Inference based Trajectory Optimization for Imitation based Grasping

Ashish Jain Ruben Martinez-Cantin Alexandre Bernardino José Santos-Victor

Abstract— **Programming by demonstration is a powerful method to solve complex, high-dimensional tasks in humanoid robots, such as reaching and grasping objects. However, a demonstration (or a set of demonstrations) are performed in particular scenarios that may be different from the ones faced by the robot during operation. Both the posture and shape of the targets, robot kinematics and dynamics constraints, presence of obstacles and other workspace contingencies often prevent a direct application of the demonstrated programs in a run-time scenario. An intelligent agent should be able to learn the task based on few demonstrations, and then, be able to generalize and solve it according to features found in novel situations. In this paper we present a framework and methodology for addressing this problem and show some results from implementations on the iCub robot.**

I. Introduction

Achieving dexterous robot grasping is a milestone in humanoid robots and mobile manipulators. It implies generating a suitable trajectory in a high dimensional space, i.e.: the robot configuration space. This trajectory depends on the task (e.g.: type of grasp); on the environment (e.g.: obstacles to avoid) and on the hand and object properties (e.g.: kinodynamic properties and contact points).

Programming by demonstration techniques allow tackling the first problem: high dimensionality. By recording human demonstrations of the execution of certain tasks one can extract motor programs (or motor primitives) that span a reduced part of the configuration space and can be used as priors for the generation of robot trajectories. Several approaches to obtain motor primitives from one or multiple demonstrations have been proposed in the literature such as gaussian mixture models [4], attractor dynamics [8] and fuzzy time clustering [20]. Such approaches represent the demonstrated trajectories using a low-dimensional set of parameters that somehow encode the relevant features of the human skill. By design, the above mentioned motor primitives can be parameterized by some aspects of the task, e.g., the position of the target. This provides the ability to reach for targets once their position is known.

Human demonstrations, however, cannot be directly imitated by the robot due to unavoidable kinodynamic differences between the human and robot motor systems. Straightforward approaches represent human trajectories in cartesian space and command the robot with inverse kinematics solutions but this approach presents high sensitivity to calibration error and low robustness to disturbances when close to singularities. Other approaches try to blend cartesian and operational spaces in order to jointly minimize trajectory errors in cartesian and operational space [5]. A problem with such model based approaches is that they require a good knowledge of the robot kinematics and dynamics which, in some cases, is hard and expensive to acquire. Techniques based on reinforcement learning can be employed in these cases, as they explore the motor space in order to optimize the motor control laws for the particular robotic system and tasks at hand [10].

Uncertainty exists not only in the knowledge of the robot motor system but also (and most often) in the perception of the external objects to interact with. When the robot needs to grasp previously unknown objects, uncertainty comes from two main sources:

- Signal noise: When the robot is confronted with an unknown environment, the main source of information is sensor data, like cameras, range finders or tactile sensors. Sensor data is typically perturbed by random noise, which has to be considered when planning the grasp.
- Abstraction: A robot needs to learn robust strategies to solve the grasping problem in different scenarios in order to be able to generalize the trajectory to novel objects or configurations. Nevertheless, this learning process introduce uncertainty when it is used to realize a prediction of a previously unseen setup. For example, we can learn to grasp a *cup*, but the concept cup is uncertain because not all the cups has the same colors, size or weight.

A standard formulation to cope with uncertainty in this kind of problems is that of Markov Decision Processes (MDP). This formulation, allows the representation of uncertainty and also to address reinforcement learning (RL) problems by assuming a *reward* or *cost* signal which is used to evaluate the optimality of the task. In this paper we adopt an MDP formulation and present and architecture for the control of a robotic system incorporating learning from demonstrations while satisfying task constraints, and use state-of-the-art inference methods [25] to implement reaching and grasping movements on the iCub robot able to deviate from obstacles in the environment.

The paper is organized as follows. In Section II we introduce our approach in the context of inference methods over MDP's. In Section III we explain how learning from

This work was partially supported by the Portuguese Government - Fundação para a Ciência e Tecnologia (ISR/IST pluriannual funding) through the POS Conhecimento Program that includes FEDER funds, and through the EC Funded projects HANDLE and FIRST-MM.

A. Jain, A. Bernardino, R. Martinez-Cantin and J. Santos-Victor are with the Institute for Systems and Robotics, IST, Lisboa, Portugal {ajain,alex,rmcantin,jasv}@isr.ist.utl.pt

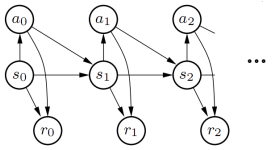

Fig. 1. A generic Markov Decision Process (MDP).

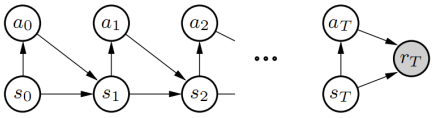

Fig. 2. A Markov Decision Process with reward on the final state.

demonstrations can be framed as learning the reward (or cost) function for an MDP. Then, in Section IV we propose an architecture to address the planning of reaching and grasping actions using the human demonstrations but also taking into account real-time contingencies. Experimental results in simulation and on the iCub robot are shown in Section V and, finally, conclusions and future work are presented in Section VI.

II. Inference based planning and control

Novel solutions to MDP and optimal control like problems are based on statistical inference [28], [6], [7], [29]. An MDP is composed by random variables defining the states (s_t), the actions (a_t) and the rewards (r_t) (see Fig. 1) The state evolution is determined by a stochastic transition model, $P(s_{t+1}|s_t, a_t)$. The actions are generated by a stochastic policy (or control law) defined as $P(a_t|s_t;\theta)$, where π is a set of parameters of the control rule that we want to optimize. The reward is given to the system at each time step according to a conditional distribution $P(r_t|s_t, a_t)$. The classical approach to solve the MDP is to search for the parameters θ that maximize the expected (average or discounted) reward. As explained in [28], assuming that the reward is only assigned in the end of the trajectory (see Fig. 2), and that $r_T = 1$ is the optimum, then we can rephrase the optimization problem in terms of maximum likelihood and solve it with probabilistic inference methods: compute the parameters θ that maximize the likelihood of $P(r_T = 1|s_t, a_t)$. This can be solved efficiently by an Expectation Maximization (EM) algorithm where the E-step computes the posterior over $a_{0:T}$ and $s_{0:T}$ conditioned in observing $r_T = 1$. This is similar to "imagining" to receive the reward and infer the trajectory to achieve that goal using the current policy. Then, the M-step computes the new policy parameters to maximize the expected complete log-likelihood of the computed trajectory conditional distribution. In [28] it is also shown that, for the unknown time horizon problem, one can compute the expectation of the accumulated reward for a trajectory by computing the reward of a mixture of trajectories with different time horizons, that is, computing the optimal trajectory is equivalent to finding the most likely time horizon in the mixture.

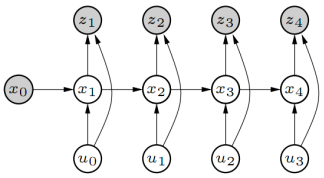

Fig. 3. A stochastic optimization problem can be reformulated as an inference problem by defining target random variables z. The cost values are mapped to the neg-log probability of $z = 1$.

The MDP formulation also resembles the problem of Stochastic Optimal Control (SOC), which replaces the *reward* function with a quadratic cost based on the robot state and action. In fact, SOC can be used for efficient robot planning through trajectory optimization [23], [28], [22]. In [25], the SOC problem is also mapped to an inference problem (see Fig. 3). More precisely, it is defined a virtual binary variable z, which has a probability distribution based on the cost of the trajectory. That is, if we define the trajectory as a set of states $x_{0:T}$ and actions $u_{0:T}$, we define the probability of $z = 1$ as

$$p(z = 1|x_{0:T}, u_{0:T}) = \exp(-C(x_{0:T}, u_{0:T})) \qquad (1)$$

where $C(\cdot,\cdot)$ is the accumulated cost of the trajectory. Basically, the probability of $z = 1$ will be high when the cost is low. Then, fast (approximate) inference methods on graphical models are employed to obtain the control sequence that maximizes the likelihood of $z = 1$, i.e. minimize the cost. The advantage of using inference is that we can use the most suitable techniques for our models and distributions. Once we have mapped the optimization problem as an inference problem, we can combine it with other learning problems that may happen simultaneously.

In [25] the trajectory is generated taking into account several criteria, including a desired target position, the obstacles in the environment and the robot kinodynamic constraints. In this work we also what to include priors from human demonstrations, i.e. the generated trajectory should be "human-like". This will be obtained by several demonstrations of humans executing a similar task. Most works on programming by demonstration typically learn a distribution of behaviors that fits the demonstrated trajectories. That distribution is conditioned on some features from the scene and the robot. For example, if we define a set of features from the object as $\mathcal{O}$ and a set of features from the hand as $\mathcal{H}$, then, we can compute the posterior distribution as

$$p(x_{0:T}, u_{0:T}|\mathcal{O}, \mathcal{H}, \mathcal{D}) \propto p(\mathcal{O}, \mathcal{H}, \mathcal{D}|x_{0:T}, u_{0:T})p(x_{0:T}, u_{0:T}) \qquad (2)$$

where $\mathcal{D}$ are the demonstrated trajectories. The likelihood of the previous equation can be decomposed as follows:

$$p(\mathcal{O}, \mathcal{H}, \mathcal{D}|x_{0:T}, u_{0:T}) = p(\mathcal{D}|x_{0:T}, u_{0:T}, \mathcal{O}, \mathcal{H})\times \qquad (3)$$

$$\times\, p(\mathcal{O}, \mathcal{H}|x_{0:T}, u_{0:T}) \qquad (4)$$

Therefore, we can take the posterior distribution of the demonstrated trajectories and use them as a prior distribution

for the trajectory optimization. Thus, instead of using the ML trajectory, we compute the Maximum a Posteriori (MAP) trajectory.

Demonstration can be useful to learn more properties of the grasping problem apart from the prior distribution of trajectories. For example, while computing the quadratic cost $C(x_{0:T}, u_{0:T})$, we have to consider different aspects of the optimal trajectory. For example:

- The end effector must reach a certain position and orientation to perform the grasp. This position will be computed using a previously learned non-parametric regression function to detect grasping points in novel objects [14].
- The robot must avoid obstacles and other objects while doing the reaching and manipulation.
- The optimal trajectory should be smooth and easy to follow by the controller, i.e., a PID controller should be able to perform the trajectory with small gains and remaining inside the kinodynamic limits [27], [3].
- The trajectory must remain within the robot workspace.

In fact, it is simpler to consider the imitation of the trajectory as a constraint in the trajectory [4], [9]. Therefore, it can be included in the cost function as one item more on the list. We can define the imitation constraint in terms of task variables, robot variables or environment variables, like objects or references, just by increasing the state variable to consider all those elements.

These constraints should be all considered in a single cost function, that is,

$$C(x_{0:T}, u_{0:T}) = \sum_{i=1}^{N} \theta_i c_i(x_{0:T}, u_{0:T}) \tag{5}$$

where $c_i(x_{0:T}, u_{0:T})$ is the cost of every task constraint and θ_i is the corresponding weight. For example, if we want to guarantee that the trajectory is collision free, we just need to set

$$\theta_{coll} \gg \theta_i \quad \forall i \neq coll \tag{6}$$

However, other weights might be trickier to set. Some constraints might have similar importance for the global task, but changing those weights produce completely different trajectories. Therefore, having the optimal relative weights is of paramount importance.

III. Inverse planning

In the past few years, there has been several works addressing the problem of Inverse Reinforcement Learning (IRL) [17], [30], [1] or Inverse Optimal Control (IOC) [23], [30], which basically study the problem of learning a *reward* or *cost* function[1], based on several demonstrations from an optimal policy or control law.

In all cases, the algorithms assume that the reward function can be obtained as the weighted sum of a set of features. Using the linearity property of the expected value, we can define the expected reward as the linear combination of expected values

$$R(x,u) = \mathbb{E}[\sum_{i=0}^{N} \theta_i r_i(x,u)] \tag{7}$$

$$= \sum_{i=0}^{N} \theta_i \mathbb{E}[r_i(x,u)] \tag{8}$$

$$= \sum_{i=0}^{N} \theta_i \phi_i(x,u) \tag{9}$$

Once we extend the reward to all the trajectory and we change the sign, equations (5) and (9) are equivalent. Interestingly, there is evidence that humans also *understand* the actions and the tasks by mentally processing the problem of inverse planning [2].

The main difficulty of IRL is twofold. First, we may not have complete demonstrations of the policy or control law. We just have partial demonstrations or trajectories of a desired behavior from an initial configuration. This effect is critical when the policy is stochastic, meaning that the output (action) from a given state is not deterministic, therefore, different realizations might have different outcomes. In fact, even in the deterministic case, it is impossible to explore all the states and actions. For that reason, Lopes et al. [12] designed an algorithm that actively searches and queries the most informative demonstrations and trajectories.

The method of active IRL from Lopes et al. is based on the Bayesian IRL [18] which tries to find a posterior distribution on the reward parameters conditioned on the demonstrations $p(R|\mathcal{D})$. However, this method is very inefficient since it relies on MCMC sampling of the complete MDP in order to estimate the posterior distribution. Another interpretations of the IRL problem are based on the optimization of a *similarity* function between the demonstration and the reward function.

$$R^* = \arg\min_R J(R|\mathcal{D}) \tag{10}$$

Thus, depending on the definition of the dissimilarity function $J(R|\mathcal{D})$, we have projection algorithms [1], max-margin planning [19], policy matching[2] [15] and maximum entropy [30]. An extended review on this classification can be found in [16]. The advantage of these methods is that they are usually more efficient that the Bayesian IRL. However, the optimization problem might be tricky and it may require some tuning depending on the application.

In Section II we have seen how to map an optimization algorithm to an inference problem. Analogously the similarity optimization problem is naturally addressed using inference methods. That gives us a complete and uniform framework for all the learning steps in our architecture.

IV. Architecture

In figure 4, we show the interdependencies between the different modules composing our architecture. The goal is

[1]One can easily transform a reward in a cost and vice versa by changing the sign of the function.

[2]In fact, this work may have a double interpretation, since the similarity function resembles a likelihood function. Therefore, it can be interpreted as the Maximum Likelihood version of the Bayesian IRL [18]

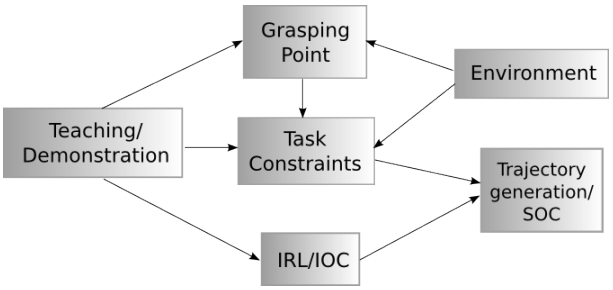

Fig. 4. Diagram of the different modules used to generate an optimal trajectory based on inference. The output of the system is the trajectory generated by the Stochastic Optimal Control module. The cost function is extracted from the task constraints based on the weights learned by Inverse Reinforcement Learning. Such constraints might include: reaching a grasping point, perform obstacle avoidance and imitating a trajectory.

to generate a reaching and grasping trajectory satisfying several constraints. Not only we desire similarity to the human motion but also to take into account the robot kinodynamic aspects and avoid collisions. The weights of the different constraints (deviations from human trajectory, collision proximity and robot limit violations) are learned based on inverse planing (IRL or IOC) performed during a set of previous demonstrations of a *good behavior*. We assume the availability of a diverse enough set of human demonstrations, not only to compute motor primitives synthesizing the average human skill but also to extract other important aspects of the human policy.

The Stochastic Optimal Control (SOC) module is a solver that will compute the trajectory that maximize the joint criteria: similarity to the human motor primitive, proximity of the end effector to a desired grasping point configuration, collision with objects and deviation from robot joint limits. Internally the SOC module contains a simulator of the robot kinodynamics and a collision detection engine. This is used to compute the cost of violating the robot limits and collisions to objects.

The grasping point module also plays an important role in the architecture. It is in charge of computing a desired posture for the hand close to an object selected for grasping. Classical graping methods assume a good 3D reconstruction of the object shape to plan the grasp but 3D object reconstruction is still difficult to achieve with off-the-shelf sensors. Instead we use a learning scheme to compute the likelihood of good grasps based on raw pixel neighborhood operations on the acquired images. This also requires some teaching period where we learn a regression function from visual features to configurations of the hand that guarantees high probability of successful grasping [14]. During execution, novel objects can be grasped based on the detection of their visual appearance features.

V. Results

In this section, we provide some preliminary results on implementing the trajectory optimization using inference on the iCub platform. Our implementation is based on *libSOC*, an open source library based on the work from [25]. The implementation also includes a simplified simulator, both for computing the kinodynamic models of the robot for the inference module and for debugging and testing.

In this work, every model is predefined manually including the grasping point. The task constraints used are: a) reaching the grasping point, b) avoid obstacles and self collisions during the whole trajectory and c) guarantee that the joint limits are satisfied. At present, the control module we implemented for the iCub platform is sensitive to large changes in adjacent steps of the trajectory. Therefore, the trajectory illustrated in Figures 5 and 7 is fairly large containing 600 steps and took about 13.5 seconds to compute on an old PC -Intel Pentium D with 2 GB RAM-. In Figure 5 we see the simulated iCub reaching the red ball (target for the end effector), while avoiding the table (yellow plane). In the starting position, the hand is placed below the table. Due to the amplitude in the motion, the iCub is forced to move the torso to be able to reach the target. This results in a very natural movement on the real robot (see Figure 7). When the collision criterion is not considered, the simulated robot collides with the table (yellow plane) as illustrated in Figure 6.

This design had in mind one goal for the near future. We want the robot to be able to react in real–time. For example, the robot should be able to adapt while reaching for a target when an obstacle is introduced. To achieve this goal we are working on two related aspects. Firstly, a Time-of-Flight (TOF) camera sensor will be used to obtain in real-time potential obstacles introduced in the environment. Secondly, trajectory can be modified/recomputed in near real time. Our approach considers the path deformation strategies extensively used in mobile robotics, such as [11]. Using our architecture, one can compute small trajectories extremely fast using [25] (the trajectory illustrated in Figures 5 and 7 when computed for only 40 steps takes only 0.51 seconds on a old PC.). At the moment, we are implementing a control module for the iCub which can follow a given trajectory accurately even if there are large changes in adjacent steps of the trajectory. Furthermore, we are simultaneously implementing dynamic time warping to reduce/increase the size of the computed trajectory if necessary.

One of the major limitations of these approaches is the high dependency on accurate models of the robot and the environment. In this work, every model is predefined manually. However, in future work, we are going to use sensor data to obtain an accurate model from the environment [21] and the robot [13]. Also, we plan to explore different algorithms for high-dimensional trajectory optimization which learn a parametric model based on random exploration [22].

VI. Conclusions

In this work, we present an architecture for generating reaching and grasping motions. The architecture includes different modules to integrate task constraints from human imitation, sensor driven grasping selection and physical restrictions such as, obstacle avoidance and workspace limits.

Using a graphical model like a Bayesian network to model the coupling of the different variables has several advantages.

Fig. 5. Trajectory generated to reach the goal (red ball) with the end effector of the left hand. The trajectory is optimized so that the board (yellow plane) is avoided. Note that the torso is also moved to help the movement.

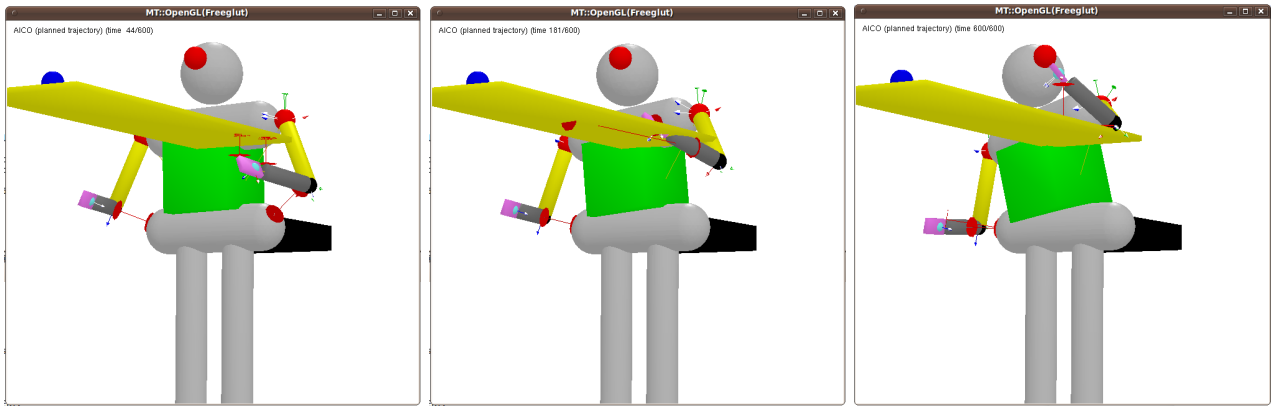

Fig. 6. Trajectory generated to reach the goal (red ball) with the end effector of the left hand. In this case, obstacle avoidance is not considered, resulting in a trajectory with collision.

The most important one is that the classical optimization problem is replaced by a Bayesian inference problem [26]. Therefore we can use and combine the different tools for Bayesian inference that are available for different models and representations. For example, using Monte Carlo methods we can combine continuous and discrete variables in the same framework [6], [7], [29]. Also, fancier models such as truncated Gaussians are suitable for robot planning with obstacles [24].

Finally, the probabilistic model also has a relationship with classical reinforcement learning [28]. Thus, we can apply a second inference layer to deal with the importance or the priority of the different task constraints, using inverse reinforcement learning algorithms [1], [15]. The inverse reinforcement learning setup allows to generalize classical apprenticeship learning and imitation learning by adding an intermediate abstraction level on the Markov decision process.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning,. In *Proc. of International Conference on Machine Learning (ICML)*, 2004.

[2] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, Dec 2009.

[3] Jonas Buchli, Evangelos Theodorou, Freek Stulp, and Stefan Schaal. Variable impedance control - a reinforcement learning approach. In *Robotics: Science and Systems Conference (RSS)*, 2010.

[4] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298, 2007.

[5] Micha Hersch and Aude Billard. Reaching with multi-referencial dynamical systems. *Autonomous Robots*, 25(1–2):71–83, 2008.

[6] Matt Hoffman, Nando de Freitas, Arnaud Doucet, and Ayai Jasra. Bayesian policy learning with trans-dimensional mcmc. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[7] Matt Hoffman, Nando de Freitas, Arnaud Doucet, and Jan Peters. An expectation maximization algorithm for continuous markov decision processes with arbitrary rewards. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[8] A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proc. of International Conference on Robotics and Automation (ICRA)*, 2002.

[9] Rainer Jäkel, Sven R. Schmidt-Rohr, Zhixing Xue, Martin Lösch, and Rüdiger Dillmann. Learning of probabilistic grasping strategies using programming by demonstration. In *IEEE International Conference on Robotics and Automation*, 2010.

[10] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *Proc. of Robotics Science and Systems conference (RSS)*, 2010.

[11] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Transactions on Robotics*, 20(6):967–977, 2004.

[12] Manuel Lopes, Francisco S. Melo, and Luis Montesano. Active learning for reward estimation in inverse reinforcement learning. In

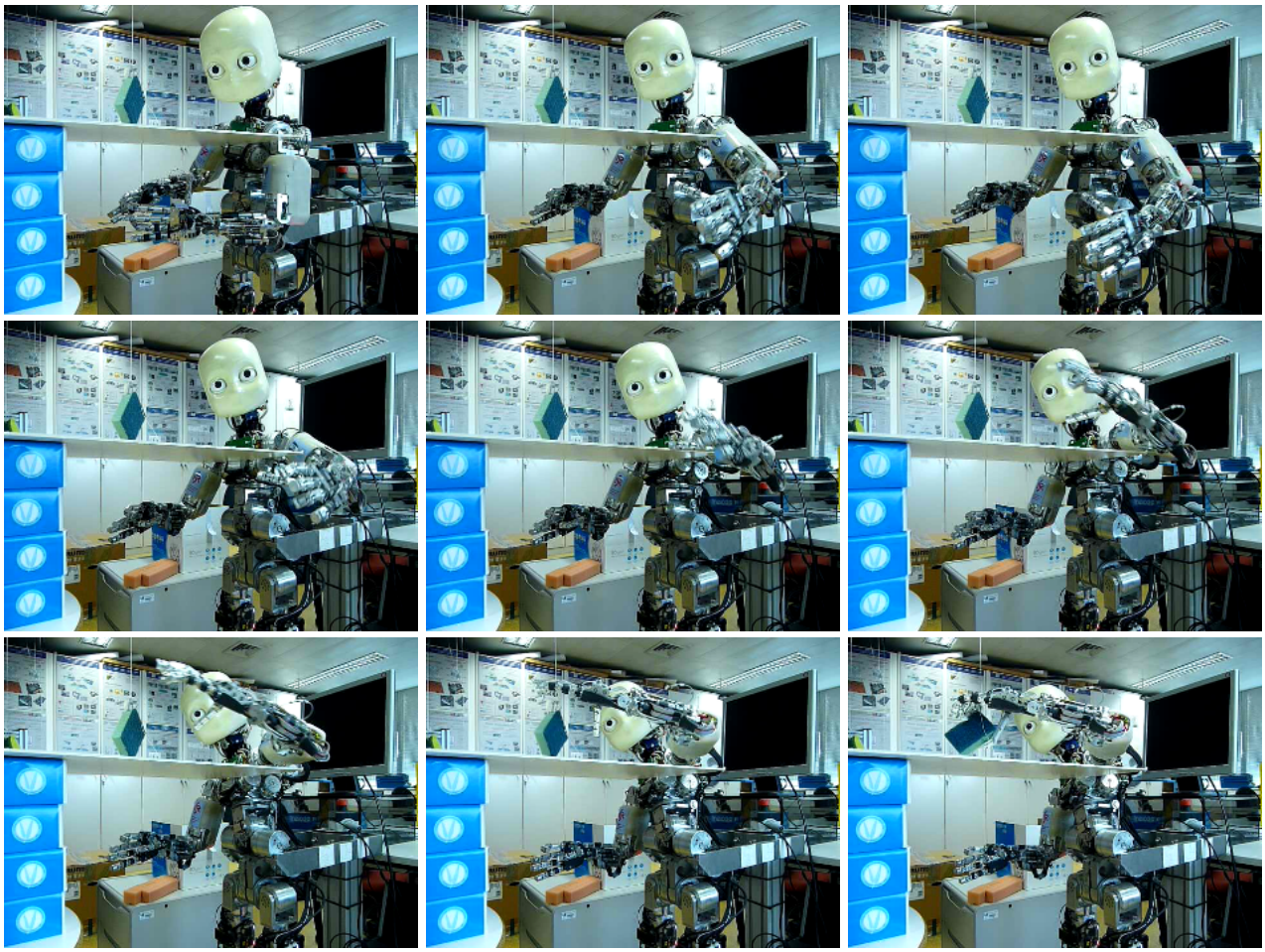

Fig. 7. The iCub humanoid robot performing the reaching and grasping motions while avoiding the white board. Due to the complexity of the motion, not only the left arm, but also the torso is moved.

European Conference on Machine Learning (ECML/PKDD), Bled, Slovenia, 2009.

[13] Ruben Martinez-Cantin, Manuel Lopes, and Luis Montesano. Body schema acquisition through active learning. In *IEEE International Conference on Robotics and Automation*, Alaska, USA, 2010.

[14] Luis Montesano and Manuel Lopes. Learning grasping affordances from local visual descriptors. In *IEEE International Conference on Development and Learning (ICDL)*, 2009.

[15] G. Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Uncertainty in Artificial Intelligence (UAI)*, pages 295–302, 2007.

[16] G. Neu and Csaba Szepesvári. Training parsers by inverse reinforcement learning. *Machine Learning*, 77:303–337, 2009.

[17] Andrew Y. Ng and Stuart J. Russel. Algorithms for inverse reinforcement learning. In *17th International Conference on Machine Learning (ICML)*, USA, 2000.

[18] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *20th Int. Joint Conf. Artificial Intelligence*, India, 2007.

[19] Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. Maximum margin planning. In *International Conference on Machine Learning*, July 2006.

[20] Alexandre Skoglund, Boyko Illiev, and Rainer Palm. Programming-by-demonstration of reaching motions: A next state planner approach. *Robotics and Autonomous Systems*, 58(5):607–621, 2010.

[21] Jürgen Sturm, Kurt Konolige, Cyrill Stachniss, and Wolfram Burgard. Vision-based detection for learning articulation models of cabinet doors and drawers in household environments. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[22] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *International Conference of Robotics and Automation (ICRA)*, 2010.

[23] E. Todorov and W . Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference*, pages 300–306, 2005.

[24] Marc 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.

[25] Marc Toussaint. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning (ICML)*, 2009.

[26] Marc Toussaint and Christian Goerick. A bayesian view on motor control and planning. In Olivier Sigaud and Jan Peters, editors, *From motor to interaction learning in robots*. Springer, 2010.

[27] Marc Toussaint, Nils Plath, Tobias Lang, and Nikolay Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[28] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *23nd International Conference on Machine Learning (ICML)*, 2006.

[29] Nikos Vlassis, Marc Toussaint, Georgios Kontes, and Savas Piperidis. Learning model-free robot control by a monte carlo em algorithm. *Autonomous Robots*, 27:123–130, 2009.

[30] Brian Ziebart, Drew Bagnell, and Anind Dey. Modeling interaction via the principle of maximum causal entropy. In *Proc. of International Conference on Machine Learning (ICML)*, 2010.