Motor Planning with Monte Carlo Sampling Methods

Prevenhueber Oliver, BSc.
prevenhueber@student.tugraz.at

Institute for Theoretical Computer Science (IGI)
Graz University of Technology
Inffeldgasse 16b
8010 Graz, Austria

Master Thesis

Supervisor: o.Univ.-Prof.Dr.Wolfgang Maass
Assessor: o.Univ.-Prof.Dr.Wolfgang Maass

May, 2013

I hereby certify that the work presented in this thesis is my own work and that to the best of my knowledge it is original except where indicated by reference to other authors.

Ich bestätige hiermit, diese Arbeit selbständig verfasst zu haben. Teile der Diplomarbeit, die auf Arbeiten anderer Autoren beruhen, sind durch Angabe der entsprechenden Referenz gekennzeichnet.

Prevenhueber Oliver, BSc.

Acknowledgements

First, I would like to thank my advisor Prof. Wolfgang Maass for the opportunity to write my master thesis in the exciting field of computational intelligence. Special thanks to Dipl.-Ing. Elmar Rückert for his patience answering my countless questions and his helpful advices.

I would also thank my family for their help and support during my study, especially my mother and grandparents who believed in me all time.

This master thesis was supported by the AMARSi project, funded by the European Union and part of the Seventh Framework Programme (FP7).

Oliver Prevenhueber, May 2013

Abstract

Motor planning algorithms are essential for the development of robust autonomous robot systems. Various approaches exist to compute movement trajectories efficiently by applying quadratic control costs. However, with quadratic costs hard constraints cannot be adequately modelled. In this thesis I choose the Monte Carlo (MC) sampling approach to investigate how dynamic motor planning tasks, considering hard constraints can be solved efficiently. For efficient sampling, Gibbs sampling, rejection sampling, and importance sampling are combined. Two different sampling methods are investigated. The first and simpler method does not consider the dynamic state transition model of a robot. The second method is more sophisticated and considers a linearised approximation of this dynamic model. The experiments range from simple tasks on a 2-link robot arm to tasks using a more complex 4-link robot arm. To enhance the performance of the investigated methods, they are extended by a via point approach. Finally, in a novel trajectory mixing approach complex planning scenarios are solved by mixing multiple trajectories, which are computed in parallel.

Keywords: motor planning, hard constraints, Monte Carlo sampling, stochastic optimal control, Gibbs sampling, rejection sampling, importance sampling.

Kurzfassung

Bewegungsplanungsalgorithmen sind für die Entwicklung von robusten autonomen Robotersystemen essentiell. Es existieren verschiedene Ansätze, um Bewegungstrajektorien mit quadratischen Kosten effizient zu berechnen. Allerdings können harte Beschränkungen mit quadratischen Kosten nicht adäquat modelliert werden. In dieser Arbeit habe ich den Monte Carlo (MC) sampling Ansatz ausgewählt, um zu untersuchen wie dynamische Bewegungsplanungsaufgaben mit harten Beschränkungen effizient gelöst werden können. Um effizient Stichproben zu ziehen, werden Gibbs sampling, rejection sampling und importance sampling kombiniert. Dazu werden zwei unterschiedliche Stichprobenverfahren untersucht. Die erste und simplere Methode berücksichtigt das dynamische Zustandsmodell eines Roboters nicht. Die zweite Methode ist komplexer und berücksichtigt eine linearisierte Approximation dieses dynamischen Modells. Die Experimente umfassen einfache Aufgabenstellungen an einem zweigliedrigen Roboterarm und Aufgabenstellungen, welche einen komplexeren viergliedrigen Roboterarm verwenden. Um die Leistungsfähigkeit der untersuchten Methoden zu steigern, werden diese durch einen via point Ansatz erweitert. Zuletzt werden komplexe Planungsszenarien mit einem neuen Trjaektorienmischungs-Ansatz mit mehreren parallel berechneten Trajektorien gelöst.

Contents

List of Figures

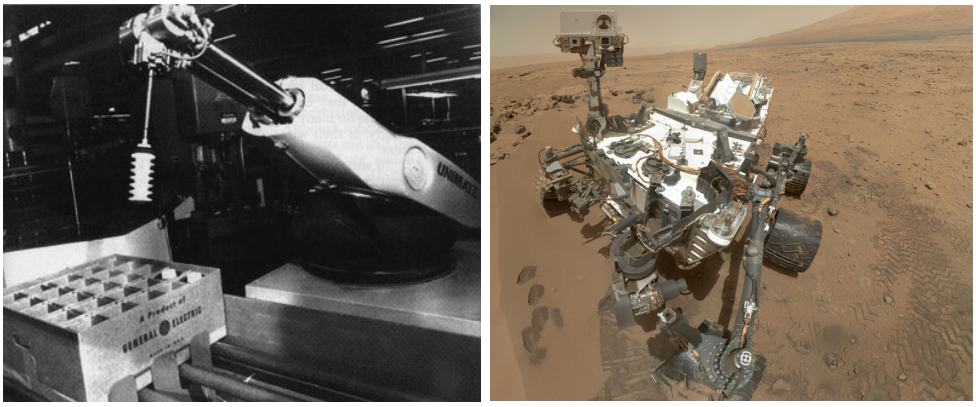
List of Tables

1. Introduction

Warum beglückt uns die herrliche, das Leben erleichternde, Arbeit ersparende Technik so wenig? Die einfache Antwort lautet: weil wir noch nicht gelernt haben, einen vernünftigen Gebrauch von ihr zu machen. Im Kriege dient sie dazu, daß wir uns gegenseitig verstümmeln. Im Frieden hat sie unser Leben hastig und unsicher gestaltet. Statt uns weitgehend von geisttötender Arbeit zu befreien, hat sie die Menschen zu Sklaven der Maschine gemacht, die meist mit Unlust ihr eintöniges, langes Tagewerk vollbringen und stets um ihr armseliges Brot zittern müssen.

Albert Einstein (1879-1955)

During the last decades robots evolved from simple systems with one special purpose to highly complex systems including numerous sensors, communication devices and the ability for performing complex movements. The development process is illustrated by the first industrial robot *Unimate* described in Engelberger [9] and the actual mars rover *Curiosity*, illustrated in Figure 1.1.
To extend the scope of application of complex robot systems, scientists are developing algorithms to enhance their autonomy. These autonomous robot systems are very useful for numerous different fields and become even more important in the future. They help reducing risk of human life by replacing people in dangerous environments and take over important processing parts in industry due to the fact that these systems need no permanent control. The more sophisticated autonomous systems are, the more tasks can be handed over to them and the higher their advantages will be. Integrating autonomous systems in human environment also raise challenges. For instance, losing control over highly complex autonomous systems cause more drastic consequences to the environment than losing control over non-autonomous systems. For example, the loss of control over a self-steering car affect the environment more drastically compared to an industrial robot without control. Hence the responsibility of system engineers is high and their aim is to



(a) (b)

Figure 1.1: The first industrial robot *Unimate* [12] described in Engelberger [9] is shown in (a). In (b) the NASA's mars rover *Curiosity* on the Mars is depicted, published by National Aeronautics and Space Administration (NASA) [23].

construct reliable systems. Motor planning tasks are essential for developing sophisticated autonomous robot systems to act appropriate, even in unknown environments. As a result, numerous research disciplines are investigating efficient and robust motor planning algorithms.

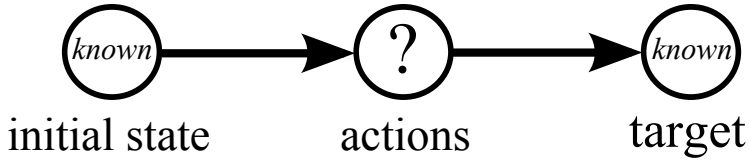


Figure 1.2: In motor planning the initial state and the target are known. The goal of motor planning is to compute the actions to reach the target.

Motor planning in general means the computation process in order to execute a series of motions. Motor planning is simple for humans, but it takes years until a series of complex movements are generated by interconnected brain areas intuitively [6]. Recent experiments in psychological science, presented in Griffiths et al. [10] suggest that statistical predictions determined by the human brain are close to Bayesian models. In Buesing et al. [5] and Pecevski et al. [24] it was shown that networks of spiking neurons can perform probabilistic inference. The applied sampling approach (neural dynamics sampling) is closely related to Gibbs sampling, which is used in this thesis. Due to this close relationship it is possible to use the obtained results from this thesis for neural dynamic sampling on similar tasks.

In motor planning typically the initial and the final state are known and the corresponding actions have to be computed (see Figure 1.2). In general, motor planning tasks can be solved by different movement trajectories, e.g. reaching a target with a robot

arm can be performed by different arm constellations. Motor planning algorithms are applied to determine the optimal movement trajectory solving the task, considering defined features and goals. This features and goals constrain the task solution. For instance, avoiding an obstacle on the path to the target or restricting the movement speed of the robot arm is implemented by features that limit possible solutions. This task constraints are implemented in general by a cost function to prioritise different features. A graphical model including multiple features implemented by a cost function is illustrated in Figure 1.3. Here, the costs are denoted by $c_1, \ldots, c_K$ and might be different for individual states or actions.

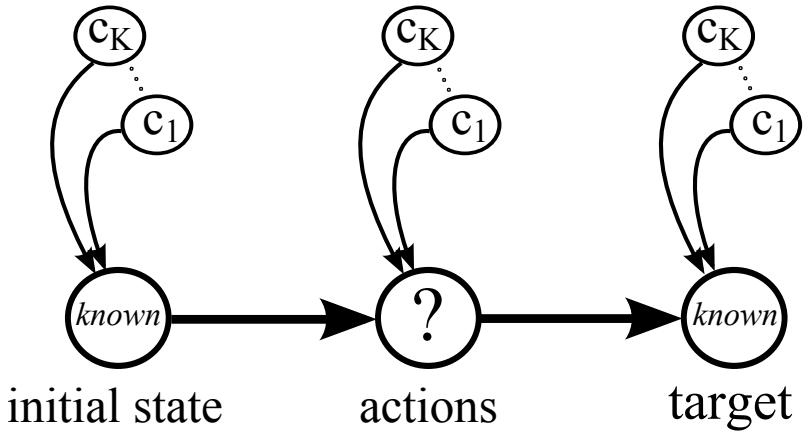


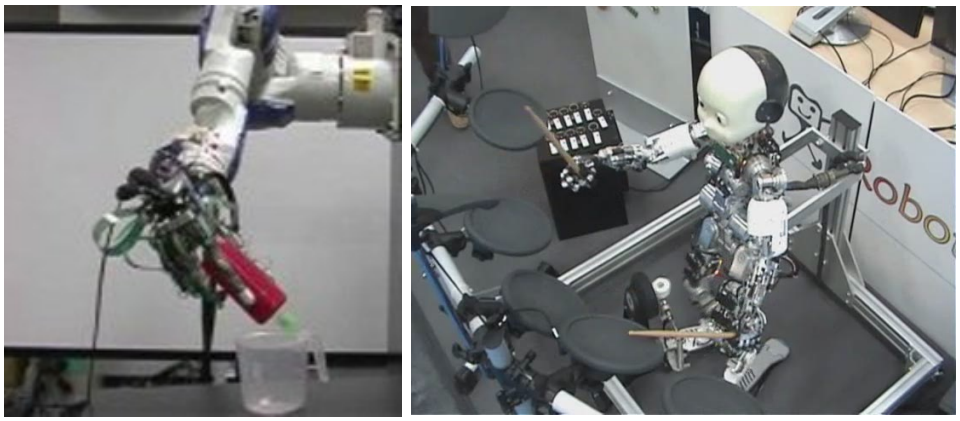
Figure 1.3: The graph shows a general motor planning task considering multiple features encoded by costs $c_{1:K}$. In general, features are implemented by a cost function.

1.1 Motivation

State of the art Stochastic Optimal Control (SOC) algorithms, i.e. Iterative Linear Quardratic Regulator (ILQR) [27] or Approximate Inference Control (AICO) [28] are very efficient planning methods, assuming quadratic costs. Most real world tasks, however, include obstacles which can only be modelled by hard constraints and therefore cannot be solved properly by these methods [20]. The following examples of advanced autonomous robot systems illustrate the importance of hard constraints in practical tasks appearing in various fields of application. The requirements of autonomous robot systems dealing with hard constraints depend on the purpose of the autonomous system.

In various real world scenarios movements are restricted by static constraints. These static constraints are represented by buildings, walls, or other fixed objects while processing and executing a movement. Therefore, the definition of static constraints depends on the time horizon of the task. Due to their temporal and spacial invariance, static constraints must be recognised only once. Slopping a ball from a cup into a jug described in

Kroemer et al. [19], illustrates a task including hard constraints by grasping the cup and the restrictions by the jug, showed in Figure 1.4 (a). The drumming task illustrated in Figure 1.4 (b) represents a complex task including a combination of rhythmic and discrete movements considering hard constraints. Drums are the hard constraints that need to be hit with a certain force. This scenario was used in Degallier et al. [7] to test a modular architecture for movement generation.



(a) (b)

Figure 1.4: In (a) a grasping task including hard constraints is illustrated. The goal of the task is to slop a ball from a cup into the jug [19]. The *iCub* robot performing the drumming task is shown in (b) [3]. For successful drumming, the robot has to combine rhythmic and discrete movements using drumsticks.

In contrast, tasks with temporal and spacial changing constraints are more challenging due to limited processing time and additional sensors necessary for permanent constraint recognition. The tasks become even more difficult with a increasing task constraint variability in time. In a table tennis task this fast variability is represented by the position of the ball. The constraints are recognised in real time by a visual ball tracking system. The task is illustrated in Figure 1.5, taken from Muelling et al. [22].

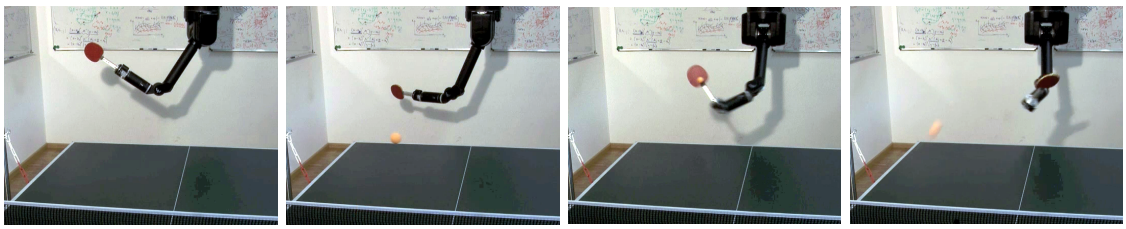


Figure 1.5: The series of figures show a robot arm playing table tennis applying a mixture of movement primitives. To be able to fulfil the requirements a high speed motion robot arm and a visual ball tracking system are used [22].

1.2 My Contributions to existing Methods

The goal of the thesis is to investigate sampling methods for motor planning tasks focussing on hard constraints. Therefore, a combination of Gibbs sampling, importance sampling, and rejection sampling from Bishop [4] are applied to draw appropriate samples. In the existing literature only Gibbs sampling is used to model distributions with sample based methods. Two sampling approaches are investigated in this thesis. First, the model-free sampling approach (described in Section 4.1) draws samples from a Gaussian distribution without considering the model of the robot arm. Second, a model-based sampling approach (discussed in detail in Section 4.2) is applied to draw samples, considering the dynamic model of the robot arm. Both investigated sampling approaches are discussed in the context of optimal control methods for movement planning using quadratic cost functions defined in state and task space [31].

In the evaluated implementation, the necessary parameters for both approaches are optimised by applying a learning framework similar to the framework presented in Rückert et al. [25]. The framework applied in this thesis (described in Section 4.4) does not include parallel model learning and uses a linear feedback controller instead of a nonlinear feedback controller. To investigate the practical relevance of the applied sampling-based methods, the performance and trajectories are compared to an implementation of AICO [28]. Finally, in a novel trajectory mixing approach complex planning scenarios are solved by mixing multiple trajectories, which are computed in parallel.

1.3 Notation

The notation used in this thesis follows the one in the book *Pattern Recognition and Machine Learning* [4].

x	...	scalar value
$\mathbf{x} = (x_1, x_2, \ldots, x_N)$	...	column vector containing N elements
x_i	...	i-th element of vector $\mathbf{x}$
$\mathbf{x}^T$	...	transposed vector $\mathbf{x}$ equals a row vector
X	...	random variable
$\mathbf{X}$	...	matrix
$\mathbf{X}^T$	...	transposed matrix $\mathbf{X}$
$\mathbf{X}^{-T}$	...	inverted and transposed matrix $\mathbf{X}$
$x_{i,j}$	...	element of matrix $\mathbf{X}$ on i-th row and j-th column
$\text{diag}(\mathbf{X})$	...	diagonal elements of matrix $\mathbf{X}$
$\mathbf{I}$	...	unit matrix (all elements are zero except the diagonal elements with value one)
$p(.)$	...	probability distribution

1.4 Outline

The following section covers background information about related models and describes state of the art methods for solving motor planning tasks. Section 3 explains graphical models, probability theory basics, the applied graphical model for planning, and probabilistic inference methods, followed by a description of the used sampling methods in general. A detailed discussion about motor planning tasks applying the chosen sampling methods are given in Section 4. In Section 5 the performed experiments and the obtained results are explained. The work of the thesis and the results are discussed in Section 6 and are concluded in Section 7.

2. Related Work

In Kappen [17] stochastic control problems are distinguished based on their time horizon which can be either finite or infinite. This thesis focuses on the special case of control problems with a finite time horizon and linear controls at every time step applying quadratic control costs, which can be solved efficiently. The methods applied to solve this class of control problems are called Stochastic Optimal Control (SOC) methods [18]. One well studied approach originating from system theory implements motor planning as graphical model inference problem [31]. Here, the system dynamics are approximated by Linear Quadratic Gaussian (LQG) systems. LQG systems approximate non-linear stochastic control problems by linearised models using Gaussians. The advantage of this approach is that optimal controls can be approximated efficiently by approximate inference algorithms [18]. For inference in graphical models numerous algorithms exist, i.e. Laplace approximation, variational methods, Expectation Propagation (EP), and Monte Carlo (MC) sampling. For completeness the mentioned algorithms are summarised briefly.

The basic idea of Laplace approximation is to compute the optimal control trajectory by solving deterministic equations, approximating the path integral [16]. The path integral describes the probability of all possible paths from a starting point to a target. The efficiency and simple usage are the main advantages of the Laplace approximation. The drawbacks of the Laplace approximation are that a large set of data points is necessary for good approximation results and that the search behaviour is local [4].

Variational methods transform the existing inference problem into an optimisation problem [13]. For that, variational methods introduce a family of probability distributions to approximate a desired probability distribution. This family of probability distributions should be simple enough to be tractable but must also scale with the complexity of the problem [13]. The approximation quality is measured by the Kullback-Leibler divergence [2, 13]. One distribution is chosen out of the family of distributions by minimising the Kullback-Leibler divergence with respect to the variational parameters. By choosing all approximation distributions of the family, the best approximation within the family is obtained. The advantages of variational methods are their fast computation and that variational methods are not limited to Gaussian distributions [14]. Variational methods can be also combined with other approximation methods, e.g. sampling methods [2, 14, 13]. However, the obtained variational approximation model is not applicable in general but

only for the specific inference problem, solved by variational approximation [13].

The EP algorithm described in Minka [21] is a special case of variational methods. EP approximates a desired probability distribution also by a family of approximating distributions but compute the Kullback-Leibler divergence by applying assumed-density filtering. In experiments described in Mensik et al. [20], EP demonstrated remarkable performance compared to MC sampling techniques.

Two sophisticated SOC methods applying probabilistic inference are the Iterative Linear Quardratic Regulator (ILQR) [27] and Approximate Inference Control (AICO) [28] algorithms. Both approaches yield promising results, also applying high dimensional tasks [28]. Therefore, the methods investigated in this thesis are compared to AICO. ILQR is based on iterative Sequential Quadratic Programming (SQP), optimising the cost function of the "*global trajectory ... over the full time interval*", [28]. By contrast, AICO applies approximate inference to optimise several local messages to optimise a distribution of trajectories. Due to quadratic costs, the probability hitting an obstacle with the trajectory must be non-zero for these methods [20]. Therefore, a trajectory will probable move through obstacles instead of passing around. To be able to deal with hard constraints using AICO, truncated Gaussian EP can be used as described in Toussaint [29]. With this extension AICO is able to avoid collisions successfully. However, for each obstacle one truncated Gaussian is needed, which is computational demanding [29].

As an alternative, MC sampling methods are proposed for stochastic control problems with hard constraints [20]. MC sampling methods approximate a complex probability distribution by only drawing samples from that distribution. Thus, the shape of the complex distribution does not need to be known. MC sampling methods and possible applications are discussed in detail in Doucet and Johansen [8]. The main drawback of these methods are their computational effort by drawing sufficient samples for an appropriate approximation. Another issue arises if sampling from the desired distribution is difficult. In Mensik et al. [20] the applied MC sampling methods suffer from infinite costs resulting from hard constraints. In this thesis a combination of MC sampling methods is used to circumvent these problems. A more detailed description of the applied sampling methods is given in Section 4.

While the discussed approaches in general can be applied to complex motor control tasks, a common strategy to simplify the control problem is to use Movement Primitives (MPs) [26]. Dynamic Movement Primitives (DMPs) [26] generate a movement trajectory with fixed shape, determined by a parameterised dynamical system and followed by a linear feedback controller [25]. Planning Movement Primitives (PMPs) were introduced in Rückert et al. [25] which build on AICO. In comparison to DMPs, with PMPs the intrinsic costs have to be determined for each time step. The shape of the trajectory is updated at each time step considering the intrinsic costs and applying a time-varying linear feedback controller. Therefore, the trajectory has no fixed shape as in the DMPs case and hence is able to react on a changing environment [25]. I also apply this concept of using MPs.

3. Background on Probability Theory and Inference Methods

In this section probabilistic models for motor planning tasks are discussed. First, graphical models in general are described. Second, probabilistic inference methods for graphical models are discussed. The third part is dedicated to the definition of the cost function used for motor planning. The following information about graphical models and inference are presented more detailed in Bishop [4].

3.1 Graphical Models and Probability Theory

Graphical models are convenient to represent information in a structured form to simplify necessary computations. In graphical models, random variables are represented by nodes and the connecting links define the probabilistic relations between the random variables. Graphical models involve various representations (for instance, Bayesian Networks, Markov Random Fields, and Factor Graphs) with different characteristics, illustrated in Figure 3.1. Bayesian Networks are directed graphical models (a), were Markov Random Fields are undirected graphical models (b). Factor Graphs introduce factor nodes in addition to nodes and links (c). Graphical models can be transformed into different representations, but the transformation must not be unique (for instance, different factor graphs exist for the same undirected graph in Figure 3.1).

In probability theory a probability distribution of a random variable X which takes the value x_i $(i = 1, \ldots, N)$, is defined as $p(X = x_i)$. The probability distribution over multiple random variables, i.e. X, Y that take the values x_i and y_j $(j = 1, \ldots, M)$ is defined as $p(X = x_i, Y = y_j)$ and called joint distribution. The probability over one random variable X given multiple random variables X, Y is obtained by integrating out all other random variables, called marginal distribution

$$p(X = x_i) = \int_y p(X = x_i, Y = y_j) dy.$$

The conditional probability distribution of a random variable $X = x_i$ given $Y = y_j$ is defined as

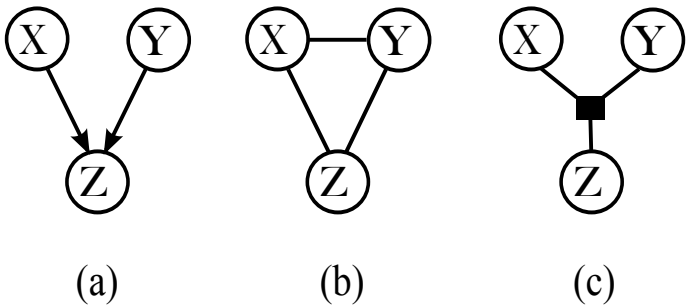


Figure 3.1: (a) A directed graph with three random variables. (b) The corresponding undirected graph. (c) One equivalent factor graph with a factor node, represented by the black square. [4]

$$p(X = x_i|Y = y_j) = \frac{p(X=x_i,Y=y_j)}{p(Y=y_j)},$$

where the distributions are normalised to probability 1, i.e. $\sum_{X,Y} p(X,Y) = 1$. A more convenient way to specify a probability distribution $p(X = x)$ for a particular value x is the notation $p(x)$, whereas the probability distribution over random variable X is denoted as $p(X)$.
If two random variables are independent, the following expressions are implied:

$$\begin{aligned} p(X,Y) &= p(X)p(Y), \\ p(X|Y) &= p(X). \end{aligned}$$

The following two equations are the fundamental rules of probability theory:

$$\text{sum rule} \qquad p(X) = \int_Y p(X,Y)$$

$$\text{product rule} \quad p(X,Y) = p(Y|X)\ p(X)$$

The Bayes Theorem is derived from the product rule in combination with the symmetry property $p(X,Y) = p(Y,X)$ and is defined as

$$p(Y|X) = \frac{p(X|Y)\ p(Y)}{p(X)}.$$

An alternative representation of Bayes Theorem is given by

$$posterior = \frac{likelihood \times prior}{evidence}.$$

The *posterior* defines the probability of an event with a *likelihood* based on the *prior*

knowledge and the *evidence* variables. This representation of Bayes Theorem is convenient to understand the process of computing the *posterior* given the known variables. Explaining inference, different types of variables must be defined referring to the second representation of Bayes Theorem.

$E_{i=1:m}$ ... observed variables, defining the *evidence*
$H_{i=1:n}$ ... hidden (unobserved) variables
$Y_{i=1:k}$ ... unobserved variables, the *posterior* is requested

The Bayes Theorem can be rewritten using the introduced variables as

$$p(Y_{1:k}|E_{1:m}) = \frac{p(E_{1:m}|Y_{1:k})\ p(Y_{1:k})}{p(E_{1:m})},$$

and approximated by

$$p(Y_{1:k}|E_{1:m}) \propto \sum_{H_{1:n}} p(E_{1:m}, H_{1:n}|Y_{1:k})\ p(Y_{1:k}),$$

which is called probabilistic (or approximate) inference. Probabilistic inference is discussed in more detail in Bishop [4].

3.2 Inference in Graphical Models for Motor Planning

Probabilistic inference in combination with graphical models are described in Jordan and Weiss [15] and are widely applied in state of the art methods. The advantage of this methods is a possible complexity reduction compared to unstructured representations, where information is added in form of the structure of the graphical model. Therefore, stochastic processes for motor planning in general are described briefly. Due to the focus on Stochastic Optimal Control (SOC) methods, approximate inference implemented by message passing and approximate inference implemented by Monte Carlo (MC) sampling methods are discussed in detail afterwards. The stochastic process for motor planning (illustrated in Figure 3.2) is given as:

$$p(\mathbf{q}_{1:T}, \mathbf{u}_{1:T}, z_{1:T}) = p(\mathbf{q}_1) \prod_{t=1}^{T-1} p(\mathbf{u}_t|\mathbf{q}_t) \prod_{t=1}^{T-1} p(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{u}_{t-1}) \prod_{t=1}^{T-1} p(z_t|\mathbf{q}_t, \mathbf{u}_t),$$

The robot states are denoted by the state vector $\mathbf{q}_{1:T}$, whereas controls are defined as $\mathbf{u}_{1:T-1}$, constraints are denoted by $z_{1:T}$, and the time horizon is specified by T. The term $p(\mathbf{q}_1)$ denotes the initial state distribution and $p(\mathbf{u}_t|\mathbf{q}_t)$ defines the prior distribution of the controls. The state transition model is defined as $p(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{u}_{t-1})$ and the distribution $p(z_t|\mathbf{q}_t, \mathbf{u}_t)$ denotes the forward kinematic model. To determine the state vetor $\mathbf{q}_{1:T}$ con-

sidering constraints $z_{1:T}$, the posterior distribution $p(\mathbf{q}_{1:T}|\mathbf{u}_{1:T}, z_{1:T})$ must be computed.

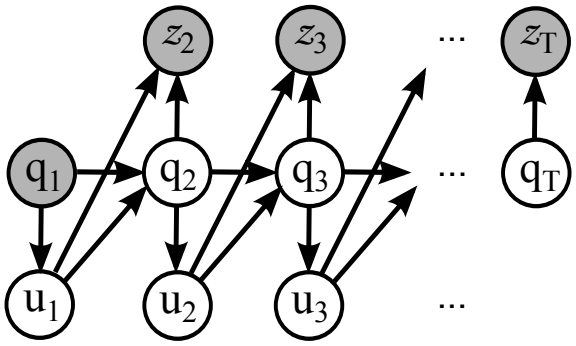


Figure 3.2: Graphical model used for motor planning tasks with finite time horizon T. Robot states are represented by nodes, with grey shaded nodes for known states. Dependencies between states are represented by arrows, connecting the nodes.

3.3 Approximate Inference implemented by Message Passing

To efficiently compute probabilistic inference in arbitrary graphs, the message passing algorithm for chain graphs and the sum-product algorithm for tree graphs were invented. To correctly compute probabilistic inference with these methods, the graph must not contain loops. The probability $p(X)$ considering the model in Figure 3.3 is computed by

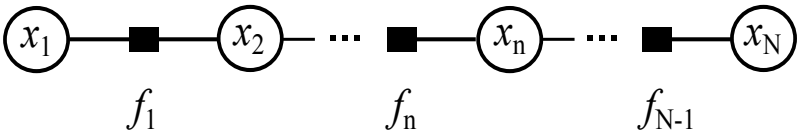


Figure 3.3: The factor graph represents a chain of values $x_{1:N}$ of the random variable X connected by factors $f_{1:N-1}$

$$p(X) = \frac{1}{Z} f_1(x_1, x_2) f_2(x_2, x_3) \dots f_{N-1}(x_{N-1}, x_N).$$

The unknown normalisation constant Z is only necessary if the factor graph was derived from an undirected graphical model. The marginal distribution $p(x_n)$ is computed by summing out all other possible values of X:

$$p(x_n) = \sum_{x_1} \dots \sum_{x_{n-1}} \sum_{x_{n+1}} \dots \sum_{x_N} p(X).$$

By rearranging the order of sums the term can be written as the product of two parts.

The forward message $\mu_\alpha(x_n)$ includes the sums over $x_{1:n-1}$ and the backward message $\mu_\beta(x_n)$ includes the sums over $x_{n+1:N}$ (see Figure 3.4).

forward message:

$$\begin{aligned}\mu_\alpha(x_n) &= \sum_{x_{n-1}} f_{n-1}(x_{n-1}, x_n) \dots \sum_{x_2} f_2(x_3, x_2) \sum_{x_1} f_1(x_2, x_1), \\ &= \sum_{x_{n-1}} f_{n-1}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}),\end{aligned}$$

backward message:

$$\begin{aligned}\mu_\beta(x_n) &= \sum_{x_{n+1}} f_{n+1}(x_n, x_{n+1}) \dots \sum_{x_N} f_{N-1}(x_{N-1}, x_N), \\ &= \sum_{x_{n+1}} f_{n+1}(x_n, x_{n+1}) \mu_\beta(x_{n+1}).\end{aligned}$$

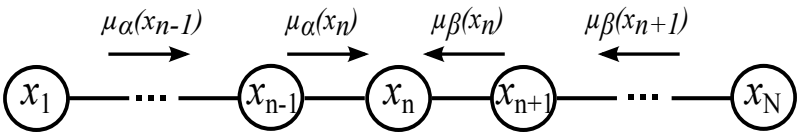


Figure 3.4: The graphical model depicts the recursive message passing through the chain for the value x_n [4].

Now the sum-product algorithm is applied to compute the messages for the whole graphical model, dividing the model into subtrees connected to the node x as illustrated in Figure 3.5. The joint distribution $p(\mathbf{x})$ is computed by the product over all subtrees connected to the node x and written as

$$p(\mathbf{x}) = \prod_{s \in ne(x)} F_s(x, X_s).$$

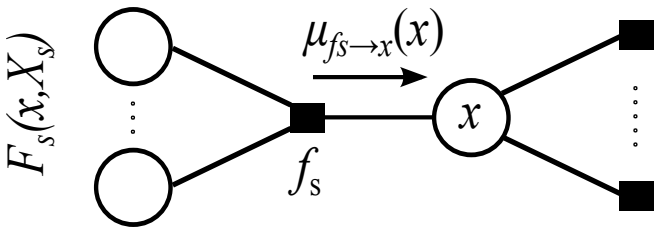


Figure 3.5: The graphical model is divided into various subtrees connected to the node x [4].

The term $ne(x)$ denotes the set of factor nodes connected to variable node x and X_s denotes the set of all variables connected to node x by the factor node f_s of the subtree s. $F_s(x, X_s)$ denotes the product over all factors connected to factor f_s of the subtree s. By

summing out all variables X_s we obtain the marginal distribution

$$\begin{aligned} p(x) &= \prod_{s \in ne(x)} \left[\sum_{X_s} F_s(x, X_s) \right], \\ &= \prod_{s \in ne(x)} \mu_{f_s \to x}(x). \end{aligned}$$

The term $\mu_{f_s \to x}(x)$ denotes the message from the factor node f_s to variable node x. The subtree can be again divided into subtrees until the leaf nodes are reached. The messages of the leaf nodes are defined as

$$\mu_{f \to x}(x) = f(x) \quad \text{if the leaf node is a factor node,}$$

$$\mu_{x \to f}(x) = 1 \quad \text{if the leaf node is a variable node.}$$

The sum-product algorithm evaluates all messages starting from the leaf nodes of the model to the node x. The algorithm is discussed in more detail in Bishop [4]. Applying these general methods on the graphical model of motor planning, we obtain the following equations and the corresponding Figure 3.6.

$$p(x_t) = \alpha_t(x_t)\beta_t(x_t)\rho_t(x_t),$$

forward message:

$$\alpha_t(x_t) = \sum_{x_{t-1}} p(x_t|x_{t-1})\rho_{t-1}(x_{t-1})\alpha_{t-1}(x_{t-1}),$$

backward message:

$$\beta_t(x_t) = \sum_{x_{t+1}} p(x_{t+1}|x_t)\rho_{t+1}(x_{t+1})\beta_{t+1}(x_{t+1}),$$

actual message:

$$\rho_t(x_t) = p(y_t|x_t).$$

The conditional probability $p(y_t|x_t)$ denotes the probability of the forward kinematic model and $p(x_t|x_{t+1})$ denotes the state transition probability from state x_t to state x_{t+1}. By using Gaussian distributions specified by Equation B.1 in the Appendix and applying a linear state space model with following equations

$$\begin{aligned}
&\text{initial state distribution} && p(\mathbf{x}_0) && = \mathcal{N}(\mathbf{x}_0|\mathbf{a}_0, \mathbf{Q}_0),\\
&\text{state transition model} && p(\mathbf{x}_{t+1}|\mathbf{x}_t) && = \mathcal{N}(\mathbf{x}_{t+1}|\mathbf{A}_t\mathbf{x}_t + \mathbf{a}_t, \mathbf{Q}_t),\\
&\text{observation model} && p(\mathbf{y}_t|\mathbf{x}_t) && = \mathcal{N}(\mathbf{y}_t|\mathbf{D}_t\mathbf{x}_t + \mathbf{d}_t, \mathbf{C}_t),
\end{aligned}$$

we obtain the following (canonical Gaussian) messages:

$$\begin{aligned}
\alpha_t(\mathbf{x}_t) &= \mathcal{N}[\mathbf{x}_t|\mathbf{s}_t, \mathbf{S}_t],\\
&\text{with } \mathbf{S}_t = (\mathbf{Q}_{t-1} + \mathbf{A}_{t-1}(\mathbf{R}_{t-1} + \mathbf{S}_{t-1})^{-1}\mathbf{A}_{t-1}^T)^{-1},\\
&\text{and } \mathbf{s}_t = \mathbf{S}_t(\mathbf{a}_{t-1} + \mathbf{A}_{t-1}(\mathbf{R}_{t-1} + \mathbf{S}_{t-1})^{-1}(\mathbf{s}_{t-1} + \mathbf{r}_{t-1})),\\
\beta_t(\mathbf{x}_t) &= \mathcal{N}[\mathbf{x}_t|\mathbf{v}_t, \mathbf{V}_t],\\
&\text{with } \mathbf{V}_t = (\mathbf{A}_t^{-1}(\mathbf{Q}_t + (\mathbf{R}_{t+1} + \mathbf{V}_{t+1})^{-1})\mathbf{A}_t^{-T})^{-1},\\
&\text{and } \mathbf{v}_t = \mathbf{V}_t\mathbf{A}_t^{-1}((\mathbf{R}_{t+1} + \mathbf{V}_{t+1})^{-1}(\mathbf{r}_{t+1} + \mathbf{v}_{t+1}) - \mathbf{a}_t),\\
\rho_t(\mathbf{x}_t) &= \mathcal{N}[\mathbf{x}_t|\mathbf{r}_t, \mathbf{R}_t],\\
&\text{with } \mathbf{R}_t = \mathbf{D}_t^T\mathbf{C}_t^{-1}\mathbf{D}_t,\\
&\text{and } \mathbf{r}_t = \mathbf{D}_t^T\mathbf{C}_t^{-1}(\mathbf{y}_t - \mathbf{d}_t).
\end{aligned}$$

The state noise model is denoted by $\mathbf{Q}$ and $\mathbf{C}_t$ denotes the model precision matrix. The marginal probability distribution $p(\mathbf{x}_t)$ is computed by the product of the three messages:

$$\begin{aligned}
p(\mathbf{x}_t) &= \mathcal{N}[\mathbf{x}_t|\mathbf{s}_t, \mathbf{S}_t]\mathcal{N}[\mathbf{x}_t|\mathbf{r}_t, \mathbf{R}_t]\mathcal{N}[\mathbf{x}_t|\mathbf{v}_t, \mathbf{V}_t]\\
&= \mathcal{N}(\mathbf{x}_t|\mathbf{b}_t, \mathbf{B}_t)\\
&\text{with } \mathbf{B}_t = (\mathbf{S}_t + \mathbf{R}_t + \mathbf{V}_t)^{-1}\\
&\text{and } \mathbf{b}_t = \mathbf{B}_t(\mathbf{s}_t + \mathbf{r}_t + \mathbf{v}_t)
\end{aligned}$$

This message passing approach with Gaussians is implemented by the AICO algorithm in Toussaint [28].

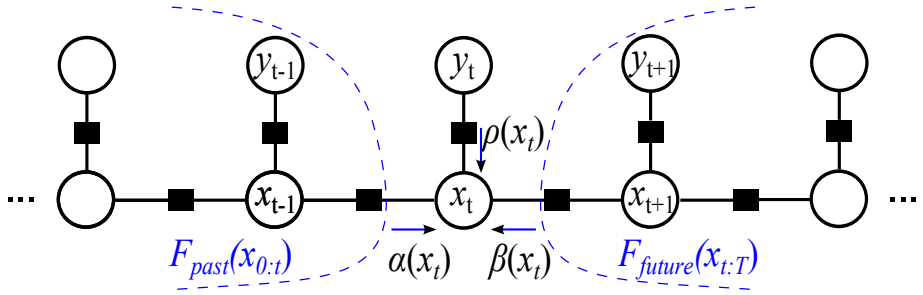


Figure 3.6: The example illustrates message passing by the sum-product algorithm for node x_t. The forward message $\alpha(x_t)$ includes all messages from nodes $x_{1:t-1}$ of the subtree $F_{past}(x_{0:t})$, whereas the backward message $\beta(x_t)$ includes the messages from nodes $x_{t+1:T}$ of the subtree $F_{future}(x_{t:T})$. The message $\rho(x_t)$ considers the defined constraints y_t of node x_t.

3.4 Approximate Inference implemented by Monte Carlo Sampling

In comparison to other SOC methods, sampling methods do not approximate the optimal trajectory by Gaussian kernels but by sampling from Gaussian distributed samples. The following section discusses the applied sampling methods to solve motor planning tasks.

3.4.1 Gibbs Sampling

Markov Chain Monte Carlo (MCMC) sampling methods are an alternative method to solve integration and optimisation problems [1]. The idea behind Monte Carlo (MC) sampling, is to approximate a target distribution $p(x)$ by drawing independent and identically distributed (i.i.d) samples from $p(x)$. In most cases, sampling from the target distribution $p(x)$ is not feasible. Hence, samples must be drawn from a proposal distribution $q(x)$ and be evaluated by a normalising constant. To be able to apply MCMC sampling methods, the used graphical model must be a Markov chain with finite state space $\mathbf{x}_t = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$. A stochastic process is a Markov chain if the Markov property is fulfilled: $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_1) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ [1]. Figure 3.2 shows that the stochastic process is a Markov chain, where state $\mathbf{x}_t$ depends only on the state $\mathbf{x}_{t-1}$ and controls $\mathbf{u}_{t-1}$. MCMC sampling is implemented in this thesis by Gibbs sampling, which is a special case of the popular Metropolis-Hastings algorithm. Gibbs sampling is used to update each value of the stochastic process $p(\mathbf{x}) = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ with the Markov property. In each update step the value $\mathbf{x}_t$ is updated by a value drawn from the distribution $p(\mathbf{x}_t|\mathbf{x}_{\setminus t})$. With $\mathbf{x}_t$ as t-th value of $\mathbf{x}$ and $p(\mathbf{x}_t|\mathbf{x}_{\setminus t}) = p(\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T)$. The T update steps can be performed in a defined order or in random sequence. Due to the focus on hard constraints in this thesis, samples violating the constraints are rejected by rejection

sampling before assigning probabilities. As a result, problems with zero probabilities are avoided, in contrast to MC sampling implemented in Mensik et al. [20]. Subsequently, to obtain optimal update states, the drawn and valid samples are weighted by importance sampling.

3.4.2 Rejection Sampling

Rejection sampling is applied, if a direct model by sampling from a desired distribution $p(z)$ is difficult to generate. Assume that $p(z)$ can be evaluated for any value z up to an unknown normalising constant Z_p by

$$p(z) = \frac{1}{Z_p}\tilde{p}(z).$$

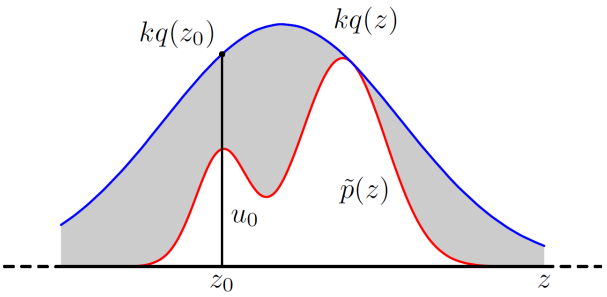


Figure 3.7: The figure taken from Bishop [4] illustrates the idea of rejection sampling. The samples drawn from the grey shaded area, bounded by the approximated desired distribution $\tilde{p}(z)$ and the proposal distribution $q(z)$ are rejected. Only samples fulfilling the condition $u_0 \leq \tilde{p}(z_0)$ are accepted.

To approximate the complex distribution $p(z)$, a second distribution (proposal distribution) $q(z)$ is introduced. Sampling from the proposal distribution is simple and $kq(z) \geq p(z)$, with the constant k. The constant k should be as small as possible under the permission that $kq(z)$ must be always larger than $\tilde{p}(z)$. As first step, z_0 is drawn from the proposal distribution $q(z)$ and u_0 is drawn from the uniform distribution $[0, kq(z_0)]$. The values z_0 and u_0 are uniformly distributed under the distribution $kq(z)$. The second step includes the rejection of u_0 if $u_0 > \tilde{p}(z_0)$. Not rejected pairs of z_0 and u_0 approximate the distribution $\tilde{p}(z)$ proportional to the desired distribution $p(z)$. The probability of acceptance of a drawn sample is defined as

$$p_{accept} = \frac{1}{k}\int \tilde{p}(z).$$

3.4.3 Importance Sampling

The idea behind importance sampling, is to sample from a distribution $p(z)$ with respect to maximising the expectation of a function $f(z)$. That means, samples should be drawn from the region with maximal values of the product of $p(z)$ and $f(z)$. The expectation can be evaluated by the sum of the product over L samples drawn from $q(z)$.

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z})\ d\mathbf{z}, \\ &= \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})\ d\mathbf{z}, \\ &\simeq \frac{1}{L}\sum_{l=1}^{L}\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)}), \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} r_l f(\mathbf{z}^{(l)}).\end{aligned}$$

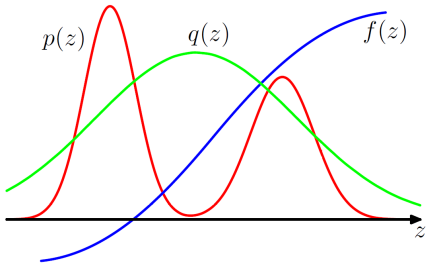


Figure 3.8: Importance sampling maximises the expectation of function $f(\mathbf{z})$ with respect to distribution $p(\mathbf{z})$. Because sampling from $p(\mathbf{z})$ is difficult, the samples are drawn from a simple proposal distribution $q(\mathbf{z})$. This figure is taken from Bishop [4].

The variable r_l denotes the importance weights and defines the weight of each drawn sample. If samples cannot be drawn from the distribution $p(\mathbf{z})$ directly, $p(\mathbf{z})$ must be approximated by $\tilde{p}(\mathbf{z})$ of the form

$$p(\mathbf{z}) = \frac{1}{Z_p}\tilde{p}(\mathbf{z}).$$

The distribution $\tilde{q}(z)$ follows the same argumentation, with the normalisation constant Z_q. In this case the expectation is represented by

$$\begin{aligned}\mathbb{E}[f] &= \frac{Z_q}{Z_p}\int f(\mathbf{z})\frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})}q(\mathbf{z})\ d\mathbf{z}, \\ &\simeq \frac{Z_q}{Z_p}\frac{1}{L}\sum_{l=1}^{L}\tilde{r}_l f(\mathbf{z}^{(l)}), \\ &\simeq \sum_{l=1}^{L} w_l f(\mathbf{z}^{(l)}), \quad \text{with } w_l = \frac{\tilde{r}_l}{\sum_m \tilde{r}_m}.\end{aligned}$$

3.5 Cost Functions for Motor Planning

In Section 3.2 the stochastic process and the corresponding model for motor planning were described. Further, both probabilistic inference methods focussed in this thesis, were

discussed in detail. In this section, the cost functions applied in AICO and both sampling methods are described. Referring to the implementation of AICO from Toussaint [28], the term $p(z_t|\mathbf{q}_t, \mathbf{u}_t)$ from Section 3.2 is used to introduce the cost function as

$$p(z_t = 1|\mathbf{q}_t, \mathbf{u}_t) \quad = \quad \exp(-c_t(\mathbf{q}_t, \mathbf{u}_t)).$$

The term $c_t(\mathbf{q}_t, \mathbf{u}_t)$ denotes the intrinsic cost function at time step t. The sum over all T time steps act as performance criteria of the computed trajectory τ

$$L(\tau) = \sum_{t=1}^{T} c_t(\mathbf{q}_t, \mathbf{u}_t). \tag{3.1}$$

Equations (11) - (13) in Toussaint [28] explain the relation of costs minimisation and finding the optimal trajectory τ^*. A trajectory τ in state space, represents a sequence of state-control pairs $\langle \mathbf{q}_{1:T}; \mathbf{u}_{1:T} \rangle$ to reach the defined target over time horizon T. The state vector $\mathbf{q}_t$ is represented by pairs of angles q and velocities $\dot{q}$ for all M joints $\mathbf{q}_t = [q_{1,t},\ \dot{q}_{1,t},\ \ldots,\ q_{M,t},\ \dot{q}_{M,t}]^T$.

The implementation of AICO from Toussaint [28] defines a quadratic cost function $c_t(\mathbf{q_t}, \mathbf{u_t})$ in state space by the term

$$\begin{aligned} c_t(\mathbf{q}_t, \mathbf{u}_t) &= (\mathbf{q}_t - \mathbf{q}_{target})^T \mathbf{R}_t (\mathbf{q}_t - \mathbf{q}_{target}), \\ &= (\mathbf{q}_t^T \mathbf{R}_t - \mathbf{q}_{target}^T \mathbf{R}_t)\ (\mathbf{q}_t - \mathbf{q}_{target}), \\ &= \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_{target} - \mathbf{q}_{target}^T \mathbf{R}_t \mathbf{q}_t + \mathbf{q}_{target}^T \mathbf{R}_t \mathbf{q}_{target}), \end{aligned}$$

due to identical form of $\mathbf{q}_t$ and $\mathbf{q}_{target}$, the term $\mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_{target}$ is equal to $\mathbf{q}_{target}^T \mathbf{R}_t \mathbf{q}_t$,

$$\begin{aligned} c_t(\mathbf{q}_t, \mathbf{u}_t) &= \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - 2(\mathbf{q}_{target}^T \mathbf{R}_t \mathbf{q}_t) + \underbrace{\mathbf{q}_{target}^T \mathbf{R}_t \mathbf{q}_{target}}_{constant}, \\ &\quad \text{with } \mathbf{r}_t = \mathbf{q}_{target}^T \mathbf{R}_t, \\ &\approx \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - 2\mathbf{r}_t^T \mathbf{q}_t, \end{aligned}$$

including the controls, the following approximation is obtained

$$c_t(\mathbf{q}_t, \mathbf{u}_t) \quad \approx \quad \mathbf{q}_t^T \mathbf{R}_t \mathbf{q}_t - 2\mathbf{r}_t^T \mathbf{q}_t + \mathbf{u}_t^T \mathbf{H}_t \mathbf{u}_t.$$

The matrices $\mathbf{R}_t$ and $\mathbf{H}_t$ denote the precision for state space vector $\mathbf{q}_t$ and controls $\mathbf{u}_t$.

For the evaluated MC sampling methods, the identical quadratic cost function as used

in AICO is applied

$$c_t(\mathbf{q}_t) \quad = \quad (\mathbf{q}_t - \mathbf{q}_{target})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{target}),$$

neglecting the control dependent part. Therefore, the cost function $c_t(\mathbf{q}_t)$ in state space of the model-based approach is defined by the term

$$c_t(\mathbf{q}_t) = \begin{cases} (\mathbf{q}_t - \mathbf{q}_{t-1})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{t-1}) + (\mathbf{q}_t - \mathbf{q}_{target})^T \mathbf{R}_T (\mathbf{q}_t - \mathbf{q}_{target}) & \text{if } t = T, \\ (\mathbf{q}_t - \mathbf{q}_{t-1})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{t-1}) + (\mathbf{q}_t - \mathbf{q}_{t+1})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{t+1}) & \text{else.} \end{cases} \tag{3.2}$$

The precision matrices $\mathbf{R}$ and $\mathbf{R}_T$ are invariant of time step t in contrast to precision matrices applied in AICO. Matrix $\mathbf{R}$ denotes the transition probability between two subsequent states $\mathbf{q}_t$ and $\mathbf{q}_{t+1}$. Precision matrix $\mathbf{R}_T$ denotes the transition probability between an arbitrary state $\mathbf{q}_t$ and the target $\mathbf{q}_{target}$. Hence, the costs $c_t(\mathbf{q}_t)$ represent the costs for transition from state $\mathbf{q}_{t-1}$ to $\mathbf{q}_t$ and from state $\mathbf{q}_t$ to $\mathbf{q}_{t+1}$. Transition matrix $\mathbf{R}_T$ is introduced to be able to sample in a wider range during the time sequence while ensuring a high precision at the target. Introducing $\mathbf{R}_T$ results in higher sampling flexibility and increased robustness against local minima and obstacles.

The corresponding cost function in task space with kinematic model $\mathbf{x}_t = \Phi(\mathbf{q}_t)$ is computed by the Euclidean distance between the state $\mathbf{x}_t$ and target $\mathbf{x}_{target}$

$$(\mathbf{x}_t - \mathbf{x}_{target})^T \mathbf{C}_T (\mathbf{x}_t - \mathbf{x}_{target}).$$

The task space vectors $\mathbf{x}_t$ and $\mathbf{x}_T$ are defined by x/y-coordinate pairs, specifying the endeffector positions of the joints and the endeffector position of the target. $\mathbf{C}_T$ denotes the corresponding precision matrix in task space.

$$c_t(\mathbf{x}_t) = \begin{cases} (\mathbf{q}_t - \mathbf{q}_{t-1})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{t-1}) + (\mathbf{x}_t - \mathbf{x}_{target})^T \mathbf{C}_T (\mathbf{x}_t - \mathbf{x}_{target}) & \text{if } t = T, \\ (\mathbf{q}_t - \mathbf{q}_{t-1})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{t-1}) + (\mathbf{q}_t - \mathbf{q}_{t+1})^T \mathbf{R} (\mathbf{q}_t - \mathbf{q}_{t+1}) & \text{else.} \end{cases} \tag{3.3}$$

The model-free approach applies the same cost function as the model-based approach, only replacing the precision matrices $\mathbf{R}$ and $\mathbf{R}_T$ by $\mathbf{R}_{trans}$ and $\mathbf{R}_{joint}$.

4. Motor Planning with Sampling Methods

4.1 Model-free Sampling Approach

The considered joint probability distribution of the model-free sampling approach

$$p(z_t, \mathbf{q}_t, \mathbf{q}_{t-1}) = p(z_t|\mathbf{q}_t)p(\mathbf{q}_t|\mathbf{q}_{t-1})p(\mathbf{q}_{t-1}),$$

is equivalent to the joint distribution of the kinematic motor planning, denoted by Equation (1) in Toussaint and Goerick [31] and corresponds to Figure 3.2. The distribution $p(\mathbf{q}_{t-1})$ denotes the probability distribution of state $\mathbf{q}_{t-1}$. The state transition probability $p(\mathbf{q}_t|\mathbf{q}_{t-1})$ denotes the multivariate Gaussian distribution from which the samples are drawn, specified by the covariance matrix $\mathbf{W}$

$$p(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{W}) = \mathcal{N}(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{W})$$

The forward kinematic model $p(z_t|\mathbf{q}_t)$ is identical to Equation (3) in Toussaint and Goerick [31]

$$p(z_t|\mathbf{q}_t) = \mathcal{N}(z_t|\mathbf{\Phi}(\mathbf{q}_t), \mathbf{R}^{-1}).$$

Matrix $\mathbf{R}^{-1}$ is representative for precision matrices $\mathbf{R}^{-1}_{trans}$ or $\mathbf{R}^{-1}_{joint}$, depending on time step t. The function $\mathbf{\Phi}(\mathbf{q}_t)$ denotes the two dimensional transformation function between the endeffector representation of task space and state space of a robot arm with M links

$$\mathbf{\Phi}(\mathbf{q}_t) = \begin{bmatrix} l_1 \cos(\phi_1) & l_1 \cos(\phi_1) + l_2 \cos(\phi_1 + \phi_2) & \dots & \sum_{i=1}^{M} l_i \cos\left(\sum_{j=1}^{i} \phi_j\right) \\ \\ l_1 \sin(\phi_1) & l_1 \sin(\phi_1) + l_2 \sin(\phi_1 + \phi_2) & \dots & \sum_{i=1}^{M} l_i \sin\left(\sum_{j=1}^{i} \phi_j\right) \end{bmatrix}$$

The link lengths of the robot arm are denoted by $l_{1:M}$ and the corresponding angles of the links are denoted by $\phi_{1:M}$. The first row of $\mathbf{\Phi}$ represents the x-coordinates of all M links,

whereas the y-coordinates are represented by the second row. For illustration assume a three link robot arm. The intermediate endeffector of the first arm link is represented by the x/y-coordinates of the first column of $\mathbf{\Phi}$. The coordinates of the intermediate endeffector of the first two joints are represented by the second column of $\mathbf{\Phi}$. And the endeffector coordinates of the entire arm is represented by the third column. Instead of computing the Jacobian as in the kinematic case in Toussaint and Goerick [31], rejection sampling is applied to enforce task constraints.

To infer state $\mathbf{q}_t$ the posterior distribution $p(\mathbf{q}_t|z_t, \mathbf{q}_{t-1})$ is computed. The posterior distribution is computed by applying importance sampling on the remaining samples. The posterior is determined by the Maximum A Posteriori (MAP) solution, choosing the sample with lowest costs $c_t(\mathbf{q}_t)$. The posterior must be computed for time steps $t = 2, \ldots, T$ to generate the trajectory τ. To improve the quality of trajectory τ, numerous rollouts are performed, each rollout starting with the previous trajectory as initial state vector. The number of rollouts performed by the algorithm depends on the task complexity and on computational resources.

4.2 Model-based Sampling Approach

In the model-based sampling approach a joint probability distribution similar to the model-free approach is applied

$$p(z_t, \mathbf{q}_t, \mathbf{q}_{t-1}, \mathbf{q}_{t+1}) = p(z_t|\mathbf{q}_t)p(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{q}_{t+1})p(\mathbf{q}_{t+1})p(\mathbf{q}_{t-1}).$$

The probability distribution of state $\mathbf{q}_{t-1}$ and the forward kinematic model $p(z_t|\mathbf{q}_t)$ are identical to the model-free approach. In addition, the probability distribution of $\mathbf{q}_{t+1}$ is considered. In comparison to the model-free approach, a linearised approximation of the non-LQG model by adapting the state transition probability $p(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{q}_{t+1})$ is included, according to the linearised model. The linearisation is implemented by Taylor series expansion, where the applied linear model is defined as $\mathbf{A}_t\mathbf{q}_t + \mathbf{B}_t\mathbf{u}_t$.

The following mathematical explanation describes how the sampling distribution is computed, considering a linear model. The samples are drawn each time step from the probability distribution

$$p(\mathbf{q}_t|\mathbf{q}_{t-1}, \mathbf{q}_{t+1}) = \mathcal{N}(\mathbf{q}_t|\mathbf{q}_{t-1})\ \mathcal{N}(\mathbf{q}_t|\mathbf{q}_{t+1}),$$

which is the product of two Gaussians representing the state transitions from state $\mathbf{q}_{t-1}$ to $\mathbf{q}_t$ and $\mathbf{q}_t$ to $\mathbf{q}_{t+1}$. The state at time step $t = 1$ is fixed as initial state $\mathbf{q}_1$ and therefore sampling is not applied.

$$\begin{aligned}
\mathcal{N}(\mathbf{q}_t|\mathbf{q}_{t-1}) &= \int_{\mathbf{u}_{t-1}} \mathcal{N}(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1} + \mathbf{a}_{t-1} + \mathbf{B}_{t-1}\mathbf{u}_{t-1}, \mathbf{Q})\ p(\mathbf{u}_{t-1})\ d\mathbf{u}_{t-1}, \\
&= \int_{\mathbf{u}_{t-1}} \mathcal{N}(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1} + \mathbf{a}_{t-1} + \mathbf{B}_{t-1}\mathbf{u}_{t-1}, \mathbf{Q})\ \mathcal{N}[\mathbf{u}_{t-1}|\mathbf{h}_{t-1}, \mathbf{H}]\ d\mathbf{u}_{t-1}. \\
\mathcal{N}(\mathbf{q}_{t+1}|\mathbf{q}_t) &= \int_{\mathbf{u}_t} \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q})\ p(\mathbf{u}_t)\ d\mathbf{u}_t, \\
&= \int_{\mathbf{u}_t} \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q})\ \mathcal{N}[\mathbf{u}_t|\mathbf{h}_t, \mathbf{H}]\ d\mathbf{u}_t.
\end{aligned}$$

$\mathbf{Q}$ ... constant state noise
$\mathbf{H}$ ... constant control weight matrix
$\mathbf{h}_t$ ... linear control cost term

The mathematical terms are now transformed to normal form by using Equation B.4 and integrated according to the propagation rule defined in Equation B.7.

$$\begin{aligned}
\mathcal{N}(\mathbf{q}_t|\mathbf{q}_{t-1}) &= \int_{\mathbf{u}_{t-1}} \mathcal{N}(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1} + \mathbf{a}_{t-1} + \mathbf{B}_{t-1}\mathbf{u}_{t-1}, \mathbf{Q}) \\
&\quad \mathcal{N}(\mathbf{u}_{t-1}|\mathbf{H}^{-1}\mathbf{h}_{t-1}, \mathbf{H}^{-1})\ d\mathbf{u}_{t-1}, \\
&= \mathcal{N}(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1} + \mathbf{a}_{t-1} + \mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{h}_{t-1}, \mathbf{Q} + \mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{B}_{t-1}^T), \\
&= \mathcal{N}(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1} + \mathbf{a}_{t-1} + \mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{h}_{t-1}, \mathbf{Q}^*_{t-1}), \\
&\quad \text{with } \mathbf{Q}^*_{t-1} = \mathbf{Q} + \mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{B}_{t-1}^T. \\
\mathcal{N}(\mathbf{q}_{t+1}|\mathbf{q}_t) &= \int_{\mathbf{u}_t} \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q})\ \mathcal{N}(\mathbf{u}_t|\mathbf{H}^{-1}\mathbf{h}_t, \mathbf{H}^{-1})\ d\mathbf{u}_t, \\
&= \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t, \mathbf{Q} + \mathbf{B}_t\mathbf{H}^{-1}\mathbf{B}_t^T), \\
&= \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t, \mathbf{Q}^*_t), \\
&\quad \text{with } \mathbf{Q}^*_t = \mathbf{Q} + \mathbf{B}_t\mathbf{H}^{-1}\mathbf{B}_t^T,
\end{aligned}$$

which can be written as (due to the symmetry property of Gaussian distributions)

$$\mathcal{N}(\mathbf{q}_{t+1}|\mathbf{q}_t) = \mathcal{N}(\mathbf{A}_t\mathbf{q}_t + \mathbf{a}_t + \mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t|\mathbf{q}_{t+1}, \mathbf{Q}^*_t).$$

After linear transformation using the Equation B.6 we obtain

$$\begin{aligned}\mathcal{N}(\mathbf{q}_{t+1}|\mathbf{q}_t) &= \tfrac{1}{|\mathbf{A}_t|}\mathcal{N}(\mathbf{q}_t|\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t-\mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t),\mathbf{A}_t^{-1}\mathbf{Q}_t^*\mathbf{A}_t^{-T}),\\ &\propto \mathcal{N}(\mathbf{q}_t|\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t-\mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t),\mathbf{A}_t^{-1}\mathbf{Q}_t^*\mathbf{A}_t^{-T}).\end{aligned}$$

To be able to multiply both distributions according to Equation B.5, the distributions are transformed into canonical form using Equation B.4

$$\begin{aligned}\mathcal{N}(\mathbf{q}_t|\mathbf{q}_{t-1}) &= \mathcal{N}(\mathbf{q}_t|\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1}+\mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{h}_{t-1},\mathbf{Q}_{t-1}^*),\\ &= \mathcal{N}[\mathbf{q}_t|\mathbf{Q}_{t-1}^*(\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1}+\mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{h}_{t-1}),\mathbf{Q}_{t-1}^{*-1}],\\ \mathcal{N}(\mathbf{q}_{t+1}|\mathbf{q}_t) &= \mathcal{N}(\mathbf{q}_t|\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t-\mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t,\mathbf{A}_t^{-1}\mathbf{Q}_t^*\mathbf{A}_t^{-T}),\\ &= \mathcal{N}[\mathbf{q}_t|\tilde{\mathbf{Q}}_t^{-1}(\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t-\mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t)),\tilde{\mathbf{Q}}_t^{-1}],\\ &\quad \text{with } \tilde{\mathbf{Q}}_t=\mathbf{A}_t^{-1}\mathbf{Q}_t^*\mathbf{A}_t^{-T}.\end{aligned}$$

Using Equation B.3 the resulting distribution is transformed back to normal form

$$\begin{aligned}\mathcal{N}[x|a,\mathbf{A}]\,\mathcal{N}[x|b,\mathbf{B}] &= \mathcal{N}[\mathbf{q}_t|\mathbf{Q}_{t-1}^{*-1}\,\mathbf{m},\mathbf{Q}_{t-1}^{*-1}]\,\mathcal{N}[\mathbf{q}_t|\tilde{\mathbf{Q}}_t^{-1}\,\mathbf{n},\tilde{\mathbf{Q}}_t^{-1}],\\ &\propto \mathcal{N}[\mathbf{q}_t|\mathbf{Q}_{t-1}^{*-1}\,\mathbf{m}+\tilde{\mathbf{Q}}_t^{-1}\,\mathbf{n},\mathbf{Q}_{t-1}^{*-1}+\tilde{\mathbf{Q}}_t^{-1}],\\ &\quad \text{with } \mathbf{m}=\mathbf{A}_{t-1}\mathbf{q}_{t-1}+\mathbf{a}_{t-1}+\mathbf{B}_{t-1}\mathbf{H}^{-1}\mathbf{h}_{t-1},\\ &\quad \text{and } \mathbf{n}=\mathbf{A}_t^{-1}(\mathbf{q}_{t+1}-\mathbf{a}_t-\mathbf{B}_t\mathbf{H}^{-1}\mathbf{h}_t).\end{aligned}$$

Finally, the state transition distribution is represented by the following Gaussian

$$p(\mathbf{q}_t|\mathbf{q}_{t-1},\mathbf{q}_{t+1}) = \mathcal{N}(\mathbf{q}_t|(\mathbf{Q}_{t-1}^{*-1}+\tilde{\mathbf{Q}}_t^{-1})^{-1}(\mathbf{Q}_{t-1}^{*-1}\,\mathbf{m}+\tilde{\mathbf{Q}}_t^{-1}\,\mathbf{n}),(\mathbf{Q}_{t-1}^{*-1}+\tilde{\mathbf{Q}}_t^{-1})^{-1}).$$

Task constraints in the forward kinematic model are enforced by rejection sampling. The posterior distribution $p(\mathbf{q}_t|z_t,\mathbf{q}_{t-1},\mathbf{q}_{t+1})$ to infer $\mathbf{q}_t$, is computed by the MAP solution as importance sampling method. The posterior distribution is, identical to the model-free approach also computed for each time step except the initial one and improved by performing several rollouts.

4.3 Extended Methods combined with Gibbs Sampling

In this section, extensions of the presented sampling based algorithms for solving challenging motor planning problems in robotics, i.e. a via point approach and a trajectory mixing method are discussed.

4.3.1 Via Point Approach

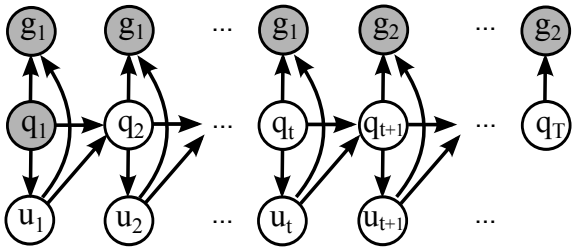


Figure 4.1: The graphical model with state vector $\mathbf{q}_{t=1:T}$ and controls $\mathbf{u}_{t=1:T-1}$ is divided into two sub-tasks, defining two sub-goals $\mathbf{g}_1$ and $\mathbf{g}_2$. The state transitions are represented by arrows connecting the states which are represented by nodes. Grey shaded nodes represent known and also fixed states.

The via point approach applied on graphical models is introduced in Rückert et al. [25]. The via point approach intent to divide a complex task into simpler sub-tasks by introducing sub-goals as illustrated in Figure 4.1. The computation of the sub-trajectories should be simple and therefore the computation of the overall trajectory is simplified. Number and positions of the sub-goals are important for maximal gain in performance. Therefore, the number of sub-goal and their positions should be learned to ensure optimal via points. The tradeoff between performance gain and additional computational effort, necessary to learn the sub-goal positions, is an additional challenge. For simplicity, the number and positions of via points are set manually in this thesis and stated as prior knowledge. To extend the applied via point approach, additional features e.g. maximum costs, or maximum simulation time for reaching a goal can be defined.

4.3.2 Trajectory Mixing Approach

The proposed trajectory mixing approach combines two trajectories computed in parallel to generate an optimal trajectory. The first trajectory is starting at the initial position and evolves towards the target position. The second trajectory is starting at the target position and evolves towards the initial position. These two trajectories are combined. Therefore, one additional joint angle constellation, the target position, must be known to initialise the second trajectory. This approach is illustrated in Figure 4.2. After each round the minimum Euclidean distance between both trajectories at each time step is used to

find an optimal mixing point. If the minimum Euclidean distance d is below a threshold δ, i.e. $d \leq \delta$, both trajectories are merged. Merging two trajectories is applied only once during a rollout to reduce computational effort. The threshold depends on the task complexity and is defined as the length of two links $\delta = 2l$ of the robot arm. The time steps indicating the minimum endeffector distance between both trajectories are denoted by t_1 for the first trajectory and by t_2 for the second trajectory. When mixing both trajectories only the states at $t = 1, \ldots, t_1$ of the first trajectory and the states at $t = 1, \ldots, t_2$ of the second trajectory (with $t_2 = 1$ at the target location), are combined. The resulting trajectory contains t_1 states of the first trajectory and t_2 states of the second trajectory. For simplicity the resulting trajectory replaces the first trajectory after the mixing process. The following example explains the mixing process.

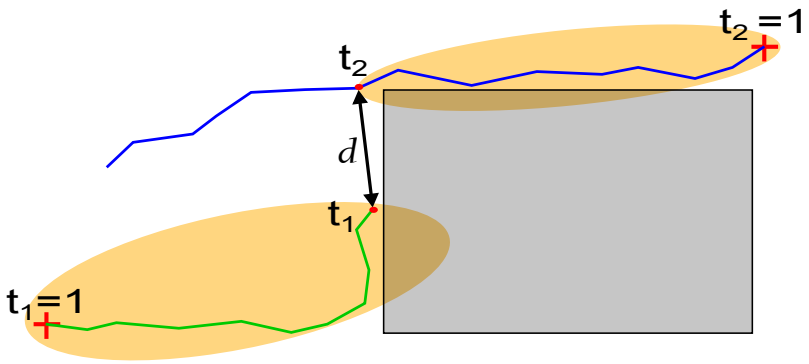


Figure 4.2: The example illustrates the trajectory mixing approach with one obstacle. The start position and the target position are indicated by red crosses and the grey shaded area defines the obstacle. The minimal Euclidean distance d between both trajectories is reached at time step t_1 and t_2 and must be smaller as a threshold δ for mixing the trajectories. The resulting trajectory is the combination of the orange shaded regions enclosing the two trajectory parts.

Let us imagine two trajectories with each 40 time steps and an optimal mixing point at $t_1 = 33$ for the first trajectory and $t_2 = 25$ for the second trajectory. Let us suppose that the minimum Euclidean distance between the endeffector positions at this time steps is smaller than the threshold δ. Then the resulting trajectory contains the first 33 states of the first trajectory and the first 25 states in reverse order of the second trajectory. Thus, illustrated in the example the resulting trajectory must not have the same length as the two original trajectories. To achieve the equal length of the resulting trajectory, a shorter trajectory is expanded with additional states at t_1 of the first trajectory. A longer resulting trajectory is resampled to reduce its length.

4.4 Motor Planning Framework with Parameter Optimisation

The results obtained by the sampling methods, depend heavily on the task environment and the parameterisation of the implemented Gaussians, i.e. the precision matrices. Therefore the parameters must be adapted for each task separately. Adjusting the parameterisation manually is demanding and in most cases does not lead to optimal results. In order to optimise the parameterisation for a given task the parameters are learned. In this thesis the parameter optimisation is implemented by the policy search algorithm Covariance Matrix Adaption - Evolution Strategy (CMA-ES) described in Hansen et al. [11]. CMA-ES is chosen due to performance advantages compared to other policy search methods and its simple implementation [25]. A motor planning framework including parameter optimisation is illustrated in Figure 4.3.

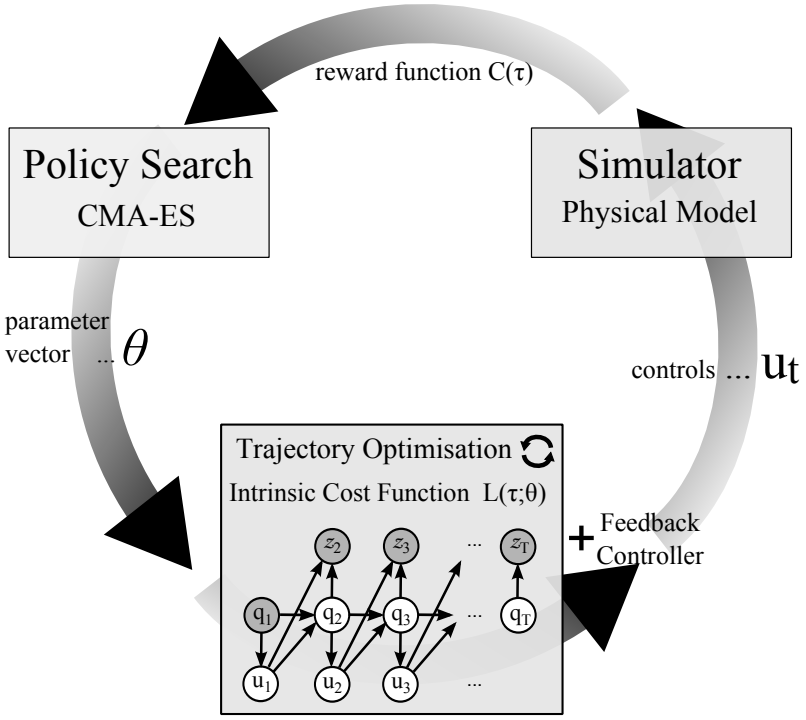


Figure 4.3: The described learning framework with parameter optimisation consists of three parts, the policy search, the trajectory optimisation, and the simulation on the physical model. The policy search algorithm CMA-ES returns the optimised parameter vector $\boldsymbol{\theta}$ to the motor planning model. The motor planning model optimises the parameterised trajectories regarding their intrinsic costs $L(\tau; \boldsymbol{\theta})$. The controls $\mathbf{u}_t$ are obtained by applying the trajectory τ on the feedback controller. Subsequently the simulator applies the obtained result on the physical model and influence the policy search behaviour by a reward function $C(\tau)$.

In the motor planning framework similar to the framework described in Rückert et al. [25], the parameter vector $\boldsymbol{\theta}$ obtained by the policy search method is applied on the motor planning model to generate a trajectory τ, minimising the parameterised intrinsic cost function $L(\tau;\boldsymbol{\theta})$. After computing the trajectory for several rollouts, the obtained data is applied on the linear feedback controller. The obtained controls $\mathbf{u}_t$ from linear feedback controller are simulated on the physical model of a robot arm. The simulator includes a reward function, acting as performance criteria of the executed movements. The policy search method adapts its policy search behaviour for generating a new parameter vector $\boldsymbol{\theta}$ according to the extrinsic cost function $C(\tau)$.

$$C(\tau) = \sum_{t=1}^{T} dt((\mathbf{x}_t - \mathbf{x}_{target})^T \mathbf{C}_{sim}(\mathbf{x}_t - \mathbf{x}_{target}) + \mathbf{u}_t^T \mathbf{H}_{sim}\mathbf{u}_t). \tag{4.1}$$

The extrinsic cost function is computed similar to the cost function $ct(\mathbf{q}_t, \mathbf{u}_t)$ but includes also the controls $\mathbf{u}_t$ and is weighted with discretisation step dt. The difference between the states $\mathbf{x}_t$ and $\mathbf{x}_{target}$ in task space is computed by the Euclidean distance. Both diagonal matrices $\mathbf{C}_{sim}$ and $\mathbf{H}_{sim}$ of size M x M ($M \ldots$ number of joint angles) are constant with identical values for both physical models (2-link and 4-link robot arm). The diagonal elements of $\mathbf{C}_{sim}$ are specified by ($1e^4$ $1e^2$ $1e^4$ $1e^2$) and diag($\mathbf{H}_{sim}$) is specified by ($1e^{-1}$ $1e^{-1}$ $1e^{-1}$ $1e^{-1}$), for the 2-link model.

Now the motor planning framework for parameter optimisation, applied to obtain the task specific optimal parameter vector $\boldsymbol{\theta}^*$, is discussed in detail. The parameter vectors for model-based $\boldsymbol{\theta}_{mb}^T$ and model-free $\boldsymbol{\theta}_{mf}^T$ sampling are defined as

$$\boldsymbol{\theta}_{mb}^T = \begin{bmatrix} \mathbf{gain} \\ \mathbf{w}_q \\ \mathbf{w}_u \\ \mathbf{w}_T \end{bmatrix}, \text{ and } \quad \boldsymbol{\theta}_{mf}^T = \begin{bmatrix} \mathbf{gain} \\ diag(\mathbf{W}) \\ diag(\mathbf{R}_{trans}) \\ diag(\mathbf{R}_{joint}) \end{bmatrix}.$$

The row vector **gain** denotes the gain values included in the computation of controls of the linear feedback controller. The values indicate for each joint angle and velocity separately, how fast the feedback controller change controls on differences between states $\mathbf{q}_t$ and $\mathbf{q}_{t+1}$. The logarithmic scaled vectors $\mathbf{w}_q$, $\mathbf{w}_T$, and $\mathbf{w}_u$ specify the diagonal elements of matrices $\mathbf{R}$, $\mathbf{R}_T/\mathbf{C}_T$, and $\mathbf{H}$, introduced in Section 3.5. The dimension of vector $\mathbf{w}_T$ is different for the intrinsic costs in state space or task space. In state space, the vector $\mathbf{w}_T^{state}$ has a size of 1 x 2M, representing the target weights for each joint angle and corresponding velocity. Due to the endeffector representation of the target in task space, the vector $\mathbf{w}_T^{task}$ has only a size of 1 x 2, representing the target weights of the x/y-coordinate pairs. In case of model-free sampling the learning parameters are also specified by **gain** and

the diagonal elements of matrices $\mathbf{W}$, $\mathbf{R}_{trans}$, and $\mathbf{R}_{joint}$ described in Section 4.1. For learning the via points instead of setting them manually, the parameters of via points can be included in the parameter vector $\boldsymbol{\theta}$. The parameters $\boldsymbol{\theta}$ are evolved by CMA-ES over 300 generations with learnrate 0.1. For each generation a number of offspring's depending logarithmically on the number of parameters is executed. The search space of the CMA-ES is limited between $0.9\ \boldsymbol{\theta} \leq \boldsymbol{\theta} \leq 1.1\ \boldsymbol{\theta}$. The boundaries are set manually, depending on the quality of the initial parameters. For the first parameter learning attempt in an unknown environment a larger search space is advantageous to avoid local optimal results. With more knowledge about the environment the search space boundaries can be tightened up. After obtaining parameter vector $\boldsymbol{\theta}$ and controls $\mathbf{u}$, the outputs are executed in the simulator. As simulator, a physical model of a 2-link or 4-link robot arm is used. For simplicity both physical models consider neither gravity nor friction and the link lengths and masses are set to 1. However, inertias are modelled, which renders the dynamic task challenging. For realistic conditions, additive Gaussian noise with standard deviation 0.1 is added to the controls by the feedback controller.

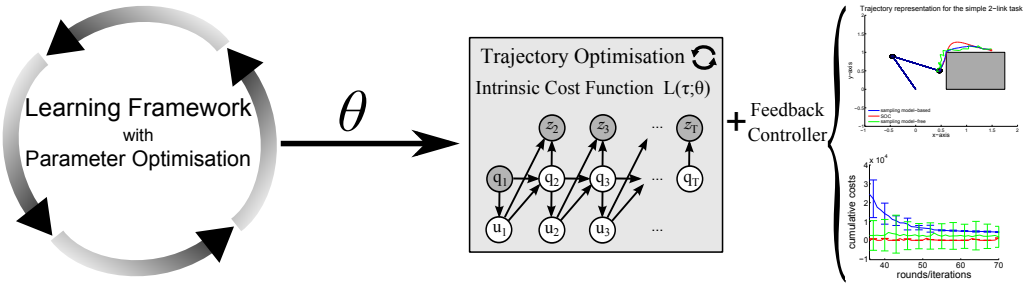


Figure 4.4: The results presented in the experiments are obtained by applying the optimal parameters $\boldsymbol{\theta}$ from the learning framework on the motor planning model, using a linear feedback controller.

After the policy search step, the obtained parameters $\boldsymbol{\theta}$ are used for motor planning simulation according to Figure 4.4. Specific parameters (e.g. simulation time steps, number of samples) must be adjusted manually, depending on the task. Finding appropriate parameter values manually, is not simple and therefore optimal values cannot be guaranteed.

5. Experiments and Results

In the following experiments the advantages and drawbacks of sampling based methods are addressed and compared to Approximate Inference Control (AICO) [28, 25]. For this purpose, the computed trajectory τ, the intrinsic costs $L(\tau)$ of the trajectory (defined in Equation 3.2 and 3.3), the convergence behaviour of $L(\tau)$, and the extrinsic costs $C(\tau)$ (defined in Equation 4.1) are chosen as performance metrics. The experiments are performed on dynamic models, a 2-link arm representing a low dimensional task and a 4-link arm representing a higher dimensional task. In both scenarios the experiments include obstacles to increase the task complexity. To emphasise hard constraints, the placed obstacles are rectangular. Furthermore, model-based sampling extended by a via point and a multiple trajectory approach is evaluated. To be able to make general predictions, for every experiment the results of the applied methods are averaged over 10 runs.

5.1 Parameter Settings

As mentioned in Section 4.4, the parameter vector $\boldsymbol{\theta}$ depends on the task specifications. The values for task specific parameters are listed for each task and method separately. Global parameters, i.e. simulation time or discretisation steps are defined for a class of tasks (e.g. experiments applying the 2-link arm model). Model parameters, for instance the parameterisation of the Covariance Matrix Adaption - Evolution Strategy (CMA-ES) algorithm do not change in all experiments. The parameterisation of CMA-ES, the level of Gaussian noise added to the controls of the feedback controller, and the state noise model $\mathbf{Q}$, are defined as model parameters. The state noise model $\mathbf{Q}$ expresses the reliability about the observed state. In the model-free approach, the state noise model $\mathbf{Q}$ is included in the covariance matrix $\mathbf{W}$. For model-based sampling and Approximate Inference Control (AICO) the 2-link state noise models are specified by

$$\mathbf{Q}_{mb} = \begin{bmatrix} 10^{-2} & 0 & 0 & 0 \\ 0 & 10^{-3} & 0 & 0 \\ 0 & 0 & 10^{-2} & 0 \\ 0 & 0 & 0 & 10^{-3} \end{bmatrix}, \text{ and } \quad \mathbf{Q}_{AICO} = \begin{bmatrix} 10^{-12} & 0 & 0 & 0 \\ 0 & 10^{-11} & 0 & 0 \\ 0 & 0 & 10^{-12} & 0 \\ 0 & 0 & 0 & 10^{-11} \end{bmatrix}.$$

For the 4-link experiments the state noise model $\mathbf{Q}$ is a 8 x 8 diagonal matrix with identical

values as used in the 2-link arm reaching experiments.

5.2 Two Link Arm Experiments

A simple 2-link arm model is used to demonstrate the behaviour of model-free sampling, model-based sampling and Approximate Inference Control (AICO). All 2-link arm experiments are performed in state space, hence the target vector $\mathbf{q}_{target}$ is represented by angles and velocities. The state vector $\mathbf{q}_{1:T}$ is initialised with values of the starting position $\mathbf{q}_1$ by default. The following model parameters are set manually and remain unchanged for all experiments performed on the 2-link model.

parameter	value
T_s	2
dt	5e^{-2}
sim_{dt}	5e^{-3}

Table 5.1: Simulation Parameters for the 2-link model experiments.

The simulation time T_s in seconds denotes the time horizon of the arm movement and the discretisation of time steps is given by dt in seconds. The number of time steps n results from $n = rounding(T_s/dt)$. The function *rounding* denotes the round function to the nearest integer. The variable sim_{dt} specifies the discretisation steps in seconds, applied on the physical model of the 2-link arm.

parameter	value
iterations	80
$\mathbf{w}_T$	$[5e^2\ 5e^2]^T$
w_o	$1e^5$
$\mathbf{w}_u$	$[1\ 1]^T$

Table 5.2: Parameters of AICO for the 2-link experiments.

The parameters for the AICO algorithm are invariant for all 2-link experiments. The number of iterations performed by AICO are specified by the variable *iterations*. Vector $\mathbf{w}_T$ specifies the importance of reaching the target, whereas the scalar w_o weights obstacle boundary violations. Vector $\mathbf{w}_u$ denotes the diagonal elements of control precision matrix $\mathbf{H}$ introduced in Section 3.5.

5.2.1 Task with a single Obstacle

In the first task, the arm is initialised with $\mathbf{q}_1$ (with angles represented in radiant and zero velocities) and controls $\mathbf{u}_1$. The target position $\mathbf{q}_{target}$ for the endeffector is marked with a red cross and lies beside a rectangular obstacle in Figure 5.1. For this task the arm

must pass the edge of the obstacle in order to reach the target position. The task focus on the ability of bypassing the edge without moving through the obstacle.

$$\mathbf{q}_1 = \begin{bmatrix} 117\frac{\pi}{180} \\ 0 \\ -140\frac{\pi}{180} \\ 0 \end{bmatrix}, \quad \mathbf{u}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ and } \quad \mathbf{q}_{target} = \begin{bmatrix} 60\frac{\pi}{180} \\ 0 \\ -49\frac{\pi}{180} \\ 0 \end{bmatrix}$$

parameter	model-free	model-based
rounds	80	80
samples	500	500
gain	$[0.100\ 10.000\ 24.570\ 0.100]^T$	$[49.986\ 1.046\ 49.452\ 6.316]^T$
$\mathbf{w}_q$		$[3.441\ 1.750\ 3.606\ 1.661]^T$
$\mathbf{w}_u$		$[-0.008\ -0.008]^T$
$\mathbf{w}_T^{state}$		$[5.105\ 5.375\ 4.950\ 5.300]^T$
$diag(\mathbf{R}_{joint}^{-1})$	$[1e^3\ 1e^3\ 1e^3\ 25]$	
$diag(\mathbf{R}_{trans}^{-1})$	[25 0.230 23.256 0.241]	
$diag(\mathbf{W})$	[0.239 0.028 0.312 0.001]	

Table 5.3: Task specific parameters of the sampling methods for the task setting with one obstacle.

Table 5.3 contains the parameter values of the model-free and model-based approach, applied on the simple 2-link task. The number of rollouts and drawn samples are denoted by *rounds* and *samples* and set manually. The parameters for model-based sampling and model-free sampling, described in Section 4.4 are learned applying the framework from Section 4.4.

Figure 5.1 compares the trajectories of the different methods. AICO and the model-based approach compute both smooth trajectories reaching the target position. The trajectory computed by the model-free approach jitters and is not able to reach the target precisely. Zooming into Figure 5.1 reveals the inability of AICO avoiding movements through the obstacle, shown in Figure 5.1 in the right panel. The intrinsic costs obtained by the different methods are illustrated in Figure 5.2. The intrinsic costs are averaged over 10 independent trials, applying the cost function specified in Equation 3.1 with the standard deviation represented by error bars. For a better comparison of the convergence behaviour, the intrinsic costs obtained by the model-based sampling approach are scaled by multiplying with a factor of $1e^{-3}$. Due to different precision matrices and weights applied on the different methods, the intrinsic costs between the different methods cannot be compared directly. Therefore, the intrinsic costs can only be used to compare the performance of one method for different tasks. Nevertheless, the convergence behaviour of the intrinsic costs for different methods can be used to compare their performances.

AICO converges in less than 10 iterations with low standard deviation. Model-free

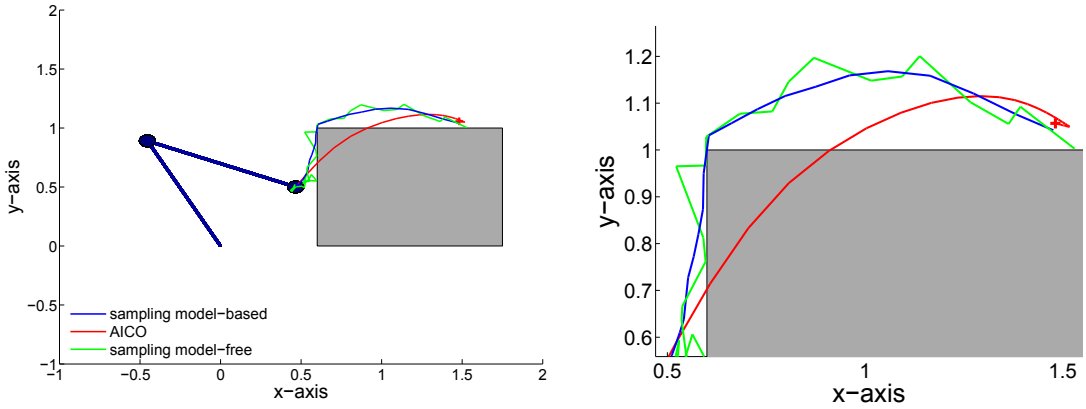


Figure 5.1: The different trajectories computed for the task with one obstacle are compared, focussing on the enlarged edge of the obstacle. The initial arm constellation is shown by the blue two link arm and the target position is marked by a red cross. The trajectories of model-free sampling (green line), model-based sampling (blue line), and AICO (red line) are compared. The optimal trajectory is a smooth trajectory without violating the obstacle (grey shaded) boundaries.

sampling converges also in less than 20 rounds but shows a high standard deviation at the beginning and also relatively large standard values over the whole iterative process. Large values of the standard deviation are due to representing different trajectories simultaneously, where the method is not able to converge to a single solution. In contrast, the model-based approach needs more rounds to converge. The values of standard deviation are small compared to model-free sampling.

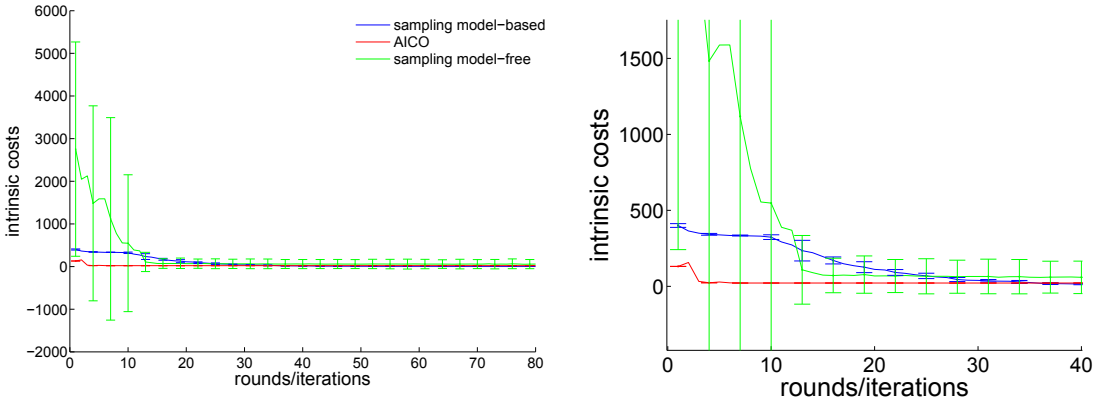
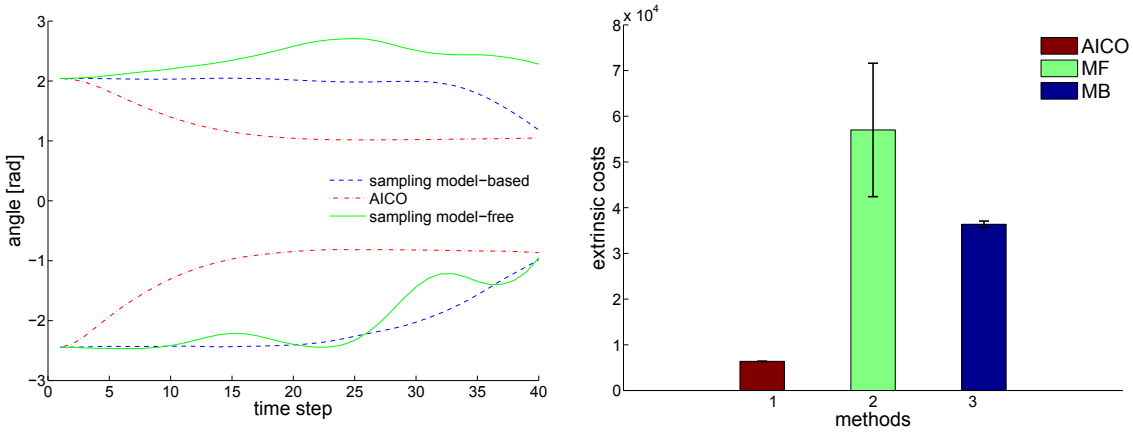


Figure 5.2: The convergence behaviours of the intrinsic costs $L(\tau)$ for the task with one obstacle are compared applying the different methods. Note that the intrinsic cost definitions are different for all methods. Therefore, the convergence behaviour is chosen to compare the performance of the different methods. For a better comparison the intrinsic costs for model-based sampling are scaled with factor $1e^{-3}$.

In Figure 5.3 (a) both joint angles are shown over time. To obtain a trajectory with minimum costs both angles should evolve smoothly over time. AICO and model-based

sampling are able to compute smooth angle trajectories, whereas the model-free sampling approach is not. The progress of joint angles, velocities, and controls over the time horizon T determines the movement of the physical 2-link model returned by the simulator. The extrinsic costs computed by the reward function $C(\tau)$ specified in Equation 4.1, are presented in Figure 5.3 (b). As expected, the extrinsic costs $C(\tau)$ and the standard deviation of model-free sampling are high compared to model-based sampling and AICO.



(a) (b)

Figure 5.3: In (a) the joint angles are illustrated over the simulation process in radiant. Smooth trajectories result in lower extrinsic costs $C(\tau)$ obtained by the simulator applying the results from the feedback controller on the physical model, shown in (b).

5.2.2 Task with two Obstacles

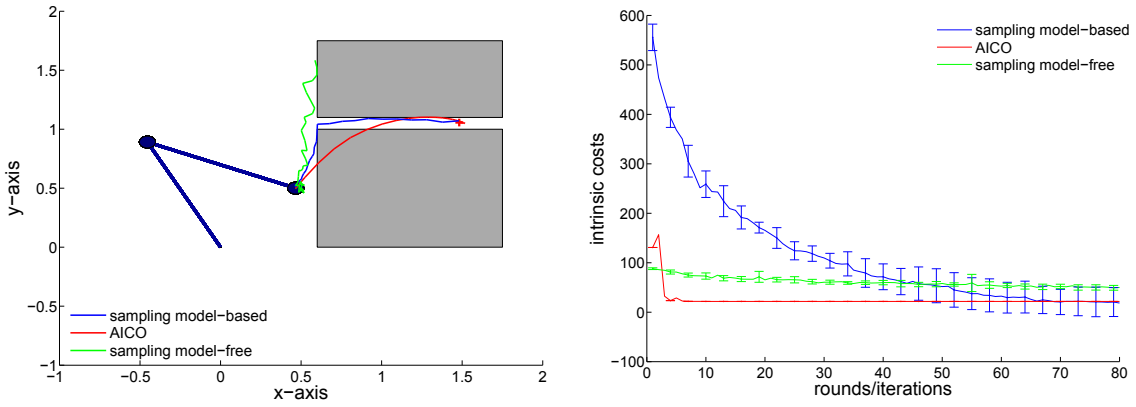
To increase the task complexity a second obstacle is introduced. The target is placed between both obstacles to generate a narrow passage. This task focuses on the ability to pass through the narrow passage without hitting the surrounding obstacles.

$$\mathbf{q}_1 = \begin{bmatrix} 117\frac{\pi}{180} \\ 0 \\ -140\frac{\pi}{180} \\ 0 \end{bmatrix}, \quad \mathbf{u}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \text{ and } \quad \mathbf{q}_{target} = \begin{bmatrix} 60\frac{\pi}{180} \\ 0 \\ -49\frac{\pi}{180} \\ 0 \end{bmatrix}$$

To compute the optimal trajectory for this task, the parameter vector $\boldsymbol{\theta}$ has to be relearned for sampling methods, which are shown in Table 5.4. Applying the framework for parameter optimisation, the whole parameter vector $\boldsymbol{\theta}$ is relearned. This setting may result in completely different parameter vectors for similar tasks. It is also possible to learn arbitrary parameter constellations by reusing specific parameters, e.g. reuse vector **gain** and learn only $\mathbf{w}_q, \mathbf{w}_u$, and $\mathbf{w}_T$. Only model-based sampling is able to reach the target position without moving through obstacles as shown in Figure 5.4 (a). Compared to

parameter	model-free	model-based
rounds	80	80
samples	500	500
gain	$[50.000\ 17.210\ 10.000\ 0.100]^T$	$[3.100\ 1.750\ 3.606\ 1.661]^T$
$\mathbf{w}_q$		$[3.531\ 1.901\ 3.398\ 1.508]^T$
$\mathbf{w}_u$		$[-0.007\ -0.007]^T$
$\mathbf{w}_T^{state}$		$[4.916\ 5.082\ 4.850\ 5.703]^T$
$diag(\mathbf{R}_{joint}^{-1})$	[26.316 0.0229 23.256 0.241]	
$diag(\mathbf{R}_{trans}^{-1})$	[83.333 90.909 100 100]	
$diag(\mathbf{W})$	[0.033 0.003 0.027 0.003]	

Table 5.4: Task specific parameters of the sampling methods for the 2-link arm reaching task with two obstacles and without a via point.

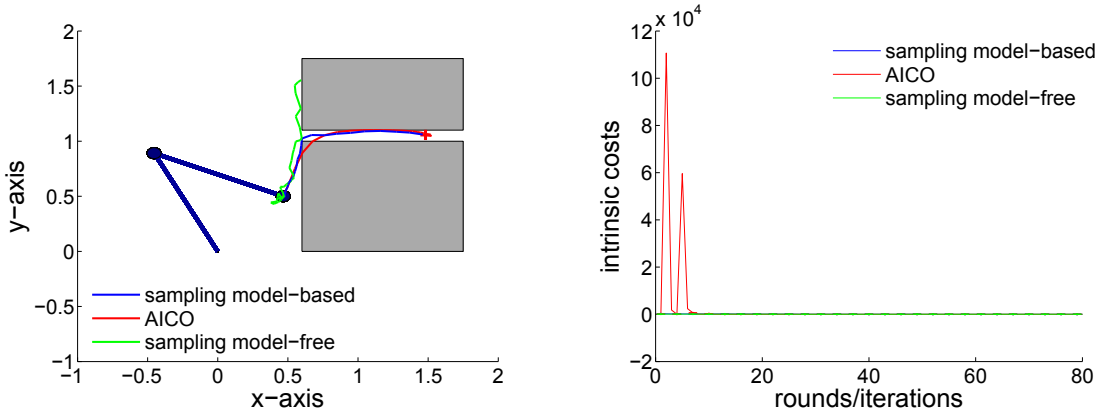


(a) (b)

Figure 5.4: In (a) the trajectories of the three methods are illustrated. The intrinsic costs of the different approaches and their convergence behaviour is shown in (b).

the previous task, the standard deviation values are larger and the intrinsic costs converge slower as shown in Figure 5.4 (b). AICO is not able to find a trajectory passing through the narrow passage without hitting an obstacle and therefore, the intrinsic costs are higher compared to the task with only one obstacle, shown in Figure 5.4. However, AICO is still able to converge within 10 iterations. By increasing the weight for obstacle violations, $w_o = 1e^8$, the trajectories generated with AICO are repelled stronger by the obstacle illustrated in Figure 5.5 (a). Nevertheless, AICO is not able to avoid obstacle violations completely and the trajectory becomes unstable indicated by peaks of the intrinsic cost function in Figure 5.5 (b). However, obstacle weights $w_o \geq 1e^8$ causes singular matrices during the computations. Thus the choice of the obstacle weight is a tradeoff between a stable algorithm and the ability of avoiding obstacles.

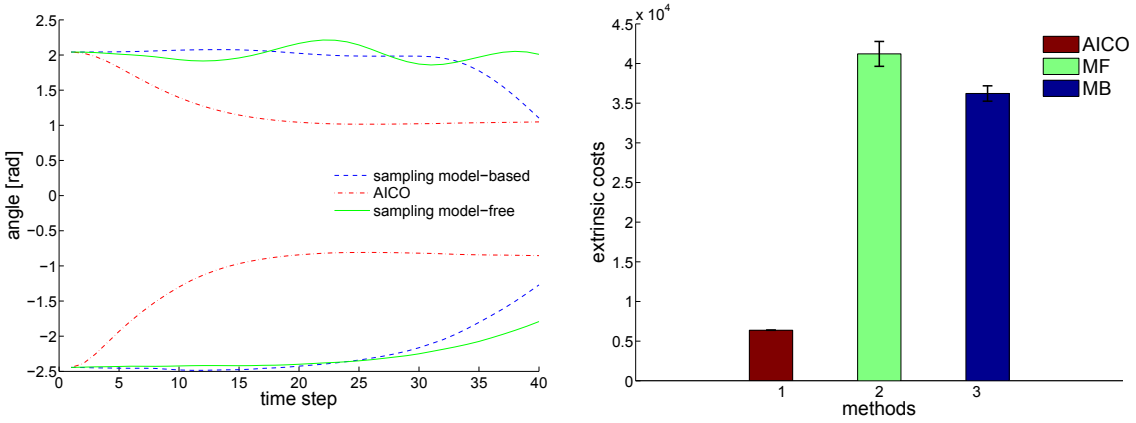
The model-free sampling method is also not able to find an appropriate trajectory. The standard deviation of the intrinsic costs in Figure 5.4 (b) are low as always similar trajectories are chosen. Figure 5.6 shows again the joint angle trajectories and the extrinsic



(a) (b)

Figure 5.5: In (a) the different trajectories applying an obstacle weight of $1e^8$ for AICO are compared. In (b) the changed convergence behaviour of intrinsic costs are illustrated. As illustrated in (a) the trajectory generated by AICO repels stronger from the obstacle but the method becomes unstable causing high peaks of intrinsic costs in (b).

costs for all applied methods. Only the extrinsic costs returned by the model-free sampling method in Figure 5.6 (b) are lower compared to the costs in Figure 5.3 (b) as model-free sampling generates simple trajectories which do not reach the target.



(a) (b)

Figure 5.6: In (a) the joint angle trajectories of the 2-link arm generated by the different methods are presented. In (b) the corresponding extrinsic costs obtained by the reward function are compared.

5.2.3 Task with two Obstacles and one Via Point

To improve the performance of sampling methods, one via point

$$\mathbf{q}_{via} = \begin{bmatrix} 117\frac{\pi}{180} & 0 & -107\frac{\pi}{180} & 0 \end{bmatrix}^T$$

at time step $n_{via} = 10$ is introduced as intermediate target position, see Figure 5.7. The initialisation vectors $\mathbf{q}_1, \mathbf{u}_1$ and the target position $\mathbf{q}_{target}$ are identical as for the previous task without a via point.

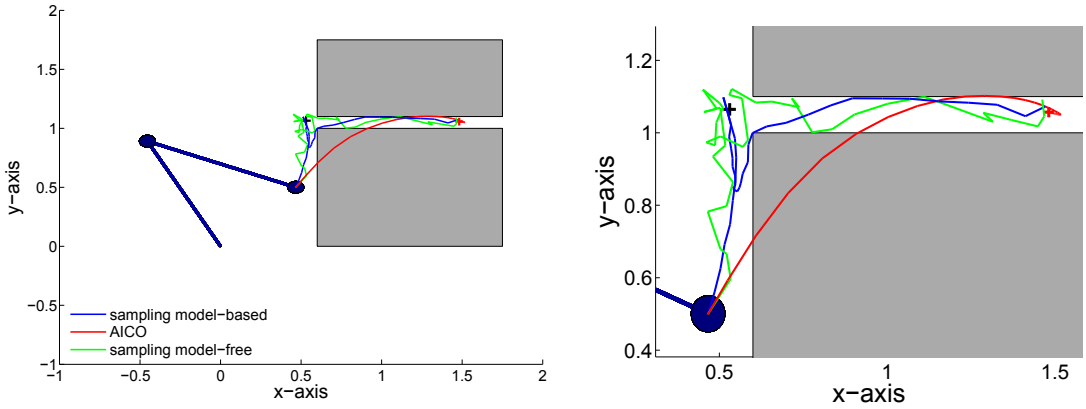


Figure 5.7: Illustrated are the movement trajectories for all three methods using two obstacles. The via point is marked by a black cross, whereas the target is marked by a red cross.

In this scenario the trajectories have to reach the via point first and afterwards the original target position. The via point is chosen to divide the original trajectory into two simpler trajectories. Hence, the position of the via point is crucial for an optimal performance. For simplicity, the position is manually chosen in front of the narrow passage to support the moving into the passage. However, it could also be learned. Due to the simplification obtained by the via point, the parameter vector $\boldsymbol{\theta}$ of the previous experiment with one obstacle, shown in Table 5.3 can be reused. The via point is not used within AICO as standard SOC methods are able to solve such simple motor planning tasks already with via points [28].
With the via point, also the model-free sampling approach is able to reach the target position. The trajectories of the model-free sampling reach the via point and the target not exactly and jitters heavily in the first half of the movement. Note that the precision of reaching the target and the via point depends on the values of $diag(\mathbf{R}_{joint}^{-1})$. In comparison model-based sampling reaches the via point, but does not move straight to the target position. This behaviour is discussed later in experiments with a 4-link model.

The intrinsic costs $L(\tau)$ are shown in Figure 5.8. Model-free sampling has initially high intrinsic costs with high standard deviation, which converge within 30 rounds. However, the intrinsic costs and standard deviation remains higher compared to the other methods. Compared to the experiment without a via point, the final intrinsic cost values for model-based sampling are higher. The reason for this behaviour results from the not optimal position of the via point. Therefore, introducing a learned via point at an optimal position

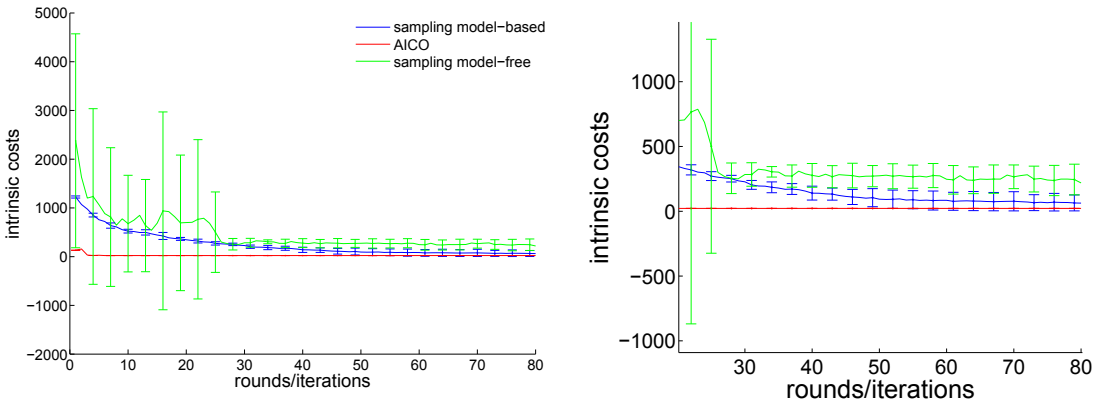
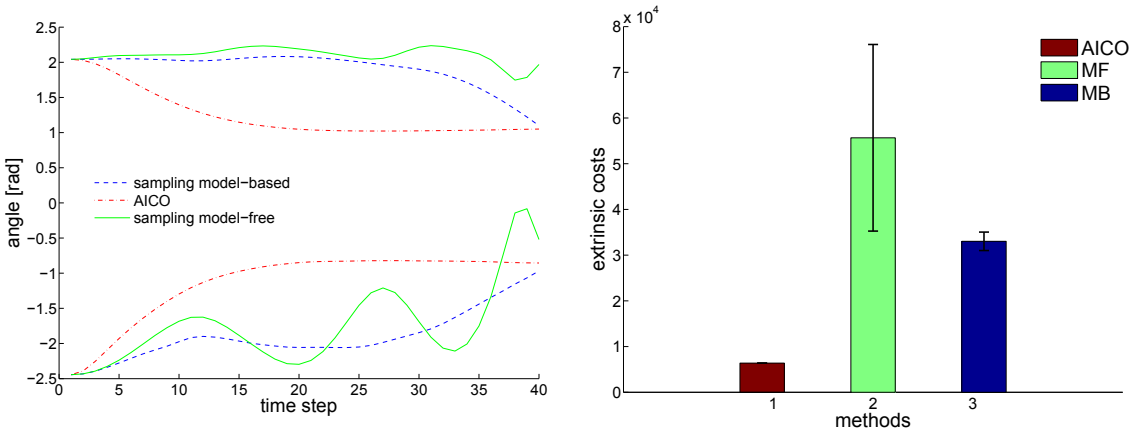


Figure 5.8: The convergence behaviour of intrinsic costs for the 2-link task applying one via point is compared with the focus on the different convergence behaviour.

would reduce the peak. The trajectory and the costs obtained by AICO are identical to the previous scenario because the via point was only used in the sampling methods. The joint angles and the extrinsic costs $C(\tau)$, are illustrated in Figure 5.9.



(a) (b)

Figure 5.9: In (a) the joint angle trajectories of the 2-link arm for the different methods are shown. In (b) the corresponding extrinsic costs obtained by the reward function are illustrated.

5.2.4 Summary

The experiments show that all applied methods are able to solve simple tasks using appropriate parameters. More complex tasks can be solved by extending the basic methods, applying the via point approach. As listed in Table 5.5 both sampling methods obtain remarkable increasing intrinsic costs and standard deviation values, solving more complex tasks. Compared to model-free sampling, model-based sampling is more robust solving complex tasks. Therefore, the model-free sampling method is not applied on higher di-

task	model-free	model-based	AICO
task with a single obstacle	24 ± 4	4.8 ± 0.6	$21.8 \pm 3.7\text{e}^{-15}$
task with two obstacles	target not reached	20 ± 26	$21.8 \pm 3.7\text{e}^{-13}$
task with one via point	218 ± 116	62 ± 60	$21.8 \pm 3.7\text{e}^{-14}$

Table 5.5: The intrinsic costs and standard deviation values for different 2-link tasks are compared, considering the applied methods.

mensional tasks. Extending the sampling approaches with an appropriate via point is able to decrease the intrinsic costs. Due to the manually chosen via point the resulting costs for model-based sampling are higher. AICO yields very good results in general but is not able to avoid the obstacles even for simple tasks.

5.3 Four Link Arm Experiments

A 4-link arm model is used to demonstrate the behaviour of the model-based sampling method on a higher dimensional dynamic task. Due to suboptimal trajectories generated by model-free sampling, especially for complex 2-link tasks, the model-free sampling approach is not applied on 4-link tasks. Due to numerical instabilities, AICO is also not applied on the 4-link tasks. These numerical problems result from the increased number of possible arm constellations to reach the target. Such numerical problems can be circumvented by applying inverse kinematic control as initial solution, which is beyond the scope of this work.

In the 4-link experiments, different initialisations are investigated. Also the challenge of multiple arm reaching solutions in motor planning is addressed and how model-based sampling deals with these challenging constellations. To evaluate multiple arm reaching solutions, task space planning is applied representing the intrinsic costs as a two-dimensional coordinate vector, introduced in Equation 3.3. To avoid numerical problems when sampling from a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the symmetry of the matrix representing $\boldsymbol{\Sigma}$ must be enforced, which is typically not guaranteed for the dynamics of the 4-link model.

The simulation parameters for the 4-link experiments are identical to the values applied for the 2-link arm. Only the simulation steps sim_{dt} for simulating the movements on the physical model are refined to enhance the ability of the simulator to deal with the increased complexity. These parameters are specified in Table 5.6.

parameter	value
T_s	5
dt	5e^{-2}
sim_{dt}	5e^{-4}

Table 5.6: Simulation Parameters for the 4-link arm experiments.

5.3.1 Comparison of different Initialisation Methods

This scenario investigates the advantages of different initialisation methods. The goal of this task is again to bypass an obstacle in order to reach the target, which is illustrated in Figure 5.10 (a). The applied parameters for model-based sampling are listed in Table 5.7. The task specifications applying the cost representation in task space are listed below.

$$\mathbf{q}_1 = \begin{bmatrix} 90\frac{\pi}{180} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -45\frac{\pi}{180} \\ 0 \end{bmatrix}, \quad \mathbf{x}_{via} = \begin{bmatrix} 0.637 \\ 0.796 \end{bmatrix}, \text{ and } \quad \mathbf{x}_{target} = \begin{bmatrix} 1.851 \\ 0.674 \end{bmatrix}.$$

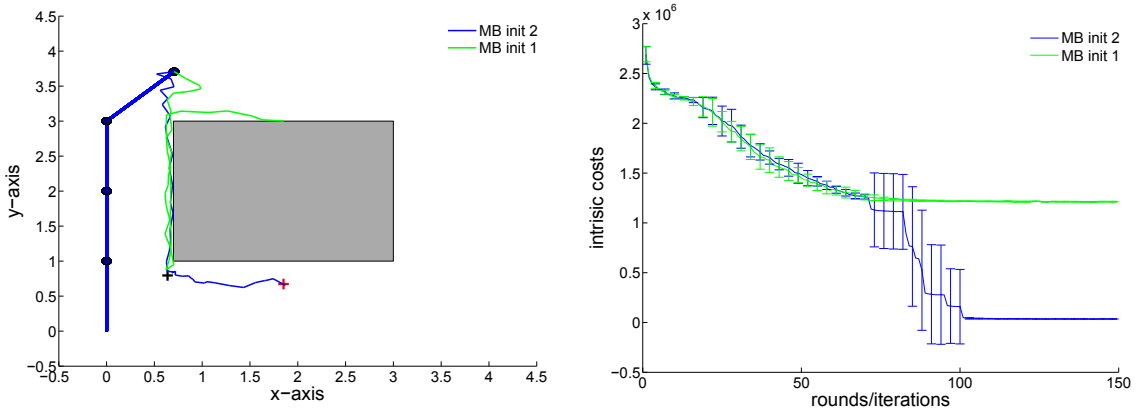
parameter	model-based
rounds	150
samples	500
gain	$[0.009\ 9.991\ 48.725\ 0.013\ 49.869\ 0.082\ 46.729\ 0.673]^T$
$\mathbf{w}_q$	$[4.053\ 2.117\ 4.160\ 2.155\ 4.039\ 2.138\ 4.033\ 1.989]^T$
$\mathbf{w}_u$	$[-0.09\ -0.09\ -0.09\ -0.09]^T$
$\mathbf{w}_T^{task}$	$[4.250\ 4.425]^T$

Table 5.7: Parameters for the model-based sampling approach comparing different initialisation methods.

The trajectory denoted by the green line, shown in Figure 5.10 (a) is generated by initialising $\mathbf{q}_{1:T}$ with $\mathbf{q}_1$. The trajectory reaches the via point, however, does not move towards the target subsequently. Instead of moving towards the target the arm returns to the initial position as the method is not able to bypass the obstacle. Due to the initialisation of $\mathbf{q}_{1:T}$ with $\mathbf{q}_1$, the costs for moving from the via point to the target are higher compared to the movement costs returning to the starting point. A similar effect was observed in the 2-link arm reaching task in Section 5.2.2.

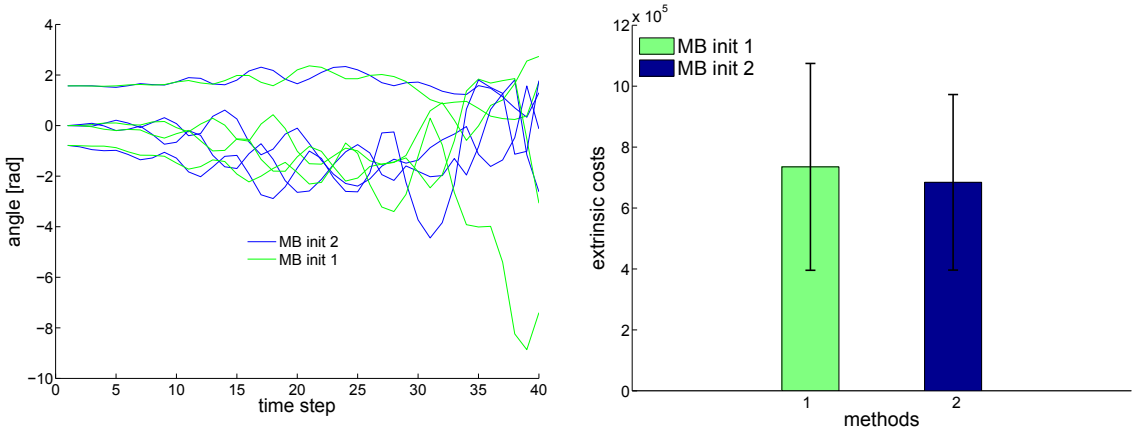
To avoid this behaviour an alternative online re-initialisation procedure is applied. The state vector $\mathbf{q}_{1:T}$ is also initialised with $\mathbf{q}_1$ identical to the first method. Reaching the via point at time step t, the state vectors of the remaining time steps $\mathbf{q}_{t:T}$ are re-initialised with the state vector $\mathbf{q}_t$. Now the model-based sampling method is able to reach the target as indicated by the blue line in Figure 5.10 (a). The via point is reached if the Euclidean distance d_{max}, between the arm endeffector and the via point reaches $d_{max} \leq l/10$ with link length l.

The intrinsic costs $L(\tau)$ of both initialisation methods converge similar for the first



(a) (b)

Figure 5.10: In (a) the trajectories of model-based sampling with different initialisation methods are shown. Applying the initialisation method 1 (green line), the state vector $\mathbf{q}_{1:T}$ is initialised with $\mathbf{q}_1$. The second initialisation method (blue line) indicates the trajectory, applying the online re-initialisation. In (b) the intrinsic costs of the trajectories and their convergence behaviour are compared, applying the different initialisation methods.



(a) (b)

Figure 5.11: In (a) the joint angle trajectories of the 4-link arm for model-based sampling are illustrated, using different state vector initialisations. In (b) the corresponding extrinsic costs are compared.

half of rounds, as shown in Figure 5.10 (b). However, the intrinsic costs of the second initialisation method decrease rapidly, whereas the intrinsic costs of the first initialisation method remain stable in the second half. The high standard deviation during the drop is caused by the exploration of the optimal trajectory from the via point to the target. Hence, model-based sampling applying the second initialisation method is able to find a movement trajectory with low intrinsic costs, reaching the target. In the following experiments always the second initialisation method is applied.

The joint angles and the extrinsic costs are shown in Figure 5.11. The noisy joint angle trajectories are supposed to result from the increased complexity of the 4-link model and the cost representation in task space. The joint angle trajectories of the following experiment confirm this observation.

5.3.2 Comparison of different Cost Representations

In this scenario, the differences between the state space cost representation and the task space cost representation are investigated. The state space cost representation applies the state vector $\mathbf{q}_{1:T}$, the via point $\mathbf{q}_{via}$, and the target $\mathbf{q}_{target}$ as joint angle constellations and compute the intrinsic costs in the state space, see Equation 3.2. In contrast, the task space cost representation defines the via point $\mathbf{x}_{via}$ and the target $\mathbf{x}_{Target}$ as two dimensional coordinate vectors, defined in Equation 3.3. The difference between the state space and the task space is the representation and cost computation of the via point and the target position. However, the state vector $\mathbf{q}_{1:T}$ represents joint angle constellations in both, the state space and the task space cost representation. The state space cost representation restricts possible trajectories for reaching a target, however, it also simplifies the task by limiting the search space. In contrast, a task space cost representation does not restrict the possible arm constellations to reach the target. This task is solved by applying model-based sampling in state space and task space cost representation. The task specifications in state space and the corresponding task space vectors are presented below.

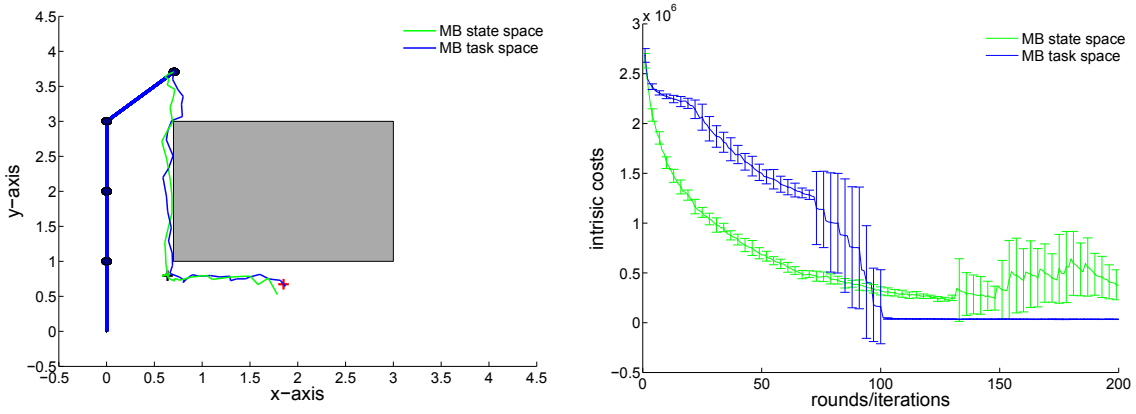
$$\mathbf{q}_1 = \begin{bmatrix} 90\frac{\pi}{180} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -45\frac{\pi}{180} \\ 0 \end{bmatrix}, \quad \mathbf{q}_{via} = \begin{bmatrix} 100\frac{\pi}{180} \\ 0 \\ 0 \\ 0 \\ -110\frac{\pi}{180} \\ 0 \\ -80\frac{\pi}{180} \\ 0 \end{bmatrix}, \quad \mathbf{q}_{target} = \begin{bmatrix} 120\frac{\pi}{180} \\ 0 \\ -90\frac{\pi}{180} \\ 0 \\ -90\frac{\pi}{180} \\ 0 \\ 70\frac{\pi}{180} \\ 0 \end{bmatrix}.$$

$$\mathbf{x}_{via} = \begin{bmatrix} 0.637 \\ 0.796 \end{bmatrix}, \text{ and } \quad \mathbf{x}_{target} = \begin{bmatrix} 1.851 \\ 0.674 \end{bmatrix}.$$

Due to the identical task constellation to the previous task, the parameters can be reused. Applying the cost computation in state space requires an additional learned weight vector $\mathbf{w}_T^{state}$, defined in Table 5.8. To illustrate the behaviour of the model-based sampling applying the state space cost representation appropriately, the number of rounds are increased to 200.

parameter	model-based
rounds	200
samples	500
gain	$[0.009\ 9.991\ 48.725\ 0.013\ 49.869\ 0.082\ 46.729\ 0.673]^T$
$\mathbf{w}_q$	$[4.053\ 2.117\ 4.160\ 2.155\ 4.039\ 2.138\ 4.033\ 1.989]^T$
$\mathbf{w}_u$	$[-0.09\ -0.09\ -0.09\ -0.09]^T$
$\mathbf{w}_T^{task}$	$[4.250\ 4.425]^T$
$\mathbf{w}_T^{state}$	$[4.4354.4094.4054.4194.5834.3894.2514.468]^T$

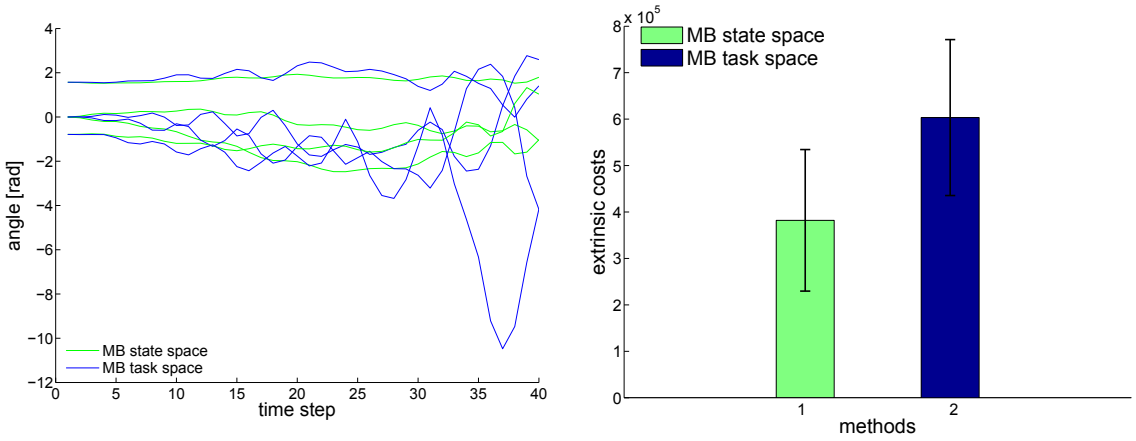
Table 5.8: Parameters for the model-based sampling approach comparing different cost representations.



(a) (b)

Figure 5.12: In (a) the trajectories of model-based sampling with different representations are shown. The trajectory obtained by the state space cost representation is represented by the green line, whereas the blue line represents the trajectory obtained by the task space cost representation. In (b) the intrinsic costs of the different representations and their convergence behaviour are compared.

The movement trajectories, illustrated in Figure 5.12 (a) are similar for both representations, but the trajectory applying the state space cost representation is not able to reach the target precisely. The intrinsic cost function obtained by the cost representation in task space is identical to the second initialisation method of the previous task and converges



(a) (b)

Figure 5.13: The joint angle trajectories of the 4-link arm for state and task space cost representation are illustrated in (a). In (b) the corresponding extrinsic costs obtained by the reward function are shown.

to low costs with low standard deviation. Also the intrinsic cost function obtained by the cost representation in state space converges in the first half. As a result of the applied initialisation method, the intrinsic costs and the standard deviation values increase in the second half by re-initialising the remaining state vectors $\mathbf{q}_{t:T}$. The joint angle trajectories of the cost function in state space, illustrated in Figure 5.13 (a) is smoother compared to the trajectories of the cost function in task space. Therefore, the corresponding extrinsic costs shown in Figure 5.13 (b) are remarkably lower for the cost function in state space compared to the cost function in task space.

5.3.3 Learning Multiple Arm Reaching Solutions

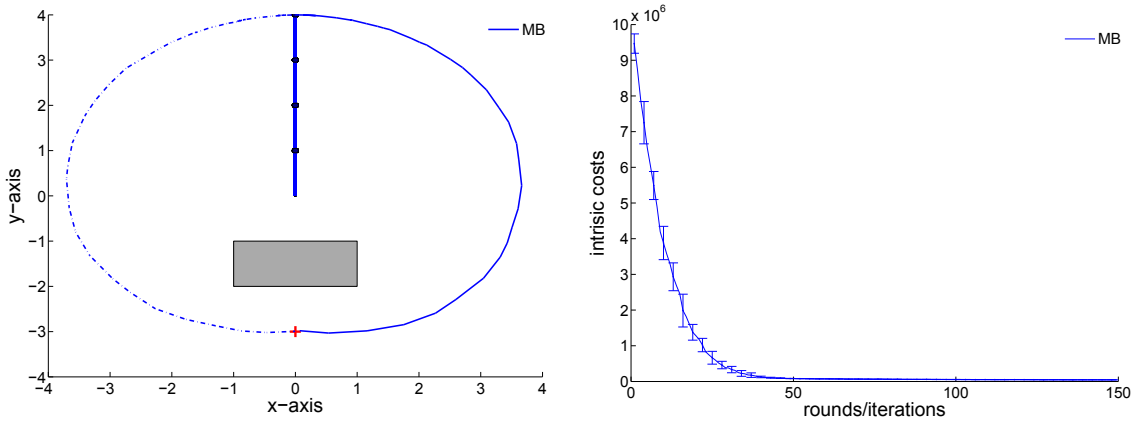
This task applies the task space cost representation to allow multiple arm reaching solutions. This scenario was chosen as classical motor planning methods have difficulties with such constellations (if the trajectory is not well chosen) [31]. The initial arm constellation and the target are presented below, followed by the learned task specific parameters listed in Table 5.9.

$$\mathbf{q}_1 = \begin{bmatrix} 90\frac{\pi}{180} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \text{ and } \quad \mathbf{x}_{target} = \begin{bmatrix} 0 & -3 \end{bmatrix}^T.$$

The symmetric task in Figure 5.14 (a) illustrates a task constellation with multiple path solutions. In this task multiple trajectories with identical intrinsic costs can be generated by moving the arm either to the left or to the right side. To avoid moving straight from the initial position to the target, an obstacle is placed between the positions. Model-based sampling is able to find both possible arm reaching solutions in different runs with minimum costs performing several rollouts. The different trajectories are represented by a

parameter	model-based
rounds	150
samples	500
gain	$[0.000\ 1.082\ 11.459\ 0.319\ 5.569\ 3.884\ 0.097\ 4.484]^T$
$\mathbf{w}_q$	$[4.063\ 2.083\ 4.457\ 2.260\ 4.061\ 1.998\ 3.792\ 2.069]^T$
$\mathbf{w}_u$	$[-0.081\ -0.081\ -0.081\ -0.081]^T$
$\mathbf{w}_T^{task}$	$[4.536\ 4.163]^T$

Table 5.9: Task specific parameters of the model-based sampling method for the task considering multiple arm reaching solutions.



(a) (b)

Figure 5.14: In (a) the different arm reaching paths, generated by model-based sampling using the task space cost representation are shown. Two possible paths are represented by a solid and a dotdashed blue line, generated in the same experiment but in different runs. The corresponding intrinsic costs and the convergence behaviour generating multiple paths is illustrated in (b).

solid line moving to right side and a dotdashed line moving to the left side. The target can be reached by the endeffector of the arm without difficulties with fast converging intrinsic costs, shown in Figure 5.14 (b).
Due to a limited amount of samples, the computed endeffector trajectories differs slightly from each other indicated by the standard deviation of intrinsic costs. One joint angle trajectory is illustrated in Figure 5.15. In contrast to the smooth endeffector trajectory, the joint angle trajectories are jittering and result in extrinsic costs of $2.36e^6$ with standard deviation $1.71e^5$.

5.3.4 Summary

Model-based sampling is able to solve higher dimensional tasks, illustrated by the experiments applying the 4-link model. Tasks including multiple obstacles can be solved as well as tasks with multiple arm reaching solutions by extending the standard method applying

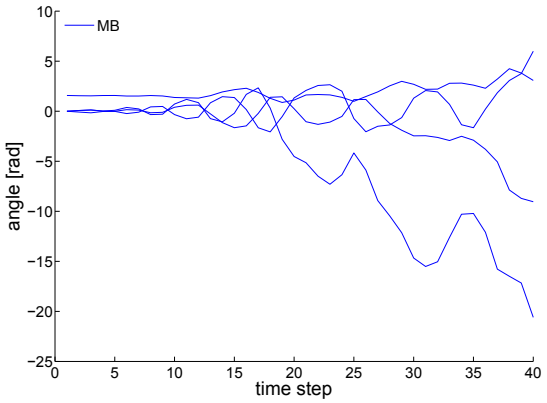


Figure 5.15: The joint angle trajectory of one run of the model-based sampling method is shown, solving the task with multiple arm reaching solutions in task space.

task	model-based
task with a single obstacle	$3.5e^4 \pm 1.5e^3$
task with multiple arm reaching solutions	$4.6e^4 \pm 3.2e^3$
task applying the trajectory mixing approach	$2.2e^6 \pm 2.6e^5$

Table 5.10: The intrinsic costs and standard deviation values for different 4-link tasks are compared. The first two tasks are solved by model-based sampling with a single trajectory. The third task is solved by model-based sampling with trajectory mixing.

trajectory mixing or via points. Although, the computed trajectories are less smooth and the costs are remarkable higher compared to 2-link tasks, shown by the intrinsic costs in Table 5.5 and Table 5.10.

5.4 Trajectory Mixing Approach

This task applies a similar task constellation as in the previous task but includes multiple obstacles, which is used to demonstrate the performance gain of the trajectory mixing approach (introduced in Section 4.3.2). For the trajectory mixing approach the task space cost representation is used. In addition, one joint angle constellation for the target position in state space must be known, to initialise a second trajectory starting at the target.

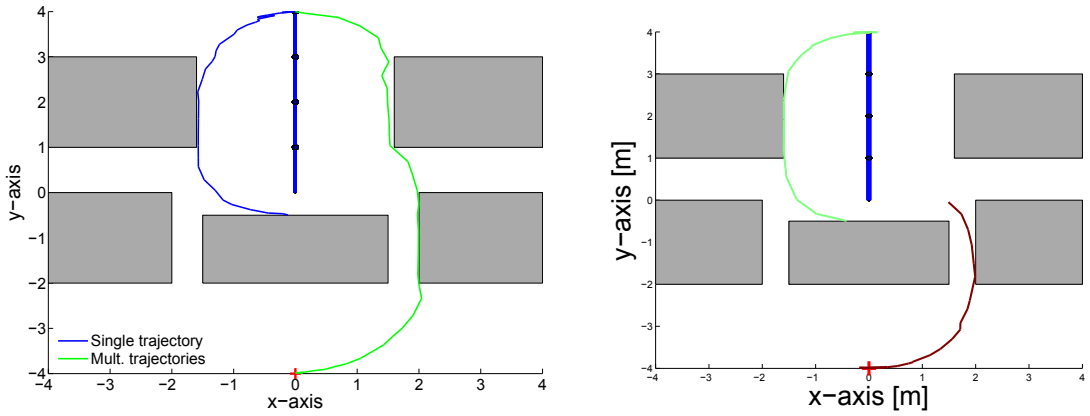
$$\mathbf{q}_1 = \begin{bmatrix} 90\frac{\pi}{180} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_{target} = \begin{bmatrix} 0 \\ -4 \end{bmatrix}, \quad \mathbf{q}_{target} = \begin{bmatrix} -90\frac{\pi}{180} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The learned parameters for model-based sampling are presented in Table 5.11. Due to the increased task complexity, the number of rounds is set to 200.

parameter	model-based
rounds	200
samples	500
gain	$[31.488\ 4.475\ 3.710\ 0.107\ 1.854\ 7.628\ 15.406\ 9.902]^T$
$\mathbf{w}_q$	$[4.361\ 1.938\ 3.858\ 2.179\ 3.975\ 2.149\ 4.303\ 1.815]^T$
$\mathbf{w}_u$	$[-0.081\ -0.081\ -0.081\ -0.081]^T$
$\mathbf{w}_T^{task}$	$[4.014\ 4.051]^T$

Table 5.11: Parameters for model-based sampling including multiple obstacles.

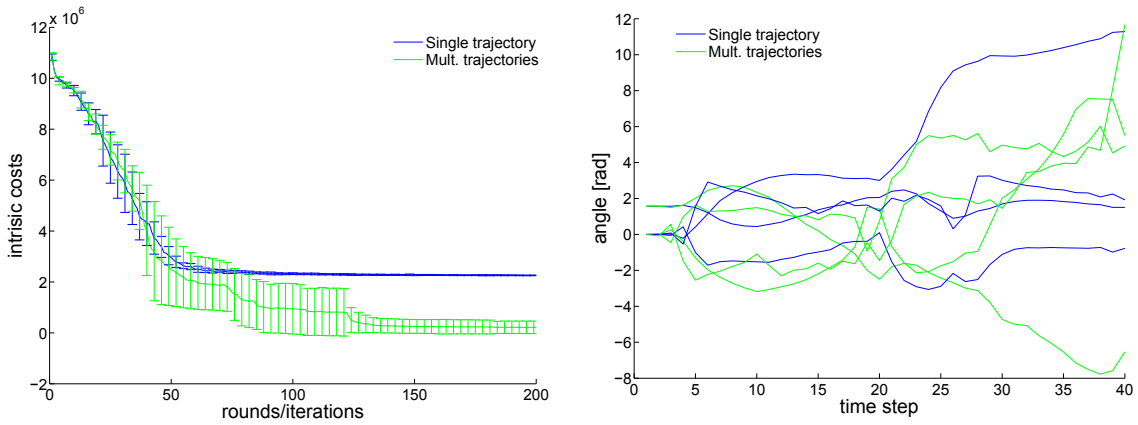
Two model-based sampling methods with different initialisations (one starts at $\mathbf{q}_1$ and one at $\mathbf{q}_{target}$) are implemented. The resulting final trajectory is indicated by the green line, shown in Figure 5.16 (a).



(a) (b)

Figure 5.16: In (a) the trajectories of standard model-based sampling (blue line) and model-based sampling with trajectory mixing (green line) are compared. The target position is marked by a red cross and the obstacles are represented by grey shaded regions. The two trajectories of the trajectory mixing approach are shown in (b) at the time of mixing ($d \leq 2l$).

The trajectory mixing approach is compared to model-based sampling with a single trajectory, denoted by the blue line. The single trajectory of the standard model-based approach is not able to move around the central obstacle.



(a) (b)

Figure 5.17: The intrinsic costs, generated by model-based sampling with a single trajectory and the costs obtained by model-based sampling with trajectory mixing, are shown in (a). The high standard deviation values in the central part of the evolution process for the trajectory mixing approach result from the different mixing points, obtained by different runs. In (b) the corresponding joint angle trajectories are illustrated.

In Figure 5.16 (b) both trajectories of the trajectory mixing approach are presented at the time of mixing (the minimum distance between both trajectories is smaller than δ). Both trajectories separately perform similar to the single trajectory of the standard model-based sampling approach but only the combination is able to solve the task.

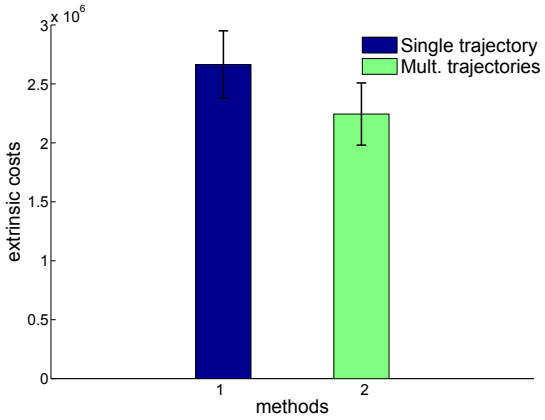


Figure 5.18: The extrinsic costs for standard model-based sampling (blue bar) and trajectory mixing (green bar) are compared.

The maximum mixing distance is crucial for solving the task and depends on the task

complexity. A small distance may prevent mixing both trajectories. With a large distance, mixing is more likely but both trajectories are not able to evolve properly. Hence, the mixing is not very efficient and the result may jump through obstacles. Therefore, the threshold represents a tradeoff between mixing efficiency and the ability to solve the task. The threshold is set manually to $\delta = 2l$.

The intrinsic costs and the corresponding joint angle trajectories are illustrated in Figure 5.17. The high standard deviation values in the central part of the evolution process for the trajectory mixing approach result from the different mixing points, obtained by different runs. The resulting extrinsic costs shown in Figure 5.18 are remarkable lower compared to the 4-link tasks before. This fact is interesting because the task is more complex by considering multiple obstacles constraining possible solutions. The trajectory mixing approach is an alternative extension to the via point approach to solve complex task constellations. Applying the trajectory mixing approach does not influence the solution as introducing a via point. The drawback compared to the via point approach is that the target must be additionally defined and known in state space.

6. Discussion

In this thesis Monte Carlo (MC) sampling methods [4] were investigated to solve Stochastic Optimal Control (SOC) problems with a finite time horizon and quadratic control costs. Two sampling approaches were evaluated. First, a simple model-free sampling approach is evaluated. Second, a model-based approach considering a linearised approximation of the dynamic model is investigated. The sampling methods are an alternative approach to classical SOC methods [16, 28] to solve motor planning tasks, especially when dealing with hard constraints. However, their computational effort prevent their usage in time-critical applications. The extraordinary computational effort compared to Approximate Inference Control (AICO), results from the necessity of adapting parameters for each task and drawing an appropriate number of samples. Especially, considering multiple obstacles leads to a high computational effort, where each drawn sample must be evaluated if it violates any obstacle. However, several important aspects should be considered to solve motor planning tasks efficiently. Due to their local search behaviour, sampling methods require an appropriate initialisation to find the optimal trajectory. This is also confirmed by the obtained results from the experiments. However, finding the optimal trajectory can only be guaranteed drawing infinitely many samples from the distribution which is practically infeasible. Therefore, a tradeoff must be found between an optimal approximation of the desired distribution and computational feasibility. Another crucial aspect to solve a task efficiently is to find appropriate values for the policy parameter vector $\boldsymbol{\theta}$. Typically optimal values for $\boldsymbol{\theta}$ depend on a specific task. Adjusting the parameters manually is demanding and learning optimal values is not guaranteed to converge to the global optimum. In the experiments, the parameter vector $\boldsymbol{\theta}$ is learned for each task separately by applying the Covariance Matrix Adaption - Evolution Strategy (CMA-ES) algorithm [11]. However, the parameters of the via point are set manually for each task and the results show that the chosen parameters are in most cases not optimal. For further experiments I recommend learning the parameters of via points as well. Extending the model-based sampling method with the via point or trajectory mixing approach results in a noticeable performance improvement and tasks with higher complexity can additionally be solved. For further experiments, hierarchical sampling may be promising to decrease the computational effort by introducing several layers for different time resolutions. Also hybrid methods, applying sampling methods only in regions near to obstacles and using

traditional SOC methods elsewhere may be able to decrease the computational effort. The acquired knowledge from this thesis may be also relevant for further research on motor planning with spiking neural networks, e.g. the differences between task space planning and state space planning.

7. Conclusion

In this thesis a model-free [4] and a model-based Monte Carlo (MC) sampling approach are investigated for motor planning tasks with hard constraints. To improve the performance of these sampling methods, a via point approach and a trajectory mixing approach were applied. The methods were evaluated on tasks using dynamic 2-link and 4-link robot arm models. The basic behaviour using hard constraints, different initialisation methods, as well as differences between state and task space cost representations, and the performance improvements applying different extensions were evaluated. On a 2-link model the sampling methods were compared to a standard implementation of Approximate Inference Control (AICO) [28]. Both sampling methods do not meet the performance of AICO but can generate trajectories without obstacle violations. Further model-based sampling is able to solve complex 4-link motor planning tasks using either a state space cost representation or a task space cost representation for planning. With the novel trajectory mixing extension, model-based sampling is able to solve complex 4-link tasks with an increased performance, where single trajectory based approaches fail.

Appendix A. Abbreviations

AICO Approximate Inference Control

CMA-ES Covariance Matrix Adaption - Evolution Strategy

DMPs Dynamic Movement Primitives

EP Expectation Propagation

ILQR Iterative Linear Quardratic Regulator

LQG Linear Quadratic Gaussian

MAP Maximum A Posteriori

MC Monte Carlo

MCMC Markov Chain Monte Carlo

MPs Movement Primitives

PMPs Planning Movement Primitives

SOC Stochastic Optimal Control

SQP Sequential Quadratic Programming

Appendix B. Gaussian Identities

This appendix defines the used gaussian identity equations in this thesis. More detailed information about gaussian identities can be found in [30].

Gaussian distribution over x with mean a and covariance matrix $\mathbf{A}$

$$\mathcal{N}(x|a,\mathbf{A}) \quad = \quad \tfrac{1}{|2\pi\mathbf{A}|^{1/2}} \exp(-\tfrac{1}{2}(x-a)^T\mathbf{A}^{-1}(x-a)) \tag{B.1}$$

Canonical form of a Gaussian distribution with belief a and precision matrix $\mathbf{A}$

$$\mathcal{N}[x|a,\mathbf{A}] \quad = \quad \tfrac{\exp(-\frac{1}{2}a^T\mathbf{A}^{-1}a)}{|2\pi\mathbf{A}^{-1}|^{1/2}} \exp(-\tfrac{1}{2}x^T\mathbf{A}x + x^Ta) \tag{B.2}$$

with properties

$$\mathcal{N}[x|a,\mathbf{A}] \quad = \quad \mathcal{N}(x|\mathbf{A}^{-1}a,\mathbf{A}^{-1}) \tag{B.3}$$

$$\mathcal{N}(x|a,\mathbf{A}) \quad = \quad \mathcal{N}[x|\mathbf{A}^{-1}a,\mathbf{A}^{-1}] \tag{B.4}$$

Product of two Gaussians in canonical form

$$\mathcal{N}[x|a,\mathbf{A}]\,\mathcal{N}[x|\mathbf{b},\mathbf{B}] \quad \propto \quad \mathcal{N}[x|\mathbf{a}+\mathbf{b},\mathbf{A}+\mathbf{B}] \tag{B.5}$$

Linear transformation

$$\mathcal{N}(\mathbf{F}x+f|a,\mathbf{A}) \quad = \quad \tfrac{1}{|\mathbf{F}|}\mathcal{N}(x|\mathbf{F}^{-1}(a-f),\mathbf{F}^{-1}\mathbf{A}\mathbf{F}^{-T}) \tag{B.6}$$

Propagation rule

$$\int_y \mathcal{N}(x|a+\mathbf{F}y,\mathbf{A})\ \mathcal{N}(y|b,\mathbf{B})\ dy \quad = \quad \mathcal{N}(x|a+\mathbf{F}b,\mathbf{A}+\mathbf{F}\mathbf{B}\mathbf{F}^T) \tag{B.7}$$

Bibliography

[1] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2):5–43, 2003.

[2] D. Barber and W. Wiegerinck. Tractable Variational Structures for Approximating Graphical Models. In *NIPS*, pages 183–189, 1998.

[3] Biorobotics Laboratory École Polytechnique Fédérale de Lausanne (EPFL). Control architecture for discrete and rythmic movements. `http://biorob.epfl.ch/page-36370-en.html`. Visited on May 3rd 2013.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

[5] Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7(11):e1002211, 11 2011.

[6] O. Ciccarelli, A. T. Toosy, J. F. Marsden, C. M. Wheeler-Kingshott, C. Sahyoun, P. M. Matthews, D. H. Miller, and A. J. Thompson. Identifying brain regions for integrative sensorimotor processing with ankle movements. In *From Motor Learning to Interaction Learning in Robots*, volume 166 of *Experimental Brain Research*, pages 31–42. Springer-Verlag, 2005.

[7] S. Degallier, L. Righetti, L. Natale, F. Nori, G. Metta, and A. Jispeert. A modular bio-inspired architecture for movement generation for the infant-like *iCub*. In *Proceedings on the 2nd Biennial IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2008.

[8] A. Doucet and A. M. Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen years later, December 2008.

[9] J. F. Engelberger. Historical Perspective and Role in Automation. In *Handbook of Industrial Robotics*, pages 3–10. John Wiley & Sons, Inc., 1999.

[10] Thomas Griffiths, Thomas L. Griffiths, and Joshua B. Tenenbaum. Optimal predictions in everyday cognition. *Psychological Science*, 17:767–773, 2006.

[11] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.*, 11(1):1–18, March 2003. ISSN 1063-6560.

[12] Famento Inc. xtimeline. `http://www.xtimeline.com/evt/view.aspx?id=113072`. Visited on May 2nd 2013.

[13] T. S. Jaakkola. Tutorial on Variational Approximation Methods. In *Advanced Mean Field Methods*, pages 129–159, 2001.

[14] M. I. Jordan. An introduction to variational methods for graphical models. In *Machine Learning*, pages 183–233. MIT Press, 1999.

[15] M. I. Jordan and Y. Weiss. Probabilistic inference in graphical models. *The Handbook of Brain Theory and Neural Networks*, page 17, 2002. URL `http://www.cs.iastate.edu/~honavar/jordan2.pdf`.

[16] H. J. Kappen. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics-theory and Experiment*, 2005, 2005. doi: 10.1088/1742-5468/2005/11/P11011.

[17] H. J. Kappen. An Introduction to stochastic control theory, path integrals and reinforcement learning. In *Proceedings 9th Granada Seminar on Computational Physics: Computational and Mathematical Modeling of Cooperative Behavior in Neural Systems*, September 2006.

[18] H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Computing Research repository (CoRR)*, abs/0901.0633, 2009.

[19] O. Kroemer, E. Ugur, E. Oztop, and J. Peters. A Kernel-based Approach to Direct Action Perception. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2012. URL `http://www.ias.informatik.tu-darmstadt.de/publications/Kroemer_ICRA_2012.pdf`.

[20] T. Mensik, J. Verbeek, and B. Kappen. EP for Efficient Stochastic Control with Obstacles. In *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 675–680, 2010.

[21] T. P. Minka. Expectation Propagation for approximate Bayesian inference. In *Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

[22] K. Muelling, J. Kober, and J. Peters. Learning Table Tennis with a Mixture of Motor Primitives. In *International Conference on Humanoid Robots*, 2010.

[23] National Aeronautics and Space Administration (NASA). Curiosity. `http://www.nasa.gov/mission_pages/msl/news/msl20121106.html`. Visited on May 2nd 2013.

[24] Dejan Pecevski, Lars Buesing, and Wolfgang Maass. Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Comput Biol*, 7(12):e1002294, 12 2011.

[25] E. A. Rückert, G. Neumann, M.and Toussaint, and W. Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in Computational Neuroscience (Special Issue on Modularity in motor control: from muscle synergies to cognitive action representation)*, 6(97), 2013.

[26] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research (ISRR2003)*, pages 561–572. Springer-Verlag, 2003.

[27] E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005 In American Control Conference*, pages 303–306, 2005.

[28] M. Toussaint. Robot Trajectory Optimization using Approximate Inference. In *25th International Conference on Machine Learning (ICML)*, pages 1049–1056, 2009.

[29] M. Toussaint. Pros and cons of truncated Gaussian EP in the context of approximate inference control. In *NIPS-Workshop on Probabilistic Approaches for Robotics and Control*, 2009.

[30] M. Toussaint. Lecture Notes: Gaussian identities, 2011.

[31] M. Toussaint and C. Goerick. A Bayesian View on Motor Control and Planning. In *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 227–252. Springer-Verlag, 2010.