Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference

Marc Toussaint, Nils Plath, Tobias Lang, Nikolay Jetchev

***Abstract*— A new approach to planning and goal-directed behavior has recently been proposed using probabilistic inference in a graphical model that represents states, actions, constraints and goals of the future to infer appropriate actions and controls. The approach has led to new algorithms on the control and trajectory optimization level as well as for high-level rule-based planning in relational domains. In this paper we integrate these methods to a coherent control, trajectory optimization, and action planning architecture, using the principle of planning by inference across all levels of abstractions. Our scenario is a real blocks world: using a 14DoF Schunk arm and hand with tactile sensors and a stereo camera, the goal is to manipulate a set of objects on the table in a goal-oriented way. For high-level reasoning, we learn relational rule-based models from experience in simulation.**

I. Introduction

Autonomous robots deal with information on different levels of abstraction: they process low-level sensory input to gain the perceptual information they are interested in, reason about their high-level goals and actions, and translate abstract actions into low-level motor control. A central problem of modern robotics is how to integrate these different levels of abstraction for decision-making, planning and control, which requires a coherent principle of information processing.

A general framework for information processing is provided by inference in graphical models which provides a principled way to define the couplings of variables with the corresponding uncertainties. Over the recent years, a new approach to reasoning and goal-directed planning has emerged which is based on probabilistic inference in such models. Using graphical models to specify the dependencies of variables across multiple time-steps, one can reason about the effects of actions in the now and the future. Inference can be viewed as internal simulation for control, planning and decision making. In previous work, we have applied this approach on different levels of abstraction, in low-level motor control [1], [2], [3] as well as in high-level planning [4], [5], where we performed successful experiments in simulated environments. In this paper, we integrate these methods to a full control architecture across levels of abstraction. We show the feasibility of this approach in a real-world scenario where an autonomous robot manipulates multiple objects in a goal-directed way.

Our target scenario is a real blocks-world (Fig. 1): a 14DoF Schunk arm and hand with tactile sensors and a stereo camera manipulates objects on top of a table. By addressing this scenario we want to bring the blocks-world, perhaps the most popular scenario in classical A.I. since the 1970s, to real life. We decompose the problem of acting in the real blocks-world according to the different levels of abstraction and apply appropriate algorithms based on approximate inference on the level of motor control, trajectory optimization, as well as for high-level planning.

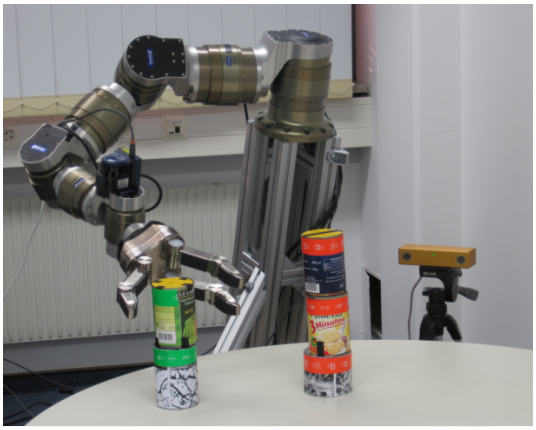

Fig. 1. The robot has successfully put objects with green and red labels into separate piles, using probabilistic inference on different levels of abstraction for planning and control.

After discussing related work in the next section, we describe our target scenario in more detail in Section III. We introduce the different components of our approach in Section IV. In Section V, we present our experiments on a real robot, before we conclude and give an outlook to future research in the last section. A video of the experiments is accompanying this paper and additional material such as source code can be found at the web-page `http://cs.tu-berlin.de/~mtoussai/10-ICRA/`.

II. Related Work

Research in the blocks world scenario has a great tradition in the A.I. planning and reinforcement learning community [6]. Over the last years, the blocks world has been made more interesting by incorporating stochastic actions and investigating generalization over situations, which has led to the emerging field of relational reinforcement learning [7], [8]. Realistic simulations of the blocks world have only very recently been approached. [9] were the first to employ a simulator of the blocks world using a physics engine, for which they developed a rule-based world model which can be learned from experience. We introduced a goal-directed

This work was supported by Honda RI Europe and DFG Emmy Noether grant TO 409/1-3.

MT, NP, TL, NJ are with TU Berlin, Machine Learning and Robotics Group, Franklinstraße 28/29, 10587 Berlin, Germany. {mtoussai,nilsp,lang,jetchev}@cs.tu-berlin.de

planning approach based on approximate inference using learned rules [4] in such a realistically simulated blocks world. This work aims to demonstrate such A.I. methods in the real world.

A core problem within the blocks world is grasping objects. Most existing literature on grasp optimization focuses on the grasp itself, isolated from the reaching movement. For instance, [10] reviews the various literature on defining grasp quality measures, [11] learn which grasp positions are feasible for several objects, [12] efficiently compute good grasps depending on how the objects shall be manipulated, and [13] simplify the grasp computation based on abstracting objects into shape primitives. The coupling to the problem of reaching motion optimization is rarely addressed. A recent approach [14] makes a step towards solving the coupled problem by including a "environment clearance score" in the grasp evaluation measure. In that way, grasps are preferred which are not prohibited by immediate obstacles directly opposing the grasp. In [15] a method for simultaneous grasp and reach trajectory optimization was presented based on a sequence of attractor representations. The method we use is similar but based on a new trajectory optimization method involving probabilistic inference.

Concerning the trajectory optimization, recently there has been growing interest in the possibility to frame the general stochastic optimal control problem as an inference problem [16], [17], [18], [3]. In practice, the resulting algorithms are closely related to differential dynamic programming (DDP) [19], [20], [21], [22] but differ in computational aspects. For instance, in [3] a message passing scheme is used instead of iterated Ricatti sweeps to find a posterior distribution over the trajectories. To our knowledge, this work is the first to demonstrate such inference based trajectory optimization methods on real and high-dimensional hardware. The way we formulate the control and trajectory optimization in terms of multiple concurrently active task variables is very similar to the Whole Body Control concept of [23]. Further related work is discussed in the context of the methods' descriptions in Section IV.

III. Target Scenario

Our overall goal is autonomous goal-directed manipulation in environments with multiple objects. In [4] we presented methods for planning in stochastic relational worlds and demonstrated these methods in physically simulated blocks world problems like clearing the desktop or building towers from objects of different sizes and shapes. In this paper we want to address similar scenarios, but on a real robotic platform. To solve the scenario we require a series of methods for learning, perception, planning and control: Eventually, the robot will need to

1) learn a high-level stochastic model of the effects of actions like grabbing and placing an object,
2) use vision to identify and localize objects,
3) use a stochastic relational planner to compute a sequence of actions,
4) use trajectory optimization to compute dynamically smooth reaching and pre-grasp motions,
5) use a controller to follow the computed trajectories,
6) and use a tactile feedback controller to execute the grasp.

A. Hardware

Our robotic platform is shown in Figure 1 and includes the following hardware components:

- Schunk Light Weight Arm (LWA) with 7DoF
- Schunk Dextrous Hand (SDH) with 7DoF
- 6×14 tactile arrays on each of the 6 finger segments
- Bumblebee stereo camera

The arm and hand use different control protocols: We control the LWA arm by sending positioning commands at 100Hz. These positions determine the reference point of the on-board PID controller in each of the 7 motor modules of the LWA. The exact behavior and parameters of these on-board PID controllers are not known to us – but they behave approximately like a position smoothing with half-decay time about 20 msecs. We control the SDH hand by sending velocity commands and querying actual velocities and positions at about 10Hz (the CAN interface currently does not allow for a higher control rate). The SDH on-board velocity controller tries to reach these velocities with constant acceleration. Concerning the tactile sensor, we neglect the spatial resolution of the signal and compute 6 scalar values y_i for each finger segment. For the ith finger segment we use the equation $y_i = (\text{integral over } i\text{th array})^{0.7}$; the effect of taking the power 0.7 is higher sensitivity to small pressure contacts. The Bumblebee stereo camera provides images at resolution 1024×768 pixels at approximately 2–4 frames per second. We downscale these images by a factor 2 before processing.

IV. Methods

A. Control

Our control framework follows in detail the approach presented in [24]. We control the robot on a dynamic level. That is, let $q_t \in \mathbb{R}^{14}$ be the vector of all joint angles in the LWA arm and SDH hand at time t. The control operates on the phase state

$$x_t = (q_t, \dot{q}_t) \in \mathbb{R}^{28} \tag{1}$$

comprising joint angles and velocities.

The control framework is based on having many *task variables* concurrently active with various precisions: We assume we have m different task variables $y_1, .., y_m$, where the dimension of the ith task variable is d_i. A basic example is the 3D endeffector position $y_i \in \mathbb{R}^3$ in world coordinates. Below we will define in detail all the task variables that we use for control in our scenario. Generally, a task variable is defined by its kinematic mapping $\phi_i : \ q \to y_i$ and its Jacobian $J_i(q) = \frac{\partial \phi_i(q)}{\partial q}$ such that $\dot{y}_i = J_i(q)\dot{q}$. For each task variable we assume we have a desired state $y_{i,t} \in \mathbb{R}^{d_i}$ and state precision $\varrho_{i,t} \in \mathbb{R}$ and a desired velocity $\dot{y}_{i,t} \in \mathbb{R}^{d_i}$

and velocity precision $\nu_{i,t} \in \mathbb{R}$ at time t. We also assume to know the state $x_{t\text{-}1}$ at time t-1. The problem of control is to compute a new state x_t which accounts for all the constraints of the system dynamics, all the desired task variables, and the control costs. The new state x_t implies the positioning commands q_t sent to the LWA arm as well as the velocity commands $\dot{q}_t$ sent to the SDH hand. We compute x_t^* such that it corresponds to optimal "pseudo-dynamic" control accounting for all task variables. Let us explain what we mean by "pseudo-dynamic": Generally, we assume the discrete time system dynamics

$$x_t = Ax_{t\text{-}1} + a + Bu_t + \xi \ , \quad \langle \xi\xi^\top \rangle = Q \ , \tag{2}$$

$$A = \begin{pmatrix} 1 & \tau \\ 0 & 1 \end{pmatrix} , \quad B = \begin{pmatrix} \tau^2 M^{\text{-}1} \\ \tau M^{\text{-}1} \end{pmatrix} , \quad a = \begin{pmatrix} \tau^2 M^{\text{-}1} F \\ \tau M^{\text{-}1} F \end{pmatrix} , \tag{3}$$

where M is the system inertia tensor and F the force vector in state $x_{t\text{-}1}$, τ the time discretization interval, $u_t \in \mathbb{R}^{14}$ the control signal, and ξ Gaussian noise with covariance matrix Q. Since our hardware does not allow for torque control anyway, we make the simplifying assumption of "pseudo-dynamics": namely $M = 1$ and $F = 0$. In other words, we assume that the control signal u_t corresponds directly to joint angle accelerations. Under these assumptions on the dynamics we compute a new state x_t^* using the following equations,

$$r = \sum_{i=1}^{m} \begin{pmatrix} \varrho_{i,t} J_i^\top (y_{i,t} - \phi_i(q_{t\text{-}1}) + J_i q_{t\text{-}1}) \\ \nu_{i,t} J_i^\top \dot{y}_{i,t} \end{pmatrix} , \tag{4}$$

$$R = \sum_{i=1}^{m} \begin{pmatrix} \varrho_{i,t} J_i^\top J_i & 0 \\ 0 & \nu_{i,t} J_i^\top J_i \end{pmatrix} , \tag{5}$$

$$S = Q + BH^{\text{-}1}B^\top \ , \tag{6}$$

$$s = a + Ax_{t\text{-}1} \ , \tag{7}$$

$$x_t^* = (S^{\text{-}1} + V^{\text{-}1} + R)^{\text{-}1}(S^{\text{-}1}s + V^{\text{-}1}v + r) \ . \tag{8}$$

The matrix H defines a quadratic cost $u_t^\top H u_t$ (or, in other terms, a Gaussian prior) on the control signal (namely, accelerations). The matrix V and vector v are explained in the next section and may comprise terms that are related to following a pre-computed optimal trajectory. We refer to [24], [25] for a derivation of these equations. Our control method is strongly related to many well-known classical control methods: If we have only one task variable, $m = 1$, and take the limit of infinite precisions, $\varrho \to \infty$, $\nu \to \infty$, one can show that our control law is equivalent to optimal dynamic control (and thereby operational space control for a certain choice of H) [25], [26]. If we have multiple task variables, $m > 1$, and take a hierarchical limit of infinite precisions, our control law corresponds to prioritized inverse kinematic control [27]. For $m = 1$, having non-infinite precisions is equivalent to introducing a regularization for singularity robust inverse kinematics [28]. When all precisions are non-infinite, our control equation is singularity free even for many concurrent task variables, $m \gg 1$. Equation (8) might seem computationally expensive due to the matrix inversions in 28 dimensions. However, all matrices are symmetric and (when using appropriate LAPACK routines) these computational costs are fully negligible compared to the cost of collision detection.

1) Task variables used for control: The control, the grasping, and the trajectory optimization algorithm are all based on defining a set of relevant task variables and conditioning them appropriately for the task. For our scenario we define the following task variables:

- $y_{\text{EFF}} \in \mathbb{R}^3$ is the endeffector position (the center of the hand) in world coordinates.
- $y_{\text{COL}} \in \mathbb{R}$ is the collision cost: a scalar task variable which measures collision danger. More precisely, if d_j is the shortest distance between a pair j of collidable geometric shapes, then $y_{\text{COL}} = \sum_j \theta(d_j - \epsilon)^2$, with the heavy-side function θ and margin $\epsilon = 0.03$ meter. We use SWIFT++ to compute the mapping $\phi_{\text{COL}} : \ q \to y_{\text{COL}}$.
- $y_{\text{LIM}} \in \mathbb{R}$ is the limit cost: a scalar task variable which measures the danger of violating joint limits. Similar to the collision costs we define $y_{\text{LIM}} = \sum_j \theta(d_j - \epsilon)^2$, summing over all joints j, where d_j is the distance to the joint limit and $\epsilon = .1$ radians is the margin.
- $y_{\text{VEL}} \in \mathbb{R}^{14}$ are the joint angles: that is, a task vector that is directly equal to the joint angles themselves, $\phi_{\text{VEL}} = \mathbf{Id}$. We will use this to penalize high velocities.
- $y_{\text{TAC}} \in \mathbb{R}^6$ are the 6 scalars of the tactile sensors (see section III-A). We do not have a kinematic function $\phi_{\text{TAC}} : q \to y_{\text{TAC}}$ explicitly, but can query the state y_{TAC} from the hardware. We approximate the Jacobian J_{TAC} by setting the rows equal to the normal vector of each tactile sensor array. See section IV-C for more details.
- $y_{\text{UP1}} \in \mathbb{R}$ and $y_{\text{UP2}} \in \mathbb{R}$ are two scalars which measure the declination of the hand (or object in hand) with respect to the horizontal. For instance, conditioning both to zero will align the object vertically.
- Furthermore, control variables represent features of the finger configuration, namely two scalars which measure whether the normals of opposing fingers are aligned.

The general control equation (8) together with these definitions of multiple task variables provides great flexibility. In practice, setting precisions ϱ_i and ν_i to zero means to deactivate a task variable since it drops out of Eq. (8). Depending on the overall task we can turn on and off task variables as desired and associate variable precisions with them. In a typical control mode, we always condition y_{COL}, y_{LIM} and $\dot{y}_{\text{VEL}}$ to zero and impose high precisions ϱ_{COL}, ϱ_{LIM} and low but non-zero velocity precision ν_{VEL}.

B. Trajectory Planning

In [3] we presented an algorithm called *Approximate Inference COontrol* (AICO) for solving a stochastic optimal control problem based on probabilistic inference. The algorithm is closely related to differential dynamic programming (DDP) [19], [20], [21], [22], but uses a message passing scheme to compute a posterior over the whole trajectory conditioned on all desired task variables. The method is introduced in exactly the same framework we described in the previous section – in fact, Eq. (8) is the solution to

a stochastic optimal control problem with a 1-step time horizon $T = 1$; the quantities (s, S) correspond to the forward message, (r, R) to the task message, and (v, V) to the backward message [3].

Stochastic optimal control means to find a control law that minimizes the expectation of the cost

$$C(x_{0:T}, u_{0:T}) = \sum_{t=0}^{T} c_t(x_t, u_t) \tag{9}$$

over the time interval $t = 0, .., T$ under the stochastic process (2). In general, the cost terms $c_t(x_t, u_t)$ are arbitrary cost functions in each time step. In our case we assume that these costs comprise the quadratic cost $u_t^\top H u_t$ of control and cost terms for each conditioned task variable. With the definitions made in the previous section, a local quadratic approximation of the task costs induced by all task variables is given by

$$c_t(x_t, u_t) = x_t^\top R_t x_t - 2 r_t^\top x_t + u_t^\top H u_t \ , \tag{10}$$

where r_r and R_t are defined in (8) and comprise all costs implied by the conditioned task variables.

1) Following an Optimized Trajectory: All the stochastic optimal control methods (AICO, DDP) compute a quadratic potential function $J_t(x) = x^\top V_t x - 2 v_t^\top x$ in each time slice t. This potential (classically the cost-to-go or value function; in the probabilistic framework the log backward message) implies the optimal (feedback) control law $x_{t\text{-}1} \mapsto u_t$ in time step t. In our case, we use AICO to compute the quantities (v_t, V_t) for a given time interval $t = 0, .., T$. Once AICO has converged we use these quantities in Eq. (8) to compute control signals for the real robot. Executing the control law in Eq. (8) for T time steps with the potentials (v_t, V_T) will make the robot "trace" approximately the trajectory that was implicitly optimized by AICO (the MAP trajectory computed by AICO). However, note that the potentials (v_t, V_t) really only define a (usually low-gain) feedback control law, that is, we do *not* replay a deterministic optimal trajectory with high gains. Also, Eq. (8) implies that even when "tracing" a pre-computed trajectory the controller will additionally account for all currently active task variables, in particular y_{COL} and y_{LIM} to avoid collisions and joint limits.

2) Optimizing the Reach and Pre-Grasp Trajectory: To optimize a trajectory we need to specify the desired task variables and precisions for the respective problem. In the case of the reaching and pre-grasp motion we consider $T = 400$ time steps with $\tau = 0.01$sec and condition the task variables as follows: The collision and limit variables y_{COL}, y_{LIM} are conditioned to zero throughout the trajectory; the endeffector variable y_{EFF} is conditioned to be at the object position at the end of the trajectory; the joint angles y_{VEL} are conditioned to zero-velocity at the end of the trajectory; opposed fingers are conditioned to be aligned at the end of the trajectory and with sufficient distance to the object. The optimized trajectory is dynamically smooth and generates a reaching motion which at the same time ends in a good pre-grasp posture. Figure 3 (left) shows some illustrations of start and end postures of such reach and pre-grasp motions.

3) Optimizing the Place Trajectory: Once the object is grasped we need to generate a motion to place it onto another object. Again, we assume $T = 400$ and $\tau = 0.01$sec and constantly conditioned y_{COL}, y_{LIM}. For the placing movement we do not condition any finger features but keep the hand posture constant. We condition y_{EFF} and y_{VEL} as above and in addition a task variable which measures whether the object in hand is upright. Figure 3 (right) shows some illustrations of start and end postures of such "place object" motions.

C. Grasping and Releasing the Object

The pre-grasp posture in which the reach motion ends is already very close to the object, wrapping the fingers around the object with about ~ 3cm distance. The grasping itself can then easily be executed using a tactile feedback loop. In our control framework we can realize this by conditioning the tactile task variable y_{TAC} to a desired non-zero pressure value and then iterate the control (8) (without (v_t, V_t)) until this task variable reaches its desired state. We condition the pressure on the three finger tips to be a non-zero constant which results in the closing of the hand until the object is grasped. Figure 5 displays the change in tactile signals y_{TAC} during such a closing of the hand.

Similarly, when releasing the object after the optimized placing motion we condition the y_{TAC} variable to zero which results in the opening of the hand until no pressure is measured. In addition, we condition y_{COL} to zero which leads to further opening of the hand until fingers have distance to the object below the collision margin of $\epsilon = 3$cm.

D. Vision

The identification and localization of objects is based on SURF interest points and descriptors [29] using the OpenSURF library [30]. Prior to our experiments, for each object O we took several sample images from different view angles and pre-computed a set $\mathcal{F}_O$ of SURF features. Given a new stereo image from the Bumblebee camera we compute sets $\mathcal{F}_L$ and $\mathcal{F}_R$ of keypoints and SURF descriptors for the left and the right image, respectively. The identification and localization of objects is now based on finding subsets of descriptors in $\mathcal{F}_O$, $\mathcal{F}_L$, and $\mathcal{F}_R$ which match (using approximate nearest neighbor (ANN library)[1]) and which are consistent with respect to a homography from the left to the right image (using the RANSAC homography implementation of OpenCV[2]). The homography constraint implies that only keypoint subsets are detected which have similar disparity on the stereo image. If, for a given object identifier O, we find keypoints in $\mathcal{F}_O$, $\mathcal{F}_L$, and $\mathcal{F}_R$ which are consistent, we use the average disparity of the matched keypoints in $\mathcal{F}_L$ and $\mathcal{F}_R$ to estimate the distance of the object and the center of mass of these keypoints to estimate the object coordinates within the image. Using the camera calibration we can compute the 3D position of the object in world coordinates. For each stereo image we loop over all possible object identifiers O to see if we can find and localize the object in the image.

[1] `http://www.cs.umd.edu/mount/ANN/`
[2] `http://opencv.willowgarage.com/wiki/`

Fig. 2. Identifying and locating objects using SURF interest points and a stereo camera. Each row depicts the recognition of an object. The left and right column correspond to the left and right camera, respectively.

Since computer vision is not the primary scope of this work, we tried to simplify the general problem by keeping lighting conditions constant (as far as possible) and use objects with significant textures.

E. High-level Relational Planning

An autonomous robot needs to reason about its potential actions to achieve its overall goals. Similarly as in our low-level motor control, we pursue a model-based approach based on inference in graphical models for high-level planning. Symbolic action models are a common approach in A.I. to describe how the world changes with the execution of actions [31]. In a real environment, such models need to account for uncertain action outcomes, e.g., a tower of blocks may topple over when trying to place an object on its top. Furthermore, action models have to generalize over situations and objects to enable planning in unseen situations with new objects. Recent work in A.I. has led to the development of abstract stochastic action models using relational representations which account for these requirements. In our work, we use a rule-based model, namely noisy indeterministic rules [9] which are particularly appealing, as they can be learned effectively from experience. Although abstract action models capture the world dynamics compactly, using them for planning is challenging: the state space in relational domains is exponential in the number of objects, the search space of action sequences is huge, and reasoning about actions is aggravated by the their stochasticity. We apply the PRADA planning algorithm presented in [4], [5], which tackles these difficulties by converting abstract stochastic relational rules into partially grounded dynamic Bayesian networks and applying approximate inference to find suitable action sequences.

We employ the same symbolic representation as in [4] to describe our domain, consisting of predicates such as $inhand(\cdot)$, $upright(\cdot)$, $on(\cdot,\cdot)$ and functions such as $size(\cdot)$ to describe world states and predicates such as $grab(\cdot)$ and $puton(\cdot)$ to describe actions. A major challenge in developing intelligent robots is how to couple high-level reasoning with sensors and low-level motor control. We approach the first problem by a set of simple heuristics to translate object information from vision and tactile sensors into our symbolic representation. For instance, we derive $on(a,b)$ if the x/y-coordinates of objects a and b are sufficiently similar while b's z-coordinate is slightly smaller, and $inhand(b)$ if b is closest to the robot's fingers and we get significant tactile feedback. To translate the high-level action symbols to concrete robot action, we set them to trigger execution of the corresponding low-level motor control routines described above.

V. Experiments

We evaluate our approach in the real blocks world scenario described in Section III. Our experiments are designed to focus on qualitative aspects: we investigate whether the different methods can indeed be successfully applied in a real world domain and our approach is suitable to fully control an autonomous robot to achieve its goals. For quantitative studies with respect to the individual methods, we refer the reader instead to the respective papers, i.e., for a comparison and discussion of different low-level robot control approaches to [3], [1] and of different high-level relational planning approaches to [4], [5].

In our experimental setup, the Schunk robot is placed in front of a table with cylindric objects of two different sizes and colors. In the scenario of the video accompanying this article, the goal is to "clear up" the desktop: to stack objects of the same color onto each other. There are two big and one small red object and two big green objects stacked in three piles, where two piles contain two objects of different colors. To achieve the goal of a cleared desktop, the robot needs to grab and place 3 objects in total.

The first problem is to localize the objects using the stereo camera which is placed next to the robot arm. The objects have individual patterns which are used to identify them, see Fig. 2 for an example. Once the objects are recognized, their coordinates are calculated and a symbolic world state representation is derived. Then, the robot uses the PRADA algorithm to derive a high-level plan of actions to achieve a cleared desktop. PRADA is based on stochastic relational rules. A set of 11 abstract rules has been learned beforehand using the algorithm of [9] with the same parameter settings from a set of 500 experiences of state transitions. We have generated these experiences in a 3D rigid-body dynamics simulator (ODE) of the scenario including the robot, the objects and the table by performing random actions with a slight bias to build high towers.

After a suitable action plan has been found, the single abstract actions trigger the respective low-level motor control routines, namely AICO, to generate grasp and placing trajectories. Fig. 3 (left) illustrates the start posture (central hand position) and the end posture of an optimized reach and pre-grasp trajectory. The pictures are taken from the simulator that is internally used by AICO. The red distance markers

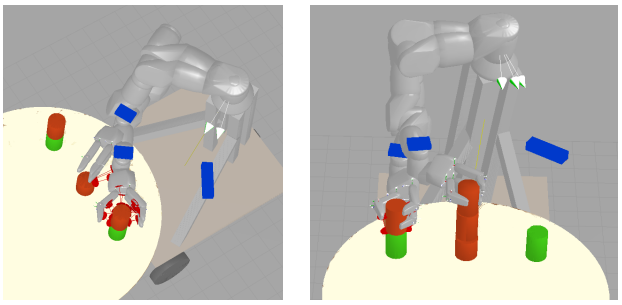

Fig. 3. Visualization of start and end postures of calculated trajectories. The blue bars are the stereo camera (to the right of the robot) and a laser scanner (not used in these experiments, mounted on the arm close to the hand). *(left)* A grab action moving the hand from the center to the right pile. *(right)* A place action moving the hand with the object from the left to the center pile. The red distance markers indicate critical proximities between collidable objects.

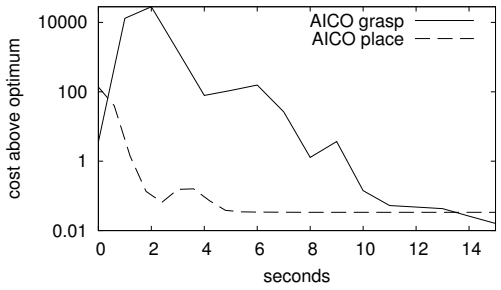

Fig. 4. The costs of intermediate solutions during trajectory optimization using AICO for a typical grasp and a place action. The reaching motion of the grasp action is more difficult to compute due to the many critical distance proximities in the pre-grasp posture.

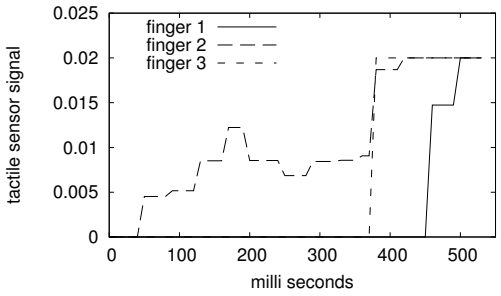

Fig. 5. Tactile signals of the finger sensors used to guide an exemplary grasp action. Finger 2 touches the target object first. After half a second, the object is fixed in hand, indicated by the maximum signals of all fingers.

illustrate the output of the collision (distance proximity) detection engine. The end posture is a good pre-grasp with the fingers wrapping around the object with about $\sim$ 3cm distance. The large number of critical distance proximities in such a pre-grasp movement makes the optimization computationally expensive (the collision engine queries generate the major computational cost). This can be seen in Fig. 4; it takes about 10 seconds to find a dynamically smooth trajectory which at the same time does not violate the collision margin of 3 centimeters but still ends wrapping the fingers around the object.

When this pre-grasp posture has been reached the grasp is executed based on the tactile feedback loop. Fig. 5 displays the three relevant components of the y_{TAC} task variable, that is, the integrated pressure feedback from the sensor arrays at the tip of the three fingers. As described before, we generate the grasp by conditioning the y_{TAC} to be non-zero, namely $(0.2, 0.2, 0.2)$, and iterating the control equation (8). The pressure curves show how this task variable behaves during the closing of the hand.

Finally, Fig. 3 (right) displays the start posture (hand at left pile) and end posture (hand at center pile) of an object placing movement. As can be seen in Fig. 4, this movement is much easier to optimize (in about 2 seconds) since collisions and finger movement do not play a critical role.

The video shows a complete demonstration to solve the clearing up task with five objects. PRADA found the correct sequence of actions necessary to stack the two objects with green labels on one pile. To reach this state, two objects with red labels first have to be removed from them by placing them on the center pile for the red-labeled objects. Fig. 1 shows the end posture after the whole trial.

For simplicity, in this demonstration, all computations and control are done on a single 2-processor laptop. This has a number of limitations. For instance, we found that the parallel communication with all four hardware components slows down the communication with the LWA to a level that in 10% of the control steps the 100Hz control frequency is not met. Further, for safety we always optimized trajectories until full convergence. Given the limited computational power we performed this optimization before executing the whole movement. A better computational infrastructure and heuristics to start the movement immediately when the quality of the trajectory is sufficient (e.g., the cost below a threshold) might allow to parallelize the trajectory optimization and movement execution in the future. The same is true for the object localization, given the significant computational cost of the computation and processing of the image features. The experiment also revealed limitations on the precision of our kinematic model of the arm and the object localization, causing some placed objects not to align as perfectly as they do in the simulation.

VI. Conclusions

The general aim of this work is to bridge the gap between methods and theory developed in the context of Machine Learning and A.I. and their application in real-world autonomous robotics. Given the recent work on probabilistic inference methods as a tool not only for sensor information processing but also for reasoning about actions and control under uncertainty, we believe that these methods have the potential to provide a more coherent and integrated view on motor control, motion planning and decision making on all levels of abstraction. In this work we demonstrated the feasibility of approximate inference methods for control and trajectory optimization on the motor level as well as high-level planning in an integrated real-world robotics problem. We chose the block-world scenario in analogy to typical

A.I. benchmarks to demonstrate that the methods, such as PRADA, can be transferred to real world.

We mentioned limitations of our current architecture in the previous section. Beyond these technical issues, in future work we intend to apply our approach to scenarios that are more challenging for both the high-level planning as well as the low-level control: scenarios which require longer action sequences, include different types of objects and more cluttered scenes with big obstacles the robot has to avoid.

In principle, inference in graphical models provides a natural way to account for sensor uncertainty. In this paper, this has not been investigated yet as so far the perceptual modalities (vision, tactile sensing) are not fully integrated in the probabilistic framework. Future work will examine how our approach can handle perceptional uncertainty which could lead to movement failures such as accidentally pushing objects off piles. In particular, we will work on more tightly coupling uncertainties across modalities and levels, e.g. between the object localization and grasp planning.

The source code for low level motor control as well as high level planning used in this paper are publicly available at `http://cs.tu-berlin.de/~mtoussai/10-ICRA/`.

REFERENCES

[1] M. Toussaint and C. Goerick, "Probabilistic inference for structured planning in robotics," in *Int Conf on Intelligent Robots and Systems (IROS 2007)*, 2007, pp. 3068–3073.

[2] N. Vlassis, M. Toussaint, G. Kontes, and S. Piperidis, "Learning model-free robot control by a monte carlo em algorithm," *Autonomous Robots, special issue on Robot Learning*, 2009, submitted.

[3] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proc. of the 26rd Int. Conf. on Machine Learning (ICML 2009)*, 2009.

[4] T. Lang and M. Toussaint, "Approximate inference for planning in stochastic relational worlds," in *Proc. of the 26rd Int. Conf. on Machine Learning (ICML 2009)*, 2009.

[5] ——, "Relevance grounding for planning in relational domains," in *Proc. of the European Conf. on Machine Learning (ECML 2009)*, 2009.

[6] J. Slaney and S. Thiébaux, "Blocks world revisited," *Artificial Intelligence*, vol. 125, no. 1-2, pp. 119–153, 2001.

[7] S. Dzeroski, L. de Raedt, and K. Driessens, "Relational reinforcement learning," *Machine Learning*, vol. 43, pp. 7–52, 2001.

[8] M. van Otterlo, *The Logic of Adaptive Behavior*. IOS Press, Amsterdam, 2009.

[9] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Artificial Intelligence Research*, vol. 29, 2007.

[10] R. Suárez, M. Roa, and J. Cornellà, "Grasp quality measures," Universitat Politècnica de Catalunya, Institut d'Organització i Control de Sistemes Industrials, Tech. Rep. IOC-DT-P 2006-10, 2006. [Online]. Available: https://upcommons.upc.edu/e-prints/bitstream/2117/316/1/Roa.pdf

[11] D. Schwammkrug, J. Walter, and H. Ritter, "Rapid learning of robot grasping positions," in *Proceedings of the International Symposion on Intelligent Robotics Systems (SIRS)*, 1999. [Online]. Available: http://www.techfak.uni-bielefeld.de/ags/ni/publications/media/SchwammkrugWalterRitter1999-RLO.ps.gz

[12] R. Haschke, J. Steil, I. Steuwer, and H. Ritter, "Task-oriented quality measures for dextrous grasping," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 6 2005, pp. 689 – 694.

[13] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *Proceedings of the IEEE International Conference of Robotics and Automation (ICRA)*, 2003, pp. 1824 – 1829.

[14] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots*, 12 2007.

[15] M. Gienger, M. Toussaint, N. Jetchev, A. Bendig, and C. Goerick, "Optimization of fluent approach and grasp motions," in *8th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids 2008)*, 2008.

[16] E. Todorov, "General duality between optimal control and estimation," in *proceedings of the 47th IEEE Conf. on Decision and Control*, 2008.

[17] B. Kappen, V. Gomez, and M. Opper, "Optimal control as a graphical model inference problem," 2009. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:0901.0633

[18] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, and Y. Bengio, Eds. Cambridge, MA: MIT Press, 2009.

[19] P. Dyer and S. R. McReynolds, *The Computation and Theory of Optimal Control*. Elsevier, 1970.

[20] D. M. Murray and S. J. Yakowitz, "Differential dynamic programming and newton's method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, pp. 395–414, 1984.

[21] C. G. Atkeson, "Using local trajectory optimizers to speed up global optimization in dynamic programming," in *NIPS*, 1993, p. 663670.

[22] Y. Tassa, T. Erez, and W. Smart, "Receding horizon differential dynamic programming," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 1465–1472.

[23] M. Gienger, H. Janssen, and C. Goerick, "Task-oriented whole body motion for humanoid robots," in *Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots*, 12 2005.

[24] M. Toussaint, "A Bayesian view on motor control and planning," in *From motor to interaction learning in robots*, O. Sigaud and J. Peters, Eds. Springer, 2010, in print.

[25] ——, "Lecture notes: Stochastic optimal control," http://ml.cs.tu-berlin.de/~mtoussai/notes/, 2009.

[26] 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.

[27] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *Int. Journal of Robotics Research*, vol. 6, 1987.

[28] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of Dynamic Systems, Measurement and Control*, vol. 108, 1986.

[29] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008.

[30] C. Evans, "Notes on the opensurf library," University of Bristol, Tech. Rep. CSTR-09-001, January 2009. [Online]. Available: http://www.cs.bris.ac.uk/Publications/Papers/2000970.pdf

[31] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 2003.