Sc.
## Organizational

<table>
<thead>
<tr>
<th>Time</th>
<th>Sunday 13th August</th>
<th>Monday 14th August</th>
<th>Tuesday 15th August</th>
<th>Wednesday 16th August</th>
<th>Thursday 17th August</th>
<th>Friday 18th August</th>
<th>Saturday 19th August</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00-9:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:30-10:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00-10:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:30-11:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00-11:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:30-12:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00-12:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:30-13:00</td>
<td>Free Time for Excursions, Sight-Seeing and Self-Study</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:00-13:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:30-14:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:00-14:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:30-15:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:00-15:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:30-16:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:00-16:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:30-17:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:00-17:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:30-18:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18:00-18:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18:30-19:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:00-19:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:30-20:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20:00-20:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20:30-21:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21:00-21:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21:30-22:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lunch Break at “Bistro Kerres”</td>
<td></td>
<td></td>
<td>City Excursion to Cologne</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lunch Break at “Bistro Kerres”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Practical Unit: Introduction to Linux and Robot Operating System</td>
<td>Practical Unit: Programming using Python</td>
<td>Practical Unit: Robot Learning of an Assembly Task</td>
<td>Show: Let the robot learn!</td>
<td>Free Time for Excursions, Sight-Seeing and Self-Study</td>
</tr>
</tbody>
</table>
Robots as an Example for Intelligent Machines

The central questions of robotics

Three main questions:

- Where am I?
- Where should I go?
- How do I go there?

- Easy in Industrial Robotics
- Hard in Mobile Robotics

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

This week's focus!

- 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

Motion Planning

This weeks topic!

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

Sensing

Motion Control and Execution

This weeks topic!
Motion Planning: Configuration Space

Motion Planning Pipeline within $C_{space}$

- **Transform workspace into configuration space**
- **Discretize configuration space**
- **Connect collision-free state transitions**
- **Perform graph or tree search**

[Diagram showing the steps of the pipeline]
Robots as an Example for Intelligent Machines

How do I (the robot) go there?

Model of the Robot and Environment

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

Sensing

Motion Control and Execution

Think about flexible objects!

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

Think about contact-situations!
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!

![Diagram of Reinforcement Learning](image-url)
Robots as an Example for Intelligent Machines

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

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

\[
\begin{align*}
\min_{\mathbf{u}_1, \ldots, \mathbf{u}_T} & \quad c(\mathbf{x}_1, \mathbf{u}_1) + c(f(\mathbf{x}_1, \mathbf{u}_1), \mathbf{u}_2) + \cdots + c(f(\ldots), \mathbf{u}_T) \\
f(\mathbf{x}_t, \mathbf{u}_t) &= F_t \mathbf{x}_t + f_t \\
c(\mathbf{x}_t, \mathbf{u}_t) &= \frac{1}{2} \mathbf{x}_t^T C_t \mathbf{x}_t + \mathbf{x}_t^T c_t
\end{align*}
\]

linear \hspace{2cm} quadratic
Linear Quadratic Regulator

Pseudocode Algorithm

Backward recursion
(Get linear equations for $u$)

for $t = T$ to 1:

$Q_t = C_t + F_t^T V_{t+1} F_t$
$q_t = c_t + F_t^T V_{t+1} f_t + F_t^T v_{t+1}$

$Q(x_t, u_t) = \text{const} + \frac{1}{2} \left[ x_t \right]^T Q_t \left[ x_t \right] + \left[ u_t \right]^T q_t$

$u_t \leftarrow \arg\min_{u_t} Q(x_t, u_t) = K_t x_t + k_t$

$K_t = -Q_{u_t,u_t}^{-1} Q_{x_t,u_t}$

$k_t = -Q_{u_t,u_t}^{-1} q_{u_t}$

$V_t = Q_{x_t,x_t} + Q_{x_t,u_t} K_t + K_t^T Q_{u_t,u_t} x_t + K_t^T Q_{u_t,u_t} K_t$

$v_t = K_t^T Q_{u_t,u_t} k_t + Q_{x_t,u_t} k_t + q_{x_t} + K_t^T q_u$

$V(x_t) = \text{const} + \frac{1}{2} x_t^T V_t x_t + x_t^T v_t$

Forward recursion
(Use known initial state to get values for $u$)

for $t = 1$ to $T$:

$u_t = K_t x_t + k_t$

$x_{t+1} = f(x_t, u_t)$
Motivation

Why do we want to learn the dynamics?

• If $x_{t+1} = f(x_t, u_t)$ is known, we can do trajectory optimization
  – In the stochastic case $p(x_{t+1} | x_t, u_t)$

  Learn $f(x_t, u_t)$ with subsequent backpropagation (i.e. iLQR)

Modelbased Reinforcement Learning Version 0.5

1. Execute initial policy $\pi_0(u_t | x_t)$ (i.e. a random policy) and collect data $\mathcal{D} = \{(x, u, x')\}_i$
2. Learn dynamics $f(x, u)$ that minimizes $\Sigma_i \|f(x_i, u_i) - x'_i\|^2$
3. Backpropagate $f(x, u)$ and calculate sequence of actions (i.e. iLQR)
Learning Dynamic Models

Does Version 0.5 work?

No! (in general)

1. Execute initial policy \( \pi_0(u_t | x_t) \) (i.e. a random policy) and collect data \( D = \{(x, u, x')_i\} \)
2. Learn dynamics \( f(x, u) \) that minimizes \( \sum_i \| f(x_i, u_i) - x'_i \|^2 \)
3. Backpropagate \( f(x, u) \) and calculate sequence of actions (i.e. iLQR) \( \pi_f(u_t | x_t) \)

\[ p_{\pi_0}(x_t) \neq p_{\pi_f}(x_t) \]

(Distribution Mismatch Problem)

Distribution Mismatch Problem increases if expressive classes of models are used (i.e. neural networks)
Learning Dynamic Models

Can we do better?

Can we make $p_{\pi_0}(x_t) = p_{\pi_f}(x_t)$?

Need to collect data from $p_{\pi_f}(x_t)$!

Modellbasiertes Reinforcement Learning Version 1.0

1. Execute initial policy $\pi_0(u_t|x_t)$ (i.e. a random policy) and collect data $\mathcal{D} = \{(x, u, x')\}_i$

2. Learn dynamics $f(x, u)$ that minimizes $\sum_i \|f(x_i, u_i) - x'_i\|^2$

3. Backpropagate $f(x, u)$ and calculate sequence of actions (i.e. iLQR)

4. Execute those actions and add the resulting data $\{(x, u, x')\}_i$ to $\mathcal{D}$
Learning a Policy

\[ p(x_{t+1}|x_t, u_t) = \mathcal{N}(f(x_t, u_t), \Sigma) \]

\[ f(x_t, u_t) \approx A_t x_t + B_t u_t \]

\[ A_t = \frac{\partial f}{\partial x_t} \quad B_t = \frac{\partial f}{\partial u_t} \]

run \( p(u_t|x_t) \) on robot, collect \( D = \{\tau_i\} \)

Linearize local dynamic models \( p(x_{t+1}|x_t, u_t) \)

Next iteration

Optimize \( p(u_t|x_t) \) subject to cost-function
### Local models

#### Linearized local dynamics

Goal: get the system dynamics $p(x_{t+1}|x_t, u_t)$ for each timestep $t$

Data: samples generated by the previous controller $\hat{p}_i(u_t|x_t) \rightarrow \{(x_t, u_t, x_{t+1})_i\}$

Linear Gaussian Dynamics are defined as

$$p(x_{t+1}|x_t, u_t) = \mathcal{N}(f_{xt}x_t + f_{ut}u_t + f_{ct}), F_t)$$

How can we determine linear Gaussian dynamics from few samples?
What kind of models can we use?

**Gaussian process**
- Input: \((x, u)\) and output \(x'\)
- Pro: very data-efficient
- Con: not great with non-smooth dynamics
- Con: very slow when dataset is big

**Neural Network**
- Input is \((x, u)\), output is \(x'\)
- Pro: very expressive, can use lots of data
- Con: not so great in low data regimes

**GMM over \((x, u, x')\) tuples**
- Train on \((x, u, x')\), condition to get \(p(x' | x, u)\)
- For \(i\)'th mixture element, \(p_i(x, u)\) gives region where the mode \(p_i(x' | x, u)\) holds
- Pro: very expressive, if the dynamics can be assumed as piecewise linear
Combining global and local models

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

Local Model
- Would require lots of training data
- Uses \( (x_t, u_t, x_{t+1}) \) tuples of samples from previous iterations at timestep \( t \)

\[
p(x_t, u_t, x_{t+1})
\]

Condition on given \((x_t, u_t)\)

\[
p(x_{t+1} | x_t, u_t)
\]

Local linear Gaussian dynamics
Learning Local and Global Models

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

Linearize: **Local Dynamic Model**
- Uses prior from local dynamic model
- Uses data from last iteration at timestep $t$
- Condition on given $(x_t, u_t)$

**Train GMM on all** $\{\tau_i\} = \{(x_t, u_t, x_{t+1})\}$

Obtain prior parameters from GMM (mean $\mu_0$, covariance $\Phi$, degree of freedom $n_0$, $m$ (number of datapoints))

Get normal-inverse-Wishart $(\mathcal{NIW})$ prior from GMM for a given tuple $(x_t, u_t, x_{t+1})$

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

Calculate posterior mean using sampled data

Condition on given $(x_t, u_t)$

$p(x_{t+1}|x_t, u_t)$ Local linear Gaussian dynamics

Data contains samples from all previous iteration

run $p(u_t|x_t)$ on robot, collect $\mathcal{D} = \{\tau_i\}$

Local data $(x_t, u_t, x_{t+1})$ From last iteration
Thank you for your attention!