Probabilistic inference for structured planning in robotics

Marc Toussaint and Christian Goerick

Abstract— **Real-world robotic environments are highly structured. The scalability of planning and reasoning methods to cope with complex problems in such environments crucially depends on exploiting this structure. We propose a new approach to planning in robotics based on probabilistic inference. The method uses structured Dynamic Bayesian Networks to represent the scenario and efficient inference techniques (loopy belief propagation) to solve planning problems. In principle, any kind of factored or hierarchical state representations can be accounted for. We demonstrate the approach on reaching tasks under collision avoidance constraints with a humanoid upper body.**

I. Introduction

Planning in high-dimensional robotic systems is a fundamental and critical problem when the goal is more autonomous, problem-solving robots in natural environments. Many behaviors that we would like robots to exhibit, such as autonomously reasoning about tools to manipulate objects, considering different paths to a goal, and generally coping with unexperienced situations, cannot be based on purely reactive behavior and require reasoning about the future.

Most approaches to planning try to directly cope with the high-dimensional state space, using smart heuristics to find paths in this space, e.g., using probabilistic road maps, or Rapidly Exploring Random Trees [1], [2]. Although these approaches provide practical solutions to many real-world applications (such as foot step planning), the fundamental question of scalability remains unsolved when the full high-dimensional state space has to be considered. In particular, structural knowledge about the system is not exploited.

What we aim for is a new approach to planning based on a factored representation of state, i.e., based on models of the environment and the robot that use multiple variables to describe the current state. These variables might represent different "parts" of the state (such as different body parts, different objects, or different constraints), but they may also provide different levels of abstractions of the current state ("situations") as in hierarchical models (see below and the discussion). Reasoning and inference on such factored models is a challenge that has been extensively addressed in the Machine Learning community [3], [4], [5], [6]. Applied to the realm of planning, these techniques provide a fundamentally new perspective on decomposing planning processes and exploiting the inherent structure in the coupling between random variables.

Marc Toussaint is with the Machine Learning group at Technical University Berlin, Franklinstr. 28/29, 10587 Berlin, Germany; Christian Goerick is with the Honda Research Institute Europe, Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany.
mtoussai@cs.tu-berlin.de
christian.goerick@honda-ri.de

We base our approach on a recently developed framework for planning based on probabilistic inference [7], where Dynamic Bayesian Networks (DBNs) are used to infer and emit optimal control signals rather than to model observed data. The general idea is that Machine Learning has developed very rich techniques to learn, e.g., hierarchical or factored structured probabilistic models (latent variable models) from data as a method to analyze and represent the inherent structure (e.g., hierarchical Hidden Markov Models [8], factorial HMMs [9]). We use exactly such structured models for the *generation* of control signals rather than as a model of observed data. For instance, hierarchical probabilistic models provide a framework for motor (or motion) primitives or macro policies (see, e.g., [10] and [11]); factored models provide a framework for multivariate (e.g., multiple object) state representations.

We will discuss this in more detail in the concluding section. As a first demonstration of the approach we address the scenario of planning in a redundant kinematic system (a humanoid upper body) under task and collision constraints. The system is described by multiple variables, including the joint state, the endeffector position and collision variables. Inference in this model allows us to compute a posterior distribution over possible trajectories that fulfill the constraints – i.e., we compute a whole distribution over trajectories rather than a single trajectory. The theoretical grounding of the approach guarantees that likelihood maximization w.r.t. the control parameters is equivalent to solving the planning problem in the sense of maximizing a corresponding global cost function.

The next two sections will first introduce DBNs as a structured model description and then the general methodology of using inference for planning in such models. Section IV describes the inference technique we used (loopy belief propagation on factor graphs) and section V the two experiments we performed.

II. Dynamic Bayesian Networks to represent structured planning problems

Real-world robotic problems are highly structured and one should exploit this structure when reasoning about possible trajectories. Graphical Models have provided a very powerful understanding of 'structure' in the realm of Machine Learning. Roughly, one assumes that the full state of the system is described by *multiple* random variables, the coupling of which is described by a graphical model. This is contrary to the view that the system's state is just a point is a high-dimensional unstructured state space.

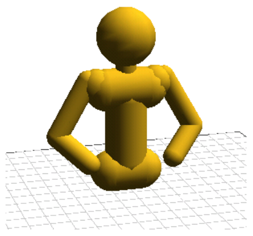


Fig. 1. A humanoid upper body with $n = 13$ hinge joints. The hip is fixed, the right hand is used as endeffector.

Transferred to the realm of robotics, the random variables we are concerned with are typically dynamic variables which are coupled via kinematic or dynamic laws or imposed constraints. Such structured dynamic processes are captured in terms of Dynamics Bayesian Networks (DBNs); see [12] for an excellent overview on how hierarchical, factored, or otherwise structured processes can be captured.

In this paper the experiments refer to a humanoid upper body (Figure 1) with $n = 13$ joints. If the hip is assumed fixed, the configuration is determined by the joint state[1] $q \in \mathbb{R}^n$. With the position $x \in \mathbb{R}^3$ of the right hand as the endeffector, this defines the non-linear kinematics $\phi :\ q \mapsto x$.

Figure 2 captures the scenario in terms of a Dynamic Bayesian Network. Here, x and q are treated as dynamic random variables. The transitions $P(q_{t+1} \,|\, q_t)$ are the free parameters of the problems and relate to the control signals that we will learn via probabilistic inference. The coupling $P(x_t \,|\, q_t)$ is actually deterministic and given by the forward kinematics ϕ. Note though that its inverse $P(q_t \,|\, x_t)$ is not at all deterministic but, via Bayes' rule, a distribution over the null-manifold (the set $\{q : \phi(q) = x_t\}$). In this model we added an extra dependency $P(x_{t+1} \,|\, x_t)$ which will allow us later to impose extra constraints on the endeffector trajectory such as smoothness and collision avoidance.

III. General approach: Probabilistic Inference for reasoning about trajectories

In [7] it was shown that a general planning problem (maximization of an arbitrary global reward function) can equivalently be translated into a problem of likelihood maximization in a corresponding Dynamic Bayesian Network description of the problem. In the context of planning problems, likelihood maximization amounts to an EM-algorithm which yields an optimal (control) policy. The core of this EM-algorithm is an inference procedure, deriving a posterior probability distribution over possible trajectories. We refer the reader to [13] for theoretical result and demonstrations on discrete and continuous Markov Decision Processes. Here, we want to give a more informal introduction of the approach of using probabilistic inference for trajectory generation in the context of robotics. The outline of the approach is as follows:

(i) First formulate the scenario in terms of a Dynamic Bayesian Network, using a multi-variate description of state and structured graphical models to describe possible inter- and intra-temporal couplings.

(ii) Constraints such as smoothness or feasibility of transitions or consistency constraints between variables are introduced by including additional factors in the model. For instance, some of these define the *prior* probability of transitions. This is elegantly done in factor graph representations of the DBN, see below.

(iii) Target constraints such as reaching a target position with the endeffector are also introduced by conditioning the respective variable (effectively introducing further factors).

(iv) An inference algorithm on the factor graph will compute the *posterior distribution over all trajectories fulfilling the constraints.* (E-step)

(v) Control parameters for each time step can be extracted by investigating the *posterior* transition probabilities (typically in the joint state). (M-step)

In the example of the humanoid upper body, we already addressed step (i) in Figure 2. For the inference algorithm and also for imposing additional constraints it is favorable to represent this DBN as a factor graph [14] as in Figure 3. Formulating constraints within this framework (step (ii)) is done by specifying certain prior probabilities in terms of factors. In our case, we include a factor

$$f_1(q_{t+1}, q_t) \propto P(q_{t+1} \,|\, q_t) = \mathcal{N}(q_t, W^{-1}) \tag{1}$$

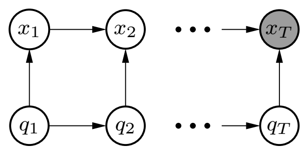
to constrain the likely joint state transitions. Here, $\mathcal{N}(c, C)$ is the normal distribution with mean c and covariance matrix C. The matrix W is a metric in joint space that weights movements for different joints differently. We used $W = \text{diag}(w_1, .., w_n)$ with $w_1, w_2 = 100$ for the two torso joints and $w_{3,..,13} = 20$ for all other joints. Further we include a prior $P(q_t)$ as a factor in the factor graph, which limits the joint range – for simplicity we use again a Gaussian potential

$$f_2(q) \propto P(q_t) = \mathcal{N}(0, 1.0) \ , \tag{2}$$

where $q = 0$ denotes the joint centers. Next we include a factor expressing our prior about the endeffector movement. Since we do not have a strong prior about this, we assume a weak potential

$$f_3(x_{t+1}, x_t) \propto P(x_{t+1} \,|\, x_t) = \mathcal{N}(x_t, 0.1) \ , \tag{3}$$

[1]Please note the unfortunate ambiguity of the word 'joint' in our field: as in the joint [combined] probability distribution of all random variables – versus the joint [hinge] state of the robot.



$$\begin{tikzcd} x_1 \arrow[r] & x_2 & \cdots \arrow[r] & x_T \\ q_1 \arrow[u] \arrow[r] & q_2 \arrow[u] & \cdots \arrow[r] & q_T \arrow[u] \end{tikzcd}$$

Fig. 2. Dynamic Bayesian Network of a redundant planning problem. q is the joint state, ξ the endeffector position. The gray shading of X_T indicates that we condition the final endeffector position, i.e., constrain it to be the target.

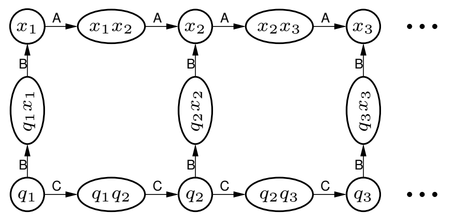


Fig. 3. The factor graph for the DBN in Figure 2. We use this to implement the generic inference algorithm. The labels A, B, C indicate the order of belief propagation as described in section IV.

implying a prior standard deviation of endeffector movements of 0.1. Finally we include the most important factor in the model: the coupling between joint state and endeffector. This coupling is given via the kinematic map ϕ and introduced by a factor

$$f_4(x_t, q_t) \propto P(x_t \,|\, q_t) = \mathcal{N}(\phi(q), 0.001) \ , \tag{4}$$

which assumes that small perturbations (standard deviations 0.001) of the kinematic map might be possible.

This completes the specification of couplings and constraints between the variables. Step (iii) then determines the actual task to be fulfilled – in our case by conditioning the final endeffector random variable x_T to be at the desired target x^*_T. Instead of a hard conditioning with a Kronecker-δ we use a softer conditioning with a Gaussian potential of small variance:

$$f_5(x_T) \propto P(x_T) = \mathcal{N}(x^*_T, 0.001) \ . \tag{5}$$

The factors $f_1, .., f_5$ we have introduced so far determine the factor graph in Figure 3. Step (iv) then uses generic probabilistic inference techniques to compute the posterior distributions on the factor graph for each variable and transitions. We use an inference technique *that exploits the structure of the problem*, namely a specific scheme of loopy belief propagation [15], which is detailed in the next section.

Finally, step (v) extracts a control law from the computed posterior transition probabilities. More specifically, if inference yields a joint posterior probability of (q_t, q_{t+1}) as

$$\xi(q_{t+1}, q_t) = \mathcal{N}(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} A & C \\ C^T & B \end{pmatrix}) \ , \tag{6}$$

then the corresponding kinematic control signal at time t is a desired displacement

$$u_t(q_t) = b - q_t + C^T A^{-1}(q_t - a) \ , \tag{7}$$

which forms an attractor signal towards $q_{t+1} \to b$. (See also [7] for details on the M-step.)

To conclude this section, first note that, to follow a proper EM-algorithm for likelihood maximization, the E- and M-steps would have to be reiterated until convergence. However, for our applications a single EM-cycle was sufficient to yield appropriate solutions. One may expect that in more complex planning problems more EM-cycles are necessary – but this remains in the scope of future research.

Second, in some applications it is not desirable to fix the total time T in advance: we might also be interested in trajectories within a certain time window, or in optimizing a *discounted future reward* that favors shorter trajectories. In [7], such problems in the case of simple Markov Decision Processes are addressed and this technique can be transferred to the redundant control scenario.

Finally, any traditionally formulated cost function can equivalently be translated into a factor of the factor graph (via a log-transform) such that global cost minimization is equivalent to likelihood maximization (which is also clear from [7]).

IV. The inference algorithm

In principle, the algorithm we use for inference in our problem is a standard approach and we could refer to the standard literature [12], [15]. However, we wish to discuss the inference process in more detail for two reasons: (1) this will allow us a more detailed comparison to traditional planning and control approaches, and (2) it will provide us with a more intuitive interpretation of the inference process: basically a process of establishing coherence between possible trajectories and all constraints, including future target constraints or collision constraints.

A. Alternating between decoupled planning and coupling

We are given the factor graph in Figure 3 which extends over T time steps and the factors $f_1, .., f_5$ defined above. The problem of inference is very closely related to finding low energy states in physical systems where only neighboring particles interact and factors $f_1, .., f_5$ define local energy potentials. This view adds some intuitive understanding of what the inference process computes: basically a distribution that describes possible trajectories that are 'coherent' with all defined constraints, be they in the future or the present.

Just as physical interactions are only local between neighboring particles, inference algorithms can be based on only local message passing between coupled random variables (loopy belief propagation, [15]). The easiest kind of inference process is a forward-backward process of message passing along a chain of coupled random variables, as in hidden Markov Models (also called the Baum-Welch algorithm) or Kalman smoothers. This principle can be generalized to other orders of message passing and coupling structures.

In our concrete model (Figure 3) we have two chains of random variables, one for the joint state $q_{1,..,T}$ and one for the endeffector $x_{1,..,T}$. Both chains are coupled 'vertically' at each time slice. The inference algorithm chosen to solve the problems is simply defined by the order of message passing we apply. Referring to the edge labels A, B, C in Figure 3, our order of message passing is:

(1) A fwd & bwd: first pass messages forward and backward along the endeffector chain $x_{1,..,T}$,
(2) B bwd: then pass messages 'vertically down' from the endeffector state x_t to the joint state q_t at each time slice,
(3) C fwd & bwd: then pass messages forward and backward along the joint state chain $q_{1,..,T}$,

(4) B fwd: then pass messages 'vertically up' from the joint state q_t to the endeffector state x_t at each time slice.

This scheme is iterated until convergence.

The inference scheme can be interpreted in an intuitive manner. Step (1) first calculates a distribution of possible endeffector trajectories that neglects the existence of joint constraints and only accounts for the endeffector state and goal conditioning. The result will describe a straight line from start to goal with variances (indicating a large variance of possible trajectories). Step (2) uses the kinematic map to induce an extra potential for the joint state (aligned with the nullspace) – this constrains (probabilistically) the possible trajectories in joint space. Step (3) then computes a distribution over possible joint state trajectories taking account of this extra potential (the endeffector constraint). Step (4) projects these possible joint state trajectories up into the endeffector space and the iterated step (1) now computes a new distribution over possible endeffector trajectories – this time accounting for the coupling to the joint state. In our examples two iterations will be enough to provide sufficient convergence.

Let us compare this procedure to a traditional redundant control scheme (see, e.g., [16] for a concise presentation). Step (1) can be interpreted as a trivial endeffector planner (generating a straight line). Step (2) corresponds to the typical nullspace identification via the (weighted) pseudo-inverse Jacobian. Now, the forward pass in step (3) is still in close analogy to the classical forward controller using the pseudo-inverse Jacobian. However the backward pass in step (3), which smoothes the joint state trajectory and 'carries back' constraints from the future to the present, does not have a classical analogue. It is exactly here, where we qualitatively depart from classical forward control. The subsequent inference process further refines the distribution over possible trajectories to globally account for the constraints and couplings.

B. Implementation and scaling

We developed our own implementation of general loopy belief propagation on factor graphs. The software and a detailed reference to factor graphs, message passing, and our implementation is freely accessible.[2]

The computational complexity of inference is basically linear in the number of message passings needed. Hence, a single forward-backward pass along one variable is linear in T. If the number of neighbors to each variable is bounded, then the total number of edges in the factor graph is $O(TK)$, i.e., linear in the number K of variables. For complex (e.g., hierarchically deep) DBNs it is however an open question how many iterations of inference sweeps one needs until convergence. For the experiments we will indicate the real computational time. (Note though that, so far, our implementation focuses on robustness and correctness rather than computational speed.)

[2]http://www.marc-toussaint.net/06-infer

	trajectory cost $C(q_{1,..,T})$
forward controller	11.19
MAP trajectory	8.14

TABLE I
GLOBAL COST OF JOINT SPACE TRAJECTORIES.

V. EXPERIMENTS

A. Reaching with a humanoid upper body

The first experiment considers the $n = 13$ joint humanoid upper body displayed in Figure 1. We take the right hand as the endeffector and plan a target reaching trajectory (of length $T = 50$) to a goal in the upper left working domain of the robot.

Figures 5(a&b) display the result after 2 iterations of the inference steps (1-4), which provided sufficient convergence. The figures display the maximum a posteriori joint configuration (MAP, the maximum of the posterior joint state distribution) and the variance of the endeffector distribution at different time steps. As we can see, the MAP endeffector trajectory is not a straight line. We can give a more quantitative measure of the quality of the trajectory: We compare the MAP joint trajectory computed via probabilistic inference with the joint trajectory that results from a standard redundant control approach. More precisely, the redundant control approach first presumes a straight endeffector line equally divided in $T = 50$ segments. Then, starting with the initial joint configuration q_0, the controller uses the weighted pseudo-inverse Jacobian to emit a desired step in joint space

$$u_t(x_t) = J^{\#}(x^*_{t+1} - x_t) \ , \quad J^{\#} = W^{-1}J^T(JW^{-1}J^T)^{-1} \ , \tag{8}$$

where $W = \text{diag}(100, 100, 20, 20, .., 20)$ weights movements in joints differently, and x^*_t is the desired intermediate endeffector position on the straight line.

Given this controller, it is appropriate to use the metric W for defining the global trajectory cost,

$$C(q_{1,..,T}) = \sum_{t=1}^{T-1} ||q_{t+1} - q_t||_W \ . \tag{9}$$

Table I displays the trajectory costs for the trajectories computed via the forward controller and the MAP trajectory computed via probabilistic inference. The MAP trajectory is clearly more efficient in terms of this cost. This stems from the fact that equation (1) imposes a prior transition likelihood $f_1(q_{t+1}q_t) \propto \mathcal{N}(q, W^{-1})$ which reflects exactly this metric in joint space. The curve of the endeffector trajectory is a result of this efficiency.

B. Coupling with collision constraints

Coupling extra constraints into the system is straightforward. To demonstrate this we augment the model with a collision risk variable c_t. In the experiment, we will consider collisions of the endeffector with a table. Figure 4 displays the factor graph augmented with this variable. Note that we

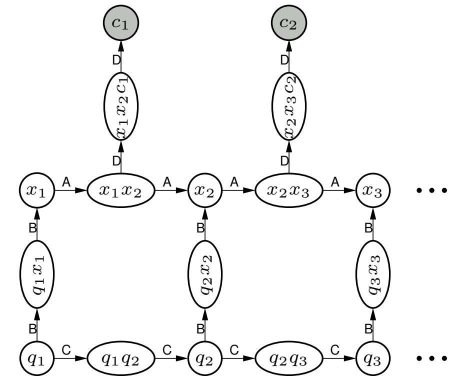


Fig. 4. The factor graph we used for belief propagation. The labels A, B, C indicate the order of belief propagation as described in the text.

decided to couple this variable *not* to a single endeffector position x_t but rather to an endeffector *transition* given by a tuple (x_t, x_{t+1}). This turned out to be more robust and accurate, particularly in view of the time discretization used.

We define the risk variable with a non-linear sigmoidal function as follows. Let $z_t, y_t \in \mathbb{R}$ be z- and y-coordinate of the endeffector position $x_t \in \mathbb{R}^3$, let z^* be the table height. Then

$$c(x_{t+1}, x_t) = \begin{cases} 0 & \text{if } z_t, z_{t+1} > z^*+\delta \text{ or } z_t, z_{t+1} < z^*-\delta \\ \psi(y_t + y_{t+1}) & \text{otherwise} \end{cases} \quad (10)$$

with $\delta = .02$. Thus, the risk is zero if x_t and x_{t+1} are both either above or below the table, and a sigmoid depending on the y-distance to the table corner otherwise. We used $\psi(y) = 1 - \frac{1}{1+\exp(-3y)}$. Just as for the kinematics, this risk function defines a coupling factor

$$f_6(c_t, x_{t+1}, x_t) \propto \mathcal{N}(c(x_{t+1}, x_t), .001) \ . \quad (11)$$

Importantly, we also impose a prior on the risk variable, effectively constraining it to be low, by including the factor

$$f_7(c_t) \propto \mathcal{N}(0, .1) \ . \quad (12)$$

The other factors $f_1, .., f_5$ remain the same as in the previous experiment, and together with f_6 and f_7 define the factor graph in Figure 4. For inference, we need to decide on an order of message passing. Referring to the edge labels indicated in Figure 4, we choose

(1) A fwd & bwd
(2) D bwd
(3) A fwd & bwd
(4) B bwd
(5) C fwd & bwd
(6) B fwd

Figures 5(c&d) display the result after two iterations of this message passing scheme for $T = 30$. In the first case (Figure (c)) the target endeffector position is slightly above the table and the generated movement avoids the obstacle. In the second case, the target is only slightly displaced but now below the table. Here, the result is a rather straight trajectory. A standard forward controller that follows a gradient of a superimposed target potential and obstacle avoidance potential would typically get trapped under the table in the first case – so we cannot present a quantitative comparison here. Also, the local target potential of a reactive controller would hardly distinguish between the first and the second case.

The experiments ran on a simple Laptop with a 1.1GHz Centrino Mobile processor. The first experiment ($T = 50$, without constraints, $k = 2$ inference sweeps) takes 3.56 seconds, the second experiment ($T = 50$, with constraints, $k = 2$ sweeps) takes 3.97 seconds.

VI. Discussion

In this paper we propose a new approach to planning in structured robotic applications using probabilistic inference. The approach is based on recent theoretical work, showing that planning (future reward maximization) can be translated into a problem of likelihood maximization – and thereby reduced to the core problem of inference in structured Dynamic Bayesian Networks. First of all, this approach departs from planning in an unstructured high-dimensional system state space and rather aims at exploiting the structure that can be expressed in a multi-variate description of state and local couplings between system variables. This has previously been addressed with traditional Reinforcement Learning techniques in the context of Markov Decision Processes (e.g., [3], [4], [5], [6]), but no demonstration on continuous robotic systems with non-linear, redundant kinematic couplings has been presented before. A hierarchical decomposition of control has been addressed in [17], which proposes a heuristic similar to our inference steps (1-3): “first plan on the endeffector, then project this in the joint space and plan there”. However, no back-coupling of the result in joint space to the endeffector is considered and there is no theoretical grounding of the method such as likelihood maximization. This approach would also lead to a non-optimal, straight endeffector trajectory in the experiment in section V-A.

DBNs provide a very powerful way to express structure. Murphy [12] gives a comprehensive overview of how factorization, hierarchies, coupling between discrete and continuous variables, variables on different time scales, etc. can be represented in DBNs. In the context of robotics, this means that the framework can encompass many levels of representation in one single model – including lower level representations as the ones addressed in the our experiments, as well as higher level symbolic representations (such as macro actions or switching variables, see also [11]). The future perspective is that using inference processes on such structured DBNs provides a principled solution to integrating the various processes of behavior generation – reactive control, trajectory planning, symbolic reasoning, which refer to different levels of representation – in one coherent framework.

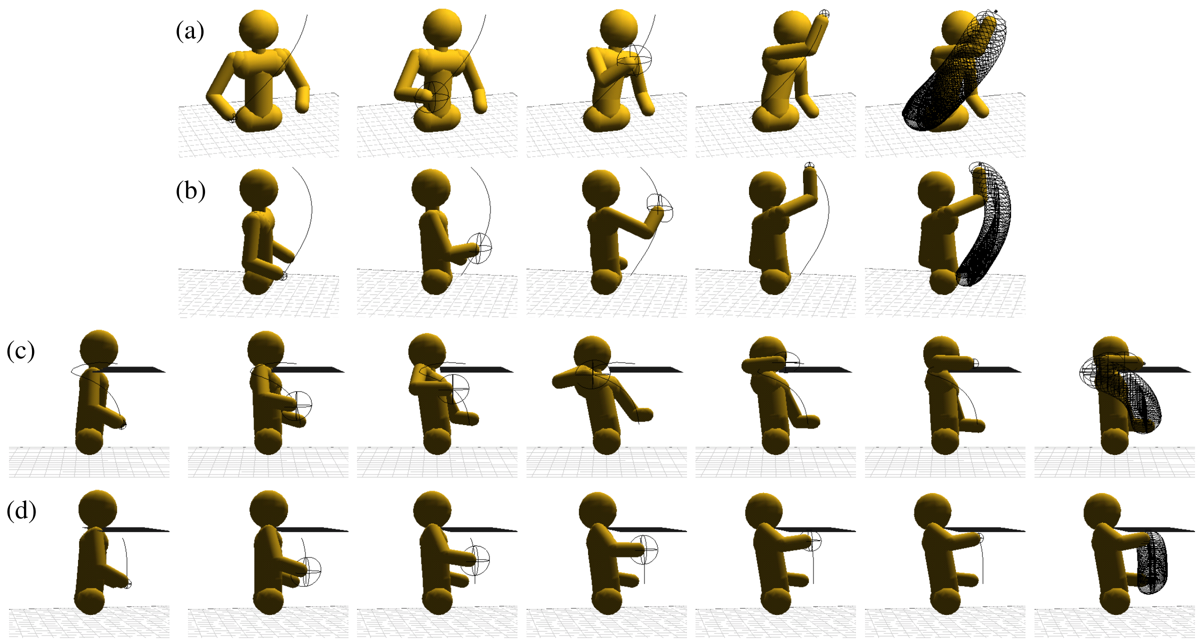


Fig. 5. Results of probabilistic inference planning with an humanoid upper body. (a&b) Reaching to a target without obstacles, displayed from two different perspectives. We see the MAP joint configuration and the Gaussian endeffector distribution (indicated as ellipsoid) at different intermediate time steps. The optimal trajectory in joint space leads to a curve trajectory in endeffector space. (c) Reaching to a target above the table and (d) below the table whilst avoiding collision.

Generally, the structure of the model will reflect the environment's structure and the internal representations used, but does not depend on the current task which is specified by conditioning some variables (step (iii) in the method's outline). For instance, changing the target position only requires one to adapt the factor conditioning the endeffector variable x_T. This provides high flexibility in using the same structural model for a large variety of tasks.

Finally, the probabilistic approach allows us to (1) correctly account for sensor and motor noise, and (2) compute exact probabilities, e.g. the probability of success or collision, which may be used for action selection on a higher level. Future work should (1) extend the current inference software to include more flexible distribution representations (including mixture of Gaussians and particle filters) and (2) address more complex robotic scenaros with more interacting variables.

Acknowledgments

MT is grateful to Honda RI Europe for their hospitality as a guest scientist. MT also acknowledges support by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3.

References

[1] L. Kavraki, P. Svestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of IEEE Int'l Conf. on Robotics and Automation*, 2000.

[3] D. Koller and R. Parr, "Computing factored value functions for policies in structured MDPs," in *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI 1999)*, 1999, pp. 1332–1339.

[4] Schuurmans and Patrascu, "Direct value-approximation for factored MDPs," in *Advances in Neural Information Processing (NIPS 2001)*, 2001, pp. 1579–1586.

[5] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman, "Efficient solution algorithms for factored MDPs," *Journal of Artificial Intelligence Research (JAIR)*, vol. 19, pp. 399–468, 2003.

[6] B. Kveton and M. Hauskrecht, "Solving factored MDPs with exponential-family transition models," in *Proc. of the 16th Int. Conf. on Planning and Scheduling*, 2006.

[7] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state Markov Decision Processes," in *23nd Int. Conf. on Machine Learning (ICML 2006)*, 2006.

[8] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden markov model: Analysis and applications," *Machine Learning*, vol. 32, pp. 41–62, 1998.

[9] Z. Ghahramani and M. I. Jordan, "Factorial hidden Markov models," in *Advances in Neural Information Processing Systems, NIPS*, vol. 8. MIT Press, 1995, pp. 472–478.

[10] C. Boutilier, R. Dearden, and M. Goldszmidt, "Exploiting structure in policy construction," in *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI 1995)*, 1995, pp. 1104–1111.

[11] G. Theocharous, K. Murphy, and L. Kaelbling, "Representing hierarchical POMDPs as DBNs for multi-scale robot localization," in *Intl. Conf. on Robotics and Automation (ICRA 2004)*, 2004.

[12] K. Murphy, "Dynamic bayesian networks: Representation, inference and learning," PhD Thesis, UC Berkeley, Computer Science Division, 2002.

[13] M. Toussaint, S. Harmeling, and A. Storkey, "Probabilistic inference for solving (PO)MDPs," Research Report EDI-INF-RR-0934, University of Edinburgh, School of Informatics, 2006.

[14] Kschischang, Frey, and Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, 2001.

[15] T. Minka, "A family of algorithms for approximate bayesian inference," PhD thesis, MIT, 2001.

[16] J. Peters, M. Mistry, F. E. Udwadia, R. Cory, J. Nakanishi, and S. Schaal, "A unifying framework for the control of robotics systems," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS 2005)*, 2005, pp. 1824–1831.

[17] W. Li, E. Todorov, and X. Pan, "Hierarchical optimal control of redundant biomechanical systems," in *26th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, 2004.