Dynamics Model Abstraction Scheme Using Radial Basis Functions

Tolu, Silvia; Vanegas, Mauricio; Agis, Rodrigo; Carrillo, Richard; Cañas, Antonio

Published in:
Journal of Control Science and Engineering

Link to article, DOI:
10.1155/2012/761019

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Dynamics Model Abstraction Scheme Using Radial Basis Functions

Silvia Tolu · Mauricio Vanegas · Rodrigo Agís · Richard Carrillo · Antonio Cañas

Received: date / Accepted: date

Abstract This paper presents a control model for object manipulation. Properties of objects and environmental conditions influence the motor control and learning. System dynamics depend on an unobserved external context, e.g., work load of a robot manipulator. The dynamics of a robot arm change as it manipulates objects with different physical properties, e.g., the mass, shape or mass distribution. We address active sensing strategies to acquire object dynamical models with a radial basis function neural network (RBF). Experiments are done using a real robot’s arm and trajectory data are gathered during various trials manipulating different objects. Biped robots do not have high force joint servos and the control system hardly compensates all the inertia variation of the adjacent joints and disturbance torque on dynamic gait control. In order to achieve smoother control and lead to more reliable sensorimotor complexes we evaluate and compare a sparse velocity-driven vs. a dense position-driven control scheme.

Keywords Dynamics Model · Sensorimotor · Radial Basis Function (RBF) · Velocity-Driven Control Scheme

1 Introduction

Current research of biomorphic robots can highly benefit from simulations of advance learning paradigms [19, 38, 43, 58, 59] and also from knowledge acquired from biological systems [36, 51, 52]. The basic control of robot actuators is usually implemented by adopting a classic scheme [20, 25, 44, 46, 54, 61]: a) the desired
trajectory, in joint coordinates, is obtained using the inverse kinematics [44, 56, 61] and b) efficient motor commands drive, e.g. an arm, making use of the inverse dynamics model or using high power motors to achieve rigid movements. Different controlling schemes using hybrid position/velocity and forces have been introduced mainly for industrial robots [20, 46]. In fact, industrial robots are equipped with digital controllers, generally of PID type with no possibility of modifying the control algorithms to improve their performance. Robust control (control using PID paradigms [44, 47, 50, 56, 61–63]) is very power consuming and highly reduces autonomy of non-rigid robots. Traditionally, the major application of industrial robots is related to tasks that require only position control of the arm. Nevertheless, there are other important robotic tasks that require interaction between the robot’s end-effector and the environment. System dynamics depend on an unobserved external context [16, 43], e.g., work load of a robot manipulator.

Biped robots do not have high force joint servos and the control system hardly compensates all the inertia variation of the adjacent joints and disturbance torque on dynamic gait control [13]. Motivated by these concerns and the promising preliminary results [57], we evaluate a sparse velocity-driven control scheme. In this case, smooth motion is naturally achieved, cutting down jerky movements and reducing the propagation of vibrations along the whole platform. Including the dynamic model into the control scheme becomes important for an accurate manipulation, therefore it is crucial to study strategies to acquire it. There are other bio-inspired model abstraction approaches that could take advantage of this strategy [7, 32–34]. Conventional artificial neural networks [17, 21] have been also applied to this issue [11, 14, 29]. In biological systems [36, 51, 52], the cerebellum [10, 23] seems to play a crucial role on model extraction tasks during manipulation [4, 24, 55, 61]. The cerebellar cortex has a unique, massively parallel modular architecture [3, 4, 37] that appears to be efficient in the model abstraction [24, 25].

Human sensorimotor recognition [37, 52, 61] continually learns from current and past sensory experiences. We set up an experimental methodology using a biped robot’s arm [4, 55] which has been equipped, at its last arm-limb, with three acceleration sensors [22] to capture the dynamics of the different movements along the three spatial dimensions. In human body, there are skin sensors specifically sensitive to acceleration [31, 52]. They represent haptic sensors and provide acceleration signals during the arm motion. In order to be able to abstract a model from object manipulation, accurate data of the movements are required. We acquire the position and the acceleration along desired trajectories when manipulating different objects. We feed these data into the radial basis function network (RBF) [17, 29]. Learning is done off-line through the knowledge acquisition module. The RBF learns the dynamic model, i.e. it learns to react by means of the acceleration in response to specific input forces and to reduce the effects of the uncertainty and nonlinearity of the system dynamics [26, 65].

To sum up, we avoid adopting a classic regular point-to-point strategy (see Fig. 1.b) with fine PID control modules [25, 44, 56] that drive the movement along a finely defined trajectory (position-based). This control scheme requires high power motors in order to achieve rigid movements (when manipulating heavy objects) and the whole robot augments vibration artefacts that make difficult accurate control. Furthermore,
Fig. 1 a) Computing scheme. An FPGA processing platform embeds all the real-time control interfaces. The robot is connected to the FPGA which receives commands from the host PC. The robot is equipped with three accelerometers in its arm. The computer captures the sensor data. A RBF is used for learning the objects models from sensorimotor complexes. b) Dense Position-Driven scheme. A motor moves from an initial position $P_1$ to a final position $P_f$ temporally targeting each intermediate point $P_i$ along this trajectory. c) Sparse Velocity-Driven scheme. For each position $P_i$ indicated above a target velocity is calculated for a motor with a time step $T$ of 0.015s. Using a fixed sampling frequency for obtaining feedback position, we calculate the distance $\Delta P$ from this current position to a position $P_{cd}$ corresponding to a change of direction. d.1) Dense position-driven trajectory. The trajectory is defined in terms of regularly sampled intermediate points (dashed line). d.2) The thinner continuous line represents the real curve executed by the robot’s arm in the space under the sparse velocity-driven control. The black arrows indicate points in which changes of direction take place. The thicker continuous line corresponds to the smooth velocity-driven trajectory executed anticipating the points in which changes of direction are applied before the arm reaches them.

these platform jerks induce high noise in the embodied accelerometers. Contrary to this scheme, we define the trajectory by a reduced set of target points (Fig. 1.c) and implement a velocity-driven control strategy that highly reduces jerks.

2 Control Scheme

The motor-driver controller and the communication interface are implemented on an FPGA (Spartan XC31500 [8] embedded on the robot) (Fig. 1.a). Robonova - I [18] is a fully articulating mechanical biped robot, 0.3 m, 1.3 kg, controlled with sixteen motors [18]. We have tested two control strategies for a specific “L” trajectory previously defined for the robot’s arm controlled with three motors (one in the shoulder and two in the arm, (see Fig. 2). This whole movement along three joints makes use of three degrees of freedom during the trajectory. The movement that involves the motor at the last arm-limb is the dominant one, as is evidenced from the sensorimotor values in figure 4.b.
Fig. 2 Diagram of the whole system architecture. The robot’s picture in the left is showing the different joints of the 3 DOF Robonova arm involved in the movement. The PC applied appropriate motor commands to each joint of the arm at all times during the movement to follow the desired trajectory. To relieve the computer from interface computation and allow real-time communication with the robot, an FPGA board containing position acquisition modules and motor-driver controller circuits with PWM (Pulse With Modulation) was connected to the robot. The communication task between devices and the operations described above were synchronised by the computer speeding up the process. Finally, the batch learning with RBF of sensorimotor data collected during the movements was performed in the PC. The dataset was split up into training data (75%) and test data (25%).

2.1 Control Strategies

The trajectory is defined in different ways for the two control strategies: **Dense Position-Driven.** The desired joint trajectory is finely defined by a set of target positions \( P_i \) that regularly sample the desired movement (see Fig. 1.b).

**Sparse Velocity-Driven.** In this control strategy, the joint trajectory definition is carried out by specifying only positions related to changes in movement direction. During all straight intervals, between a starting position \( P_1 \) and the next change of direction \( P_{cd} \), we proceed to command the arm by modulating the velocity towards \( P_{cd} \) (see Fig. 1.c). Velocities \( V_n \) are calculated taking into account the last captured position and the time step in which the position has been acquired. Each target velocity for period \( T \) is calculated with the following expression, \( V_n = (P_{cd} - P_n) / T \).

The control scheme applies a force proportional to the target velocity. Figure 1.d.2 illustrates how a smoother movement is achieved using sparse velocity-driven control. The dense position-driven strategy reveals noisy vibrations and jerks because each motor always tries to get the desired position in the minimum time, whereas it would be better to efficiently adjust the velocity according to the whole target trajectory. The second strategy defines velocities dynamically along the trajectory as described above. Therefore, we need to sample the position regularly to adapt the velocity.
2.2 Modelling Robot Dynamics

The dynamics of a robot arm have a linear relationship to the inertial properties of the manipulator joints [5, 28]. In other words, for a specific context \( r \) they can be written in the form (1),

\[
\tau = Y(q, \dot{q}, \ddot{q}) \cdot \pi_r.
\]

where \( q, \dot{q}, \) and \( \ddot{q} \) are joint angles, velocities and accelerations respectively. This Equation 1, based on fundamentals of robot dynamics [12], splits the dynamics in two terms. \( Y(q, \dot{q}, \ddot{q}) \) is a term that depends on kinematics properties of the arm such as direction of the axis of rotation of joints, link lengths, and so on. Term \( \pi_r \) is a high dimensional vector containing all inertial parameters of all links of the arm [5].

Now, let us consider that the manipulated objects are attached at the end of the arm, so we model the dynamics of the arm as the manipulated object being the last link of the arm. Then, manipulating different objects is equivalent to changing the physical properties of the last link of the arm. The first term of the equation remains constant in different models. It means that all kinematic quantities of the arm remain the same between different contexts. Under this assumption, we could use a set of models with known inertial parameters to infer a predictive model (RBF) of dynamics [17, 27, 29] for any possible context. From another point of view, the previous dynamic Equation (1) could be written in the form (2),

\[
\tau = A(q)(\ddot{q}) + H(q, \dot{q}).
\]

Each dynamic term is non linear and coupled [53] where \( A(q) \) is the matrix of inertia, symmetric and positive and \( H(q, \dot{q}) \) includes Coriolis, gravitational, and centrifugal forces. \( A(q) \) could be inverted for each robot configuration. Approximating \( A(q) = \hat{A}(q) \) and \( H(q, \dot{q}) = \hat{H}(q, \dot{q}) \) we obtain (3),

\[
\hat{\tau} = \hat{A}(q) \cdot u + \hat{H}(q, \dot{q}).
\]

Where \( u \) is a new control vector. \( \hat{A}(q) \) and \( \hat{H}(q, \dot{q}) \) are estimations of the real robot terms. So we have the following equivalence relation (4),

\[
u = \ddot{q}.
\]

Component \( u \) is influenced only by a second order term, it depends on the joint variables \((q, \dot{q})\), independently from the movement of each joint. To sum up, we replaced the nonlinear and coupled dynamic expressions with a second order linear system of non coupled equations. The inputs to the RBF network are the positions and velocities of the robot joints, while the outputs are the accelerations. The input-output relationship is in accordance with the Equations of motion. In this work, we propose and show the RBF to be useful in approximating the unknown nonlinearities of the dynamical systems [1, 2, 26, 30, 45, 48, 49, 60, 65] through the control signal (4) found for the estimation method for the dynamics of the joints of a robot.
2.3 The RBF Based Modelling

The RBF function is defined as (5):

\[ z(x) = \phi(||x - \mu||). \]  

(5)

Where \( x \) is the \( n \)-dimensional input vector, \( \mu \) is an \( n \)-dimensional vector called centre of the RBF, \( || \cdot || \) is the Euclidean distance, and \( \phi \) is the RBF outline.

We built our model as a linear combination of \( N \) RBF functions with \( N \) centres. The RBF output is given by Equation (6):

\[ \hat{y}(x) = \sum_{j=1}^{N} \beta_j z_j(x). \]  

(6)

Where \( \beta_j \) is the weight of the \( j \)th radial function, centered at \( \mu \) and with an activation level \( z_j \).

RBF has seven inputs: two per each of the three motors in the robot arm (joints) and one for the label of the trajectory which is executed (\( T_r \)). Three inputs encode the difference between the current joint position and the next target position in each of the joints (\( D_1 \), \( D_2 \), and \( D_3 \)). The other three inputs (\( V_1 \), \( V_2 \), and \( V_3 \)) encode the velocity computed on the basis of the previous three inputs. We aim to model the acceleration sensory outputs (\( S_1 \) and \( S_2 \)) (\( S_3 \) does not provide further information than \( S_2 \) for the movement trajectory defined) and we also include one output (label) to classify different kinds of objects (OBJ). See the illustrative block in Fig. 3.b). Therefore, we use two RBF networks. The first one (see figure 3.a)) aims to approximate the acceleration sensor responses to the input motor commands. The second one (see figure 3.b) aims to discriminate object models (Classification task) using as inputs the motor commands and also the actual acceleration estimations given by the sensors [64]. During learning, we feed the inputs and actual outputs of the training set in the network [17, 29]. During the test stage, we evaluate the error obtained from the sensors [22] that indicates how accurate the acquired model is and therefore, how stable,
Fig. 4 Plots a) and c) show the motor commands (in joint coordinates) and plots b), d), and e) show the sensor responses during the movement. We compare the dense position-driven vs. the sparse velocity-driven strategies for the case related to object 0 (see the experimental results). In figure b) (Sensor 1) it can be clearly seen how the sensor responses are directly related with the motor command changes in plot a). The sensor responses, obtained using the sparse velocity-driven strategy, are more stable. Motor commands increasing the derivative of joint coordinates cause increments in sensor 1 response during a certain period, while motor commands decreasing the derivative of joint coordinates cause decrements in sensor 1 response, plot b). The piece of trajectory monitored by these plots is mainly related to sensor 1. It can be seen in the right plots d) and e) that the other sensors are not so sensitive to the motions in this example. Nevertheless, sparse velocity-driven control still leads to much more stable sensor responses. Dense position-driven control causes large vibrations that highly affect the capability of the neural network to approach the sensor response function.

3 Results

Studies have shown high correlations between the inertia tensor and various haptic properties like length, height, orientation of the object, and position of the hand grasp [15, 41, 64]. The inertia tensor has the central role to be the perceptual basis for mak-
ing haptic judgments of object properties. In order to describe the approach and the pursued movements (several repetitions) and to test different dynamic models of the whole system (robot arm with object) [6, 42], the robot hand manipulated objects with different weights, shapes and position of grasp. We acquired all the data from the position and accelerator sensors and we used them for the RBF off-line training. We considered the following cases of study:

0. No object;
1. Scissors (0.05 kg, 0.14 m);
2. Monkey wrench manipulated fixing the centre (0.07 kg, 0.16 m);
3. Monkey wrench manipulated fixing an extreme (0.07 kg, 0.16 m);
4. Two Allen keys (0.1 kg, 0.10 m).

We repeated manipulation experiments ten times for each case of study collecting sixty thousand data. We applied a cross-validation method defining different datasets for the training and test stages. We used the data of seven experiments out of ten for training and three for the test stage. We repeated this approach three times shuffling the experiments of the datasets for training and test. The neural network is built using the MBC model toolbox of MATLAB [35]. We chose the following algorithms: TrialWidths [40] for the width selection and StepItRols for the Lambda and centres selection [9]. We measured the performance calculating the RMSE in degrees/s² in the acceleration sensory outputs (S₁ and S₂).

We addressed a comparative study evaluating how accurately the neural network can abstract the dynamic model using a dense position-driven and a sparse velocity-driven schemes. Plots in Fig. 4 illustrate how the motor commands are highly related with specific sensor responses (mainly in sensor 1 along this piece of trajectory). Figs. 4.a and 4.c illustrate a piece of movement. Figs. 4.b, 4.d, and 4.e show that sparse velocity-driven control leads to much more stable sensor responses.

3.1 Abstracting the Dynamic Model during Manipulation

The goal of the RBF network is to model the sensory outputs given the motor commands along well defined trajectories (manipulating several objects). The purpose is to study how accurate models can be acquired if the neural network is trained with individual objects or with groups of "sets of objects" (as a single class). When we select the target "models" to abstract in the training stage with the neural network, we can define several target cases: 5 classes for abstracting single object models (Fig. 5.a), 10 classes for abstracting models for pairs of objects (Fig. 5.b), 10 classes for abstracting models of sets of "three objects" (Fig. 5.c), 5 classes for models of sets of "four objects" (Fig. 5.d) and finally, a single global model for the set of five objects (Fig. 5.e). The neural network, trained with data of two objects, presents the minimum of RMSE using 25 neurons in the hidden layer. For three objects, we used 66-80 neurons depending on the specific case, and for four objects, 80 neurons. But even with these larger numbers of neurons, the network was able to accurately classify the objects in the case of sets of two of them. During manipulation, if the model prediction significantly deviates from the actual sensory outputs, the system can as-
Dynamics Model Abstraction Scheme Using Radial Basis Functions

Fig. 5 RMSE values for predicted data of sensors $S_1$ and $S_2$ for sparse velocity-driven control scheme. The darker colour in the top of the columns indicates the increment of RMSE in the test stage. a) Using a different class (model) for each single object. b) Using a different class (model) for each set of two objects. Cases (objects): $A$ (0 and 1), $B$ (0 and 2), $C$ (0 and 3), $D$ (0 and 4), $E$ (1 and 2), $F$ (1 and 3), $G$ (1 and 4), $H$ (2 and 3), $I$ (2 and 4), and $J$ (3 and 4). c) Using a different class (model) for each set of three objects. Cases (objects): $A$ (0, 1, and 2), $B$ (0, 1, and 3), $C$ (0, 1, and 4), $D$ (0, 2, and 3), $E$ (0, 2, and 4), $F$ (0, 3, and 4), $G$ (1, 2, and 3), $H$ (objects 1, 2, and 4), $I$ (1, 3, and 4), and $J$ (2, 3, and 4). d) Using a different class for each set of four objects. Cases (objects): $A$ (0, 1, 2, and 4), $B$ (0, 1, 2, and 3), $C$ (0, 1, 3, and 4), $D$ (0, 2, 3, and 4), $E$ (1, 2, 3, and 4). e) Using a single class for the only set of five objects. In this case the increment of RMSE in the test stage is neglectable. Cases (objects): (0, 1, 2, 3, and 4).

Assume that the model being applied is incorrect and should proceed to "switch" to another model or to "learn modifications" on the current model. In this task, we evaluate the performance of the "abstraction process" calculating the RMSE between the model sensor response and the actual sensor response along the movement. Fig. 6 compares the performance achieved between the two control schemes under study for the cases of a non-fixed (Fig. 6.a) and a fixed robot (Fig. 6.b). Results indicate that the velocity driven strategy reduces the error to very low values in both cases. Results in Fig. 5 are obtained using a sparse velocity-driven control scheme. In Fig. 5.a, they show how the RBF achieves different performance when trying to abstract models of different objects. For object 1 and 3, both sensors lead to similar RMSE values as their dynamic models are similar. Objects 2 and 4 lead to higher RMSE values since they have more inertia than others. As we can see (comparing the results of Fig. 5.a and Fig. 5.b, if the network has to learn the dynamic model of more objects in the same class (shared model learning) at the same time, the error rate increases slightly.
Comparison between dense position-driven control scheme (column A) and the sparse velocity-driven control scheme (column B) for the case related to object 0. RMSE (degrees/s²) of predicted values of the sensors S1 and S2 using a sparse velocity-driven scheme (B) are always lower than the ones obtained using dense position-driven control (A). The sensor that suffers more significantly from jerks when using dense position-driven scheme is S2. The darker colour in the top of the columns indicates the increment of RMSE in the test stage. The test results are always slightly higher. The variance value obtained with the cross-validation method for the test results is also included. Figs. 6.a and 6.b are related to the cases of a non-fixed and a fixed robot respectively.

3.2 Classifying Objects during Manipulation

In Fig. 7, the discrimination power of the RBF based on the dynamic model itself is shown. In this case, we evaluate if the RBF is able to accurately distinguish among different objects assigning the correct labels (0, 1, 2, 3, and 4) by using only the inputs of the network (i.e. sensorimotor vectors). Evaluation of the classification performance is not straightforward and is based on the difference between the "discrimination threshold" (half of the distance between the closest target label values) and the RMSE value of the label output for a specific set of objects. We have chosen a single dimension distribution of the classes, therefore, distances between the different class labels are diverse. For instance, the "distance" between labels one and two is one, while the distance between labels one and four is three and the threshold is the half of that distance. For any set of two or five objects (i.e. A (0 and 1), B (1 and 3), and C (0, 1, 2, 3, and 4)) only one threshold value exists (Th (A) = 0.5, Th (B) = 1, and Th (C) = [0.5, 0.5, 0.5, 0.5]), while sets of three or four objects (i.e., D (1, 2, and 3), E (0, 2, and 3), and F (0, 1, 2, and 4), and G(1, 2, 3, and 4)) may have two threshold values (Th(D) = [0.5, 0.5], Th(E) = [1, 0.5], Th(F) = [0.5, 0.5, 1], and Th(G) = [0.5, 0.5, 0.5]). We see that the RBF is able to correctly classify the objects when using only two of them (values above zero) but the system becomes unreliable (negative values) when using three objects or more in the case of a standing robot (Fig. 7.a),
Bars represent the object discrimination power of the network (RBF) for all combinations of two, three, four, or five objects respectively (see paragraph 3 for object classification). If the RMSE (OBJ) value of the label output for a specific set of objects lies below the discrimination threshold, an accurate classification of objects can be achieved. Graphs show the result obtained with a non-fixed robot (Fig. 7.a) and with a fixed robot (Fig. 7.b), and each bar is associated to a combination of objects listed in the legend from left to right respectively. A combination with two thresholds (i.e., 0 3 4) has two bars and two labels (0 3 4a and 0 3 4b) and bars are joined together in the graph.

Fig. 7 Bars represent the object discrimination power of the network (RBF) for all combinations of two, three, four, or five objects respectively (see paragraph 3 for object classification). If the RMSE (OBJ) value of the label output for a specific set of objects lies below the discrimination threshold, an accurate classification of objects can be achieved. Graphs show the result obtained with a non-fixed robot (Fig. 7.a) and with a fixed robot (Fig. 7.b), and each bar is associated to a combination of objects listed in the legend from left to right respectively. A combination with two thresholds (i.e., 0 3 4) has two bars and two labels (0 3 4a and 0 3 4b) and bars are joined together in the graph.

Fig. 8 Mean of RMSE for objects (OBJ) using the sparse velocity-driven control scheme.

while the RBF always correctly classifies the objects when the robot is fixed to the ground (Fig. 7.b).

3.3 Accurate Manipulation Strategy: Classification with Custom Control

We have abstracted dynamic models from objects and can use them for accurate control (Fig. 9).

a. **Direct control.** We control different objects directly. During manipulation, the sensor responses are compared with a "shared model" to evaluate the performance. We measure the RMSE comparing the global model output (model of the sensor outputs of two different manipulated objects) and the actual sensor outputs.

b. **Classification with control.** Since we are able to accurately distinguish between two objects (Fig. 7.a), we select the corresponding individual model from the RBF network (see 3.a) to obtain the acceleration values of the object being manipulated instead of using a shared model with the data of the two objects. In fact, the indi-
individual models of the five objects were previously learnt. The RMSE is always lower when single object models are used (Fig. 9) (after an object classification stage). Direct control using a global (shared) model achieves a lower performance because the applied model does not take into account the singularities of each of the individual models.

The classification with control scheme is feasible when dealing with objects easy to be discriminated and the actual trajectory of the movement follows more reliably the single object model than a "shared model". In Fig. 9, the RMSE is plotted when comparing sensor model response and the actual sensor responses. Fig. 8 shows the accuracy of the classification task when training the network with different numbers of objects (classification errors along the trajectory). Nevertheless, we have shown (Fig. 7) that with the objects used in our study, the network can only reliably classify the object (along the whole trajectory) when we focus on distinguishing between two objects.

4 Conclusions

The first conclusion of the presented work is that the applied control scheme is critical to facilitate reliable dynamic model abstraction. In fact, the dynamic models or computed torques are used for high performance and compliant robot control in terms of precision and energy-efficiency [39, 43].

We have compared a dense position-driven vs. a sparse velocity-driven control strategy. The second scheme leads to smoother movements when manipulating different
objects. This produces more stable sensor responses which allow to model the dynamics of the movement. As model abstraction engine, we have used a well known RBF network (widely used for function approximation tasks) and we compared its performance for abstracting dynamics models of the different objects using a sparse velocity-driven scheme (Fig. 5.a).

In a second part of the work, we have evaluated if the object classification is feasible. More concretely, we have checked out if we can perform it using sensorimotor complexes (motor commands, velocities and local accelerations) obtained during object manipulation (exploration task) as inputs. Figure 7.a shows that only the discrimination between any pair of two objects was possible. Robots with low gains will produce different actual movements when manipulating different objects, because these objects affect significantly their dynamic model. If the robot is controlled with high gains, the differences in the dynamic model of the robot are compensated by the high control gains. If the robot is tightly fixed but is controlled with low gains the actual movement (sensorimotor representation of an object manipulation) is different for each object (Fig. 7.b). If the robot is slightly fixed, the control commands produce vibrations that perturb the dynamic model and therefore, the different sensorimotor representations are highly affected by these noisy artefacts (motor command driven vibrations) (see figure 7.a).

Finally, by assuming that we are able to classify objects during manipulation and to switch to the best dynamic model, we have evaluated the accuracy when comparing the movement’s actual dynamics (actual sensor responses) with the abstracted ones. Furthermore, we have evaluated the gain in performance if we compare it with a specific dynamic model instead of a ”global” dynamics model. We have evaluated the impact of this two-step control strategy obtaining an improvement of 30% (Fig. 9) in predicting the sensor outputs along the trajectory. This abstracted model can be used in predicted control strategies or for stabilisation purposes.

Summarising, our results prove that the presented robot and artificial neural network can abstract dynamic models of objects within the stream sensorimotor primitives during manipulation tasks. One of the most important results of the work shown in Figs. 6.a (non-fixed robot) and 6.b (fixed robot) indicates that robot platforms can highly benefit from non-rigid sparse velocity-driven control scheme. The high error obtained with the dense position-driven control scheme (Fig. 6.a) is caused by vibrations as the slight fixation of the robot to the ground is only supported by its own weight. Furthermore, when we try to abstract a movement model, the performance achieved by the neural network (RBF) also represents a measurement of the stability of the movements (the repeatability of such trajectories when applying different control schemes).

Acknowledgements  This work has been supported in part by the EU grant SENSOPAC (FP6-IST-028056) and by the Spanish National Grants DPI-2004-07032 and TEC2010-15396.
References

symposium on learning and adaptive behavior in robotic systems


