We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



6

Robotic Grasping: A Generic Neural Network Architecture

Nasser Rezzoug & Philippe Gorce
LESP EA 3162 - Université du Sud Toulon - Var
France

1. Introduction

Reproducing human dexterity and flexibility in unknown environments is one of the major challenges of robotics. Among the issues related to the use of robots with artificial hands of varying complexity, the definition of their kinematical configuration when grasping an object is one of the most difficult problems. Indeed, it necessitates the consideration of a large number of constraints related to the structure of the hand as well as the object, the requirements of the task and the state of the environment. In this frame, sensing capability is crucial to obtain information about the robot state and the quality of sensory signals that may be noisy or of low precision has to be considered to control efficiently a robotic grasping task.

In the literature, several paradigms have been proposed to perform grasp planning. Mainly, they can be divided in three categories: knowledge based grasp planning, geometric grasp planning and sensory driven or learning based grasp planning.

Knowledge based grasp planning refers to techniques where posture definition is made according to direct or indirect knowledge in human prehension. In fact, in order to better understand the different parameters involved during grasping, several studies have been conducted to classify human prehension. Their main principle is that prototypical hand shapes are used to perform classes of tasks according to particular hand, environment and task related constraints (Napier, 1956; Iberall et al., 1986; Cutkosky, 1987; Tomovic et al., 1987; Bekey et al., 1993; Iberall, 1997; Saito & Nagata, 1999). These classifications have been intensively used to propose solutions in the robotics field, especially in the frame of artificial intelligence and expert systems (Pook & Ballard, 1993; Kang & Ikeuchi, 1997;). The concept of "assembly plan from observation" states that a robot can perform complicated grasps and manipulations by taking advantage of the human dexterity. Thus, optimal solutions can be obtained by the direct observation of human behavior and the extraction of the appropriate movement features (Jeannerod, 1984; Bard et al., 1991; Lee & Xu, 1996; Becker et al., 1999). These methods are very effective but they need sophisticated motion sensing devices such as datagloves. Moreover, the robot can learn only what is demonstrated and it is difficult to generalize from one task to another. Also, an alternative way to define hand posture is to use the concept of preshaping coming from experimental studies on prehension. In fact, it has been demonstrated (Wren & Fischer, 1995) that, prior to grasping, there is an anticipation of hand opening and orientation which is a function of object and task constraints. Therefore, the grasp planning problem is dramatically simplified by the choice of a finite number of hand preshapes. Then, hand closure schemes are defined in order to achieve grasp from the adequate preshape (Miller and al., 2003).

Open Access Database www.i-techonline.com

Source: Mobile Robots Towards New Applications, ISBN 3-86611-314-5, Edited by Aleksandar Lazinica, pp. 784, ARS/plV, Germany, December 2006

Geometric contact level grasp synthesis algorithms have been developed to find an optimal set of contacts according to grasp quality measures like force and form closure (Kerr & Roth, 1986; Nguyen, 1986; Ferrari & Canny, 1992; Gorce & Fontaine, 1994; Mirtich & Canny, 1994, Miller & Allen, 1999). In this case, the hand is represented by a set of contacts and while these algorithms are applicable to any dexterous hand they leave the problem of finding a suitable hand configuration open. Optimization based methods are then proposed to define the hand kinematical configuration (Borst et al., 2002; Guan & Zhang, 2003). The main drawback of such approaches is that they often rely on a precise knowledge of the task to be performed.

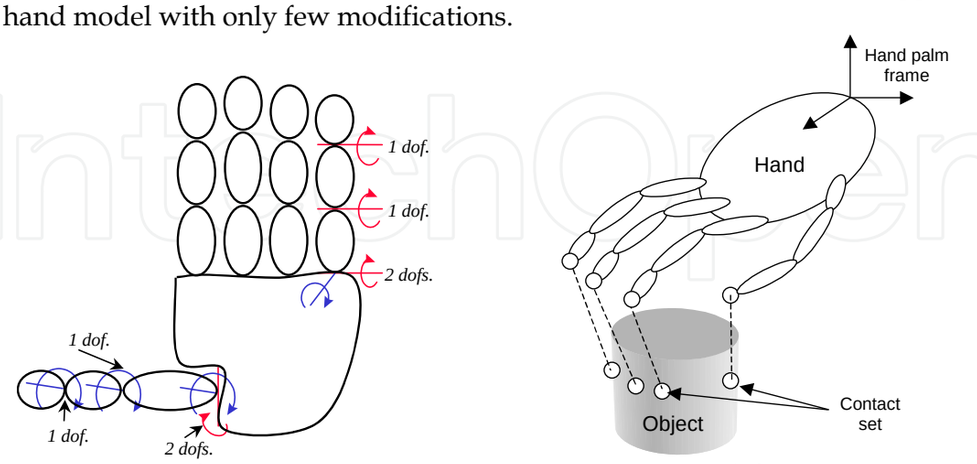
Sensory driven grasp planning tries to solve this difficulty by incrementally constructing a model of the task by exploration and learning (Coehlo & Grupen, 1997; Wheeler et al., 2002; Grupen & Coehlo, 2002; Pelossof and al., 2004). In this frame, neural network based approaches are used to learn the underlying rules adopted during grasping. For example, (Kurperstein & Rubinstein, 1989) developed a neural network based scheme called INFANT in order to perform grasping tasks with a 5 degrees of freedom gripper. (Uno et al., 1995) proposed a neural based computational scheme able to integrate an internal representation of objects as well as the corresponding hand shape. (Taha et al., 1997) developed a model of preshaping of a planar hand model (with seven degrees of freedom) for circular and rectangular objects. (Moussa & Kamel, 1998) proposed a computational architecture able to learn grasping rules called "generic grasping functions". This representation treats grasping knowledge as a mapping between two reference frames: the object body attached coordinate frame, and the gripper body attached frame. This paradigm is applied to a very simple mechanism (single jaw gripper) and can not be applied to a multifingered hand where the mapping between the hand and the object frames may not be sufficient.

In this chapter, a new method based on neural networks is proposed. It allows learning multichain redundant structures configuration during grasping in the presence of noise and uncertainty as well as unknown obstacles. Theses structures include serial manipulators equipped with a single jaw gripper or a multifingered hand. The developed model is composed of one generic module whose inputs and outputs vary according to the considered robot. A second module carries out the learning of the fingers inverse kinematics when a multifingered hand is used. It is based on a modular architecture composed of several neural networks. Using reinforcement learning, the main neural network based module optimizes the position and orientation of the grasping device taking into account noisy sensing information. Working together these two modules exchange information to define the complete robot configuration. The main interest of this architecture is that it is generic and can define the posture of hand, arm+hand or non anthropomorphic robots (e.g. the MANUS: a robotic assistant for the disabled) with noise or obstacles with few modifications. In order to illustrate the capabilities of the proposed model, simulation results are proposed using different situations considering different robot kinematical structures.

This chapter is organized as following. In section 2, we describe the model architecture. The retained scheme to learn the inverse kinematics of the fingers is described in section 3. Then, section 4 is focused on the procedure used to define and optimize robot configuration. Finally in the remaining sections, simulation results are presented to demonstrate the efficiency of the proposed tools with considering the anthropomorphic hand only (section 5), the hand + arm model (section 6) and a non anthropomorphic robot (section 7).

2. Model presentation

The adopted approach will be explained considering the hand palm configuration only (i.e. position and orientation). The resolution process is almost the same when applied to the arm + hand model with only few modifications.



a/ b/

Fig. 1. a/ 3D model of the hand, b/ Problem statement

2.1 Hand model

The hand model is composed of five articulated rigid chains representing the fingers. They are connected to a common body representing the palm (Fig. 1a/). Each finger has three links connected by three joints with a total of four degrees of freedom. The first joint of each finger has two degrees of freedom that allow to simulate the flexion-extension and abduction-adduction movements. The two other joints have one degree of freedom representing joint flexion. The complete model has 26 degrees of freedom. They are divided in two groups: 20 for the fingers joint angles and 6 for the hand palm position and orientation relative to a global frame. This architecture is generic and can be associated with many robotic hand designs directly as is or with simplifications with fewer fingers or fewer degrees of freedom. In the remainder of this article, the terms hand posture or grasping posture are used to define the numeric value of the set of parameters describing the kinematical configuration of the hand model during grasping.

2.2 Problem definition and chosen hypotheses

The definition of the problem is: *Given an object and a contact set, define all the kinematics parameters of the hand model in such a way that all the fingertips can reach a defined contact position on the surface of the object to be grasped. This task has to be executed in the presence of noise and uncertainty*. The number of contacts is between two and five (Fig. 1b/).

To define the hand posture, the following assumptions are made:

1. Only precision grasps are considered, therefore there is only one contact per finger,
2. We assume that the contact set is already defined and satisfies force closure,
3. Each contact on the surface of the object is associated with a particular finger,

The input data of the model are:

1. The hand geometry,
2. The number of contacts,
3. The finger associated to each contact,

4. The location of the contacts in the object reference frame,
5. The bounds of a 6D hand palm search space.

The outputs of the model are:

1. The location and orientation of the hand palm reference frame expressed in the world frame,
2. The configuration of the fingers in joint coordinate space.

2.3 Retained approach

Defining all these kinematics parameters in a single step is a complex task because of the large number of degrees of freedom. One can consider it as an inverse kinematics problem of a redundant multichain mechanism with multiple objectives. Indeed, several difficulties arise: several fingers are involved and a hand palm configuration may conduct to a good solution for one finger but bad for the others. The second point is relative to the fact that an infinite number of solutions may exist. To solve this problem, we start from the idea that if the hand palm configuration is held fixed it is possible to compute the configuration of the fingers in joint coordinate space by using inverse kinematics. Furthermore, if we express the desired fingertip position in the frame of the finger and construct a model of inverse kinematics by learning, it is possible to compute quickly the fingers joint angles given any hand palm configuration. Thus, the number of unknown parameters decreases from to twenty six to six. These latter correspond to the hand palm configuration which remains to be defined. The chosen mechanism has to optimize the hand palm configuration in such a way that the fingers configuration allows the fingertip to reach the contact on the surface of the object. In order to integrate sensor noise and uncertainty, a stochastic search procedure based on reinforcement learning is chosen. The process of hand grasping posture definition has two parts: the hand palm configuration is generated by a first network (called "Hand Configuration Neural Network" or HCNN) and the corresponding finger joints are obtained by inverse kinematics with a second neural network model called "Finger Configuration Neural Network" or FCNN. Then the whole grasping posture is evaluated and from the corresponding evaluation the HCNN is trained by reinforcement learning in order to generate a better hand posture at the next iteration.

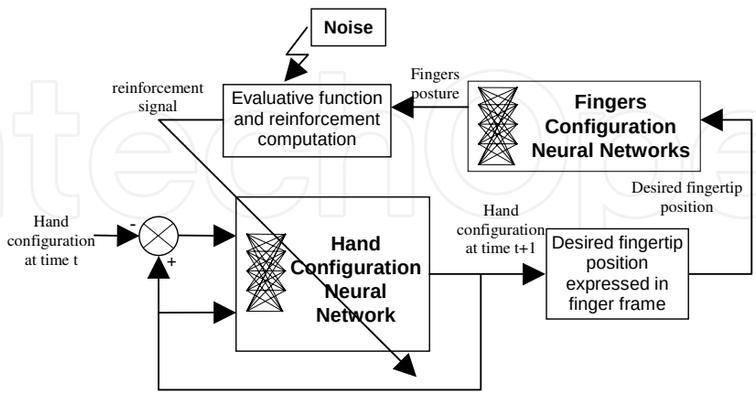


Fig. 2. Model architecture.

2.4 Model structure and principle

According to the method presented in the former section, the model global structure is composed of two neural modules working in a closed loop (Fig. 2). The first module

(HCNN) is aimed at determining the appropriate hand palm configuration (location and orientation) relative to a global frame. It is a multilayer feedforward neural network with a learning process based on the paradigm of reinforcement learning. This type of learning is achieved with resorting to a particular type of processing unit called Stochastic Real Valued (SRV) neurons. The detailed structure of the HCNN is presented in section 4. The FCNN, the second module, is devoted to the definition of the fingers configuration in joint coordinate space from the desired fingertips position. Its output data are used to evaluate the appropriateness of the hand configuration by an evaluative function. The corresponding reinforcement signal is used by the HCNN to update its internal parameters. In section 3, the FCNN is presented in details. The principle of the developed method is now described:

1. The current hand configuration and the difference between the actual and previous hand configurations (equivalent to a speed with unit time) are input to the HCNN. As output, a new hand palm configuration is obtained,
2. Thanks to this new hand palm configuration, it is possible to express the position of the contacts in the frame of each corresponding finger (on the surface of the object, a particular contact is affected to each finger),
3. Using the inverse kinematics scheme (FCNN), the joint configuration of the fingers is computed.
4. The complete hand configuration (hand palm configuration and fingers joint angles) is tested with a criterion. From this latter, a reinforcement signal is computed and is used to update the HCNN internal parameters. Before this last step and in order to introduce noise and uncertainty, the actual reinforcement is perturbed with noise with zero mean and predefined variance,
5. This procedure is repeated until a good solution is found (i.e. the reinforcement reaches a sufficiently high value and the criterion is optimized) or until the maximum number of iterations is reached

The evaluative function evaluated after each iteration is based on the computation of the sum of the distance between each fingertip desired and actual position. In step 2, the location of the contacts is expressed in the frame of the fingers and therefore the desired fingertip position is available. For each finger, forward kinematics is used to evaluate the actual fingertip position. Let

$$\mathbf{PX}_i^D = \left(x_i^D, y_i^D, z_i^D\right)^T \tag{1}$$

be the vector of the desired fingertip position expressed relative to the coordinate frame of finger i at step k and

$$\mathbf{PX}_i^M = \left(x_i^M, y_i^M, z_i^M\right)^T \tag{2}$$

be the vector of the actual fingertip position expressed relative to the coordinate frame of finger i. If n fingers are involved, the total error obtained at step k is:

$$E_k = \sum_{i=1}^{n} \left\| \mathbf{PX}_i^D - \mathbf{PX}_i^M \right\| \tag{3}$$

In order to model the effect of low quality sensors, this evaluative function is perturbed with a noise of known properties. The chosen procedure is detailed in section 4. The purpose of the next two sections is to provide further explanations on the structure and function of the two neural network based modules FCNN and HCNN.

3. Finger configuration neural networks (FCNN)

Given a contact set in the reference frame of the object and a global hand palm configuration, the problem is to compute the configuration of the fingers in joint coordinate space. To carry out this task (Rezzoug and Gorce, 2001; Gorce and Rezzoug, 2005), a slightly modified version of the modular architecture defined by Oyama and Tachi (Oyama & Tachi, 1999; Oyama & Tachi, 2000) is used. The main advantage of this method over other connectionist models is that it takes into account the discontinuity of inverse kinematics mapping of multi-joints mechanisms with joint limits by its modular nature. Indeed, it is difficult to approximate such a function by a single multi-layers neural network. The principle of the formalism defined by Oyama and Tachi (Oyama & Tachi, 1999; Oyama & Tachi, 2000) is based on three particular points. Firstly, the learning of inverse kinematics is performed by the incremental creation of multiple neural networks called experts each one spanning correctly one sub-set of the finger workspace. Secondly, when an expert is created, a representative posture in joint coordinate space is attached to it. Thirdly, after learning and for a given fingertip desired position, the finger joint angles are obtained by the appropriate competition among the created experts.
An expert is a feedforward neural network with four layers. The activation function of the input and output layers is linear and tangent sigmoid for the hidden layers. The input layer is composed of three units that correspond to the X, Y and Z position of the fingertip. The output layer has 4 units corresponding to the 4 degrees of freedom of a finger. The two hidden layers have 35 units each. The error backpropagation algorithm is used to update the neurons weights. During learning, four to six experts are generated for each finger.
In the remainder of this section, some details on the learning procedure are provided.

3.1 Learning procedure

At the beginning of the learning process, there is only one expert, with the representative posture $[0, 0, 0, 0]^T$ in joint coordinate space. For one particular finger, during one epoch of learning the following steps are performed:

1. A fingertip desired position X_D in an appropriate 3D workspace is chosen randomly. Each expert produces an output representing the finger joint angles θ_i. For each expert, the finger forward kinematics is used to compute the current fingertip Cartesian position X_C from the joint angles θ_i. The output error is the Euclidean distance between the current and the desired fingertip position ($\|X_C - X_D\|$). The posture generated by the expert with the minimal error is chosen as a start point. Then, a reaching motion represented by a straight line is constructed in Cartesian and joint spaces from the current to the desired fingertip position. The number of points that defines this motion is a function of the produced error. If the error is large, several points are stored. On the other hand, if the error is below a predefined threshold, an iterative procedure is used to improve the accuracy of the finger posture for the current training point (see section 3.2 for further details). Finally, the points of the reaching motion or the improved fingertip position and the corresponding joint angles are stored in the training set of the expert that has produced the minimal error among all the experts. Then, for a predefined number of steps, the learning of the training set is performed for the chosen expert.
2. If during the reaching motion construction, no satisfactory solution is obtained for the best expert either due to joint limits or if the finger reaches a singular position, another expert is

selected in increasing order of the predicted error. The procedure described in step 1 is repeated until a satisfactory solution is found or until all the experts are tested,

3. If no satisfactory solution is obtained during steps 1 and 2, an expert is randomly chosen and a reaching motion is constructed from the fingertip position derived from the expert representative posture to the desired fingertip position. This procedure is repeated until a solution is found or all the experts are tested,
4. If no solution is obtained during steps 1, 2 and 3, a posture is randomly computed and a reaching motion is performed between the actual and desired fingertip position. This procedure is repeated until a satisfactory motion is generated (i.e. no singular position and joints within their limits). If so, the finger configuration obtained at the end of the reaching motion is considered as the representative posture of a new expert, which is added to the set of experts.

3.2 Reaching motion and iterative improvement of fingertip position

In this section, the reaching motion definition and the iterative improvement of fingertip position (Oyama & Tachi, 1999; Oyama & Tachi, 2000) are described.

We consider $\boldsymbol{\theta(0)}$, the output of one selected expert. Then, the error between the current and the desired fingertip position is computed:

$$e(0) = |\mathbf{X_D} - g(\boldsymbol{\theta(0)})| \tag{4}$$

$\mathbf{X_D}$ represents the desired fingertip position and $g(.)$ the finger forward kinematics.

Two cases are considered:

1. if $e(0) < r_{st}$, r_{st} being a predefined threshold, perform the iterative procedure is performed and the joint angles $\theta(k+1)$ are computed according to the following equation:

$$\theta(k+1) = \theta(k) + \mathrm{J}^{+}(\theta(k)) \cdot \left(\mathrm{X_D} - g(\theta(k))\right) \text{ until } e(k) < r_e \tag{5}$$

r_e is a second predefined threshold.

2. if $e(0) > r_{st}$, a reaching motion is performed. To do this, a straight line is constructed in Cartesian space from $\mathrm{X_C} = g(\boldsymbol{\theta(0)})$ to $\mathbf{X_D}$ according to the following precdure:

Let T be an integer satisfying the following inequality:

$$T - 1 < \frac{|\mathrm{X_D} - \mathrm{X_C}|}{r_{st}} < T \tag{6}$$

The Cartesian trajectory $\mathbf{X_D}(k)$ is constructed in the following way:

$$\mathrm{X_D}(k) = \begin{cases} \left(1 - \dfrac{k}{T}\right) \cdot \mathrm{X_C} + \dfrac{k}{T} \cdot \mathrm{X_D} & (0 \le k < T) \\ \mathrm{X_D} & (k \ge T) \end{cases} \tag{7}$$

If r_{st} is sufficiently low, the set of joint angles $\boldsymbol{\theta}(k)$ that produces the trajectory of the fingertip as defined by (7) can be calculated in the following way:

$$\theta(k+1) = \theta(k) + \mathrm{J}^{+}(\theta(k)) \cdot \left(\mathrm{X_D}(k+1) - g(\theta(k))\right) \tag{8}$$

$\mathbf{J}^{+}$ represents the pseudo inverse of the finger Jacobian matrix.

During the iterative procedure, a single value of $\theta(k)$ is computed while several are determined during the reaching motion construction. This is due to the fact that, in the first case the error is considered small enough and it is just necessary to improve this particular output of the expert. Practically, at the beginning of the training phase, large reaching motions are constructed denoting the fact that the experts fail to approximate correctly the desired finger joint configuration. As training progresses, smaller reaching motions are generated and finally only the iterative procedure is performed.
The function *g(.)* representing finger forward kinematics can be obtained from a previously trained neural network or directly from its closed form expression. For simplicity the latter procedure was chosen. The main difference between the architecture of (Oyama & Tachi, 1999; Oyama & Tachi, 2000) and our implementation (Rezzoug & Gorce, 2001; Gorce and Rezzoug, 2005) is that while forward and inverse kinematics are learned by two separate modules, the closed form expression of forward kinematic is used and therefore only the inverse model is learned. At each iteration, the current generated motion only is learned by the best expert(Oyama & Tachi, 1999; Oyama & Tachi, 2000). On the other hand (Rezzoug & Gorce, 2001; Gorce and Rezzoug, 2005), we keep track of all the previous learned motions to speed up the learning. Also, every five epochs, all the training sets are tested by all the experts. If it happens that a point is best approximated by an expert, this point is transferred to the training set of the corresponding expert. Therefore, as training proceeds, we insure that the points of an expert training set are best learned by the corresponding expert.

4. Hand configuration neural network (HCNN)

4.1 Reinforcement learning with a direct method

The learning process used to define the hand palm configuration (HCNN) is based on reinforcement learning (Sutton, 1988; Watkins, 1989; Kaelbling et al., 1996; Sutton & Barto, 1998; Doya; 1999). In this frame, a learning agent has to infer appropriate actions to perform a task based solely on evaluative feedback. Unlike supervised learning, the agent does not know what action to take in a given situation but has only limited information on how well it behaves after the execution of the action. This information called reinforcement can take the form of a binary signal such as 0 (failure) or 1 (success) or be a real value. An extensive amount of relevant work has been proposed describing reinforcement learning problems (Sutton, 1988; Watkins, 1989; Kaelbling et al., 1996; Sutton & Barto, 1998; Doya, 1999). Most of the developed methods deals with discrete input and action spaces such as temporal difference learning (TD) (Sutton, 1988) or Q-learning (Watkins, 1989). The necessity to work in continuous input and action spaces, (e. g. to control robots) has led to the development of new methods based on an adequate discretization of state and action spaces or on the adaptation of the TD(□) algorithm (Doya; 1999).
Another point of interest of this technique is the choice of the process to infer the appropriate actions from evaluations. Gullapalli (Gullapalli, 1990; Gullapalli, 1995) presented the two major categories, indirect and direct methods. Indirect methods rely on the model construction of the transformation from the controller's action to the evaluations. Then, this model is used to obtain gradient information for training the controller (Gullapalli, 1990). On the other hand, direct methods perform this task directly by perturbing the process. From the produced effects on the performance evaluation the agent is able to modify its internal parameters in order to maximize the reinforcement. Usually, the perturbation takes the form of a random noise with known properties and a stochastic search procedure is conducted (Gullapalli, 1990). This latter procedure seems to be very attractive, since no model of the process is necessary. Indeed, building such a

model is very difficult especially in the presence of noise and uncertainty. In the frame of direct methods, the process itself provides the necessary training data. That is the reason why we have chosen this formalism to determine the hand palm configuration by the mean of a neural network composed of backpropagation and Stochastic Real Valued (SRV) units which are detailed in section 4.2 The main interest of SRV units is that they enable the learning of functions with continuous outputs using a connectionist network with a direct method (Gullapalli, 1995).
The neural network has 2 hidden backpropagation layers and an output layer composed of SRV units. The input layer has 12 units: 6 for the hand palm attached coordinate frame configuration and 6 for the difference between the actual and previous hand palm configuration. The 2 hidden layers have 20 units each. The association of SRV and backpropagation units allows to take advantage of both supervised and reinforcement learning. The whole network still has the benefits of reinforcement learning thanks to its stochastic search behavior. Also, the backpropagation units in the hidden layer enable to develop the right internal distributed representation of the problem as it is seen in supervised learning.
In order to better understand how the SRV units learn, a description of their input-output relation and stochastic learning behavior is now presented.

4.2 SRV unit input-output relation and learning process

An input vector i_k from $X \subseteq \Re^n$, where $\Re$ is the set of real numbers, is presented to a SRV unit at time step k. The unit produces a random output o_k selected from some internal probability distribution over the interval $O \subseteq \Re$. The SRV unit uses its input i_k to compute the parameters μ_k and σ_k of an internal normal probability distribution (μ_k is the mean and σ_k the standard deviation). These parameters are obtained by the weighted sum of the input i_k, with a particular set of weights for each parameter. We define the weight vectors (or matrices for a layer of *SRV* units) Θ and Φ respectively for μ_k and σ_k. In the case of several SRV units θ_{lj} and φ_{lj} corresponds to the weight associated with the j^{th} component of the input vector and l^{th} *SRV* unit parameter. For a single unit, the mean of the normal probability distribution is obtained as following:

$$\mu_k = \theta^T i_k \tag{9}$$

θ is a column weight vectors.
The computation of the standard deviation is done in two stages. Firstly, an expected reinforcement is computed as the weighted sum of the input vector i_k by the column vector φ:

$$\hat{r}_k = \varphi^T i_k \tag{10}$$

Then, the standard deviation is obtained as a function of the expected reinforcement as:

$$\sigma_k = s(\hat{r}_k) \tag{11}$$

where $s(.)$ is a monotonically decreasing, non negative function of $\hat{r}_k$. Moreover, $s(1.0) = 0$, so that when the maximum reinforcement is expected, the standard deviation decreases to zero 0. The expected reinforcement represents the internal estimation of the reinforcement obtained after taking a particular action. The standard deviation represents the degree of exploration around the mean. Therefore, if the expected reinforcement is high it is likely that the amount of exploration is low and therefore the standard deviation should be low.
Once μ_k and σ_k are computed, the SRV unit computes its *activation* drawn form the normal distribution defined by μ_k and σ_k:

$$a_k \sim N(\mu_k, \sigma_k) \tag{12}$$

Finally, the *output* is obtained as a function of the unit's *activation* as $o_k = f(a_k)$. In the present case the chosen function $f(.)$ is the logistic function, because the output is restricted to lie in the interval [0, 1]. In order to obtain an output vector within the desired bounds, the network output vector o_k is scaled according to the following equation:

$$\mathbf{X}_{i+1} = \mathbf{X}_{\min} + (\mathbf{X}_{\max} - \mathbf{X}_{\min}) \otimes o_k \tag{13}$$

$\mathbf{X_{i+1}}$ denotes the new output, $\mathbf{X_{min}}$ the lower bounds of the search space, $\mathbf{X_{max}}$ the upper bounds of the search space, o_k the network output vector and $\otimes$ the componentwise vector product.
The environment evaluates the new output $\mathbf{X}_{i+1}$ according to the evaluation function (1-3) and the context of i_k and returns a reinforcement signal $r_k \in \mathrm{R} = [0, 1]$, with $r_k = 1$ denoting the maximum possible reinforcement. Therefore, the reinforcement signal value is obtained as follows:

$$r_k = 1 - h(E_k) \tag{14}$$

where E_k (3) corresponds to the error at time step k. h is a monotonic increasing function of the error E_k taking values over the interval [0, 1]. If E_k is large, h tends towards 1 and therefore the network receives a maximum punishment with a reinforcement toward 0. On the contrary, if the error E_k is low, h tends towards 0 and, consequently, the system receives a higher reinforcement through equation (14). In the present case, h is the tangent sigmoid function.
In order to model low sensing quality and noise effect, the actual reinforcement is perturbed with a random noise with a zero mean and known standard deviation σ_v (15). Also, it is considered to be distributed according to a Gaussian probability distribution. This random parameter reflects the quality of hand position sensors providing information about the hand palm configuration as well as fingers joint position sensors:

$$r_k = r_k + N(0, \sigma_n) \tag{15}$$

To update the two parameters $\theta(k)$ and $\varphi(k)$ used in the computation of the mean μ_k and standard deviation σ_k, the following learning rules are used:

$$\theta(k+1) = \begin{cases} \theta(k) + \alpha\left((r_k - \hat{r}_k)\left(\frac{a_k - \mu_k}{\sigma_k}\right)\right)(i_k) & \text{if } \sigma_k > 0 \\ \theta(k) & \text{if } \sigma_k = 0 \end{cases} \tag{16}$$

$$\varphi(k+1) = \varphi(k) + \rho(r_k - \hat{r}_k) i_k \tag{17}$$

α and ρ are constant learning rates.
The update rules are designed to produce the following behavior. If the normalized perturbation added to the mean output of the unit (fraction in (16)) leads to a reward that is greater than the expected reward, then it is likely that the unit produces an output that is closer to the actual output. In other words, the mean should be changed in the direction of the perturbation that has produced a better reward and the unit should update its weights accordingly.
To update the weight vector φ (17), the following procedure is used. To each input vector is associated an expected reinforcement value. Since these two parameters are available, to learn their association, a supervised learning paradigm can be used. In this case the Widrow-Hoff LMS rule is chosen. The second important point in the

learning rule (16) is that the standard deviation depends on the expected reinforcement. Therefore, the SRV unit can control the extent of search through the standard deviation value. In fact, as the expected reinforcement increases, the standard deviation decreases and, therefore, the size of the search space is narrowed in the neighborhood of the mean output.

4.3 Integration of the layer of SRV units in the HCNN

Recalling that the HCNN is composed of two layers of neurons using backpropagation (BP) and one output layer with SRV neurons, the input vector of the SRV units is the output vector of a hidden layer of BP units. To train the BP neurons, an error vector is needed and since SRV output units are used, the error signal is not available because there is no desired output. (Gullapalli, 1990) wrote that randomly perturbing the mean output and observing the consequent change in the evaluation, enables the unit to estimate the gradient of the evaluation with respect to the output. Therefore, to train the backpropagation layers, the actual error is replaced with an estimated error of the following form:

$$\partial_n^{SRV} = \frac{(r_k - \hat{r}_k)(a_k - \mu_k)}{\sigma_k} \tag{18}$$

In order to propagate the error from the *SRV* units back to the *BP* units, we have used the following equation.

$$\partial_n^{BP} = \Theta^T \cdot \partial_n^{SRV} \tag{19}$$

Where $\Theta[i, j]$ is the weight θ used to compute the mean parameter of the i^{th} *SRV* unit from the j^{th} *BP* unit's output (considered as input *j* of the *SRV* unit). With this properly propagated error (19), we can train the *BP* layers using the standard backpropagation algorithm.

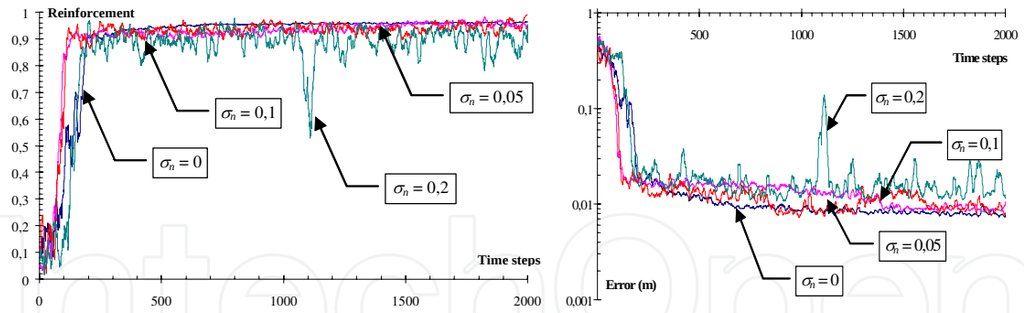
The purpose of the following sections is to study the performances of the proposed model. In a first part, the model is applied to define the posture of the MANUS manipulator used in rehabilitation robotics. Then, we evaluate the capability of the model to construct the posture of an anthropomorphic upper-limb model for a particular grasp with the incorporation of noise in the reinforcement signal and obstacles in the environment.

5. Simulation results

The purpose of this section is to study the performances of the proposed model. In a first part, we evaluate the capability of the model to construct hand posture for a particular grasp with the incorporation of noise in the reinforcement signal. In a second time, for three different tasks, proposed model is tested.

5.1 Grasping with noise and uncertainty

In this section, simulation results are presented. The hand posture to grasp a parallelepiped with five fingers is constructed. Before each test, the weights of the HCNN are initialized with random values over the interval [-0.5, 0.5] and the hand palm is placed in a random configuration within the search space. The learning is performed until a reinforcement greater than 0.99 is obtained or until the maximum number of iterations is reached.



a/ b/

Fig. 3. Evolution of learning parameters according to noise level: a/ Reinforcement signal and b/ corresponding error (m).

In order to identify the influence of uncertainty and noise reflecting low quality sensors, we have considered four levels of noise with zero mean and standard deviation σ_n of 0.2, 0.1, 0.05 and 0 (which corresponds to a deterministic reinforcement, $r_k \in [0, 1]$). Each trial has duration of 2000 steps and the bounds of the workspace are defined in Table I. In order to have a satisfying convergence, we use the following learning rates $\alpha_{BP} = 0.01$ for the BP units, $\alpha = 0.01$ and $\rho = 0.01$ for the SRV units. In Fig. 3a/, the obtained reinforcement signals with the four noise standard deviation levels are displayed as well as the corresponding deterministic error in Fig. 3b/. The algorithm finds a satisfactory solution even if the curves are more irregular for large standard deviations. We can also notice that the convergence is of the same order than the trial with deterministic reinforcement attesting the model robustness to noise. Finally, a satisfactory solution is obtained after a relatively low number of time steps.

At the beginning of the learning process the SRV units expected reinforcement is low and the standard deviation parameter is high; thus the exploratory behavior is predominant. As training proceeds, the mean of the SRV units Gaussian distribution is gradually shifted in such a direction that the reinforcement increases. Consequently, the expected reinforcement increases and the standard deviation decreases rapidly as well as the total error. Then, the exploitation behavior becomes predominant and the hand configuration is slightly modified to optimize the solution.

	X (m)	*Y (m)*	*Z (m)*	*θX (°)*	*θY (°)*	*θZ (°)*
min. value	-0.1	-0.1	0.15	-45	-45	-135
max. value	0.1	0.1	0.25	45	45	-45

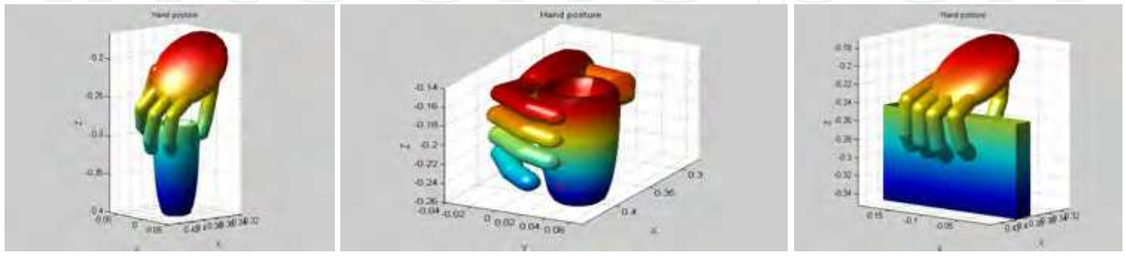
Table 1. Search space bounds for the rectangular block five fingered grasp.

The results of the methods are constructed for grasps with different number of fingers and different contact sets. MATLAB environment is used to solve the grasp planning problem (Pentium 4 HT 3.0Ghz, 512 ram, FSB 800 MHz). Three different grasps are considered in increasing number of fingers : grasping a glass with three fingers (task 1) or four fingers (task 2) and grasping a videotape with five fingers (task 3) (Fig. 4). For each task, twenty simulations are performed using the proposed model. Only the deterministic reinforcement with a zero noise level is considered. The results are summarized in Table 2. Results are displayed for each task in three columns. For task 1, in the first and second ones are indicated the total error and the detail for each fingertip after 2 and 5 seconds of learning respectively. Finally, in the third column, the error after 2000 iterations is displayed. For task 2, results are provided after 2 and 6 seconds of learning in first and second columns and after 1000 iterations in the third one, for task 3, after 3 and 7 seconds and after 1000 iterations in column 1, 2 and 3 respectively. Average values ± standard deviation are provided.

	Task 1			Task 2			Task 3		
Time (s)	2	5	18.5±12.3	2	6	14.8±5.2	3	7	18.9±3.2
Error (mm)	9.0±9.4	4.1±1.1	3.1±0.04	8.4±5.8	6.2±1.7	5.1±1.2	21.3±21.7	9.1±2.8	6.9±1.3
Thumb (mm)	2.6±1.4	1.9±0.8	1.5±0.7	3.4±2.9	2.5±1.3	2.1±0.9	7.0±9.8	3.7±1.5	3.0±0.9
Index (mm)	4.5±7.5	1.1±0.7	0.8±0.4	1.9±1.5	1.3±0.3	1.2±0.4	4.9±6.2	2.2±1.3	1.6±0.9
Middle (mm)	2.0±2.2	1.1±0.4	0.8±0.4	1.5±1.5	1.2±0.7	0.9±0.5	3.5±5.1	1.1±0.5	1.0±0.3
Ring (mm)				1.6±0.9	1.2±0.5	0.9±0.5	2.2±2.8	1.0±0.5	0.7±0.3
Pinky (mm)							3.7±7.4	1.2±1.0	0.7±0.3

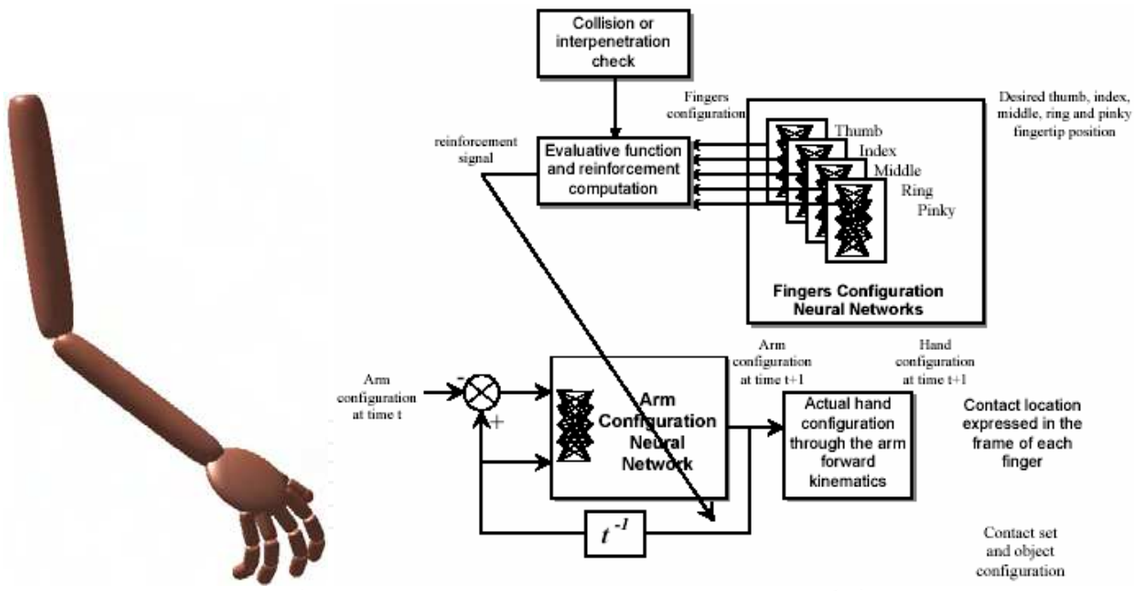
Table 2. HCNN results.

Finally, in Fig. 4, the obtained postures for task 1, 2 and 3 are displayed.



a/ Task 1 b/ Task 2 c/ Task 3

Fig. 4. Postures obtained by HCNN and optimization for the three tasks: a/ glass grasp with 3 fingers, b/ glass grasp with four fingers, c/ videotape grasp with five fingers.



a/ b/

Fig. 5. Complete arm model, Model architecture when considering the arm and obstacles in the environment.

6. Complete upper-limb posture definition with obstacle avoidance

In this section, a complete upper-limb is considered including an arm model associated with the hand model described and used in the former section. The model capabilities are extended by considering the presence of unknown obstacles in the environment (Rezzoug &

Gorce, 2006). The resolution principle is the same than the one used when considering the hand only. Instead of the hand palm position, the arm joints configuration is considered as input of the ACNN (Arm Configuration Neural Network) which replaces the HCNN of the original model. Also, the arm forward model is used to compute the contact set in the frame of the considered fingers as shown in Fig. 5b/.

6.1. Arm model

The model of the arm is composed of two segments and three joints (Fig. 5a/). The first joint, located at the shoulder (gleno-humeral joint) has three degrees of freedom. The second joint is located at the elbow and has one degree of freedom. Finally, the last joint, located at the wrist, has three degrees of freedom. The final frame of the last segment defines the orientation of the hand palm. According to this formulation, the arm posture is completely defined by the joint angles vector $\mathbf{q} = [q_1 , q_2, q_3, q_4, q_5, q_6, q_7]^T$. The chosen model has 7 degrees of freedom (Fig. 5a/).

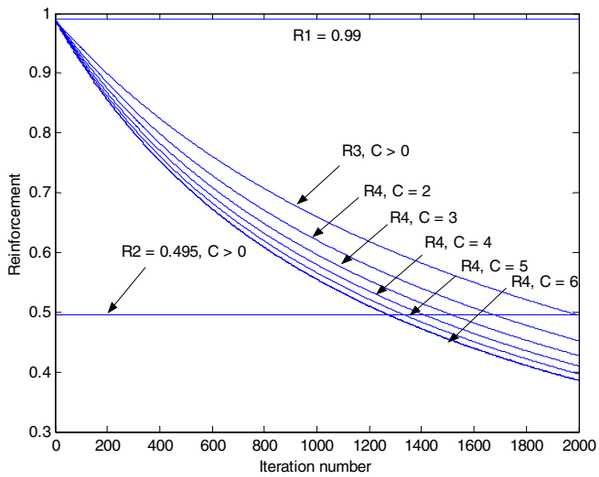


Fig. 6. Evolution of the reinforcements R_1, R_2, R_3 and R_4 relative to the iteration and collision numbers.

6.2. Improving learning performances by shaping

In order to define a suitable reinforcement signal, two aspects of the performance have to be taken into account. The first one evaluates the upper-limb positioning task while the second is relative to collision avoidance. In the following, the different steps that conduct to the definition of the appropriate reinforcement signal are described. Firstly, the positioning task is evaluated. To do this, given the arm and fingers configuration, the actual position of the fingertips is calculated using forward kinematics.

The simplest form of the reinforcement R_1 as used in section 4 gives a maximum penalty if error E_k is large and is given in (14) and recalled here (a is a positive real number):

$$R_1 = 1 - h(a.E_k) \tag{20}$$

Starting from the definition of R_1, the basic expression of the reinforcement signal that incorporates collision avoidance behavior is given by:

$$R_2 = \begin{cases} R_1 & \text{if no collision} \\ R_1 / 2 & \text{if collision} \end{cases} \tag{21}$$

In order to fulfill the secondary task i.e. collision avoidance, the reinforcement R_1 is divided by two whenever a collision is detected. Therefore, even if the principal task is accomplished with success the reinforcement is low due to the occurrence of a collision. One can notice the simplicity of the incorporation of collision avoidance behavior in the learning process. However, the criterion R_2 uses a somewhat crude strategy and the results may not be as satisfying as expected. Indeed, the learning agent has to directly discover the right strategy to satisfy two kinds of constraints at the same time. This is a more complex task than arm positioning only.

In order to circumvent this difficulty, we propose to use a technique inspired from animal training called shaping (Skinner, 1938). (Gullapalli, 1992) gave a nice definition of this concept and applied it to the frame of reinforcement learning : "The principle underlying shaping is that learning to solve complex problems can be facilitated by first learning to solve simpler problems. … the behavior of a controller can be shaped over time by gradually increasing the complexity of the task as the controller learns".

To incorporate shaping in the learning procedure, the basic idea is to let the agent learn the positioning task first and the collision avoidance behavior during a second phase. To implement this, a reinforcement signal that gradually increases over time the penalty due to collisions is defined. In this way, the agent can learn adequately the first task and modify its behavior in order to achieve the second one. The reinforcement value used in this case is the following (*i* is the current iteration number and *p* the maximum number of iterations):

$$R_3 = \begin{cases} R_1 & \text{if no collision} \\ R_1 / \left(1 + i / p\right) & \text{if collision} \end{cases} \tag{22}$$

If collisions occur, for the same value of R_1, an increase of *i* conducts to an increase of the denominator in (22) and consequently to a decrease of R_3. If $i = p$, we can notice that $R_3 = R_2$ and that there is a gradual shift from R_1 (no penalty for collision) to R_2 (full penalty for collision). This weaker definition of arm positioning with collision avoidance may be easier to learn than direct collision avoidance as defined by R_2. The evolution of R_3 with $R_1 = 0.99$ when collisions occur is displayed in Fig. 6.

The main drawback of R_3 is that the same penalty is applied whatever the number of collisions. It may be easier to learn the task successfully if the learning agent can grade differently two situations with different numbers of collision, giving more penalty to the posture conducting to more collisions or interpenetrations. In order to solve this problem, we define the reinforcement R_4:

$$R_4 = \begin{cases} R_1 & \text{if no collision} \\ R_1 / \left(1 + c^{\beta} \left(i / p\right)\right) & \text{if collision} \end{cases} \tag{23}$$

where *c* is the number of detected collision(s) or interpenetration(s) and β a positive real number.

Reinforcements R_3 and R_4 use the same strategy, except that R_4 takes into account the number of collisions. Indeed, for the same value of R_1, *i* and *p*, an increase of *c* conducts to an increase of the denominator in (23) and therefore to a decrease of the reinforcement R_4. If $c = 1$, we notice that $R_4 = R_3$. The evolution of R_4, with different values of *c* is displayed in Fig. 6.

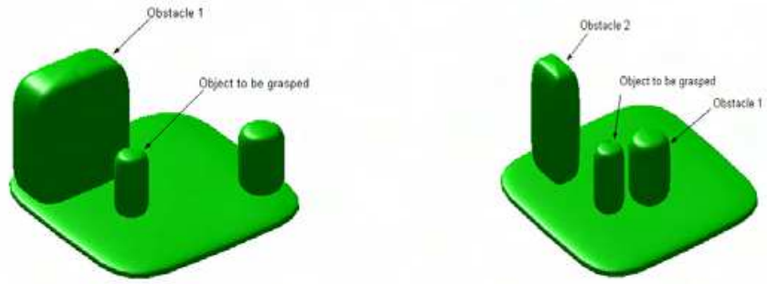


Fig. 7. Environments for the two grasping tasks.

6.3. Simulation results

The task to be performed is to grasp a cylinder with three fingers. Two different environments are considered, the first one with a big obstacle between the arm and the object and the second one with two obstacles (Fig. 7).

Reinforcement		R_1	R_2	R_3	R_4
Success		26	20	22	26
Causes of failure	Positioning task	4	10	7	4
	Collision	22	3	6	2
Mean iterations number		102	430	281	310
Standard deviation		126	271	259	245

Table 3. Simulation results for Task 1.

Reinforcement		R_1	R_2	R_3	R_4
Success		25	8	22	16
Causes of failure	Positioning task	5	22	7	14
	Collision	24	4	2	5
Mean iterations number		120	454	273	260
Standard deviation		161	266	228	223

Table 4. Simulation results for Task 2.

Thirty simulations are performed for each reinforcement and for each environment. The weights of the ACNN are initialized with random values over the interval [-0.5, 0.5] and a random arm configuration is chosen within the search space. We use the following values for the parameters of the ACNN : $\alpha_{BP} = 0.03$, $\alpha_{SRV} = 0.03$, $\rho = 0.01$, $n_1 = 14$, $n_2 = n_3 = 20$, $n_4 = 7$, $a = 1.5$, $p = 2000$ and $\beta = 0.25$ The learning has to be completed for each task and environment and is performed until a reinforcement greater than 0.99 is obtained or until the maximum number of iterations is reached. A FCNN is constructed off line for each finger before the simulations. Collision or interpenetration check is implemented with a two steps scheme. Axis aligned bounding boxes are constructed for each element of the environment to make a first check. If it is positive, the distance between any pair of solids that are likely to collide is computed. This is done by minimizing the distance between any pair of points on the surface of two elements of the scene modeled with superquadrics.

In Tables 3 and 4 are displayed the obtained results. In the first row, the number of successes is indicated for each reinforcement. This corresponds to the case where the reinforcement is greater than 0.99. In the second and third rows is indicated the number of cases for which a failure is due either to the positioning task or to collisions. Finally, for the successful cases, the last two rows indicate the mean and standard deviation of the required number of iterations to obtain a suitable reinforcement.

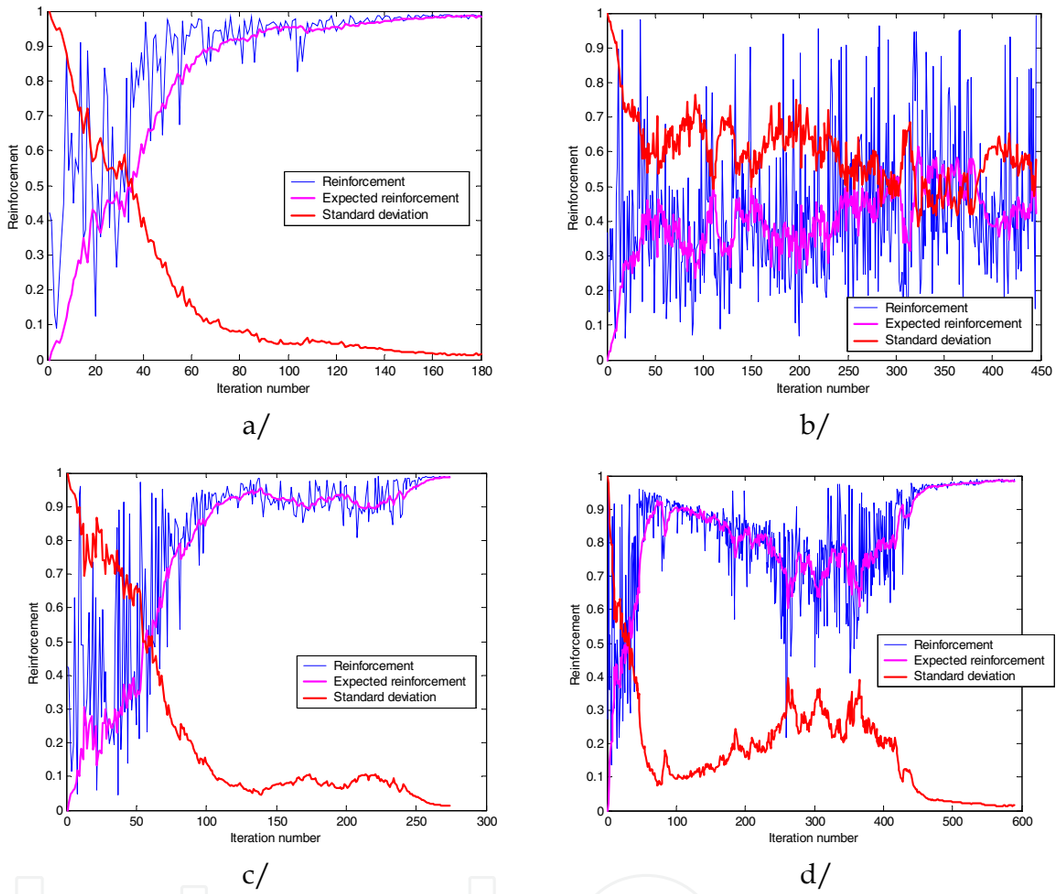


Fig. 8. Examples of learning parameters evolution for task 1 obtained with reinforcement a/ R_1 , b/ R_2, c/ R_3 and d/ R_4.

Reinforcement R_1 is used as a reference in order to demonstrate that the other reinforcements R_2, R_3 and R_4 have effectively an effect on collision avoidance.

The first observation is that the incorporation of collision avoidance behavior in the reinforcement signal effectively leads to collision avoidance even if the positioning task is not achieved. Using R_1, 22 solutions out of 26 valid ones are obtained with collisions between the upper limb and the environment for the first task and 24 out of 25 for the second task. This number falls to 3, 6 and 2 for R_2, R_3 and R_4 for the first task and 4, 2 and 5 for the second task respectively. Also, one can notice that there is an increase of the number of successes when shaping is incorporated compared to the case where crude collision avoidance reinforcement is used (R_2). This is particularly obvious for the second task (8 successes with R_2 compared to 22 using R_3). This suggests that the strategy of shaping conducts to a higher number of valid solutions and therefore that the learning is enhanced.

In Fig. 8 is presented the evolution of the learning parameters using reinforcements R_1, R_2, R_3 and R_4. These curves give interesting information on the learning process. Indeed, using R_1 (Fig. 8a/), the evolution of the expected reinforcement which is an internal parameter representing the learning performance is "monotonic" and increasing, suggesting that the learning is performed in one sequence and that there is a gradual shift from exploration to exploitation. This parameter using R_2 (Fig. 8b/) has a far less regular evolution, denoting the increased difficulty to find a solution due to the strict nature of R_2. The evolution of R_3 and R_4 (Fig. 8c/ and 8d/) show the effect of shaping. At the beginning of the learning, the expected reinforcement follows the same pattern than using R_1, i.e. a monotonic increasing function on average. As the number of iterations (NOI) increases, the occurrence of collision(s) is more penalized and there is a decrease of the expected reinforcement. This causes an augmentation of exploration evidenced by the augmentation of the standard deviation (bold dashed curve). Then, taking advantage of the previous learning of the positioning task, a solution is found that exhibits collision avoidance behavior. This is particularly visible in Fig.9 where the evolution of the number of collisions using R_3 is shown (this case corresponds to the results of Fig. 8c/). The agent has learned the first task without considering the collisions. This is evidenced by the high number of collisions at the beginning of the learning. Then, as the number of iterations increases this behavior is more penalized and using the previous acquired knowledge the agent is able to learn the second aspect of the task (collision avoidance) without a deterioration of the performance concerning the first aspect (i.e. grasping).

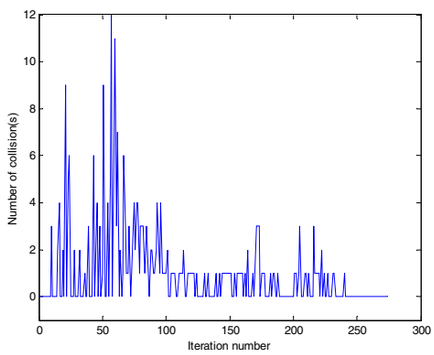


Fig. 9. Evolution of the number of collision(s) using reinforcement R_3.

To determine if the use of the different reinforcements has an effect on the NOI, a one way analysis of variance (ANOVA) (Miller, 1997) on the number of iterations to complete the task is conducted. A Bonferoni post-hoc test is used to perform multiple comparisons between means. The ANOVA evidences a significant difference between four groups means ($F(3, 161) = 14.33$, $p<0.0001$). Also, the post-hoc tests show a significant increase of the NOI using R_2 compared to the NOI using R_3 and R_4 ($p<0.05$). Also, a significant increase of the NOI using R_2, R_3 and R_4 compared to the NOI using R_1 is noticed ($p<0.05$). There is no significant difference between the NOI using R_3 and R_4. These results suggest that learning the positioning task is easier than the positioning task with collision avoidance because, on average, more iterations are needed whatever the chosen reinforcement. Secondly, the incorporation of shaping in the learning process reduces significantly the required number of iterations to reach the goal. Finally, taking into account the number of collisions in the reinforcement definition does not seem to improve significantly the learning performances. Therefore, among all the reinforcement signals proposed in this study, we can consider that R_3 is the best one to perform grasping posture definition with obstacles in the frame of the considered model. In Fig. 10, the posture obtained to grasp a cylinder surrounded by 3 obstacles is shown.

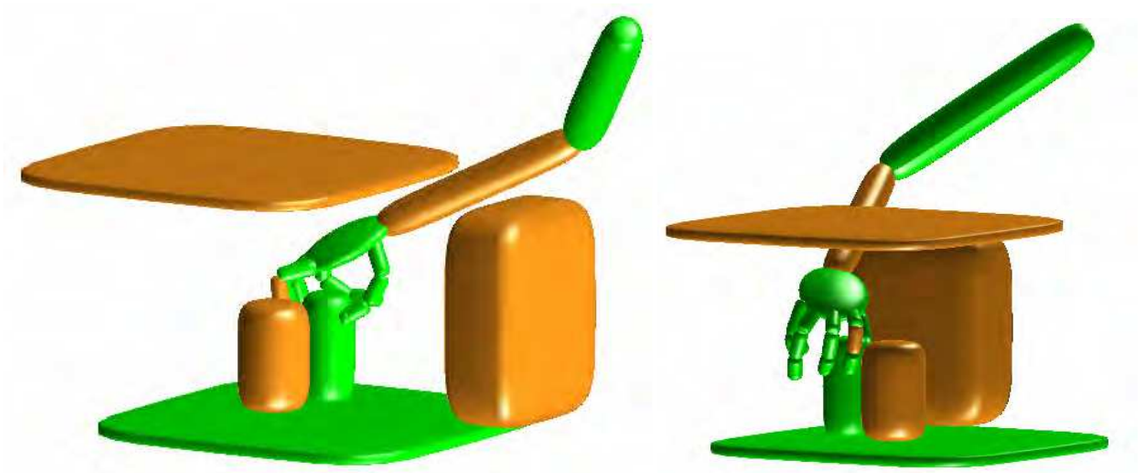


Fig. 10. Upper-limb posture to grasp a cylinder surrounded by 3 obstacles.

In the following section, a non anthropomorphic arm is considered. The method is slightly modified to handle this case.

7. MANUS robot configuration definition

Several movements are needed to grasp an object: an approach phase during which the end-effector is brought in the vicinity of the object followed by a grasping phase that implies precise adjustments in order to orient properly the end-effector. This latter phase can necessitate fine motions that are tedious if executed in manual control. To reduce the difficulty of this task, we propose to automate partially the grasping phase working in the vicinity of the arm configuration chosen by the user at the end of the approach phase. More precisely, we define the angular configuration of the robot joints in order to place the end-effector in an adequate configuration.

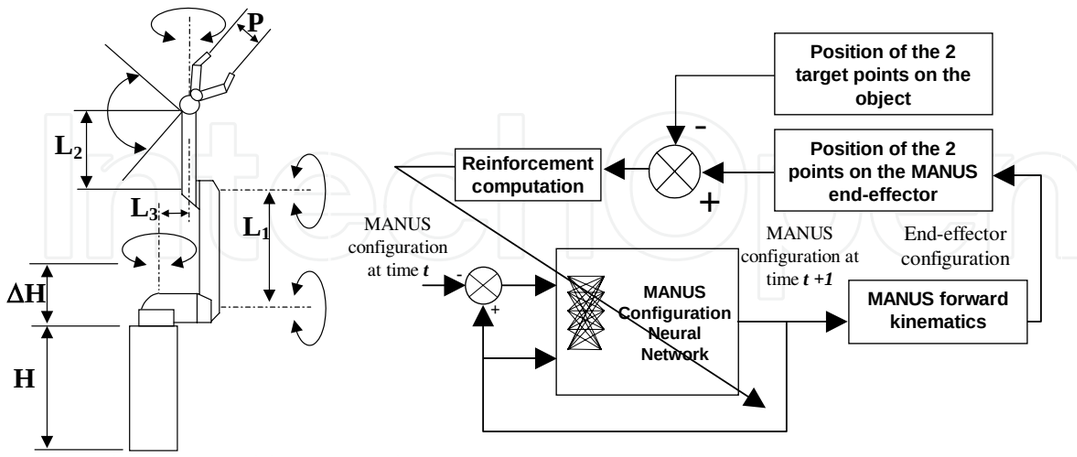


Fig. 11. a/ Degrees of freedom of the MANUS robot , b/ Model architecture.

7.1. Robot arm model

To define the mobile platform configuration 9 degrees of freedom have to be controlled (6 for the MANUS and 3 for the mobile base). This task is made difficult due to the existence of an infinite

number of solutions to put the platform and the MANUS arm in a given configuration (Fig. 11a/).
In order to identify the object to be grasped, it is necessary to obtain information from a camera. It seems important to define the needed amount of information to achieve the task considering the trade off between efficacy and complexity.
During the grasping phase, two points defined on the end-effector of the MANUS arm are associated with two points on the surface of the object. The goal of the grasping task is to bring the MANUS arm in such a configuration that the two pairs of points overlap. In this way, the constraints relative to the position as well as the orientation of the end-effector are treated. Furthermore, the amount of the needed information is limited to the position of four points in 3D. The corresponding model architecture is displayed in Fig. 11b/.
The inputs of the model are the location of the two points of interest both on the MANUS gripper and on the object and also the arm joint limits. The output is the mobile platform configuration that optimizes a particular cost function composed of two parts:

1. A first term that insures that the axes defined by the two points on the surface of the object and on the MANUS gripper are collinear.
2. A second term to minimize the distance between each couple of points.

The first part evaluates the orientation of the gripper relative to the object. Its expression is given in (24).

$$R_1 = \left| \mathbf{n}_{\mathbf{Gripper}} \cdot \mathbf{n}_{\mathbf{Object}} \right| \tag{24}$$

$\mathbf{n}_{\mathbf{Gripper}}$ is the unit vector aligned with the axe defined by the two points on the MANUS gripper and $\mathbf{n}_{\mathbf{Object}}$ is the unit vector defined by the two points on the object. The maximum value of R_1 is reached when the two vectors are collinear, then $R_1 = 1$. On the other hand, when the two vectors are orthogonal $R_1 = 0$.
The second part of the cost function evaluates the distance between the couples of points. It is defined as following:

$$R_2 = 1 - \tanh(d) \tag{25}$$

where

$$d = \min(d_1, d_2) \tag{26}$$

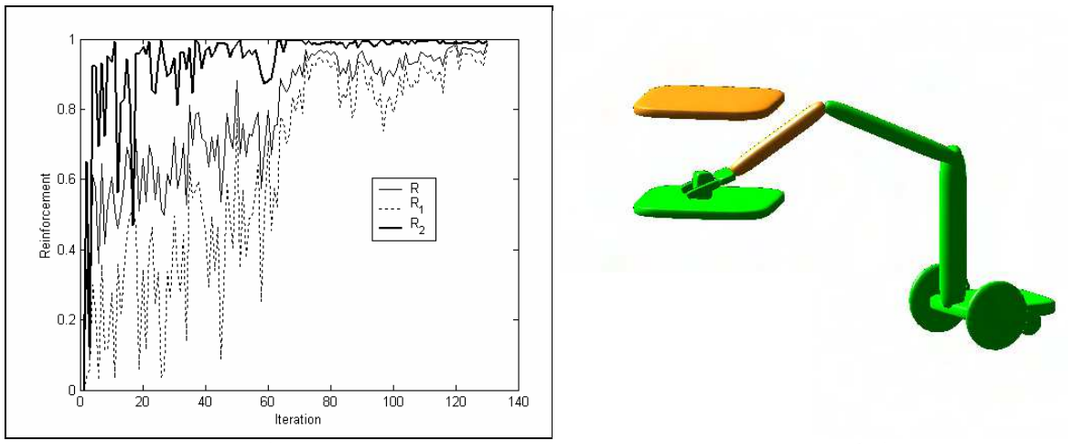
With

$$d_1 = \left\| \mathbf{X}_1^{\mathbf{Object}} - \mathbf{X}_1^{\mathbf{Gripper}} \right\| + \left\| \mathbf{X}_2^{\mathbf{Object}} - \mathbf{X}_2^{\mathbf{Gripper}} \right\| \tag{27}$$

$$d_2 = \left\| \mathbf{X}_2^{\mathbf{Object}} - \mathbf{X}_1^{\mathbf{Gripper}} \right\| + \left\| \mathbf{X}_1^{\mathbf{Object}} - \mathbf{X}_2^{\mathbf{Gripper}} \right\| \tag{28}$$

$\mathbf{X}_{\mathbf{i}}^{\mathbf{Object}}$ and $\mathbf{X}_{\mathbf{i}}^{\mathbf{Gripper}}$ (i = 1, 2) represent the 3D position of the points attached to the object and to the gripper in. d represents the minimum of the summed distances between the couple of points on the gripper and the object.
The function *tanh* insures that R_2 lies in the interval [0, 1]. R_2 is minimum if d is high and maximum ($R_2 = 1$) if d = 0. Combining the two criteria, the cost function R (29) is obtained.

$$R = \frac{R_1 + R_2}{2} \tag{29}$$



a/ b/

Fig. 12. a/Criterion evolution and b/simulated MANUS robot configuration.

7.2. Simulation results

In this section, simulation results are presented. The grasp of a cube placed at different locations is considered. The contact points location and the position of the object are known and considered as input data. The arm starts from a random position within its workspace. The graphs of Fig. 12a/ displaying the evolution of the learning criteria R_1, R_2 and R during learning show the ability of the model to discover the adequate configuration while satisfying orientation and position constraints. Also, to evaluate the performances of the method over a workspace, 30 simulations are performed for each of 64 object positions equally distributed in a 0.6 m x 0.6 m x 1m workspace. The obtained mean d (3) is 9.6 ± 4.1 mm. For one situation, the platform configuration is shown in Fig. 12b/.

8. Conclusion

In this chapter, a new model was proposed to define the kinematics of various robotic structures including an anthropomorphic arm and hand as well as industrial or service robots (MANUS). The proposed method is based on two neural networks. The first one is dedicated to finger inverse kinematics. The second stage of the model uses reinforcement learning to define the appropriate arm configuration. This model is able to define the whole upper limb configuration to grasp an object while avoiding obstacles located in the environment and with noise and uncertainty. Several simulation results demonstrate the capability of the model. The fact that no information about the number, position, shape and size of the obstacles is provided to the learning agent is an interesting property of this method. Another valuable feature is that a solution can be obtained after a relatively low number of iterations. One can consider this method as a part of a larger model to define robotic arm postures that tackles the "kinematical part" of the problem and can be associated with any grasp synthesis algorithm. In future work, we plan to develop algorithms based on unsupervised learning and Hopfield networks to construct the upper-limb movement. In this way, we will be able to generate an upper-limb collision free trajectory in joint coordinate space from any initial position to the collision free final configuration obtained by the method described in this article.

9. References

Bard, C.; Troccaz, J. & Vercelli, G. (1991). Shape analysis and hand preshaping for grasping, *Proceedings of the 1991 IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pp 64-69, 1991, Osaka, Japan

Becker, M.; Kefalea, E.; Mael, E.; von der Malsburg, C.; Pagel, M.; Triesch, J.; Vorbruggen, J. C.; Wurtz, R. P. & Zadel, S. (1999). GripSee: A gesture-controlled robot for object perception and manipulation. *Autonomous Robots*, Vol. 6, No 2, 203-221

Bekey, G. A.; Liu, H.; Tomovic, R. & Karplus, W. J., (1993). Knowledge-based control of grasping in robot hands using heuristics from human motor skills. *IEEE Transactions on Robotics and Automation*, Vol. 9, No 6, 709-721

Borst, C.; Fischer, M. & Hirzinger, G. (2002). Calculating hand configurations for precision and pinch grasps, *Proceedings of the 2002 IEEE Int. Conf. on Intelligent Robots and Systems*, pp 1553-1559, 2002, Lausanne, Suisse

Coehlo, J. A. & Grupen, R. A. (1997). A control basis for learning multifingered grasps. *Journal of Robotic Systems*, Vol. 14, No. 7, 545-557

Cutkosky, M. R. (1989). On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE Transactions on Robotics and Automation*, Vol.5, 269-279

Doya, K., (2000). Reinforcement learning in continuous time and space. *Neural Computation*, Vol. 12, 243 - 269

Ferrari, C. & Canny, J. (1992). Planning optimal grasps, *Proceedings of the 1992 IEEE Int. Conf. on Robotics and Automation*, pp 2290-2295, 1992, Nice, France

Gorce, P. & Fontaine J. G. (1996). Design methodology for flexible grippers. *Journal of Intelligent and Robotic Systems*, Vol. 15, No. 3, 307-328

Gorce, P. & Rezzoug, N. (2005). Grasping posture learning from noisy sensing information for a large scale of multi-fingered robotic systems. *Journal of Robotic Systems*, Vol. 22, No. 12, 711-724.

Grupen, R., & Coelho, J. (2002). Acquiring State form Control Dynamics to learn grasping policies for Robot hands. *International Journal on Advanced Robotics*, Vol. 15, No. 5, 427-444

Guan, Y. & Zhang, H. (2003). Kinematic feasibility analysis of 3-D multifingered grasps. *IEEE Transactions on Robotics and Automation*, Vol. 19, No. 3, 507-513

Gullapalli, V. (1990). A stochastic reinforcement learning algorithm for learning real valued functions. *Neural Networks*, Vol. 3, 671-692

Gullapalli, V. (1992) *Reinforcement learning and its application to control. PhD Thesis,* University of Massachusetts, MA, USA

Gullapalli, V. (1995). Direct associative reinforcement learning methods for dynamic systems control, *Neurocomputing*, Vol. 9, 271-292

Iberall, T. (1997). Human prehension and dexterous robot hands. *International Journal of Robotics Research*, Vol. 16, No 3, 285-299

Iberall, T.; Bingham, G. & Arbib, M. A. (1986). Opposition space as a structuring concept for the analysis of skilled hand movements. *Experimental Brain Research*, Vol. 15, 158-173

Jeannerod, M. (1984). The timing of natural prehension. *Journal of Motor Behavior*, Vol. 13, No 3, 235-254

K. Pook and Ballard, Recognizing teleoperated manipulations, *Proceedings of the 1996* IEEE Int. Conf. on Robotics and Automation , pp 578-583, 1993, Atlanta, GE, USA

Kaelbling, L. P.; Littman, M. L. & Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, Vol. 4, 237-285

Kang, S. B. & Ikeuchi, K., (1997). Toward automatic robot instruction from perception - Mapping human grasps to manipulator grasps. *IEEE Transactions on Robotics and Automation*, Vol 13, No 1, 81-95

Kerr, J. & Roth, R. (1986). Analysis of multifingered hands. *The International Journal of Robotics Research*, Vol. 4, 3-17

Kuperstein, M. & Rubinstein, J. (1989). Implementation of an adaptative controller for sensory-motor condition, in: *Connectionism in perspective*, Pfeiffer, R.; Schreter, Z.; Fogelman-Soulie, F. & Steels, L., (Ed.), pp. 49-61, Elsevier Science Publishers, ISBN 0-444-88061-5 , Amsterdam, North-Holland

Lee, C. & Xu, F. Y., (1996). Online, Interactive learning of gestures for human/robot interface, *Proceedings of the 1996 IEEE Int. Conf. on Robotics and Automation*, pp 2982-2987, 1996, Minneapolis MI, USA

Miller, A. T. & Allen, P. K. (1999). Examples of 3D grasps quality measures, *Proceedings of the 1999 IEEE Int. Conf. on Robotics and Automation*, pp. 1240-1246, 1999, Detroit MI, USA

Miller, A. T.; Knoop, S.; Allen, P. K. & Christensen, H. I. (2003). Automatic grasp planning using shape primitives, *Proceedings of the 2003 IEEE Int. Conf. on Robotics and Automation*, pp. 1824-1829, 2003, Taipei Taiwan

Miller, R.G., (1997). *Beyond Anova, Basics of applied statistics*, Chapman & Hall/CRC, Boca Raton, FL, USA

Mirtich, B. & Canny, J. (1994). Easily computable optimum grasps in 2D and 3D, *Proceedings of the 1994 IEEE Int. Conf. on Robotics and Automation*, pp 739–747, 1994, San Diego CA, USA

Moussa M. A. & Kamel, M. S. (1998). An Experimental approach to robotic grasping using a connectionist architecture and generic grasping functions. *IEEE Transactions on System Man and Cybernetics*, Vol. 28, 239-253

Napier, J.R. (1956). The prehensile movements of the human hand. *Journal of Bone and Joint Surgery*, Vol. 38, 902-913

Nguyen, V. (1986). Constructing force-closure grasps, *Proceedings of the 1986 IEEE Int. Conf. on Robotics and Automation*, pp 1368-1373, 1986, San Francisco CA , USA

Oyama, E. & Tachi, S. (1999). Inverse kinematics learning by modular architecture neural networks, *Proceedings of the 1999 IEEE Int. Joint Conf. on Neural Network*, pp 2065-2070, 1999, Washington DC, USA

Oyama, E. & Tachi, S. (2000). Modular neural net system for inverse kinematics learning, *Proceedings of the 2000 IEEE Int. Conf. on Robotics and Automation*, pp 3239-3246, 2000, San Francisco CA, USA

Pelossof, R.; Miller, A.; Allen, P. & Jebara, T. (2004). An SVM learning approach to robotic grasping, *Proceedings of the 2002 IEEE Int. Conf. on Robotics and Automation*, pp. 3215-3218, 2004, New Orleans LA, USA

Rezzoug, N. & Gorce, P. (2001). A neural network architecture to learn hand posture definition, *Proceedings of the 2001 IEEE Int. Conf. on Systems Man and Cybernetics*, pp. 3019-3025, 2001, Tucson AR, USA

Rezzoug, N. & Gorce, P. (2006). Upper-limb posture definition during grasping with task and environment constraints. *Lecture Notes in Computer Science / Artificial Intelligence*, Vol. 3881, 212–223.

Saito, F. & Nagata, K. (1999). Interpretation of grasp and manipulation based on grasping surfaces, *Proceedings of the 1999 IEEE Int. Conf. on Robotics and Automation*, pp 1247-1254, 1999, Detroit MI, USA

Skinner, B.F. (1938). *The behavior of organisms : An experimental analysis*, D Appleton century, New York, USA

Sutton, R. S. & Barto, A.G. (1998). *Reinforcement learning*, MIT Press, Cambridge, MA

Sutton, R. S. (1988). Learning to predict by the methods of temporal difference. *Machine Learning*, Vol. 3, 9-44

Taha, Z.; Brown, R. & Wright, D. (1997). Modelling and simulation of the hand grasping using neural networks. *Medical Engineering and Physiology*, Vol. 19, 536-538

Tomovic, R.; Bekey, G. A. & Karplus, W. J. (1987). A strategy for grasp synthesis with multifingered hand, *Proceedings of the 1987 IEEE Int. Conf. on Robotics and Automation*, pp. 83-89, 1987, Raleigh NC, USA

Uno, Y.; Fukumura, N.; Suzuki, R. & Kawato, M. (1995). A computational model for recognizing objects and planning hand shapes in grasping movements. *Neural Networks*, Vol. 8, 839-851

Watkins, C. J. C. H., (1989). *Learning form delayed reward. PhD thesis*, Cambridge University, MA, USA

Wheeler, D., Fagg, A. H. & Grupen, R. (2002). Learning prospective Pick and place behavior, *Proceedings of the 2002 IEEE/RSJ Int. Conf. on Development and Learning*, pp 197-202, 2002, Cambridge MA, USA

Wren, D. O. & Fisher, R. B. (1995). Dextrous hand grasping strategies using preshapes and digit trajectories, *Proceedings of the 1995 IEEE Int. Conf. on Systems, Man and Cybernetics*, pp. 910-915, 1995, Vancouver, Canada

IntechOpen



Mobile Robots: towards New Applications

Edited by Aleksandar Lazinica

ISBN 978-3-86611-314-5

Hard cover, 600 pages

Publisher I-Tech Education and Publishing

Published online 01, December, 2006

Published in print edition December, 2006

The range of potential applications for mobile robots is enormous. It includes agricultural robotics applications, routine material transport in factories, warehouses, office buildings and hospitals, indoor and outdoor security patrols, inventory verification, hazardous material handling, hazardous site cleanup, underwater applications, and numerous military applications. This book is the result of inspirations and contributions from many researchers worldwide. It presents a collection of wide range research results of robotics scientific community. Various aspects of current research in new robotics research areas and disciplines are explored and discussed. It is divided in three main parts covering different research areas: Humanoid Robots, Human-Robot Interaction, and Special Applications. We hope that you will find a lot of useful information in this book, which will help you in performing your research or fire your interests to start performing research in some of the cutting edge research fields mentioned in the book.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Nasser Rezzoug and Philippe Gorce (2006). Robotic Grasping: A Generic Neural Network Architecture, Mobile Robots: towards New Applications, Aleksandar Lazinica (Ed.), ISBN: 978-3-86611-314-5, InTech, Available from: http://www.intechopen.com/books/mobile_robots_towards_new_applications/robotic_grasping__a_generic_neural_network_architecture

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

© 2006 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.

IntechOpen

IntechOpen