Proceedings of the 1st IEEE Latin American Robotics Symposium – LARS 2004
Mexico city, Mexico. October 28 – 29, 2004.

On-line Learning of an Object Manipulation Behavior for Legged Robots

Ximena Olivares, Javier Ruiz-del-Solar, *Senior Member, IEEE*, and Juan Cristóbal Zagal

Abstract - **In this work we address on-line learning of behaviors for legged robots. We present a methodology for generating and adapting object manipulation behaviors using reinforcement learning. An AIBO four-legged robot carries cylinders to a desired position by pushing them with its frontal legs. The movement of a cylinder depends on where the robot exerts forces on it, and upon several physical characteristics (geometry, weight, floor roughness, etc.). Initially the robot doesn't know this dynamics and it learns it from its own experience. By continuously evaluating the action that minimizes the time to achieve the goal, and using a global configuration fitness function, the robots can complete the target configuration, and learn the tradeoff between pushing a given cylinder on a given position and moving itself to a new pushing position. Successful results on the application of this methodology are presented.**

Index terms - **Legged Robots, Reinforcement Learning, On-line learning.**

I. INTRODUCTION

Providing of learning and self-adaptation capabilities to real autonomous robots is becoming one of the major subjects of research in robotics. The engineering design process encounters a fundamental limit when trying to model the fine and tight interaction that exists between a robot and its dynamic environment [3]. The designer might introduce important bias when generating robot controllers affecting the optimality of resulting behaviors. Allowing a robot to learn from its own experience, acquired when interacting with the environment, appears to be an important requirement to produce fully autonomous agents. It is clear that both, the robot and its environment, suffer of changes in their physical properties over time, consequently affecting their corresponding models. Since these changes might occur even at short time scales, there is a fundamental need for using on-line self-adaptation methods.

Approaches such as reinforcement learning have shown to be well suited for the on-line learning of robots behaviors. There are studies in the literature were reinforcement learning shows to be a good choice for adapting behaviors with data obtained from real robots [1]. In particular the approach of *Q–learning* has some advantages over traditional algorithms [2]. Most of the experiments that have been published are based on the use of wheeled robotic platforms, and the addressed problems usually correspond to simple navigational tasks. We believe that it is important to explore the extents of similar approaches using more complex robotic platforms, such as limbed robots, performing more challenging tasks such as object manipulation.

In this paper we address the problem of learning to manipulate objects with a complex robotic platform; a legged robot receives the task of building a given spatial configuration using plastic cylinders. These cylinders are initially lying on a carpet on a random configuration, and the robot should move them until a desired configuration is formed. The target configuration corresponds to a human-meaningful entity, such as a triangle in this case. The robot will learn to construct configurations of cylinders by continuously choosing the right behavior to execute among a given set of choices, aiming at reaching a target configuration in the minimum time. If the robot performs badly, scattering the cylinders for example, it realizes that changes are necessary on its current behavior. The robot continuously decides which action to take among a set of choices. It internally predicts the resulting action profit. The robot uses the error between the expected profit and the obtained profit for modifying its internal representation of the world, and we therefore say that it learns. The robot learns also a dynamic model of its interaction with the object, which is expressed as a homogeneous transformation of the cylinders' pose related to a set of actions. The task of moving cylinders implies also to either keep on pushing a given cylinder in some place or moving to another pushing place. Deciding whether to push or change to another position is also a matter of adaptation. We let the robot to continuously evaluate the tradeoff between pushing the object and moving itself to another position.

This paper is organized as follows. In section II is described the proposed methodology for generating and adapting object manipulation behaviors. Successful results on its application are presented in section III. Finally, in section IV some conclusions of this work are given.

Department of Electrical Engineering, Universidad de Chile. E-Mail: {xolivare,jruizd,jzagal}@ing.uchile.cl.
This research was funded by the research project CONICYT/BMBF 2003 – 4 – 146.

© 2004 IEEE

II. Learning to Configure Objects

A. Experimental Settings

We develop our experiments in a rectangular arena, which also corresponds to a standard robotic soccer field, of size 3.1 x 4.6 m. In order to ensure high levels of accuracy for a first version of our experiments, we are currently obtaining the pose of the robots and cylinders by means of a global vision system. For this, a color camera is mounted right over the arena at 3.5 meters high; this camera is connected to a host computer that performs image processing of the image data. The robots are marked with a pair of colored circles allowing for a simple visual detection of their pose. We are using a set of three plastic cylinders of 80 cm length and 6 cm of diameter. Each cylinder has a particular color which allows for their simple detection and poses estimation. We have adjusted the weight of these cylinders by placing a piece of metal in the center of them. This allows the robot to rotate a cylinder by pushing in one extreme of it. The communication between the robot and the host computer is performed by means of a wireless network. Figure 1 shows an illustration of these components.

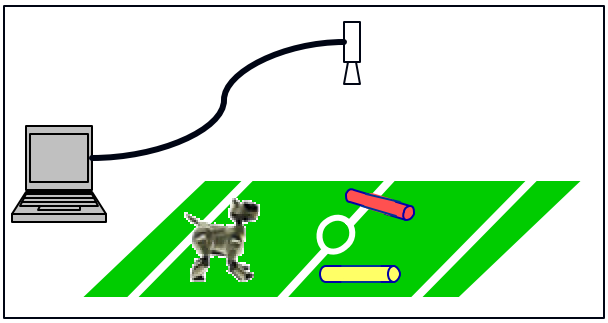


Figure 1: Illustration of the experimental settings.

B. General Idea

The robot will learn to construct configurations of cylinders by continuously choosing the right behavior to execute among a given set of choices, aiming at reaching a target configuration in the shortest time. Behaviors can be either to exert forces over one of the cylinders or to move to a different location. We measure the similarity of the current configuration with respect to the target configuration (a triangle in this case) by means of a scalar fitness function. Since motion of the robot involves the consumption of time, the selection of behaviors should satisfy a compromise between the improvement in the cylinders' configuration and the related time consumption.

C. Software Architecture

In figure 2 is shown a block diagram of the main modules of the software architecture. The *Actuation* module is in charge of controlling the robot motors, according to instructions provided by the *Behaviors* module. *Actuation* is part of the U-Chile1 software control library [5]. The *Global Vision* module is in charge of the recognition and localization of the relevant objects (robot and cylinders). It performs this task using color segmentation (the vision module of U-Chile1 is employed) and the Hough transform for determining the cylinders orientation and position. Finally, the *Behaviors* module is in charge of solving the object manipulation task by performing the high-level control of the robot. Among other functionalities this module includes behaviors' selection, trajectory generation, and learning. It will be extensively described in the next sections.

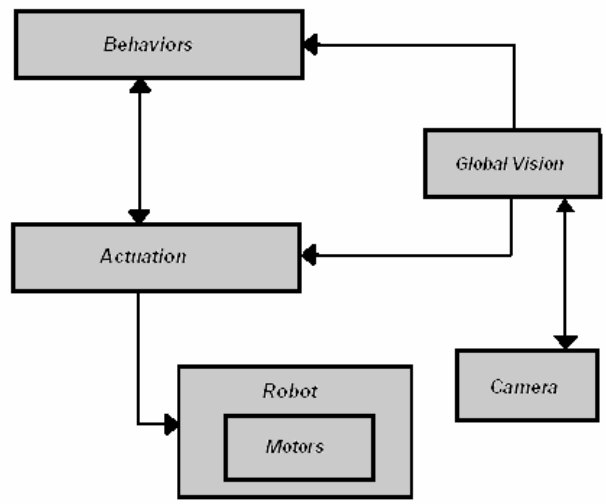


Figure 2: Main software modules.

D. Basic Definitions

The robot should perform certain *actions* over some specific portion of the cylinder body for moving it. These actions correspond to the robot pushing the object with its frontal legs. For this experiment we define a set of eight different possible actions A_j, each consists on pushing the object on the specific point j on the cylinder. Figure 3 shows the corresponding locations of these points with respect to the cylinder. It is expected that the cylinder simply translates after applying actions towards the cylinder center of mass, while actions pointing in other directions will produce a rotation of the cylinder.

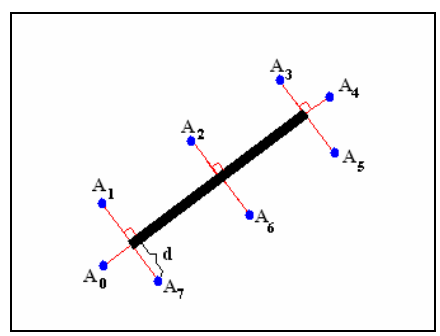


Figure 3: Eight different actions A_j that the robot is allowed to exert over the cylinder, and their corresponding application points.

We consider two types of *simple behaviors*. The so-called *behavior1* corresponds to the robot moving from the

actual robot pose P_R to a target pose $P_{b_{ij}}$, which corresponds to the position of the point *j* of the cylinder *i*. In order to execute this behavior a trajectory must be generated, taking into account possible obstacles. The so-called *behavior2* corresponds to executing a specific action A_j over the cylinder b_i (i.e. in target position b_{ij}). We consider *compound behaviors* that correspond to combinations of the simple behaviors defined previously. We define the compound behavior $C_{ij}(P_R, b_i, A_j, B)$ as going from the actual position of the robot P_R to the point *j* in the cylinder b_i, and performing the action A_j over b_i. It is important to consider the global configuration of the cylinders, denoted by B, in the process of determining the global gain or profit, obtained by executing this compound behavior.

E. Trajectory Generation

For planning the path that the robot should follow to move from one point to another, while avoiding obstacles, we will use the virtual force field method (VFF), as proposed in [4]. The VFF method uses a two-dimensional grid that divides the arena in multiple cells. The obstacles in the arena exert a virtual repulsive force towards the robot, while the target exerts a virtual attractive force that "pulls" the robot to the target. The resulting force corresponds to adding the repulsive and attractive forces. This force tells the direction the robot must follow.

Our implementation is based on creating a grid on the image obtained from the camera. Since the global vision module is able to detect different objects on the image and determine their position, orientation and the pixels that an obstacle contains, it is possible to go along the grid and assign a certainty value to each cell. This certainty value indicates how much of the cell is covered by the obstacle (see figure 4). After determining the certainty value, we calculate the repulsive force that each cell of the grid exerts on the robot. This value is obtained using the following expression

$$F_{ij} = \frac{F_{rep} WC_{ij}}{d^2(i,j)} \left(\frac{x_i - x_0}{d(i,j)} \hat{x} + \frac{y_i - y_0}{d(i,j)} \hat{y} \right), \qquad (1)$$

where, F_{rep} is a repulsive force constant; $d(i,j)$ is the distance between the cell(i,j) and the robot; C_{ij} correspond to the certainty value of cell(i,j); W is the width of the robot; (x_0, y_0) are the current robot coordinates; and (x_i, y_i) are the coordinates of cell(i,j).

The attractive force exerted by the target corresponds to

$$F_t = F_A \left(\frac{x_t - x_0}{d_t} \hat{x} + \frac{y_t - y_0}{d_t} \hat{y} \right), \qquad (2)$$

where F_A is the attractive force; d_t is the distance between the robot and the target; (x_t, y_t) are the target coordinates.

Finally the resultant force vector R is computed as follows

$$R = F_R + F_t, \qquad (3)$$

where F_R corresponds of the sum of all virtual repulsive forces.

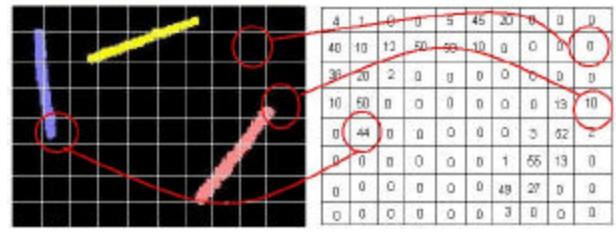
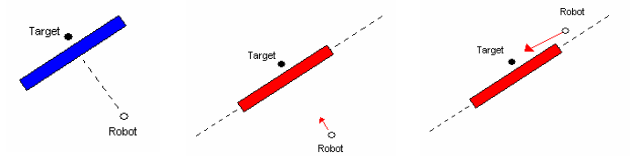


Figure 4: Certainty grid. Empty cells obtain low certainty values.



(a) (b) (c)

Figure 5: The target and the robot are in different sides of an obstacle (a). The robot starts moving using a small attractive force (b). After a while the robot and the target are in the same side of the obstacle and the robot uses a larger attractive force.

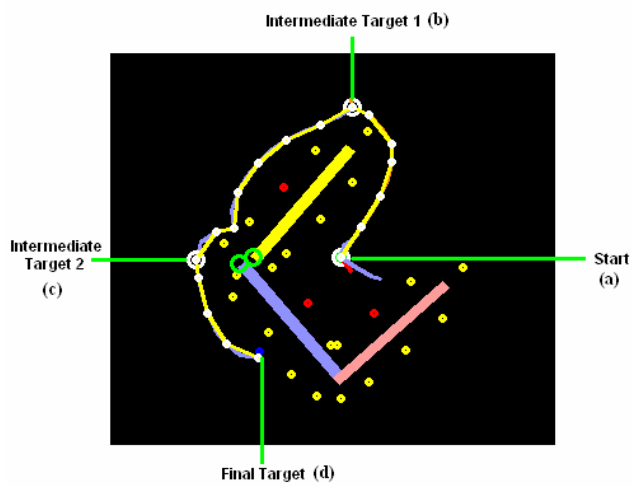


Figure 6: Trajectory generation using intermediate targets.

Since it is important that the robot moves inside the arena, our implementation considers that the borders of the image exert a repulsive force towards the robot. This repulsive force is obtained using (1), but with a smaller repulsive force constant ($F_{repBorder}$), since it is more important that the robot keeps away from the obstacles than from the borders. Another important consideration is that in some cases, which depend on the obstacle and target relative positions, the robot may get stuck on a local minimum, or the robot trajectory can produces oscillations when approaching the target. This situation is produced, for example, when the robot and the target are in different sides of an obstacle (see figure 5 (a)).

In order to overcome this problem we use attractive forces of different magnitudes. A small attractive force F_{AWeak} is used when the target and the robot are in different sides of the obstacles (figure 5 (b)). When the obstacle and the target are on the same side, we use a larger attractive force $F_{AStrong}$ (figure 5 (c)). However, this criteria is not

enough for avoiding local minima, and intermediate targets need to be defined. An intermediate target is chosen by first selecting the closest cylinder to the robot, among the ones that produces the local minimum; then, both ends of this cylinder are analyzed and the one that allows the robot to get out of the local minimum is chosen as intermediate target. In figure 6 is shown a real complex situation in which, after starting in a local minimum (a), the robot should select two intermediate target, (b) and (c). When moving from (a) to (b) F_{AWeak} is used, while when moving from (b) to (c), and from (c) to (d), $F_{AStrong}$ is employed.

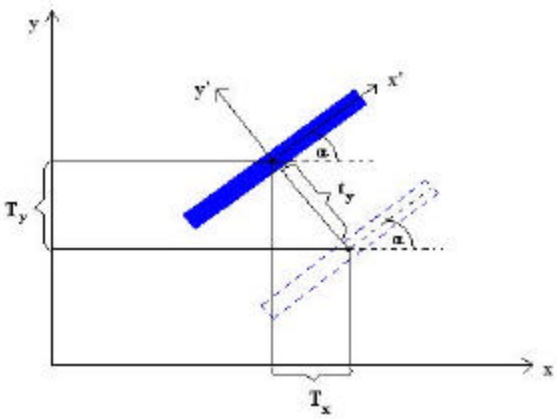


Figure 7. Global (x,y) and local (x',y') coordinate systems.

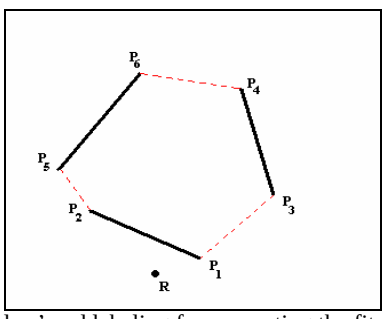


Figure 8. Cylinders' end labeling for computing the fitness value.

F. Kinematical Model

Once the robot exerts an action over the cylinder, its corresponding effect, in terms of cylinder displacements, is modeled by means of a homogeneous transformation. We call this the kinematics model of the action:

$$\begin{pmatrix} x_f \\ y_f \\ 1 \end{pmatrix} = \begin{pmatrix} \cos q & -\sin q & p_x(1-\cos q) + p_y \sin q + T_x \\ \sin q & \cos q & p_y(1-\cos q) - p_x \sin q + T_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \quad (4)$$

where (x_f, y_f) and (x_0, y_0) correspond to the final and the current position of the cylinder, respectively; (p_x, p_y) is the position of the cylinder center of mass; q is the rotation angle; T_x and T_y are the translation parameters in the global coordinate system. They are related to the translation parameters in the local coordinate system, t_x and t_y, as (see figure 7):

$$\begin{aligned} T_x &= t_x \cos a - t_y \sin a \\ T_y &= t_y \cos a + t_x \sin a \end{aligned} \quad (5)$$

with a the angle between both coordinate systems.

Through reinforced learning the parameters t_x, t_y and a are continuously adjusted.

G. Configuration Fitness Function

This function gives a measure of how similar is the current cylinder configuration with respect to the desired target configuration. In order to evaluate the configuration fitness function, it is necessary labeling the ends of each cylinder firstly. Assuming that we are using three cylinders, as figure 8 shows, the algorithm will be:

1. Labeling of each cylinder end (CE) depending on the robot position:

- P1 is assigned to the closest CE to the robot position. P2 will correspond to the other CE of this cylinder.
- The CE closer to P1 and the CE closer to P2 are determined. Two cases:

i. If both CEs are in different cylinders:
   - P3 corresponds to the (unlabelled) CE closer to P1, and P4 corresponds to the other CE of that cylinder.
   - P5 corresponds to the CE closer to P2, and P6 correspond to the other CE of this cylinder.

ii. If both CEs are in the same cylinder:
   - P3 corresponds to the (unlabelled) CE closer to P1, and P4 corresponds to the other CE of that cylinder.
   - P5 and P6 correspond to the CEs of the remaining cylinder.

2. Evaluation of the configuration fitness function, using the distances $d(P_1,P_3)$, $d(P_2,P_5)$ and $d(P_4,P_6)$, as:

$$f(B) = \frac{1}{d(P_1,P_3) + d(P_2,P_5) + d(P_4,P_6)} \quad (6)$$

H. Gain Function

We introduced a gain or profit function, in order to be able to choose between a set of possible compound behaviors (simple behaviors are never chosen alone). The best behavior to perform is the one with the highest estimated profit. The global gain function involves the time consumed while performing the behavior, and how this behavior will help in the achievement of the desired task. Thus, the estimated global gain of the compound behavior $C_{ij}(P_R, b_i, A_j, B)$ is the difference between the value of the configuration fitness of the current configuration of the cylinders B and the configuration fitness evaluated in the estimated configuration of the cylinders $\tilde{B}$, after executing the behavior, divided by the time required for carrying out this behavior.

$$Gain = \frac{\left| f(\tilde{B})_k - f(B)_{k-1} \right|}{\Delta \tilde{t}_k} \quad (7)$$

$$\Delta \tilde{t}_k = b \cdot d_T(P_R, P_{b_{ij}}) + t_{act} \quad (8)$$

d_T corresponds to the shortest trajectory (calculated using

the VFF method) from the robot current position to the target located in the point j in the cylinder b_i. t_{act} corresponds to the time that the robot employs in executing a basic action A_j. b is a parameter to be learn, which allows adjusting the relationship between the time of traveling and the time of performing an action.

I. *On-line Learning Algorithm*

We present an algorithm for performing the proposed on-line learning process. The robot continuously predicts the gain function value that is related to the execution of each possible compound behavior, and then it executes the one having the largest gain. After behavior execution, the robot is able to evaluate the resulting error between the predicted cylinders' configuration and the real one. By using this error value the robot is able to modify its internal kinematical model, i.e. the parameters which define each homogeneous transformation (t_x,t_y,θ). For adapting b, the error between the estimated time and the time employed for completing the behavior is employed. The delta rule is used as learning algorithm in both cases (var = var + learning_rate*error).

Figure 9 shows the pseudo code of this algorithm. The cylinders configuration is expressed as a set of cylinder poses (p_x, p_y, a). The function *executeBehavior* corresponds to the execution of the compound behavior $C_{ij}(P_R, b_i, A_j, B)$.

```
maxGain = SMALL_CONSTANT;
fitness = calculateFitness(B_t0);  // B_t0 initial configuration
while(task_non_Completed){
   P_R = currentRobotPosition();
   //Estimate the best Aj to be applied in bi
   for(bi: i=1 to NUMBER_OF_CYLINDERS){
      for(Aj: j=1 to NUMER_OF_ACTIONS){
         trajectory = computeTrayectory(P_R,bi,Aj);
         trajectoryLength = computeTrajectoryLenght(trajectory);
         (t_x,t_y,θ)=computeTraslationRotationParameters(Aj);
         B̃_{t+1} = estimateNextConfiguracion(t_x,t_y,θ);
         estimatedfitness=calculateFitness(B̃_{t+1});
         estimatedTime= β*trajectoryLength + T_ACTION;
         gain = (estimatedfitness –fitness)/estimatedTime;
         if (gain > maxGain){
            maxGain = gain;
            bi* = bi; Aj* = Aj; trajectory* = trajectory;
            B_{t+1}* = B̃_{t+1};
         }
      }
   }
   //After execution B_{t+1}, the real configuration, is return
   B_{t+1} = executeBehavior(bi*,Aj*,trajectory*);
   errorConf = calculateErrorConf(B_{t+1}*, B_{t+1});
   adjustParameters(t_x,t_y,θ,errorConf);
   errorTime = calculateErrorTime(elapsedTime,estimatedTime);
   adjustParameters(β,errorTime);
   fitness = calculateFitness(B_{t+1});
}
```

Figure 9: Pseudo-code version of the algorithm

III. EXPERIMENTAL RESULTS

We implemented the on-line learning methodology described in section II in two steps. In the first step all algorithms were tested in a simulator. This simulator takes real images from the robot arena, and can simulate the movement of the robot and the manipulation of the cylinders. The dynamic of the simulator is very simple, because the robot is managed as a single point, trajectory simulation in these conditions is very easy to implement. The use of this simulator allowed us to correct some small algorithms' errors, to develop the trajectory generation algorithm described in section II, and to set up some of the algorithm parameters, as for example $F_{AWeak} = 20$, $F_{AStrong} = 70$, $F_{rep} = 7$ and $F_{repBorder} = 5$.

After all single functionalities were tested in the simulator, the robot controller was easily transferred to the reality. In reality, the robot learn to adapt its reality dependant constants, i.e. the parameters defining the kinematical model of the cylinders, (t_x,t_y,θ), as well as b. As mentioned, the delta rule was employed as learning algorithm. Figure 10 shows the online learning of b, as an example of the obtained results.

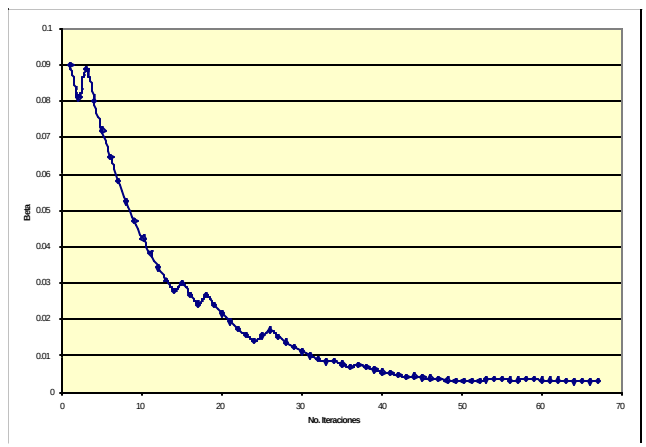


Figure 10: Learning curve of the β parameter.

In reality the robot is not anymore a single point, but a real object with four legs, each one having 3 degrees of freedom. Therefore the movement control of the robot is not a simple task. As mention in section II.C, we used the *Actuation* module of our U-Chile 1 software control library [5] for controlling the robot motors (see the software architecture block diagram in figure 2). For moving the robot the following procedure was employed. In each position is calculated the corresponding displacement vector and the robot orientation change, which is necessary to reach the next position (trajectory generation algorithm). Then, the *Actuation* module receives these values and it executes the corresponding action. For each position, the robot orientation is selected as the direction pointing towards the cylinder, but orthogonal to the cylinder axis. If the robot is rounding some cylinder extreme, the robot points towards the cylinder center. Figure 11 shows an example of a robot completing a triangle configuration. In

(a) is shown the generated trajectory for going from a given pushing position to the next one. Then, in (b), is shown the robot, ready for its next pushing action. It can be observed than in this case the final position of the robot is orthogonal to the cylinder to be pushed.

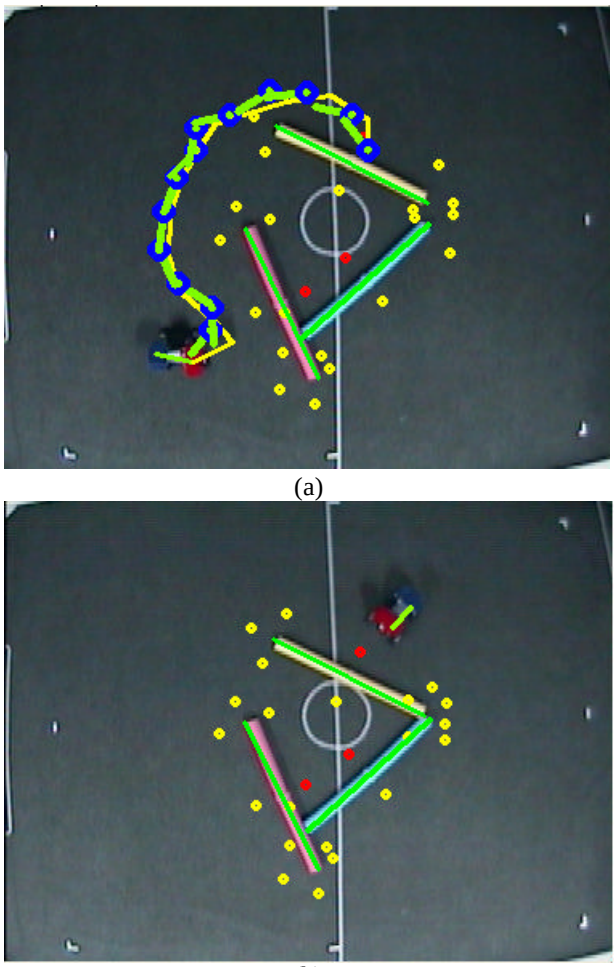


Figure 11: Robot by completing a triangle configuration. (a) Generated trajectory for going from a given pushing position to the next one. (b) The robot is ready for its next pushing action.

Many triangle configuration experiments were performed with the robots. In each case the initial position of the cylinders and the robot was randomly selected. The time required by the robot for completing the task was in the range 6 – 9 minutes. In a couple of special cases the task completion took more than 10 minutes, mainly because of the robot get stuck inside the cylinders, and by going out it scatter the cylinders. As an example of a typical evolution of the configuration fitness value, in figure 12 is shown a typical learning sequence. It can be observed that sometimes the fitness value decreases. The main reason of that is the accidentally scattering of the cylinders produced by the robot.

IV. Conclusions and Projections

We have presented a methodology for learning to configure objects with legged robots. The robot continuously predicts the gain function value that is related to the execution of all possible behaviors, and then it executes the behavior having the largest gain. It is important to notice that the adaptation of the robot is expressed in variations on the internal representation that the robot possesses of the world. This representation is expressed in the form of a kinematical model and a value which tells about the convenience of moving, with respect to maintaining the current action behavior.

As a future work we want to explore three issues: (i) the replacement of the global vision by a local vision system using the AIBO camera, (ii) the use of a neural network for learning the kinematical model of the cylinders, and (iii) the use of several robots for exploring a collaborative solution of the described task.

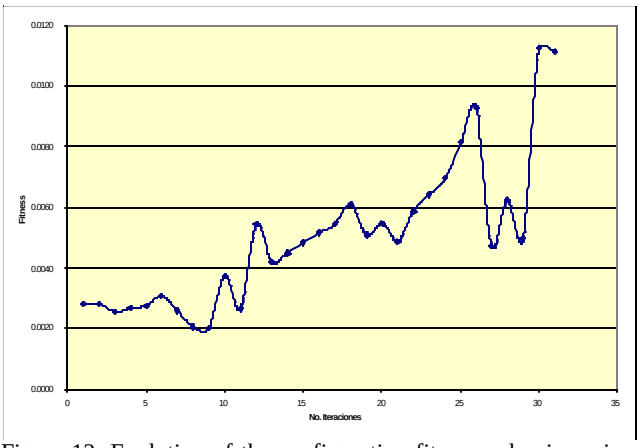


Figure 12: Evolution of the configuration fitness value in a given learning sequence.

References

[1] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning", *Machine Learning*, 23, 279 – 303 (1996).

[2] M. Carreras, P. Ridao and A. El-Fakdi, "Semi-Online Neural-Q- learning for Real-time Robot Learning". *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, USA, 2003.

[3] S. Nolfy, D. Floreano, "Evolutionary Robotics The Biology, Intelligence, and Technology of Self-Organizing Machines", Cambridge, MA, MIT Press 2000.

[4] Y.Roren and J.Borenstein, "Potencial Field Methods and Their Inherent Limitations for Mobile Robot Navigation", *Proc. of the IEEE Conf. on Robotics and Automation*, 1398-1404. Sacramento, California, April 7-12, 1991.

[5] J. Ruiz-del-Solar, P. Vallejos, J. Zagal, R. Lastra, G. Castro, C. Gortaris, I. Sarmiento, "UChile1 2004 Team Description Paper", *Proc. of the RoboCup 2004 Symposium*, July 4 – 5, Lisbon, Portugal.