Enhanced Robot Learning using Fuzzy Q-Learning & Context-Aware Middleware

Charles C. Phiri[a], Zhaojie Ju[a], Naoyuki Kubota[b] and Honghai Liu[a]

***Abstract*— In this paper we continue with previous work by the authors implementing context-aware middleware to accelerate robot learning from demonstration, LfD. Specifically, we apply Fuzzy Q-Learning, FQL, reinforcement learning strategy to enhance the learning experience of the robot. Typically, fuzzy techniques allow the robot to make decisions without the need for an exhaustive map of the world. FQL, approximates the observable configuration space allowing the robot to overcome the high dimension challenge of feature decomposition and navigation in a stochastic environment.**

I. INTRODUCTION

Robot learning from demonstration, LfD, is one of the most promising methods for intuitively teaching humanoid robots from human action skills. The human expert demonstrates a task which the robot must imitate and perform according to its embodied capabilities. In previous work by the authors [13], we presented the architecture of context-aware middleware to accelerate the LfD learning experience. Reinforcement learning, RL, strategy is applied to provide online learning without the need for an exhaustive map beforehand. For a robot operating in stochastic environments, virtually at any given moment in time, it does not have a perfect model of the world it is operating in. RL algorithms are typically applied to sequential decision-making problems called the Markov Decision Processes, MDP [15]. Fuzzy Q-Learning, FQL, combines reinforcement learning strategy and fuzzy modeling of the world [6]. Methods have been proposed to combine FQL with Lyapunov theory providing stability with relation to perturbations within the operating environment [15]. This provides future direction of our research. Current research focuses on retrofitting intelligent behavior onto a robot by providing ambient context data from available embedded smart sensors. Using context-aware middleware and FQL allows the robot to take advantage of a wide range of ambient situational data to map out an online learning and adaptive policy.

The Fuzzy Q-Learning techniques applied use embodied robot sensors and actuators to implement cognitive capabilities. Context-aware Middleware, on the other hand, allows the robot learning experience to be augmented with context data from smart sensing environments. In our work we locate both the robot and the human expert in the 3D map of the world [19]. 3D-SLAM is used for pose and landmark detection from a global perspective for both the robot and human expert [2], [16], [18], [8]. We demonstrate that by sharing the context data, the field of view of the robot is enhanced thereby speeding up the learning process.

In previous work [13], we have noted that LfD involves the human instructor intuitively teaching the robot behavior without requiring the high-level programming skills. The learning challenges in LfD can be categorized into motion planning challenges and understanding of tasks. Understanding of a task requires a higher level knowledge of the intention of the tasks. This paper does not tackle the motivation for executing the tasks. It is assumed that the motivation for the tasks is predetermined by the human instructor. LfD, on the other hand, suffers from the curse of dimensionality; at the same time, the correspondence problem due to the differences in the physical embodiment and perception subsystems of both humans and humanoid robots cannot be ignored.

This paper is organized as follows: Section II breakdown the Fuzzy Q-Learning method into its constituent parts. It shows how the reinforcement algorithm fits together with the Fuzzy Logic Controller to remove the crisp decisions resulting in smooth transitions in a high dimension kinematic data set. Section III proposes the methodology for the application of the methods described in section II and how the context is developed and distributed to the robot. Section IV discusses the results and Section V provides a summary of the results.

II. FUZZY Q-LEARNING: THEORY

A. Markov Decision Process

Markov decision process, MDP, provides a model for making decisions in situations where the outcomes are partly random and partly under the control of the robot. MDP's can be framed as an optimization problem thereby allowing the application of Reinforcement Learning, RL, and Dynamic Programming, DP, to identify an optimal solutions in an infinite horizon problem space. RL and DP are algorithmic methods for solving problems in which the actions/decisions are applied to a system over an extended period of time to achieve the desired goal. Unlike DP, RL does not require an explicit model of the behavior of the system. This, however, does not imply that RL techniques cannot take advantage of a system behavior model [6], [1].

A stochastic process has the Markov property if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state, not on the sequence of events that preceded it. The utility function is tied to the immediate sequence of rewards as opposed to the final utility which may never be reached. In directed learning, the process

must have an unequivocal goal state. In essence this reduces the infinite horizon space to the finite horizon space by creating a stopping state if we only consider the end-point matching. In motion planning however, the path has features that influence the optimal path to the end-point pose. The following equations (1,2) show the differences between Finite and Infinite Horizon spaces.

Infinite Horizon:

$$Q(s,a) = r(s,a) + \gamma\sigma[p(s'|s,a)\max_{a'} Q(s',a')] \quad (1)$$

n-Step Finite Horizon:

$$Q_n(s,a) = r(s,a) + \gamma\sigma[p(s'|s,a)\max_{a'} Q_{n-1}(s',a')] \quad (2)$$

At each time step, the robot is in some state, s, and may choose any available action, a, in the state s. MDP is a discrete time stochastic control process. At the next time step the robot responds by randomly moving into a new state, s', and gets a corresponding reward, $R_a(s,s')$. The probability that the robot moves into its new state, s, is only influenced by the chosen action and is expressed by the state transition function, $P_a(s,s')$. Thus, the next state, s', depends on the current state, s, and the actions, a; however, given s and a, the next state is conditionally independent of all preceding states and actions. In other words, the state transition of MDP satisfies the Markov Property.

A policy, π, is a mapping from each state, $s \in S$, and $a \in A(s)$, to the probability $\pi(s,a)$ of taking a when in state s. The expected value of a policy, π, for the discounted reward, with discount, γ, is defined in terms of two interrelated functions, V^π and Q^π. Where $V^\pi(s)$ is the expected value of following π in s. A policy, π, is an optimal policy if there is no policy π' and no state, s, such that $V\pi'(s) > V\pi(s)$. Let $Q^\pi(s,a)$, where s is a state and a is an action, be the expected value of doing a in state s and then following policy π. Recall that $V^\pi(s)$ where s is a state, is the expected value of following policy π in state, s. V^π and Q^π can be defined recursively in terms of each other. If the agent is in state, s, performs action a, and arrives in state, s', it gets the immediate reward of $R(s,a,s')$ plus the discounted future reward, $\gamma V\pi(s')$. When the robot is planning it does not know the actual resulting state, so it uses the expected value, averaged over the possible resulting states:

$$Q^\pi(s,a) = \Sigma s' P(s'|s,a)(R(s,a,s') + \gamma V^\pi(s')) \quad (3)$$

$V^\pi(s)$ is obtained by doing the action specified by π and then acting following π:

$$V^\pi(s) = Q^\pi(s,\pi(s)) \quad (4)$$

B. Q-Learning

Q-Learning belongs to the class of model-free value iteration algorithms in RL. Q-Learning starts from an arbitrary initial Q-function, Q_0, and continually updates the Q-function without requiring the model, using observed state

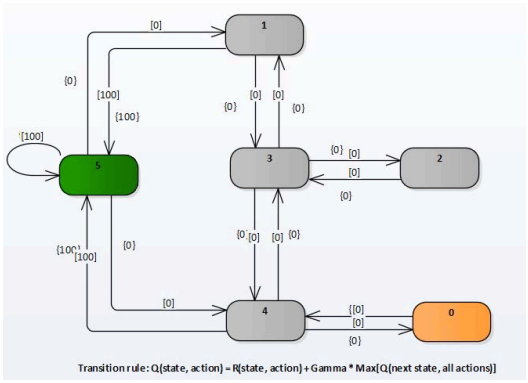

Fig. 1. Starting in State 2, the challenge is to to get to state 5, the goal state. The robot could start from any random state.

State \ Next State	5	0	1	2	3	4
	S0	S1	S2	S3	S4	S5
5 S0	[100]	N	[0]	N	N	[0]
0 S1	N	N	N	N	N	[0]
1 S2	[100]	N	N	N	[0]	N
2 S3	N	N	N	N	[0]	N
3 S4	N	N	[0]	[0]	N	[0]
4 S5	[100]	[0]	N	N	[0]	N

Fig. 2. The state space translates to this matrix where the N represents transitions that are not possible or allowed.

transition and rewards. The Q-function update strategy is captured in the expression [6]:

$$Q_{k+1}(x_k,u_k) = Q_k(x_k,u_k) + a_k[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1},u') - Q_k(X_k,u_k)] \quad (5)$$

The performance of RL and Q-learning in particular is strongly determined by the proportion of effort dedicated to exploration against exploitation. This can be shown to be linked directly to the computational burden of the robot. The parameters of the RL strategy are updated as soon as a sample of the value function is available thereby maintaining the online strategy. FQL is used to enhance the self-learning capabilities of the robot.

3D-SLAM algorithm provide an element of the prior information required by FQL. A toy problem demonstrating how RL works for a finite state space is shown in the following steps.

The Q-Learning Algorithm for solving this toy problem would is expressed as [6]:

1) Set γ, and environment rewards in matrix R
2) Initialize matrix $Q = 0$
3) For each episode:
   - Select a random initial state, s

$$R = \begin{array}{c} \\ \text{State} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \text{Action} \\ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Fig. 3. The reward matrix is formally expressed as this state and action matrix. The aim is to compute a policy that maximizes the Reward.

- Do While not in goal state
  a) Choose 1 among all possible actions for the current state
  b) Using this possible action, consider going to the next state
  c) Get maximum Q value for this next state based on all possible actions
  d) Compute: Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
  e) Set current state = next state
- End Do

4) End For

C. Fuzzy Logic Controller

The fuzzy logic is used to map an input space to an output space primarily using a list of if-then statements called rules which are evaluated in parallel. The sequence in which the rules are handled is not significant at all. The system setup involves interpreting these rules in terms of the parameters to be used and their descriptions. Fig. 4 shows a summary of the process. A membership function (MF) is a curve that defines how each of the points in the *universe of discourse* (input space) is mapped to a degree of membership bounded between 0 and 1.

The fuzzy logic algorithm is summarized as follows:

1) Define the linguistic variables and terms (initialization)
2) Construct the membership functions (initialization)
3) Construct the rule base (initialization)
4) Convert crisp input data to fuzzy values using the membership functions (fuzzification)
5) Evaluate the rules in the rule base (inference)
6) Combine the results of each rule (inference)
7) Convert the output data to non-fuzzy values (defuzzification)

A robot performing a walking task could be implemented using dead reckoning feedback from the positional sensors in the robot. With dead reckoning accuracy is hard to achieve. By definition dead reckoning is the determination, without the aid of external observations, of the position and orientation of a robot from the record of the courses traveled,

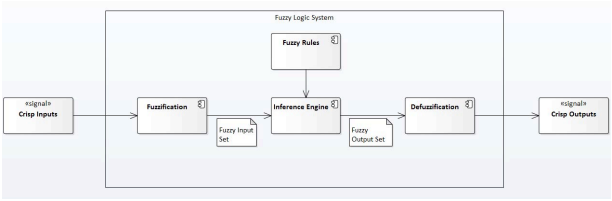

Fig. 4. Fuzzy Logic System

the distance made, and the known estimated drift [14], [17]. The method proposed in this paper allow the robot and the human instructor to both be localized on a 3D map. The map specific parameters do not contribute much to the fuzzy system. This is an implementation choice to reduce the number of features being processed in the fuzzy inference system. At each time step the the localization information is updated and reinforcement learning applied to adjust the control policy for the planning.

Fuzzy Q-Learning provides an automatic mean of self-tuning of Fuzzy Inference System based only on reinforcement signals. Continuous states are handled and continuous actions are generated by fuzzy reasoning. Prior knowledge can be embedded into the fuzzy rules, which can reduce learning time significantly. The demand for an expert to provide a complete set of rules to navigate stochastic environments is virtually unrealistic. The middleware is used in an attempt to provide an adaptive layer for adjusting both the rules and the reference knowledge.

III. Fuzzy Q-Learning: Implementation

The application of RL in a continuous state space suffers from the curse of dimensionality. Function approximators are used to implement the FQL model.

In FQL the rules of a Fuzzy Inference System (FIS) with Q-values takes the form:

$$\begin{aligned} R_i : \text{If } x_1^k \text{ is } L_1^k \text{ and ... and } x_n^k \text{ is } L_n^k \text{ then} \\ u = u_1 \text{ with } q(i,1) \\ \text{or } u = u_2 \text{ with } q(i,2) \\ ... \\ \text{or } u = u_m \text{ with } q(i,m) \end{aligned} \tag{6}$$

where L_s^i is the linguistic term for the variable x_s^k in rule R_i. The membership function $\mu_{L_s^i}.x^k = \{x_1^k, x_2^k, ..., x_n^k\}$ is the input vector for the FIS and the true-value is expressed as $R_i : \alpha_i(x^k)$. For the i-th rule, R_i, there exists m competing actions such that $U = \{u_1, u_2, ..., u_n\}$ and $q(i,1)$ is optimal [5], [15].

Fig. 5 shows the Closed Loop model for the biped locomotion in the Nao robot using center of pressure, (COP) method and the Three-Dimensional Linear Inverted Pendulum method, 3D-LIPM; the foot trajectory and the trunk information is used to calculated the inverse kinematics for the robot's stable walk. This feature is called *Stable and Omnidirectional Walk*. Apart from the intrinsic merit of the implementation it also highlights extension points for implementing novel algorithms by monitoring the feedback loop

and adjusting the control. A control policy is developed for reinforcement learning by integrating the different constraints into the cost function.

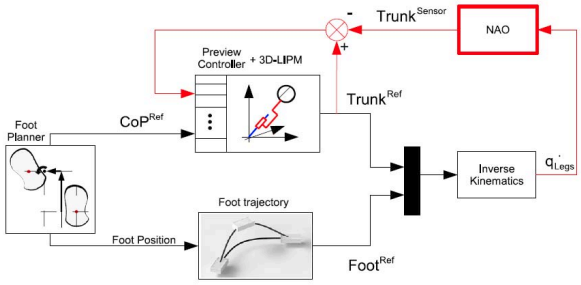

Fig. 5. NAO Closed Loop Walk [12], [17]

An analogy of humanoid learning using reinforcement learning would be the way a human being makes decisions about tasks and situations that have not been encountered before and the consequences of making a decision can only be received and applied in the next time step (iteration).The quality of the reward is evaluated after each action. If an experienced teacher is available, the number of mistakes (exploration) is bounded to achieving tasks that are within the domain of the expert with a high degree of confidence.

The prevailing assumption in LfD is that there exists a desired robot control policy, innate in a human expert, Ω^y, which perfectly describes what the robot does in every situation. The goal of LfD is then to create an analogous robot policy, Ω^x, that will enable the robot to exhibit this desired behavior. Searching for this optimal policy should, in theory, be easy and intuitive.

A policy is a plan in the motion planning context [10]. Aude, et. al., posit a metric, M, for mapping the computation of the similarity between the policies that is optimal when Ω^y and Ω^x match exactly, i.e. when the robot does exactly what the human would want in all situations. They further observe that there may be multiple Ω^x that maximize M, due in part to fundamental differences in robot and human perception and embodiment. To resolve the correspondence problem and allow for a direct comparison of the policies, the human and robot perceptions and actions are mapped into a common task-specific space, with a pair of operators ϕ_y, ϕ_x, which can be encoded into the teaching interface [4].

The case for fuzzy techniques and reinforcement learning, in LfD, can be established by analyzing the following 5 steps captured by Aude, et al and summarized here using the definition of the metric M [4], the following can be reduced to an optimization problem:

1) End Point Matching:

$$M(\mathfrak{X},\mathfrak{Y}) = -\frac{1}{\mathsf{N_x}}\sum_{n=1}^{\mathsf{N_x}}(\phi_{\mathsf{x}}(\mathbf{x}^n_{\mathsf{T}^{\mathsf{x}}_n}) - \frac{1}{\mathsf{N_y}}\sum_{m=1}^{\mathsf{N_y}}\phi_{\mathsf{y}}(\mathbf{y}^m_{\mathsf{T}^{\mathsf{y}}_m}))^2 \tag{7}$$

Where the target location is taken to be the average ending location of the demonstrations, and learning is aimed at minimizing the mean squared error of the ending location of the trials.

2) Path Matching:
In addition to (7) the trajectory matching is given by:

$$M(\mathfrak{X},\mathfrak{Y}) = -\sum_{n=1}^{\mathsf{N_x}}\sum_{t=0}^{\mathsf{T}^{\mathsf{x}}_n}(\phi_{\mathsf{x}}(\mathbf{x}^n_t) - \phi_{\mathsf{y}}(\mathbf{y}^n_t))^2 \tag{8}$$

However, this equation (8) places a burden that the human and robot generate the same number of trajectories ($\mathsf{N_x} = \mathsf{N_y}$), starting in equivalent locations ($\mathfrak{y}_0 \approx \mathfrak{x}_0$), and that the paired trajectories take the same length of time ($\mathsf{T}^{\mathsf{y}}_n = \mathsf{T}^{\mathsf{x}}_n$). These first two conditions can be met with experimental design, and the last is often achieved by resampling or time warping [4].

3) Path Similarity:
The assumptions in Path Matching (8) can be relaxed by introducing features of the paths, such as smoothness or minimum jerk. Given f as some feature of the trajectories defined in the task space, the Path Similarity is computed:

$$M(\mathfrak{X},\mathfrak{Y}) = -\left((\frac{1}{\mathsf{N_x}}\sum_{n=1}^{\mathsf{N_x}} f(\phi_{\mathsf{x}}(\mathbf{X}^n))) - (\frac{1}{\mathsf{N_y}}\sum_{n=1}^{\mathsf{N_y}} f(\phi_{\mathsf{y}}(\mathbf{Y}^n))\right) \tag{9}$$

4) Reward:
A particular case of the Path Similarity (9) considers accumulated reward along the paths. However, in this case, it is unnecessary to compare to the demonstrator. Instead, the demonstrator can be used to initialize the learning process, or provide the reward signal itself [4].

$$M(\mathfrak{X},\mathfrak{Y}) = \frac{1}{\mathsf{N_x}}\sum_{n=1}^{\mathsf{N_x}} R(\phi_{\mathsf{x}}(\mathbf{X}^n)) \tag{10}$$

5) Probabilistic:
Alternatively, the known trajectories are considered as samples from some underlying probability distribution. Where P is a density estimator:

$$M(\mathfrak{X},\mathfrak{Y}) = P(\phi_{\mathsf{y}}(\mathfrak{Y})|\phi_{\mathsf{x}}(\mathfrak{X})) \tag{11}$$

This approach can be thought of as maximizing the probability that the robot will generate the same trajectories as the human.

IV. Experiments and Simulation

Locomotion in biped robots is a an interesting and challenging field of study. As established in the preceding sections the task in this research involve learning from demonstration, LFD. With the Nao H25 v4 robot platform learning from demonstration can be as simple as repeating gestures from the instructor of more advance tasks as the Cartesian control Hula-Hoop Challenge. We build on this research and attempt to go beyond the simplistic mechanical repetitive tasks; we explore high level tasks such as how to

intuitively teach a robot such a seemingly simple command as ”come to me”? To understand the challenge in section we segment the problems and show the intermediate step toward this high-level goal. On the path to this goal, we explore cognitive behavior embodied in fuzzy system models and reinforcement learning.

A series of experiments and simulations were conducted to evaluate the theory proposed in this research. The experiments were carried out using the NAO H25 humanoid platform, the Microsoft Kinect sensor device and the simulation were conducted in the MATLAB environment. The context-aware middleware applies JSON-LD as an instance of web ontology language; the software itself was developed using C#. Due to poor support for modern development tools, we took the approach to use python for implementing the control mechanism for the NAO robot.

A. Human Action Skills Segmentation

The extraction of feature for LfD is described by the context offered from a common view point The Kinect sensor device is configured to observe both the robot and the human operator. This shows a stereoscopic stand with an adjustable head. In future we will experiment with multiple Kinect devices and and also an actuated stand to allow the head to follow the robot. Ceiling mounting has also been tried but that proved too challenging to maintain at this phase. Left: head tilted downward. Right: Normal position.

Fig. 6. The setup for the Kinect. Common View Point.

Fig. 7 shows the human skeleton superimposed on the color image. The joint angles are computed from this frame. The $37\,\mathrm{deg}$ shown in the upper right-hand corner reports the values of the joint angles. For demonstration the angles are also displayed on the canvas.

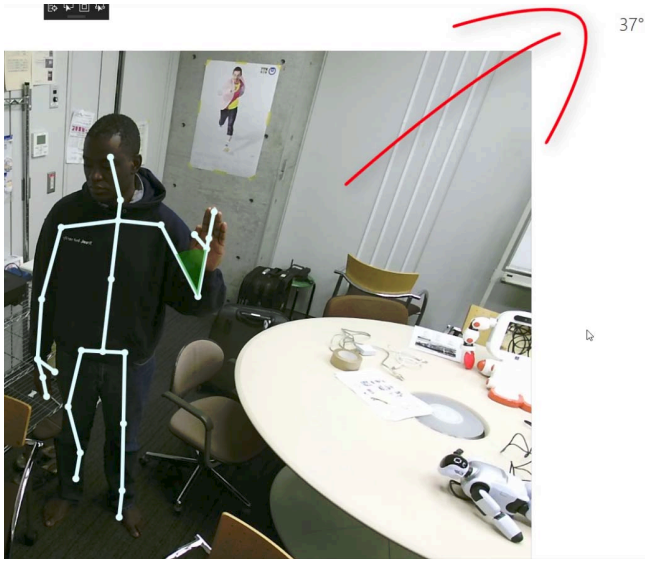

Fig. 7. The human skeleton data.

Fig. 8 shows human Pose Data from the Kinect serialized to JSON and propagated to the middleware as context for the learning. The JSON shown here does not include the ontological rule to encoded into a directed graph of constraints and knowledge for cognitive behavior. However, this should be easy to infer. The reader is encouraged to review [13].

Fig. 8. Human Pose Data.

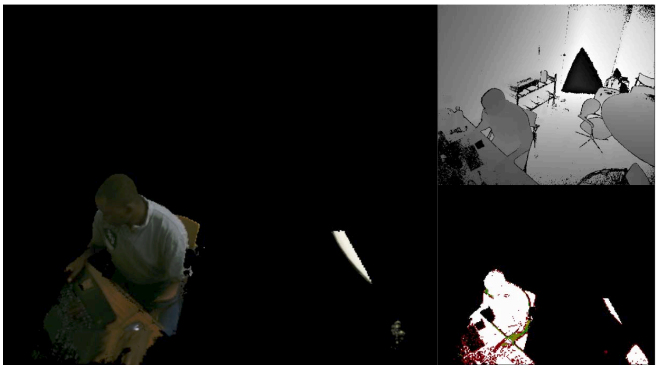

Fig. 9. 3D-Depth data from the Kinect

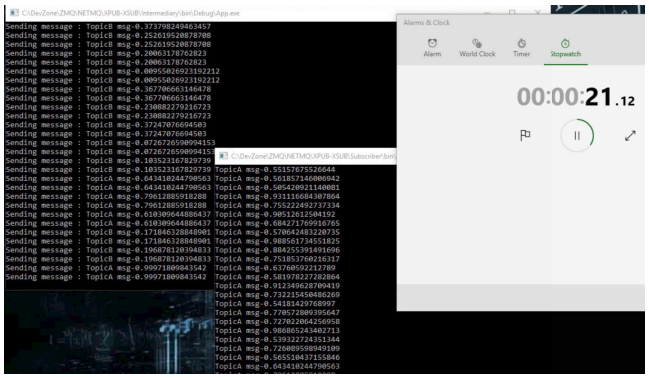

Fig. 10. Shows the Publisher and the Intermediary Application (The two are combined). A subscriber to Topic A is also shown. The publisher can disconnect as reconnect without problems. In the video we show a quick comparison of the throughput against the stopwatch. The current rate is with the throughput throttled to only allow up to 1000 messages per second.

B. Cartesian Control

The NAO robot has onboard a dedicated mechanism to control directly the Effectors of the robot in a Cartesian space. Two Inverse Kinematics, IK, solvers are implemented by the manufacturer, viz, a classical IK solver using only the joints of the effector chain to reach a target; and a generalized IK solver for whole body control using all the robot joints to reach a target [12]. For the Whole Body Balancer, for example the generalized IK are described using the classical form of a quadratic program which has a cycle time of 20ms:

$$\min \frac{1}{2}\|Y - Y^{des}\|_Q^2 \text{ such as } \begin{cases} AY + B & = 0 \\ CY + D & \geq 0 \end{cases} \qquad (12)$$

where Y: Unknown vector, Y^{des}: Desired but not necessarily feasible solution;Q: Quadratic norm; A, b, C and d: Matrices and vectors which express linear equality and inequality constraints [7]. Quadratic Programming is a problem of optimizing a quadratic function of several variables subject to some linear constraints on the variables.

Taking into account the velocity of all articulated joints and unactuated joints, the equality constraints are about keeping feet fixed or in a plane whereas the inequality constraints are:

- Joint limits.
- Balance. The Center Of Mass is constrained to stay within the support polygon.

SE3 Interpolation is used for all interpolations that are defined in Cartesian Space. It provides a spline-like interpolation which allows for initial speeds and points of passage to be taken into account, ensuring smooth trajectories that respect speed constraints. SE3 being the set of all transformations that can be applied to a rigid body in addition to composition and inversion. The classical IK solver due to robot singularity (i.e. the inverse Jacobian becomes singular (determinant = 0)) configuration could create huge joint velocity and the robot could lose balance and fall.

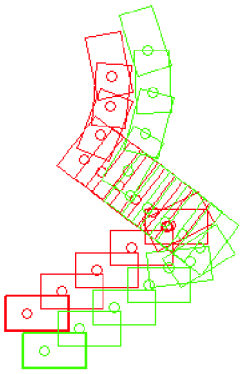

Fig. 11. Nao robot performance limits.

TABLE I
NAO GAIT PARAMETERS [12]

Name		Default	Minimum	Maximum	Settable
MaxStepX	maximum forward translation along X (meters)	0.04	0.001	0.080 [3]	yes
MinStepX	maximum backward translation along X (meters)	-0.04			no
MaxStepY	absolute maximum lateral translation along Y (meters)	0.14	0.101	0.16	yes
MaxStepTheta	absolute maximum rotation around Z (radians)	0.349	0.001	0.524	yes
MaxStepFrequency	maximum step frequency (normalized, unit-less)				yes
MinStepPeriod	minimum step duration (seconds)	0.42			no
MaxStepPeriod	maximum step duration (seconds)	0.6			no
StepHeight	peak foot elevation along Z (meters)	0.02	0.005	0.04	yes
TorsoWx	peak torso rotation around X (radians)	0	-0.122	0.122	yes
TorsoWy	peak torso rotation around Y (radians)	0	-0.122	0.122	yes
FootSeparation	alter distance between both feet along Y (meters)	0.1			no
MinFootSeparation	minimum distance between both feet along Y (meters)	0.088			no

C. The Hula-Hoop Problem

The Hula-Hoop motion problem, fig. 12, presents an interesting starting point for evaluating the learning strategy for LfD in now. The task constraints are as follows:

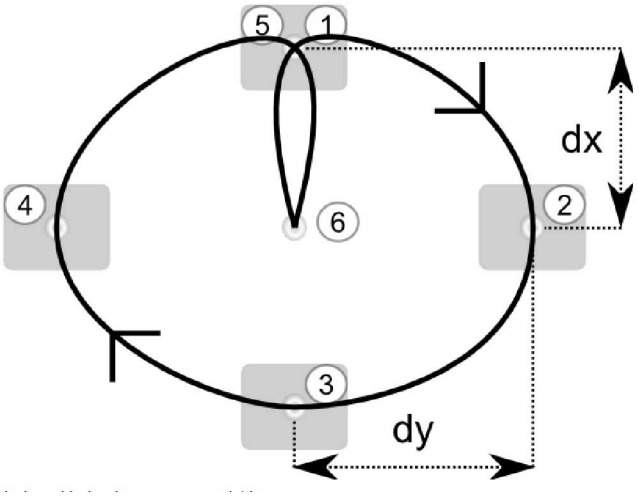

Fig. 12. Hula-Hoop Motion [12]. Basic implementation of the source is provided at the Aldebaran NAO robot website.

Motion checkpoints:

- forward / bend backward
- right / bend left
- backward / bend forward

- left / bend right

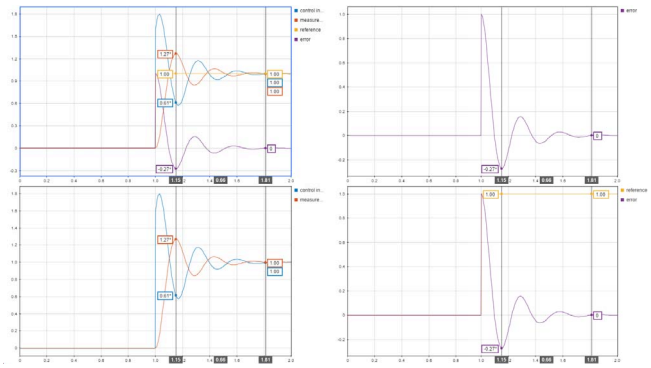

Fig. 13. Closed-Loop control of a dynamical system response curves.

[11] have shown NAO robot control using Fuzzy-PD controller to correct gait. In this research we decomposed the problem as shown in fig. 14.

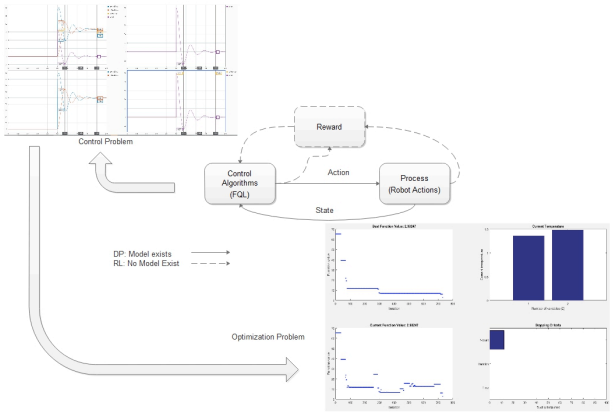

Fig. 14. Mapping control states to FQL

V. Summary

In this research, we apply Fuzzy Q-Learning, FQL, reinforcement learning strategy to enhance the learning experience of the robot. Typically, fuzzy techniques allow the robot to make decisions without the need for an exhaustive crisp model of the world. FQL, approximates the observable configuration space allowing the robot to overcome the high dimension challenge of feature decomposition and navigation in a stochastic environment. One of the features that makes humanoid robots attractive as social companions is their ability and capacity to learn. Methods advocated by LfD provide intuitive, time and computationally tractable transfer of knowledge from the human instructor to the robot. Using simulation and Aldebaran Nao H25 robot platform which has 25 degrees of freedom and a kinematics frame already computed, we show some of these mechanisms. The numerous sensors the Naos head, hands and feet, as well as sonars in the torso, have enabled researchers to build a semblance of narrow cognitive function. The robot, however, cannot faithfully reproduce the actions of the human expert due to the affordance and correspondence problem. Via the context-aware ontological middleware, we provide context data to the robot from a common point of view allowing decision support data to be referenced from a single point which enables optimization of the squared mean errors and trajectory deviations to be easily characterized augmenting the robots perception.

Acknowledgments

The authors would like to acknowledge support from DREAM project of EU FP7-ICT (grant no. 611391), Research Project of State Key Laboratory of Mechanical System and Vibration China (grant no. MSV201508), and National Natural Science Foundation of China (grant no. 51575412).

References

[1] Abhijit Gosavi, Variance-penalized Markov decision processes: Dynamic programming and reinforcement learning techniques, International Journal of General Systems, issue 43 vol. 6, pp. 649-669, 2014.
[2] Azhar Aulia Saputra, Jnos Botzheim, Indra Adji Sulistijono, Naoyuki Kubota: Biologically Inspired Control System for 3-D Locomotion of a Humanoid Biped Robot. IEEE Trans. Systems, Man, and Cybernetics: Systems 46(7): 898-911 (2016)
[3] Babu, A. K. A. and R. Sivakumar (2015). Development of Type 2 Fuzzy Rough Ontology-based Middleware for Context Processing in Ambient Smart Environment. Intelligent Computing and Applications, Springer: 137-143.
[4] Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Handbook of Robotics Chapter 59: Robot Programming by Demonstration. Handbook of Robotics. Springer.
[5] Bonarini, A., Lazaric, A., Montrone, F., & Restelli, M. (2009). Reinforcement distribution in fuzzy Q-learning. Fuzzy sets and systems, 160(10), 1420-1443.
[6] Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). Reinforcement learning and dynamic programming using function approximators (Vol. 39). CRC press.
[7] Ferreau, H. J., Bock, H. G., & Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. International Journal of Robust and Nonlinear Control, 18(8), 816-830.
[8] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., & Tanie, K. (2001). Planning walking patterns for a biped robot. IEEE Transactions on robotics and automation, 17(3), 280-289.
[9] Ju, Z. and Liu, H.Fuzzy Gaussian Mixture Models, Pattern Recognition 2012, 45(3):1146-115.
[10] LaValle, S. M. (2006). Planning algorithms: Cambridge university press.
[11] LpezCaudana E. A. & Gutirrez, C. D. G. (2016). Fuzzy PD Controller in NAO System's Platform, Automation and Control Trends, Dr. Pedro Ponce (Ed.), InTech, DOI: 10.5772/63979. Available from: http://www.intechopen.com/books/automation-and-control-trends/fuzzy-pd-controller-in-nao-system-s-platform
[12] "NAO Key Feature - Stable and Omnidirectional Walk," [Online]. Available: http://www.aldebaran.com/sites/aldebaran/files/featurepaperstableandomnidirectionalwalk.pdf [Accessed 7 July 2016]
[13] Phiri, C., Ju, Z., Liu, H. (In Press). Accelerating Humanoid Robot Learning from Human Action Skills Using Context-Aware Middleware. ICIRA2016
[14] Roston, G. P., & Krotkov, E. P. (1991). Dead reckoning navigation for walking robots (No. CMU-RI-TR-91-27). CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.
[15] Saxena, R.,& Sharma, R. (2015, March). A Hybrid Lyapunov Fuzzy Reinforcement Learning Controller. In Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on (pp. 1193-1197). IEEE.
[16] Stachniss, C. (2014). Introduction to Robot Mapping. SLAM-Course.
[17] Sugihara, T., Nakamura, Y., & Inoue, H. (2002). Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control. Paper presented at the Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on.

[18] Thrun, S., et al. (2005). Probabilistic robotics, MIT press.
[19] Wei Hong Chin, Chu Kiong Loo, Manjeevan Seera, Naoyuki Kubota, Yuichiro Toda: Multi-channel Bayesian Adaptive Resonance Associate Memory for on-line topological map building. Appl. Soft Comput. 38: 269-280 (2016)