Master's Thesis

Incorporating Uncertainty into Reinforcement Learning through Gaussian Processes

Markus Kaiser

Chair for Foundations of Software Reliability and Theoretical Computer Science
Department of Informatics
Technische Universität München

Computer Vision and Active Perception Lab
School of Computer Science and Communication
Kungliga Tekniska högskolan

Learning Systems
CT-RDA-BAM-LSY-DE
Siemens

SIEMENS

Contents

Todo list

Chapter 2

The Bicycle Benchmark

The bicycle benchmark is an example of a dynamic a computer should learn to control. It was originally defined by Randløv and Alstrøm in 1998 [RA98]. The task in this benchmark is to balance and navigate a simulated bicycle which travels at a constant speed.

The computer, or *agent*, takes the place of the rider of the bicycle. After fixed time intervals, it has to decide how to influence the bicycle by applying some torque T on the handle bars to steer or by displacing the center of mass of the bicycle via leaning over the bicycle by some distance d or both. In order to make its decision, the controller is given perfect information about the internal state of the simulation. Besides having to prevent the bicycle from falling over in the short-term, there is also the long-term goal of navigating to some predefined goal position.

The bicycle is modelled by non-linear differential equations which describe the steering and leaning behaviour of the bicycle. The speed of the back tyre is considered constant and independent of the actions of the agent. This results in two important angles to describe the system. The angle θ is measured between the front tyre and the frame of the bicycle and describes the straightness of the bicycles path. The angle ω measures the amount of tilt of the bicycle's frame compared to standing upright. If the absolute value of ω is greater than 12 degrees, the bicycle has fallen over. In addition to the two angles, their time-derivatives $\dot{\theta}$ and $\dot{\omega}$ define the dynamics of the bicycle. Together with the bicycle's position and rotation in euclidean space, this completely describes the current state of one instance of the bicycle benchmark and is summarized in table 2.1.

The conservation of angular momentum of the tyres results in interactions between θ and ω and their derivatives. The equations presented in the following describe the dynamics of the system as introduced in [RA98]. Two simplifying assumptions have been made: Firstly, the front fork is assumed to be vertical, which is unusual and makes balancing the bicycle more difficult but is not physically impossible. And secondly,

the equations are not an exact analytical description, as some second and higher order terms have been ignored.

An overview of the geometric interpretations of the state variables can be seen in figure 2.1. The interactions between tilt and lean are based on the conservation of angular momentum which is heavily influenced by the moments of inertia of the system. The moments of the tyre as displayed in figure 2.1c and the moment of the bicycle and cyclist combined I_{BC} were estimated by the original definition [RA98] as

$$\begin{aligned} I_{dc} &:= \quad M_d r^2 \\ I_{dv} &:= \frac{3}{2} M_d r^2 \\ I_{dl} &:= \frac{1}{2} M_d r^2 \\ I_{\mathrm{BC}} &:= \frac{13}{3} M_c h^2 + M_p (h + d_{\mathrm{CM}})^2. \end{aligned}$$

The dynamics also depend on multiple constants which are detailed in table 2.3.

The angular acceleration $\ddot{\omega}$ of the lean of the bicycle consists of three parts. The first part describes the gravitation acting on the bicycle and cyclist which pulls the bicycle in the direction it is already leaning. The second one are effects based on the conservation of angular momentum introduced via a cross-term dependent on $\dot{\theta}$. The centrifugal force applied because of the curved movement of the bicycle forms the last part. The center of mass can be displaced horizontally by the agent via the choice of d. The combination φ of this displacement and the lean angle of the bicycle is defined as

interpretation?

$$\varphi := \omega + \arctan\left(\frac{d}{h}\right).$$

With this, the angular acceleration $\ddot{\omega}$ can be calculated as

$$\ddot{\omega} := \frac{1}{I_{\mathrm{BC}}} \left(\sin\varphi \cdot Mgh - \cos\varphi \cdot \left(I_{dc}\dot{\sigma} \cdot \dot{\theta} + \mathrm{sgn}(\theta) \cdot v^2 \left(\frac{M_d r}{r_f} + \frac{M_d r}{r_b} + \frac{Mh}{r_{\mathrm{CM}}} \right) \right) \right).$$

The angular acceleration $\ddot{\theta}$ of the orientation of the front tyre is equal to that of the handlebars because the front fork is assumed to vertical. It is dependent on the torque T applied to the handlebars and the conservation of angular momentum introduced via a cross-term dependent on $\dot{\omega}$ and is calculated as

$$\ddot{\theta} := \frac{T - I_{dv}\dot{\sigma} \cdot \dot{\omega}}{I_{dl}}.$$

Table 2.1: Variables defining the current state of the bicycle system.

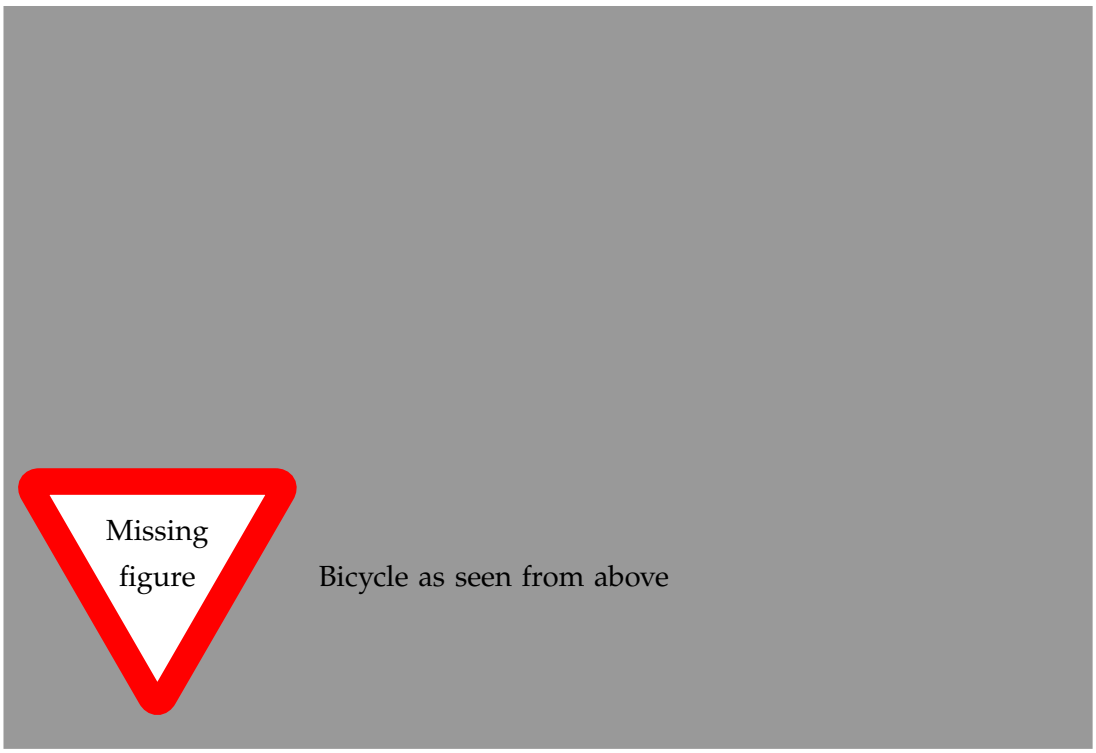
Notation	Description	Value range	Unit
θ	Angle between the frame and the front tyre	$-\pi/2$ to $\pi/2$	rad
$\dot{\theta}$	Rotational speed of the handlebars	-10 to 10	rad/s
ω	Tilt of the bicycle	$-\pi/15$ to $\pi/15$	rad
$\dot{\omega}$	Tilting speed of the bicycle	-10 to 10	rad/s
x	Global x-position of the front tyre	-100 to 100	m
y	Global y-position of the front tyre	-100 to 100	m
ψ	Global orientation of the bicycle	$-\pi$ to π	rad

Table 2.2: Actions which can be applied to the bicycle system.

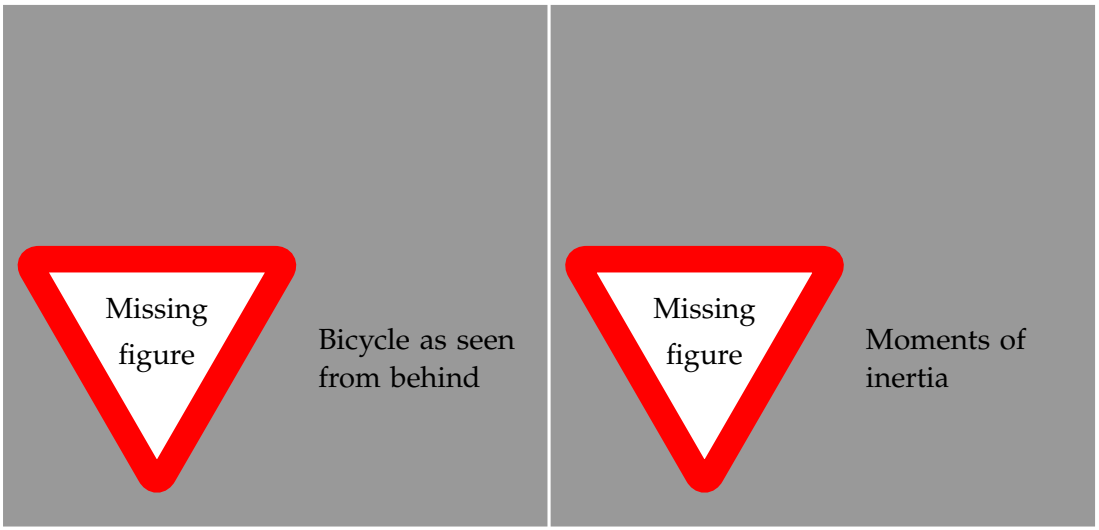
Notation	Description	Value range	Unit
d	The distance the agent leans sideways by displacing the center of mass	-0.02 to 0.02	m
T	The torque the agent applies on the handlebars	-2 to 2	N

Table 2.3: Physical constants and their values in the bicycle system [RA98].

Notation	Description	Value
CM	Center of mass of the bicycle and cyclist in total	
c	Horizontal distance between the point where the front tyre touches the ground and the saddle	66 cm
d_{CM}	Vertical distance between the centers of mass of the bicycle and the cyclist	30 cm
h	Vertical distance between the CM and the ground	94 cm
l	Distance between the points where the front and back tyres touch the ground	111 cm
M_c	Mass of the bicycle	15 kg
M_d	Mass of a tyre	1.7 kg
M_p	Mass of the cyclist	60 kg
r	Radius of a tyre	34 cm
v	Velocity of the bicycle	10 km/h
$\dot{\sigma}$	Angular velocity of the back tyre	$\dot{\sigma} = v/r$



(a) Bicycle as seen from above



(b) Bicycle as seen from behind.

(c) Moments of inertia.

Figure 2.1: Bicycle dynamics

These differential equations describe how leaning left or right as the driver and turning the handlebars interact with each other. These internal dynamics are the important factors when trying to balance the bicycle while ignoring the movement of the bicycle in space. In order to successfully navigate to the goal position, these movements have to be taken into consideration. The bicycle state contains three variables which locate the bicycle in space. The position of the bicycle is defined by the point where the front tyre touches the ground. This point is independent of the orientation of the handlebars given by the angle θ and identified via the two euclidean coordinates x and y. The last state variable ψ describes the orientation of the bicycle frame relative to this point as shown in figure 2.1a. If ψ is equal to zero, the back tyre points in x-direction if viewed from the point (x, y) and a positive ψ denotes a counter-clockwise rotation.

ψ rotation direction correct?

The back tyre of the bicycle moves at the constant speed v. The front and back tyre follow two circular paths with different radii but the same center as can be seen in figure 2.1a with the front tyre following the longer path. The radii r_f and r_b can be calculated as

$$r_f = \frac{l}{|\sin\theta|},$$
$$r_b = \frac{l}{|\tan\theta|},$$

respectively for θ not equal to zero. The singularity of θ approaching zero yields radii of arbitrary size as the bicycle's path becomes more and more straight. If θ is equal to zero, the bicycle's orientation does not change and it moves along a straight line. If it is not zero, the change of position and orientation can be calculated from the curved movement.

Since the two tyres are connected with a rigid frame and the front tyre travels on a longer path, it has to do so at a higher speed. They do however share the same angular velocity $v_\circ$ on their respective circular paths, since if θ remains constant, the bicycle rotates around the common center point. This angular velocity can be derived from the constant speed of the back tyre and is given by

$$v_\circ = \frac{v}{r_b}.$$

Combined with the direction of rotation on the circle defined by the sign of θ, this directly yields the derivative of the world orientation ψ:

$$\dot{\psi} = \operatorname{sgn}(\theta) \cdot v_\circ$$

The actual speed of the front tyre can be obtained from the common angular velocity $v_\circ$ and the radius of its path r_f. Together with the orientation of the front tyre, this gives the derivatives

$$\dot{x} = v_\circ \cdot r_f \cdot \cos(\psi + \theta)$$
$$\dot{y} = v_\circ \cdot r_f \cdot \sin(\psi + \theta)$$

of the position of the bicycle.

The original implementation of Randløv and Alstrøm uses an explicit Euler scheme to evolve the dynamics of the system for a time step. The changes of position and orientation are calculated using the exact analytical solution of moving the tyres along their circular paths. To improve accuracy, the implementation developed for this thesis implements a standard Runge-Kutta-Scheme as described in *Numerical Recipes* [Pre07]. The assumption that for a single time step, the bicycle moves along a fixed circular path is no longer correct when evaluating the dynamics with Runge-Kutta. Because of this, the changes in position and orientation are integrated into the scheme.

The goal of the bicycle benchmark is to drive the bicycle to a specific position which is assumed to be a circle at the origin of the coordinate system with a radius of 5. The bicycle starts in a position which is almost upright, with the state variables θ, $\dot{\theta}$, ω, $\dot{\omega}$ being sampled from Gaussian noise with a standard deviation of one percent of their value range.

The agent has to choose the two continuous actions d and T described in table 2.2 every 0.01 seconds, after which they are assumed constant for this time interval. After this time, the controller is once again presented with the current and exact values of the state variables and has to make a new decision. One run of the bicycle system creates a time series of bicycle states and the actions chosen by the controller. The episode ends in failure if the bicycle falls over and it ends in success if the bicycle reaches the goal. Since it is possible for the bicycle to drive in circles indefinitely, the episode also ends in failure after a certain amount of time has passed.

The next chapter introduces reinforcement learning as a general mathematical description of the problem posed with the bicycle benchmark. Model based reinforcement learning is one way to solve the bicycle benchmark which tries to learn the bicycle's dynamics using general function approximators and use them to make informed guesses about the future.

Chapter 4

Uncertainties in the Bicycle Benchmark

Chapter 3 introduced reinforcement learning as a general mathematical framework to describe the problem of controlling a bicycle in the benchmark presented in chapter 2. This thesis is concerned with model-based reinforcement learning, where the transition function of the system to be controlled is represented with some function approximation which is learnt from observations of the system and is used to make predictions about the future. Learning a model of the true system introduces model bias, where actions considered to be good with respect to the model's predictions can show bad performance in reality because the model is incorrect. In order to reduce this bias, Gaussian Processes can be used which do not only yield one specific function approximation but a distribution over all plausible models. This uncertainty about the correct model can be propagated through to predictions about specific test points and instead of a single point, Gaussian processes predict a Gaussian distribution about possible function values.

Modelling the transition dynamics allows the prediction of a state s_{t+1} given a concrete pair of state and action (s_t, a_t) for the previous time step. Assuming a deterministic model which yields exactly one posterior state, the *long-term prediction* of states multiple time steps into the future reduces to iterated one-step predictions. Given a (deterministic) reward function $\mathbf{R}$ as detailed in definition 3, it is possible to evaluate the action value function $\hat{\mathbf{V}}$ of PSO-P and thus directly use the model to extract a policy, since the expected value in equation (3.5) is a simple sum of deterministic values.

In the Bayesian context however, the transition model predicts a distribution over posterior states. This complicates long-term predictions since the uncertainty of intermediate states has to be propagated through the (non-linear) transition model to accumulate the uncertainties of multiple predictions. Additionally, the original deterministic reward function possibly has to be adapted to make it possible to evaluate the expected reward for all time steps. Once these problems are solved and the action value function can be

calculated, PSO-P can be used in the same way as in the deterministic case to choose appropriate actions.

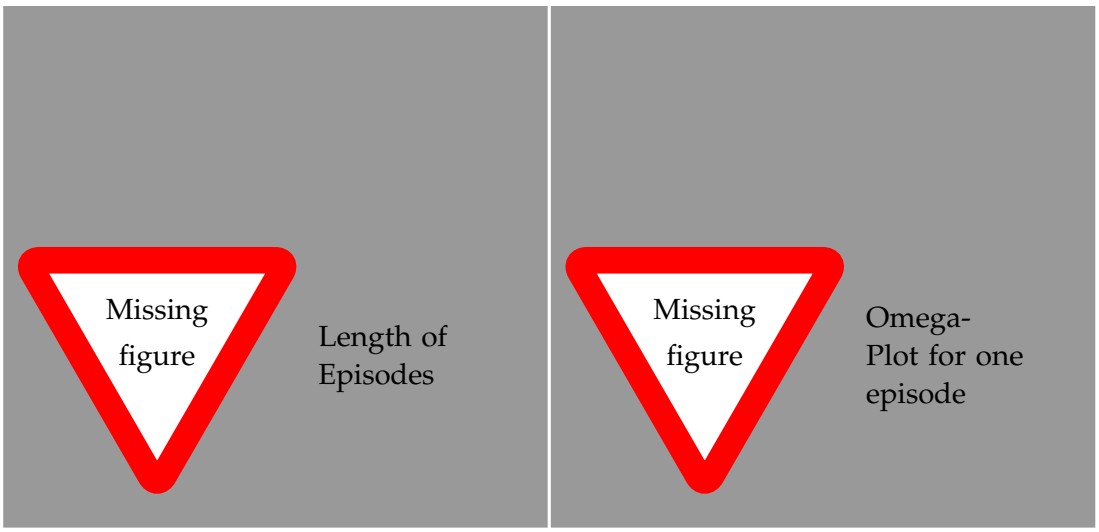
Based on the bicycle benchmark, this chapter compares the classical deterministic long-term predictions to two approaches of integrating uncertainty into the predictions. The first section describes the creation of data sets used to train the transition model and the design-choices made to obtain suitable models. These models are then interpreted as deterministic to obtain a base-line for comparison in the next section. The last two sections describe how to use uncertainties in the planning. The first approach is to use the one-step uncertainties of the Gaussian Process models in the reward function but to still create long-term predictions using deterministic states. The second fully Bayesian approach completely propagates state uncertainties both to the reward function and subsequent states.

4.1 Transition Dynamics

The goal in the bicycle benchmark is to learn to both balance a bicycle and to ride it to a target position. The state of a bicycle, as described in chapter 2, consists of the real valued vector $(\theta, \dot{\theta}, \omega, \dot{\omega}, x, y, \psi)$, where θ is the angle of the handlebars, ω is the vertical angle of the bicycle frame, x and y are euclidean coordinates and ψ is the orientation of the bicycle. The bicycle starts in almost upright position and the task of the controller is to choose the actions (d, T), where d is the horizontal leaning displacement of the driver and T is the torque applied to the handlebars at every time step. The transition dynamics are derived from a physical approximation of the system and are completely deterministic.

This thesis assumes that the actor is not allowed to interact with the system in order to try or improve its policy. In contrast, the agent is presented with a predefined data set of observations of the system obtained with a simple and sub-optimal controller. This constraint is meant to mimic industrial systems where it is comparatively cheap to obtain measurements of running systems but allowing an agent to explore is either very expensive or a security concern. It is therefore not possible to apply an on-line learning scheme on the system or to explore in specific directions in order to improve the dynamics model.

This section first describes how data sets are sampled from the bicycle benchmark in order to simulate this constraint. These data sets are then used to train the Gaussian Processes used in PSO-P to form a controller.



(a) Length of Episodes **(b)** Omega-Plot for one episode

Figure 4.1: Data Set properties

4.1.1 Data Sets

The bicycle benchmark's dynamics are introduced in chapter 2 by defining the derivatives of the state variables and choosing values for the relevant constants in table 2.3. Given a starting state and an appropriate number of actions, these derivatives can be used to approximate the future behaviour of the system using iterative numerical methods for approximating ordinary differential equations. For this thesis, the bicycle benchmark was implemented in Python [Ros95] with NumPy [WCV11] and using the classical Runge-Kutta scheme [Kut01].

In their experiments, Randløv and Alstrøm chose a time discretization of 0.01 seconds. During this time, the action applied by the agent remains constant and after one such time step, the agent can choose a new action This results in a controller frequency of 100 Hz. A high frequency gives the agent a high degree of control which is often not possible to achieve in real systems. The experiments in this thesis are based on the same time discretization of 0.01 seconds but only allow the actor to choose a new action every ten time steps, keeping it constant for the time steps in between. This yields a controller frequency of 10 Hz. Combined with the differential equations, the choice of time discretization completely defines the interface between the actor and the bicycle system. The resulting transition dynamics used for the interaction of the controller and

Algorithm 1 Sampling bicycle transitions

Let $\mathbf{f}_{\text{bicycle}}$ denote the transition function of the bicycle benchmark. The minimal and maximal values for the state variables and the actions can be found in tables 2.1 and 2.2.

function SAMPLEBICYCLESTATE
  $\left(\theta, \dot{\theta}, \omega, \dot{\omega}\right) \leftarrow \mathcal{N}(\mathbf{0}, 1/4 \cdot \text{diag}\left(\theta^{\max}, \dot{\theta}^{\max}, \omega^{\max}, \dot{\omega}^{\max}\right))$
  $x \leftarrow \mathbb{U}(x^{\min}, x^{\max})$
  $y \leftarrow \mathbb{U}(y^{\min}, y^{\max})$
  $\psi \leftarrow \mathbb{U}(-\pi, \pi)$
  return $\left(\theta, \dot{\theta}, \omega, \dot{\omega}, x, y, \psi\right)$

function SAMPLEACTION
  $d \leftarrow \mathbb{U}(d^{\min}, d^{\max})$
  $T \leftarrow \mathbb{U}(T^{\min}, T^{\max})$
  return (d, T)

function SAMPLETRANSITIONS(N)
  for $i \leftarrow 1, N$ **do**
    $\boldsymbol{s_i} \leftarrow$ SAMPLEBICYCLESTATE()
    $\boldsymbol{a_i} \leftarrow$ SAMPLEACTION()
    $\boldsymbol{s'_i} \leftarrow \mathbf{f}_{\text{bicycle}}(\boldsymbol{s_i}, \boldsymbol{a_i})$
  return $\left(\left(\boldsymbol{s_1}, \boldsymbol{a_1}, \boldsymbol{s'_1}\right), \left(\boldsymbol{s_2}, \boldsymbol{a_2}, \boldsymbol{s'_2}\right), \ldots, \left(\boldsymbol{s_N}, \boldsymbol{a_N}, \boldsymbol{s'_N}\right)\right)$

Algorithm 2 Sampling a bicycle trajectory

Let $\mathbf{f}_{\text{bicycle}}$ denote the transition function of the bicycle benchmark.

function SAMPLEBICYCLESTARTSTATE
  $(\_, \_, \_, \_, x, y, \psi) \leftarrow$ SAMPLEBICYCLESTATE()
  return $(0, 0, 0, 0, x, y, \psi)$

function SAMPLETRAJECTORY
  $\boldsymbol{s_0} \leftarrow$ SAMPLEBICYCLESTARTSTATE()
  $t \leftarrow 0$
  while $\boldsymbol{s_t}$ is not terminal **do**
    $\boldsymbol{a_t} \leftarrow$ SAMPLEACTION()
    $\boldsymbol{s_{t+1}} \leftarrow \mathbf{f}_{\text{bicycle}}(\boldsymbol{s_t}, \boldsymbol{a_t})$
    $t \leftarrow t + 1$
  return $\left(\left(\boldsymbol{s_0}, \boldsymbol{a_0}, \boldsymbol{s_1}\right), \left(\boldsymbol{s_1}, \boldsymbol{a_1}, \boldsymbol{s_2}\right), \ldots, \left(\boldsymbol{s_{t-1}}, \boldsymbol{a_{t-1}}, \boldsymbol{s_t}\right)\right)$

Algorithm 3 Sampling a bicycle data set

```
1: function SampleMixedBicycleDataSet(N)
2:     D ← ∅
3:     while |D| < N do
4:         T ← SampleTrajectory()
5:         R ← SampleTransitions(|T|)
6:         D ← D ∪ T ∪ R
7:     return D
```

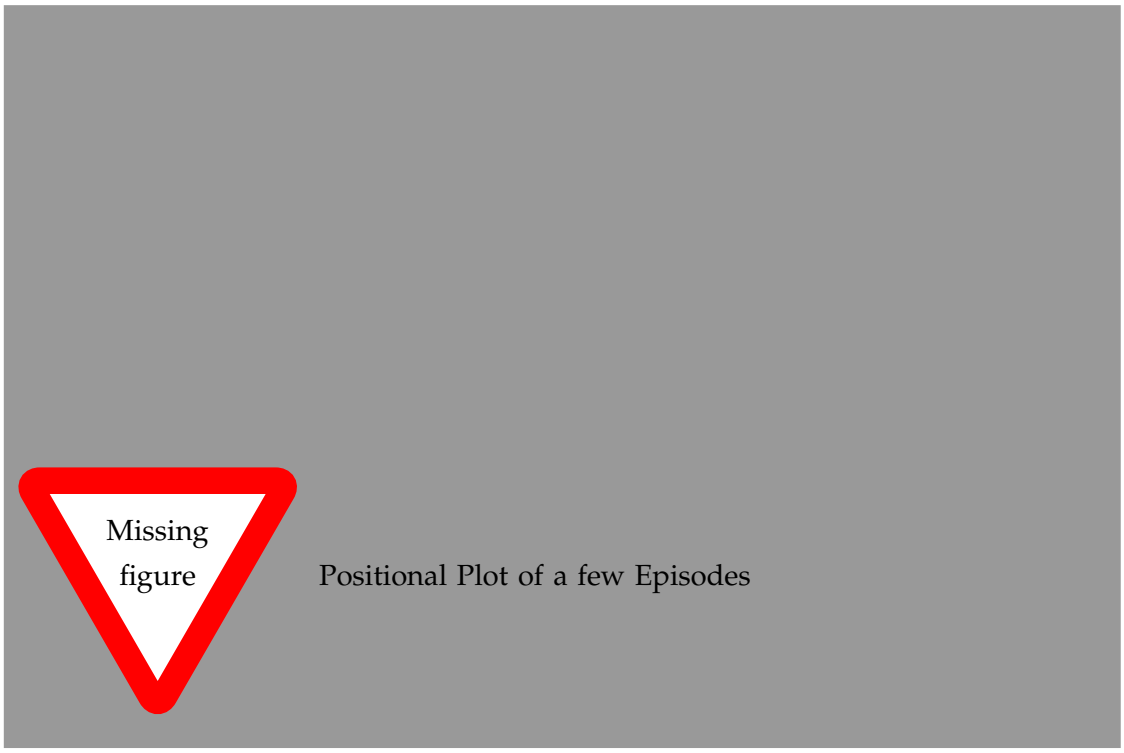


Figure 4.2: Positional Plot of a few Episodes

the system are called $\mathbf{f}_{\text{bicycle}}$ in the following.

Applying a controller to the bicycle benchmark produces time series beginning at some starting state and ending when the cyclist either falls down or reaches the goal. It is assumed that no expert knowledge is available, so the data sets available for learning transition dynamics should not be based a controller which can successfully balance the bicycle. Instead, this thesis chooses an uninformed controller which applies random actions to the system.

Algorithms 1 to 3 describe how data sets were created for the experiments. A data set consists of both complete trajectories and single random samples from the state space. A trajectory always starts in an upright position, that is, the state variables θ, $\dot{\theta}$, ω and $\dot{\omega}$ are all set to zero, while the remaining positional variables are sampled uniformly. This is both a sensible assumption and increases the mean lengths of the sampled trajectories when compared to more random starting states. As shown in figures 4.1a and 4.2, an average trajectory in the data set is quite short, since random actions are not suitable to balance the bicycle.

Figure 4.1b shows this in more detail, as it depicts the values of ω for a typical trajectory. While the bicycle starts in an upright position, it quickly starts leaning heavily towards one side and, since the controller does not choose actions to stabilize the bicycle, falls over. The sampled trajectories do not contain many state transitions where the bicycle drives straight or the actions counteract falling.

In order to reduce this bias, the data set also contains random samples from the complete state space as shown in algorithm 3. While those random samples add more balanced observations of the system, they also increase the difficulty of the learning problem. Not every combination of angles in the state space is sensible and can be reached from an upright starting state by applying actions. The transition models therefore also have to learn irrelevant information about the dynamics. A heuristic to reduce the amount of improbable states is to not sample the angles uniformly but rather to sample them from a broad random distribution around zero, resampling values which fall outside of the range of allowed values. Since terminal states are modelled separately, both the last transition of a trajectory and all samples which result in a terminal state are removed from the data set.

4.1.2 Gaussian Process Models

The Gaussian process models for the transition dynamics are trained using data sets of the form $\mathcal{D} = \{(\boldsymbol{s_i}, \boldsymbol{a_i}, \boldsymbol{s'_i}) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mid i \in [N]\}$ of pairs of states and actions and their

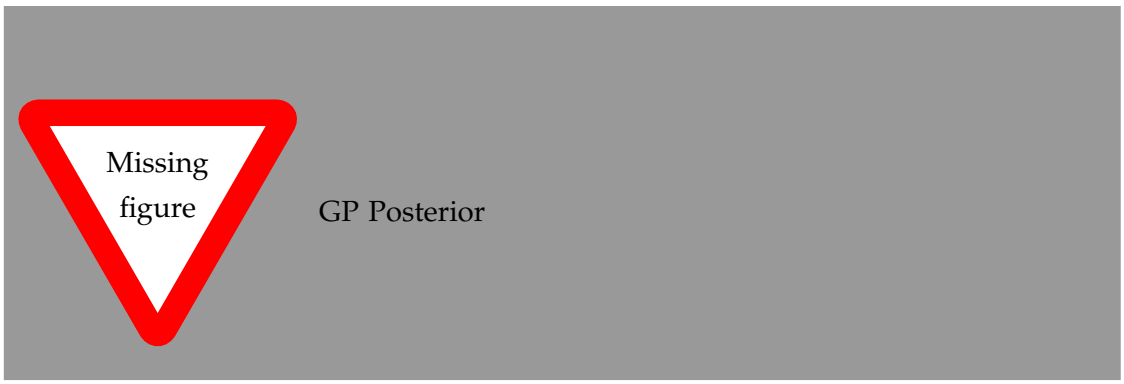


Figure 4.3: GP posterior

corresponding following state $s'_i = \mathbf{f}_{\text{bicycle}}(s_i, a_i)$. Since the transition dynamics of the bicycle benchmark are deterministic, these observations have no probabilistic element and they are not noisy. The model f of the transition dynamics is a compact statistical representation of this collected knowledge and is to be used to predict successive states of unobserved combinations of states and actions (s_*, a_*).

Besides predicting a concrete following state, the model should provide a measure about the uncertainty of its predictions. Since there is no randomness in the dynamics themselves, this uncertainty comes from the imperfect information about the true system dynamics and is dependent on the location of both the training data and the required predictions. If a query is made to the model in a part of the state space in which it has not seen many observations, the model should express its uncertainty and not assume that its best guess is close to the truth.

The Gaussian processes presented in section 3.2 represent a distribution over all plausible transition dynamics given a data set. In figure figure 4.3, the x-axis represents pairs of states and actions while the y-axis represents the successive state. Since the observations are noise-free, the GP is completely certain about predicting them and, since it assumed a smooth RBF-prior, it is also confident about predicting states closed to the observed data. Between the data points, uncertainties are higher since there are many different models which are plausible. Gaussian processes are called *non-degenerate*, since for predictions far away from the training set, the predicted uncertainty does not converge to zero. In contrast, for parts of the input space without any knowledge, the GP falls back to the prior assumptions about uncertainties and the mean function. Given a large enough data set which is spread out through the complete state and action space, the model becomes more and more confident about its predictions and

converges towards the true transition dynamics.

Gaussian processes as presented in this thesis can only model functions with univariate output. Approximating successive states requires multivariate predictions however. While there do exist extensions of Gaussian processes for multidimensional output [Raso6], a common solution is to train D separate GPs, one for every dimension in the state space $\mathbb{R}^D$. While this requires longer training time, it allows choosing a different set of hyperparameters for every dimension. Since the training set does not contain transitions which result in terminal states, the transition models do not know about them. The signature of the function represented by the transition model is $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, where $\mathcal{S} = \mathbb{R}^D$ and $\mathcal{A} = R^k$. Similar to the absence of terminal states, the transition models are also not aware of the rectangular boundaries of the state space described in table 2.1, which means that it is possible for the transition models to predict illegal states, which also have to be handled separately.

All models are trained using the squared exponential kernel presented in definition 10. The bicycle benchmark represents a physical system, which makes smoothness of the transition function a natural assumption. The RBF kernel is the standard choice in Gaussian process regression when no special knowledge about the shape of the transition function is available.

maybe mention other kernels? Is there anything to say about them?

Opposed to learning successive states directly, the training targets for the dth dimension are the differences to the current state given by

$$\Delta s_{i,d} := \mathbf{f}_{\text{bicycle}}(\boldsymbol{s_i}, \boldsymbol{a_i})_d - s_{i,d} = s'_{i,d} - s_{i,d},$$

where $i \in [N]$ and $d \in [D]$. This can be advantageous since differences vary less than the original function [Dei10]. Learning differences can also introduce independences in the data, since predicting the change in position of the bicycle only depends on the direction of movement but not on the previous position. Having learned models for the differences, the mean and the variance of the Gaussian posterior state distribution $\mathrm{p}(f_d(\boldsymbol{s}_*, \boldsymbol{a}_*))$ is given by

$$\begin{aligned}\mathbb{E}[f_d(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*] &= s_{*,d} + \mathbb{E}[\Delta s_{*,d} \mid \boldsymbol{s}_*, \boldsymbol{a}_*],\\ \operatorname{var}[f_d(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*] &= \operatorname{var}[\Delta s_{*,d} \mid \boldsymbol{s}_*, \boldsymbol{a}_*],\end{aligned}$$

respectively, since the prior state is considered constant and non-probabilistic. Uncertainties in the predictions only originate from the amount of confidence expressed by the models for the differences. The values of the expected value and variances are calculated according to lemma 11.

Since the different Gaussian processes are trained independently of each other and their training sets only contain their respective output dimension, their predictions are conditionally independent given the input. With the predictive distribution for the single dimension being Gaussian, the joint predictive state distribution is also Gaussian with a diagonal covariance matrix and is given by

$$\begin{aligned}
\mathrm{p}(f(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*) &= \mathcal{N}\big(f(\boldsymbol{s}_*, \boldsymbol{a}_*) \,\big|\, \boldsymbol{\mu}_f, \boldsymbol{\Sigma}_f\big), \text{ where} \\
\boldsymbol{\mu}_f &= \begin{pmatrix} \mathbb{E}[f_1(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*] \\ \vdots \\ \mathbb{E}[f_D(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*] \end{pmatrix} \\
\boldsymbol{\Sigma}_f &= \operatorname{diag}\left(\operatorname{var}[f_1(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*], \cdots, \operatorname{var}[f_D(\boldsymbol{s}_*, \boldsymbol{a}_*) \mid \boldsymbol{s}_*, \boldsymbol{a}_*]\right).
\end{aligned}$$

This diagonal covariance matrix illustrates the implicit independence assumption of the different output dimensions introduced by training one model per output dimension. While this assumption is not true in most cases, it can be used as an approximation and generally yields good results.

The state of the bicycle system is given by a vector $(\theta, \dot{\theta}, \omega, \dot{\omega}, x, y, \psi)$ composed of the internal dynamics of the bicycle and its position and orientation in euclidean space. During simulation with the transition model, the coordinates were transformed to polar coordinates given by

$$\begin{aligned}
\varphi(x, y) &:= \operatorname{atan2}(y, x) \\
r(x, y) &:= \sqrt{x^2 + y^2},
\end{aligned}$$

where atan2 is the arctangent function with two arguments. Polar coordinates uniquely represent a two-dimensional point by its angle to the x-axis and its distance to the origin. This representation both increases model performance and simplifies calculating the bicycle's relative position to the goal in the origin.

Additionally, representing both φ and ψ as numbers between $-\pi$ and π leads to a loss of information. While two angles with absolute value close to π but opposite signs are close together on a circle, their representations have a large euclidean distance. A Gaussian Process using the RBF-Kernel cannot recognize their similarity. In this case, it is possible to choose a specialized periodic variant of the squared exponential kernel which recognizes periodicity. Equivalently, an angle can be represented as a complex number on the unit circle, replacing it by its sine and cosine. Therefore, the internal representation of a bicycle state in the simulation is given by a vector

$$\boldsymbol{s} = (\theta, \dot{\theta}, \omega, \dot{\omega}, \varphi, r, \psi) \in \mathbb{R}^7$$

which is transformed to

$$\hat{\boldsymbol{s}} = (\theta, \dot{\theta}, \omega, \dot{\omega}, \sin\varphi, \cos\varphi, r, \sin\psi, \cos\psi) \in \mathbb{R}^9$$

when presented to the GPs. The transition model consists of seven Gaussian processes, each with nine-dimensional input.

The models are implemented in Python using Titsias's sparse variational GP regression implemented in *GPy* [GPy12] and trained using expectation maximization as presented in section 3.2.4. The optimization of the likelihood function is calculated using scaled conjugated gradients with multiple restarts to avoid local minima.

The performance of the transition models is highly dependent on the size of the training set N and the number of inducing inputs M. For N smaller than 35000, the performance of the transition models for long-term predictions was not good enough to allow PSO-P to succeed for any of the approaches presented below. Conversely, for large N and M larger than 250, the models are good enough such that PSO-P finds perfect solutions for all approaches. The experiments in this thesis focus on choices for N and M which are inbetween these extremes and where information about the model uncertainties can be used to improve performance. The next section presents the classic approach of long-term predictions without the use of uncertainty information, which is used as a baseline for comparison for the following techniques.

4.2 Predictions without Uncertainties

The transition model trained on a predefined data set allows the prediction of a successive state distribution $p(\boldsymbol{s}_1)$ given a deterministic pair of a state and an action $(\boldsymbol{s}_0, \boldsymbol{a}_0)$. To evaluate the action value function $\hat{\mathbf{V}}$, two extensions need to be made. Firstly, beyond specifying the goal, chapter 2 does not define a concrete reward function. This section introduces a variant of the reward function used by Randløv and Alstrøm in [RA98].

And secondly, for a time horizon T longer than one step into the future, the predictive state distributions $p(\boldsymbol{s}_1)$ up to $p(\boldsymbol{s}_T)$ are required. Since the GP dynamics model returns a Gaussian predictive distribution for all states beyond the starting state to account for the model uncertainty, all states beyond the starting state are no longer deterministic. In order to mimic a classic non-Bayesian model without a measure of uncertainty, the approach presented in this section discards this information and considers the maximum-a-posteriori estimation to be the deterministic prediction of the

transition model. Having established the deterministic mode of evaluating the action value function, this section finally introduces the evaluation setup used in this thesis and discusses the results of applying this technique to the bicycle system.

4.2.1 Deterministic Bicycle Reward Function

Solving the bicycle benchmark is a composite problem. An agent has to both learn to balance a bicycle and drive to the goal. Instead of having to solve the two tasks one after the other, they both have to be solved simultaneously, switching between them. While an agent is in control of the bicycle, it can try to drive towards the goal. If any action applied to the system leads to the danger of falling over however, the agent has to quickly change its focus towards preventing this.

Without expert knowledge available, the controller must learn this distinction autonomously, given the reward function. The most basic and uninformed reward function possible assigns positive reward for reaching the goal, a punishment (in the form of negative reward) for falling over and weighs all other states equally between the two extremes. While this can be enough to teach the short-term task of avoiding to fall down, the agent has no initiative of driving towards the goal besides actually hitting it. For most situations, the goal cannot be reached within the time-horizon of one PSO-P instance. In this setting, PSO-P would optimize towards a trajectory for which the chance of falling down is minimal. This trajectory is a circle with large radius [RA98].

To give the controller a chance of reaching the goal, it has to receive some hint about the correct direction to drive. Encoding this information in the reward function goes against the assumption of the complete absence of expert knowledge. If it is too detailed, it introduces the risk of significantly simplifying the learning problem or pushing the agent towards a policy which is only locally optimal. This reduction of the hard problem of finding the goal to a series of easier problems of driving in the correct direction and then going straight is called *shaping* [SB98; RA98].

The hint towards the goal encoded in the reward function should be a term which represents information which is local in the sense that its value can change considerably within the time horizon. The most simple term to consider is a punishment based on the current distance to the goal. This formulation is problematic however, since the agent should not care about the actual distance rather than the change of distance with respect to the previous state, which cannot be expressed in the reward. While an increase in reward would express movement in the correct direction, for any non-linear

punishment, the amount of increase is dependent on the current position in the input space and can lead to numerical problems if it gets to small. If it were linear, the punishment might at some point be larger than the punishment for falling. At this point, the agent's correct choice would be to fall down as quickly as possible.

In order to avoid these problems, the reward function used in this thesis uses the current angle between the frame of the bicycle and the direction towards the goal as a hint. Since the goal's position is at the origin of the coordinate system, this angle can be calculated as the difference of the current rotation of the bicycle ψ and the angular component of its polar coordinates in space φ. This difference can be scaled to an interval between zero and one and can take values in the complete range if the time horizon is long enough to contain a curve of the bicycle. The reward is scaled to be zero if the bicycle points straight away from the goal and one if directly towards it. If the bicycle has reached the goal, the agent receives a constant award of two which is double the amount which can be obtained for any state which is not in the goal. Similarly, the reward for falling is constant zero.

Definition 15 (Deterministic Bicycle Reward Function)
Given the set $\mathcal{S}^+$ of possible states of the bicycle benchmark, the *deterministic bicycle reward function* is defined as

$$\mathbf{R}_{\text{bicycle}} : \begin{cases} \mathcal{S}^+ \to \mathbb{R} \\ \quad s \mapsto \begin{cases} 2 & \text{if } s \in \mathcal{T}_{\text{goal}} \\ 0 & \text{if } s \in \mathcal{T}_{\text{fallen}} \\ 1 - \frac{|\psi_s - \varphi_s|}{\pi} & \text{otherwise} \end{cases} \end{cases}$$

where ψ_s and φ_s denote the respective angles in state s. The difference is defined to take values between $-\pi$ and π.

The reward function assigns constant reward for terminal states and reward inversely proportional to the angle between the bicycle's heading and the goal otherwise.

The agent does not receive negative reward for falling down. Instead, the reward is zero, which is equal to pointing away from the goal. The agent is still punished when falling down though, since the episode has to end and the agent is not able to collect additional reward by driving towards the goal.

Appendix A

Lists of Figures, Tables and Algorithms

List of Figures

List of Tables

List of Algorithms

Appendix B

Bibliography

[Åst71] Karl J. Åström. *Introduction to Stochastic Control Theory*. Elsevier, Feb. 27, 1971. 318 pp. ISBN: 9780080955797.

[BCF10] Eric Brochu, Vlad M. Cora, and Nando de Freitas. "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning". In: *arXiv:1012.2599 [cs]* (Dec. 12, 2010). arXiv: 1012.2599. URL: http://arxiv.org/abs/1012.2599 (visited on 02/01/2016).

[DA] David Duvenaud and Ryan P. Adams. "Black-Box Stochastic Variational Inference in Five Lines of Python". In: (). URL: http://people.seas.harvard.edu/~dduvenaud/papers/blackbox.pdf (visited on 04/03/2016).

[Dam15] Andreas Damianou. "Deep Gaussian processes and variational propagation of uncertainty". PhD thesis. University of Sheffield, 2015. URL: http://etheses.whiterose.ac.uk/id/eprint/9968 (visited on 02/01/2016).

[Dei10] Marc Peter Deisenroth. "Efficient Reinforcement Learning using Gaussian Processes". PhD thesis. KIT Scientific Publishing, 2010. URL: http://www.cs.washington.edu/research/projects/aiweb/media/papers/tmppqidj5 (visited on 04/19/2016).

[DFR15] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. "Gaussian processes for data-efficient learning in robotics and control". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 37.2 (2015), pp. 408–423. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6654139 (visited on 02/01/2016).

[DR11] Marc Deisenroth and Carl E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472. URL: http://machinelearning.wustl.edu/mlpapers/paper_files/ICML2011Deisenroth_323.pdf (visited on 02/01/2016).

[DRF] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. "Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning". In: (). URL: http://core.ac.uk/download/pdf/241164.pdf (visited on 02/01/2016).

[Eng06] Andries P. Engelbrecht. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006. URL: http://dl.acm.org/citation.cfm?id=1199518 (visited on 02/01/2016).

[ES00] Russ C. Eberhart and Yuhui Shi. "Comparing inertia weights and constriction factors in particle swarm optimization". In: *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. Vol. 1. IEEE, 2000, pp. 84–88. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=870279 (visited on 05/13/2016).

[GPy12] GPy. *GPy: A Gaussian process framework in python*. GitHub. 2012. URL: https://github.com/SheffieldML/GPy (visited on 05/20/2016).

[Hei+16] Daniel Hein et al. "Reinforcement Learning with Particle Swarm Optimization Policy (PSO-P) in Continuous State and Actionspaces". In: 7.3 (July 2016).

[Kut01] Wilhelm Kutta. "Beitrag zur näherungweisen Integration totaler Differentialgleichungen". In: (1901). URL: http://www.citeulike.org/group/1448/article/813805 (visited on 05/20/2016).

[McK10] Wes McKinney. "Data Structures for Statistical Computing in Python". In: Proceedings of the 9th Python in Science Conference. 2010, pp. 51–56. URL: https://conference.scipy.org/proceedings/scipy2010/mckinney.html (visited on 05/20/2016).

[MR16] Rowan McAllister and Carl Edward Rasmussen. "Data-Efficient Reinforcement Learning in Continuous-State POMDPs". In: *arXiv preprint* (2016). DOI: arXiv:1602.02523. URL: http://arxiv.org/abs/1602.02523 (visited on 02/28/2016).

[Mur12] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Aug. 24, 2012. 1098 pp. ISBN: 9780262018029.

[P+08] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. "The matrix cookbook". In: *Technical University of Denmark* 7 (2008), p. 15. URL: http://www.cim.mcgill.ca/~dudek/417/Papers/matrixOperations.pdf (visited on 02/01/2016).

[Pre07] William H. Press. *Numerical Recipes: The Art of Scientific Computing*. 3rd Edition. Cambridge University Press, Sept. 6, 2007. 1195 pp. ISBN: 9780521880688.

[RA98] Jette Randløv and Preben Alstrøm. "Learning to Drive a Bicycle Using Reinforcement Learning and Shaping." In: *ICML*. Vol. 98. Citeseer, 1998, pp. 463–471. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.3038&rep=rep1&type=pdf (visited on 04/15/2016).

[Ras06] Carl Edward Rasmussen. "Gaussian processes for machine learning". In: (2006). URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.3414 (visited on 02/01/2016).

[RN10] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010. 1153 pp. ISBN: 9780136042594.

[Ros95] Guido Rossum. *Python Reference Manual*. Amsterdam, The Netherlands, The Netherlands: CWI (Centre for Mathematics and Computer Science), 1995.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. 1998. URL: https://books.google.de/books?hl=de&lr=&id=CAFR6IBF4xYC (visited on 02/01/2016).

[SG05] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian processes using pseudo-inputs". In: *Advances in neural information processing systems*. 2005, pp. 1257–1264. URL: http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2005_543.pdf (visited on 02/01/2016).

[Sne07] Edward Lloyd Snelson. "Flexible and efficient Gaussian process models for machine learning". PhD thesis. Citeseer, 2007. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.4041&rep=rep1&type=pdf (visited on 02/01/2016).

[SS02] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Jan. 2002. 658 pp. ISBN: 9780262194754.

[Tit09] Michalis K. Titsias. "Variational learning of inducing variables in sparse Gaussian processes". In: *International Conference on Artificial Intelligence and Statistics*. 2009, pp. 567–574. URL: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS09_Titsias.pdf (visited on 02/01/2016).

[TL10] Michalis K. Titsias and Neil D. Lawrence. "Bayesian Gaussian process latent variable model". In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 844–851. URL: http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2010_TitsiasL10.pdf (visited on 02/01/2016).

[Wat+15] Manuel Watter et al. "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 2746–2754. URL: http://papers.nips.cc/paper/5964-embed-to-control-a-locally-linear-latent-dynamics-model-for-control-from-raw-images.pdf (visited on 04/03/2016).

[WCV11] S. van der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation". In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 22–30. ISSN: 1521-9615. DOI: 10.1109/MCSE.2011.37.