Humanoid Robot Reaching Task Using Support Vector Machine

Mirko Raković[1], Milutin Nikolić[1], Branislav Borovac[1]

[1]Faculty of Technical Sciences, University of Novi Sad,
Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia

{rakovicm, milutinn, borovac}@uns.ac.rs

Abstract. A novel approach is proposed for the realization of the humanoid robot's reaching task using Support Vector Machine (SVM). The main difficulty is how to ensure an appropriate SVM training data set. Control law is firstly devised, and SVM is trained to calculate driving torques according to control law. For purpose of training SVM, sufficiently dense training data set was generated using designed controller. During grasping, dynamics of the environment changes, so SVM coefficients were altered in order to adapt to changes that have occured. In the stage of verification, the target point to be reached by the robot's hand is assigned. The trained SVM determines the necessary torques in a very efficient way, which has been demonstrated by several simulation examples.

Keywords: Humanoid Robots, Grasping, SVM

1 Introduction

Bearing in mind that we can expect soon the ‘living and working coexistence of robots and humans’, one of the very important tasks that will be imposed on humanoid robots is to manipulate the objects in their environment. In doing this, they will have to be ready to reach the objects with the aim of their grasping, and this is expected to be realized in a way involving the trajectory shape and realization efficiency that are very similar to those of humans. Reaching is always supervised by a visual system which estimates the relative mutual position of the object and hand, since the exact positions of either the robot's hand or the object to be grasped are not known in advance. Investigations have shown [1-2] that the trajectory of the man's hand in the reaching task in a free space is approximately a straight line. The same principle is to be also applied in the realization of the robot's hand reach, and the efficiency of the realized motion is expected to be close to that of human [3-6]. There are several ways in which such reference motion can be realized, and the main problem is how to execute the motion in a sufficiently effective. Trajectory generation can be computationally intensive, and relies heavily on model accuracy. Also, dynamical properties of object that robot is grasping is rarely known, so motion has to be adaptable to changing dynamics of whole system.

The way how to realize motion of such characteristics is in the focus of this study. In order to do so, control law for reaching task was devised. Afterwards, we train SVM to calculate driving torques which are in accordance with previously devised control law. During control of robotic arm, we constantly alter acquired SVM in order to adapt to ever changing environment.

2 Control Law

We will model the robot's hand motion as a mass *m* connected to the target point by a spring *K* and a damper *C* (Fig. 1).

If it is supposed that the position of the target point is fixed, the motion of the system can be described by the following second-order differential equation:

$$m^{\cdot\cdot}\ \text{[illegible]} \qquad (1)$$

Vectors **r** and **r**$_{\mathbf{d}}$ define current and target point position, while *m*, *C* and *K* are mass, dumping coefficient and spring stiffness, respectively. It is known that the response of the second order system will not have an overshoot if the following condition is satisfied:

$$C \geq 2\sqrt{mK} \qquad (2)$$

If the equality in (2) is satisfied, the system is critically damped, which means that it has the fastest possible response without overshoot. If the damping coefficient is increased, the system will still have no overshoot, but the approach of the point mass to the desired position will last longer. Therefore, it comes out that by changing the damping coefficient *C* we can modify the time of the movement realization. It is well known, that with the robots, the overshoots of the desired position must not be allowed under any circumstances. For instance, if the desired position is near some obstacle, and if the robot overshoots the desired position, it would be driven into the obstacle.

In order for the humanoid robot's hand to move in the way shown on Fig. 1, it is necessary to determine the corresponding driving torques at all the joints. To this end, a mathematical model of the robot's hand is formed, with three degrees of freedom (DOFs) (two at the shoulder and one at the elbow), as sketched in Fig. 1.

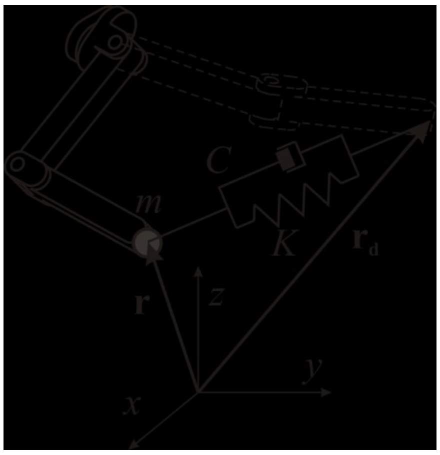

Fig. 1. Robot's hand with three DOFs

The motion of multy-body system can be described by the following differential equation:

$$\mathbf{H}\ddot{} \quad \boldsymbol{\tau} \tag{3}$$

where **H** is the inertia matrix of the system; **h0** is the vector that includes the velocity and gravitation effects; $\ddot{}$ is the vector of joint accelerations, and $\boldsymbol{\tau}$ is the vector of driving torques.

Based on the relations of direct kinematics, it is possible to find the relationship between the robot's tip and joint coordinates:

$$\mathbf{r} = \mathrm{f}(\mathbf{q}) \tag{4}$$

By differentiating equation (4) one obtains the relation between the joint velocities of the robot's hand, as well as the relation between the joint accelerations and linear acceleration of the robot's hand:

$$\dot{} \quad \dot{}, \tag{5}$$

$$\ddot{} \quad \ddot{} \quad , \tag{6}$$

where the matrix $\mathbf{J} = \partial \mathbf{r} / \partial \mathbf{q}$ represents the system's Jacobean and the vector $\mathbf{A} = (\partial^2 \mathbf{r} / \partial \mathbf{q}^2)\dot{}$ is the adjoint vector–column. The combination of equations (1), (3), (4), (5) and (6) gives an expression that allows us to calculate the corresponding driving torques at the joints:

$$\boldsymbol{\tau} = -\mathbf{H}(m\mathbf{J})^{-1}(K(\mathbf{r} - \mathbf{r}_d) + C\mathbf{J}\dot{} \quad \mathbf{h}_0 \tag{7}$$

Thus, it is important to point out the driving torques can be calculated only if **J** is a square matrix (i.e. when the system has no redundancy), which is fulfilled in the case considered. However, if the system is redundant, the Jacobi matrix is not square, so the inverse matrix $\mathbf{J}^{-1}$ does not exist. Such case is not considered here, but it is necessary to remind that the addition of certain conditions can allow the determination of a pseudo-Jacobean that could be used in expression (7).

2.1 Generating the training set

Our aim is to use the presented model to form an appropriate training set. Based on equations above, training data set for SVM training is calculated.

As evident from (7), the driving torques $\boldsymbol{\tau}$ depend on the joint coordinates $\mathbf{q}$, and velocities $\dot{\mathbf{q}}$, and distance between the current and target positions, $\mathbf{r}-\mathbf{r}_\mathbf{d}$. If we want to change the movement duration (the rate of hand motion), it suffices to change only the coefficient *C*. Hence, we will adopt the angles $\mathbf{q}$ and angular velocities $\dot{\mathbf{q}}$ at all hand joints, position vector of the hand from the target position ($\mathbf{r}-\mathbf{r}_\mathbf{d}$), as well as the damping *C* as inputs, and the driving torques at the joints corresponding to such motion as the outputs. The procedure of determining the input and output quantities is as follows:

1. An arbitrary point in the robot's workspace is selected as the starting position of the robot's hand
2. For the selected point, the inverse kinematics problem is solved.
3. An arbitrary point is then selected in the robot workspace to serve as the desired (target) position.
4. The damping coefficient is also arbitrarily selected from the predefined range.
5. The system's motion is simulated, and the driving torques in each iteration are calculated from equation (7).
6. When the distance between the current and target positions is sufficiently small, the simulation is stopped and the steps 3-6 are repeated, the last state serving as the starting one for the subsequent simulation.
7. After generating a sufficient number of movements, the procedure is stopped.

In this way the hand's motion is simulated from the point at which the previous movement terminated to the next (arbitrarily chosen) target point. For each time instant, the values of all input and output quantities are obtained. The input vector thus formed for the training set $[\mathbf{q}^\mathrm{T}\ \dot{\mathbf{q}}$ [illegible] $)^\mathrm{T}\ C]^\mathrm{T}$ is of dimension 10, whereas the dimension of the output vector $\boldsymbol{\tau}$ is 3.

3 Basics of Machine Learning

Since, we are using SVMs to generate movement of robot hand it is necessary to briefly describe what SVM represents and how it works. We will be using error backpropagation (EBP) algorithm to adapt learned weights, so we are also giving a brief insight into RBF networks and EBP algorithm itself.

3.1 SVM Regression

There are a number of algorithms for approximating the function for establishing the unknown interdependence between the input and output data, but an ever-arising question is how good is the approximation of the function $y = f(\mathbf{x})$. In determining the approximation function, it is necessary to minimize some of the error functions. The majority of the algorithms for the function approximation minimize the empirical error.

With the function approximation algorithms that minimize only the empirical error, there arises the problem of a large generalization error. The problem appears when the training set is small compared to the number of different data that can appear at the input. Structural Risk Minimization (SRM) [7] is a new technique of the statistical learning theory, which apart from minimizing the empirical errors, also minimizes the generalization errors (elements of the weight matrix **w**). Hence, it follows that the structural error will be minimized by minimizing function of the form:

$$R = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}|y_i - f_a(\mathbf{x}_i, \mathbf{w})|_\varepsilon \qquad \textbf{(8)}$$

In (8), the error function with the ε- insensitivity zone was used as the norm. The parameter C is the penalty parameter which determines the extent to which the empirical error is penalized relatively to the penalization of the large values in the weighting matrix. Network input is denoted by **x**, and desired output is denoted by **y**. Approximating function is denoted by $\mathbf{f}(\mathbf{x}, \mathbf{w})$ and it has to be chosen in advance. Since case considered is highly nonlinear, for approximating function we have chosen RBF network with Gaussian kernel, for which output is calculated by:

$$f_a(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} w_i \exp(-\gamma\|\mathbf{x} - \mathbf{c}_i\|^2) + \rho \qquad \textbf{(9)}$$

The nonlinear SVM regressions (minimization of (8)) determines the elements of the weight matrix **w** bias ρ. During the SVM training, support vectors ($\mathbf{c}_i$) are chosen from set of input training data. Design parameter ρ defines the shape of RBFs, and it is experimentally chosen to minimize VC-dimension, which provides good generalization.

3.2 RBF Network Adaptation

As a result of SVM regression we acquire RBF network with Gaussian function as kernel. Since model always contains some inaccuracies and during grasping dynamics of whole system changes it is necessary to adapt network parameters on-line. For that purpose EBP algorithm is used, which minimizes standard cost function [8]:

$$E(\mathbf{w}, \mathbf{c}, \gamma, \rho) = \sum_{i=1}^{l}(y_i - f_a(\mathbf{x}_i, \mathbf{w}, \mathbf{c}, \gamma, \rho))^2 \qquad \textbf{(10)}$$

This problem is nonlinear and nonconvex, therefore, many local minima can be expected, and EPB merely guarantees convergence to closest local minima. As a starting point for optimization we are using weights and coefficients obtained by SVM regression. Since SVM regression finds global minimum of error function (8) it is safe to assume that initial guess will be close to global minimum of EPBs cost function (10). Therefore, we can assume that EPB will find global minimum of error function (10).

For adapting coefficients and weights we have used following equations:

$$w_i^{t+1} = w_i^t + 2\eta(y^t - f_a(\mathbf{x}^t))\exp\left(-\gamma^t \left\|\mathbf{x}^t - \mathbf{c}_i^t\right\|^2\right) \tag{11}$$

$$\mathbf{c}_i^{t+1} = \mathbf{c}_i^t - 4\eta(y^t - f_a(\mathbf{x}^t))w_i^t\gamma\exp\left(-\gamma^t \left\|\mathbf{x}^t - \mathbf{c}_i^t\right\|^2\right)\left(\mathbf{c}_i^t - \mathbf{x}^t\right) \tag{12}$$

$$\gamma^{t+1} = \gamma^t - 2\eta\left(y^t - f_a(\mathbf{x}^t)\right)\sum_{i=1}^{N} w_i^t \exp\left(-\gamma^t \left\|\mathbf{x}^t - \mathbf{c}_i^t\right\|^2\right)\left\|\mathbf{x}^t - \mathbf{c}_i^t\right\|^2 \tag{13}$$

$$\rho^{t+1} = \rho^t - 2\eta\left(y^t - f_a(\mathbf{x}^t)\right) \tag{14}$$

where t stands for iteration step and η represents learning rate, which is experimentally chosen,

4 Simulation Results

In this section we describe the procedure of the SVM training for the task of the robot's hand reaching the object. The simulations of the robot's hand were carried out on the humanoid robot model [9]. Verification of the SVM control for the robot's hand motion is realized by assigning random points in the workspace to be reached by the robot's hand. Subsequently, RBF network adaptation is described, and verification of control is performed on two different cases.

4.1 Model of the robotic hand

The humanoid robot was modeled using the software that allowed the formation of a multi-body dynamics system with branched, open or closed, kinematic chains whose links are interconnected with the joints having only one DOF. Since only hand motion is simulated, Fig. 2.1 shows only the part of the humanoid robot that represents its right hand. The shoulder joint is modeled by two DOFs, whereas the elbow has only one DOF, with the upper arm and forearm being 0.4 m long, and weighing 5 kg each.

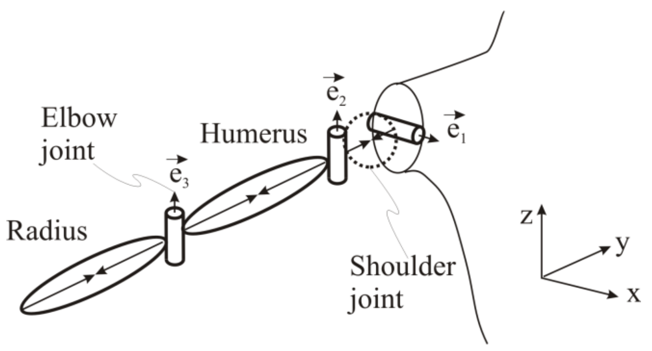

Fig. 2. Model of the robotic hand with three DOFs

4.2 The SVM Training

Let us remind that the inputs to the SVM training set are the joint coordinates $\mathbf{q}$, velocities $\dot{\mathbf{q}}$, the remaining distance to the target position $\mathbf{r}-\mathbf{r}_\mathbf{d}$, and the damping coefficient *C*, by which the hand motion speed is indirectly assigned, whereas the outputs are the joint driving torques $\boldsymbol{\tau}$. The hand's target position in the workspace is randomly assigned under the constraint that the x-coordinate of the given position is positive. In this way, the boundary of the workspace is determined by the hemisphere of a radius of 0.8 m.

In order to obtain an SVM with good generalization properties it is necessary to prepare a sufficiently dense training set, i.e. training data has to be equally distributed throughout whole input space. To do so, we have used procedure described in section 2. Mass of the virtual point mass is chosen to be *m*=26 kg, and elasticity coefficient is *K*=1000 N/m. The value of the damping coefficient *C* is randomly chosen in the span from 450 to 850 Ns/m, which ensures movement without overshoot. Driving torque $\boldsymbol{\tau}$ at each joint is limited to be between -80 and 80 Nm

Approximately one million noise–free data pairs were generated, which span whole input space. Because SVM training is very time consuming we have randomly chosen 10000 input-output data pairs to use for SVM training. Input and output data obtained this way were first normalized, and then SVM was trained by minimizing empirical risk (8) with ε- insensitivity zone of 0.01 and penalty parameter *C*=100.

4.3 Validation of the robot's hand motion for the object reaching task using SVM

To validate the control of the robot's hand motion using the trained SVM model, it was necessary to assign unseen input data. The simulation was carried out in the following way. In the beginning, the starting position of the robot's hand is assigned in the world coordinate frame. Based on the relations of the inverse kinematics, the values of joint coordinates $\mathbf{q}$ corresponding to this position were calculated. Since we assumed that the hand moves from the rest, the values of joint velocities in the beginning of the simulation are 0. Then, the target position in the workspace is selected in a random way, as well as the damping coefficient *C*, which remained constant during the realization of one movement.

Based on these data, using the trained SVM, three driving torques at the hand joints were determined and then applied in the simulation of the motion. The action of the joint driving torques causes the movement of the hand and values of the joint coordinates, joint velocities, as well as the distance to the target position changes. In each sampling period, using the information about the instantaneous state of the robot (distance from target position, joint angles and joint velocities) the trained SVM determines new driving torques to be applied at all joints, which produces further hand movement. If a driving torque exceeds predetermined maximum/minimum of ±80 Nm, maximum/minimum driving torque is applied at corresponding joint The process is repeated until the hand tip stops, After which a new target point is randomly selected, and the process is repeated.

Fig. 3 shows two different simulated movements, that differ only in respect of the damping coefficient. Starting and target positions of the hand in both cases are the same. The values of damping coefficients are *C*=450 Ns/m (first case Fig 3a) and *C*=850 Ns/m (second case, Fig 3b). It is evident that in both cases the hand's trajectory almost coincides with the linear path, and that there are no overshoots on the way to the target position. Also, hand’s speed is much greater at the beginning of the motion than at it’s end which coincides with desired movement. Positioning error in cases considered is less than 2.5 mm, and no overshoot occurred.

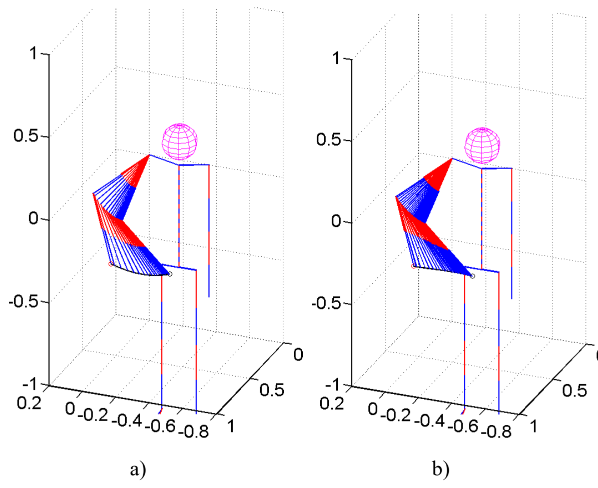

Fig. 3. Stick diagrams showing the hand's motion for the two different cases. Damping coefficients are a) *C*=450 Ns/m; b) *C*=850 Ns/m. Hand’s position is plotted every 75 ms. Black line represents trajectory of hand’s tip

As stated above, damping coefficient *C* is used to indirectly assign movement speed. When *C* is 450 Ns/m rise time (time from 10% to 90% percent of distance travelled) is 0.86 s, but when *C* is 850 Ns/m rise time is 1.8 s. Hence, rise time (and total movement time) increases as the damping coefficient *C* increases. It should also be noted that movements with smaller damping coefficient *C* produce greater deviations from desired linear path.

It would be interesting to present what happens when robot is carrying a point mass of unknown dynamical properties. For the sake of comparison we have simulated two cases with same start, target positions and damping coefficients *C*=850, but in one case, robot is carrying 1 kg weight. In both cases we will be using SVM to calculate driving torques as in previous cases. Results are displayed in Fig 4.

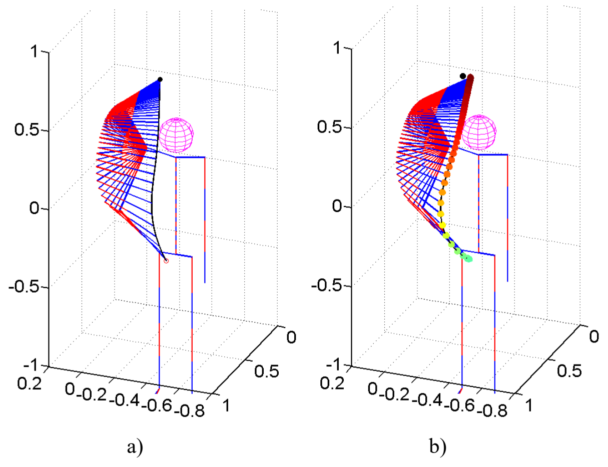

Fig. 4. Stick diagrams showing the hand's motion for the two different cases: a) movement without weight b) movement with weight. Hand's position is plotted every 75 ms. Black line represents trajectory of hand's tip. Black circle represents target position

In case depicted on Fig 4a, positioning error is less than 1 mm. Rise time is 1.82s, which is approximately equal to rise time in case from Fig 3b. This means that rise time (and total movement time) is not distance dependant, but only depends on damping parameter. When robot is holding point mass, robot's tip misses target by 55 mm, which is substantially larger than in previous case. Also it is clearly noticeable that movement is very nonlinear. This difference is due to change in dynamics of whole system with driving torque calculation remaining the same. That's why, this method is unsuitable for grasping task, and tasks during which system parameters change.

4.4 On-line RBF network adaptation

To compensate for changes that unintentionally (aging of materials) or intentionally (picking up) arise, we have to constantly adapt our control system. After SVM training we acquire RBF network, which is used to calculate driving torques, thus it controls our robotic hand. As stated before, for adapting RBF network we will use EBP algorithm, which is briefly described in section 3.2.

If we take a closer look into equations (11-14) we can notice that in order to perform EBP adaptation, pairs of input and corresponding output are needed. In our case input vector is $[\mathbf{q}^{\mathrm{T}}$ [illegible] $)^{\mathrm{T}}\ C]^{\mathrm{T}}$ and output vector is $\boldsymbol{\tau}$. For basic SVM training we have calculated $\boldsymbol{\tau}$ using equation (7). That is no longer possible since we have assumed that model has changed and it is unknown to robot, so other way of calculating desired output is needed.

Suppose that, at each time sample values of joint coordinates and velocities ($\mathbf{q}^t$ and [illegible]) are known. Damping coefficient for whole movement C is determined in advance, and stereo vision can be used to calculate distance to target position $(\mathbf{r}^i - \mathbf{r}_{\mathbf{d}})$. At each sampling time we are calculating new driving torques $\boldsymbol{\tau}^t$. When that torque is applied certain linear acceleration of robot's tip arises. Lets assume than we can measure that acceleration, either by using accelerometers or stereo vision, and that measured value is [illegible]. We can also calculate speed of robot's tip [illegible] by integrating measured acceleration. Since dynamics of the system has changed, calculated values of driving torques are not same as desired values. But it is desired that robot's tip moves in accordance with equation (1). Since we have calculated hand's speed, and measured acceleration we can calculate new distance to target position, for which measured movement is in accordance with equation (1).:

$$(\mathbf{r}^i_{NEW} - \mathbf{r}_{\mathbf{d}}) = \frac{1}{K}\left(C \text{[illegible]}\right) \tag{15}$$

Now, for input data $[\left(\mathbf{q}^t\right)^{\mathrm{T}}$ [illegible] $-\mathbf{r}_{\mathbf{d}})^{\mathrm{T}}\ C]^{\mathrm{T}}$ desired output is exactly $\boldsymbol{\tau}^t$. We are using data pairs calculated this way to adapt RBF. We were unable to determine desired output for certain input (since model is unknown), so we have determined input that fits known output.

4.5 Verification of RBF network-adaptation

Verification of control system adaptation process is similar to process verification described in section 4.3. We assign arbitrary chosen start and target positions and damping coefficient. Also, in some parts of simulation we add point mass to robot's tip. At each sampling instant, using state of the system (joint coordinates, join velocities, distance to target position and damping coefficient) driving torques are calculated. Those torques are applied at appropriate joints, resulting in acceleration of hand's tip. Using that acceleration, we calculate (equation 10) input-output data pair for RBF network adaptation, which is adapted in each time instant with learning rate $\eta = 5 \cdot 10^{-8}$.

Firstly, last two simulated cases (rising of hand with and without carrying weight (Fig 4)) were repeated, but this time control system was adapted on-line. Results of simulations are shown on Fig 5. When robot's arm is not carrying any weight, dynamics of robot are same as one used to train SVM, so it is expected that movements with and without adaptation are the same. It is evident from figures 4a and 5a, that those two motions are almost identical. When control system was adapted on-line positioning error has slightly decreased to 0.8 mm. Rise time has slightly increased form 1.8 s to 1.82 s.

On the contrary, when robot is carrying weight, differences between movements (Figs 4b and 5b) are significant. At the beginning motions look the same, since adaptation has not yet taken hold. After certain amount of time, when enough data pairs for RBF network adaptation were acquired, motions begin to differ. When there was no adaptation, tip of the hand veered off course and missed desired target position by 55 mm (Fig 4b). In contrast to that, when adaptation was performed robot's tip stopped within 1 mm from desired position, which is same as in case when no mass was carried. As expected, rise time is higher (1.95 s) than in case when there was no weight, because the control system was not sufficiently altered at the beginning of motion, and during that time hand's tip does travel in direction of target position. It should be noted that already during first hand movement RBF network has fully adapted to change that occurred.

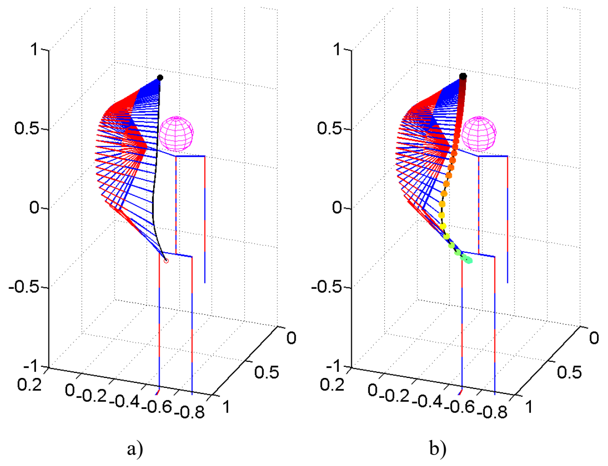

Fig. 5. Stick diagrams showing the hand's motion for the two different cases: a) movement without weight b) movement with weight. Hand's position is plotted every 75 ms. Black line represents trajectory of hand's tip. Black circle represents target position

In last case considered, we have simulated pick-and-place task performed by humanoid robot. Firstly, humanoid's hand moves without any weight to the pick-up position. Then, robot picks up object weighing 1 kg and moves it to desired 'place' position. After reaching that position hand will move to arbitrary position. This case is especially interesting since dynamics of the system changes two times, first time when robot picks up an object, and second time when robot puts down the object, and continues to move freely. Each phase is displayed separately on Fig 6.

During first phase (Fig 6a), robot moves without any mass attached to it, and it reaches desired position with accuracy of 1 mm and no overshoot. Then robot picks up an object and starts moving upwards to intermediate position depicted by square on Fig 6b. Since, positioning at this point is not of great significance, when hand's tip reaches within 50 mm of first intermediate position, it starts moving towards second intermediate position. When hand's tip reaches within 50 mm of second intermediate point, it starts moving towards 'drop off' position. During whole movement control system is constantly adapting to change that happened. The adaptation is carried out successfully, and positioning error at place where robot should place object is less than 1 mm, also hand positioned itself without overshoot. At this point robot releases object that it carried, and dynamics changes once again. Robot starts moving to arbitrary position constantly adapting RBF coefficients. It reaches its final position (Fig 6c) with accuracy of 1 mm, what tells us that adaptation was successful once again.

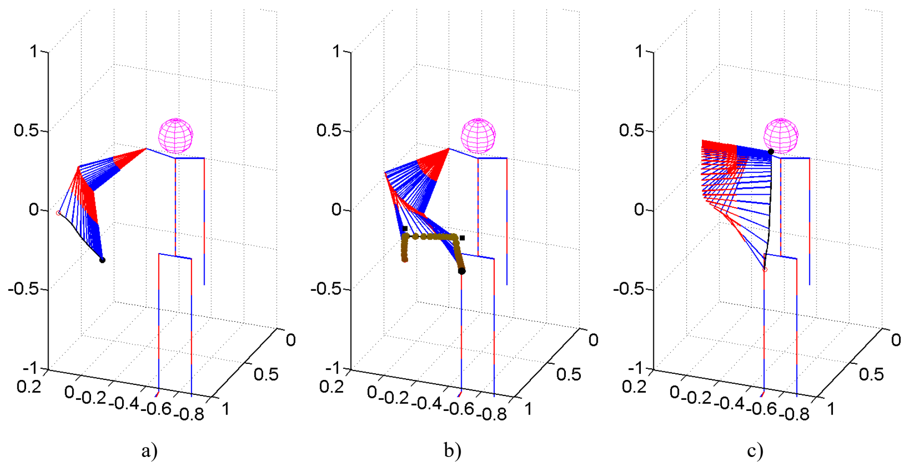

Fig. 6. Stick diagrams showing three stages of motion: a) hand movement to pick up location b) hand pick-and-place movement c) hand movement from pace to arbitrary position. Hand's position is plotted every 150 ms. Black line represents trajectory of hand's tip. Black circle represents target position. Black squares represent intermediate target positions

4 Conclusion

The work considers the possibility of using the SVM for the realization of humanoid reaching task. It is assumed that the human hand moves as a point mass m connected to the target position via a spring and a damper. The mass, the stiffness and the damping are selected so as to obtain a critically or overcritically damped rectilinear motion of the point mass. For an identical motion of the humanoid's hand, based on the known motion of the point mass, it is possible to calculate the joint driving torques. In this way, the input and output data of the training set are formed.

Since the relation between the input and output data is nonlinear, the SVM was trained using the kernel trick, the RBF serving as a kernel function. After completed training, the obtained SVM was validated.

Several examples demonstrated that the application of the driving torques obtained using the SVM produced an almost linear motion. Also, the presented examples have unequivocally shown that no overshoots arisen in the realization of the movement to the target position.

Using SVM obtained from model of humanoid arm, has some serious drawbacks if dynamics of system changes. Movement is vastly nonlinear, and robot's hand misses its target substantially. To address this issue we have employed EBP algorithm to alter existing control system in order to adapt to change that occurred. This approach was validated on several examples. Examples have shown, that adaptation to changes in the environment is performed successfully, during just one grasping motion. After adaptation, control system performs as if no change has occurred, and accuracy is even slightly improved.

All the presented examples are related to the task of the object's reaching by one hand. The humanoid robots, in a dynamic environment of humans, will be forced to realize two-handed manipulation in real time, so this will be one of the focuses of our future work. It should also be pointed out that the target position in the presented examples was related to stationary goal position. Since the objects in the human environment are mobile, it is important to realize the tasks of reaching objects that are movable. Therefore, the future investigations have to be concerned with the task of reaching mobile objects, as well as with the realization of two-handed manipulations using SVM and EBP.

Acknowledgments

This work was supported by the Ministry of Technology and Development of the Republic of Serbia under contract No. 44008.

References

1. Konczak J., Dichgans J.:The development toward stereotypic arm kinematics during reaching in the first 3 years of life, Exp Brain Res, vol .117, 346—354 (1997)
2. Konczak J., Borutta M., Dichgans J.:The development of goal-directed reaching in infants, II. Learning to produce task-adequate patterns of joint torque, Exp Brain Res, vol. 113, 465–474 (1997)
3. Grosso E., Metta G., Oddera A., Sandini G.:Robust Visual Servoing in 3-D Reaching Tasks, IEEE Tran. On Robotics And Automation, vol. 12, no. 5, 732—742 (1996)
4 Coelho J., Piater J., Grupen R.: Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot, Robotics and Autonomous Systems, vol. 37, 195—218 (2006)
5 Gaskett C., Cheng G.: Online Learning of a Motor Map for Humanoid Robot Reaching, Proc. 2nd Int. Conf. Computational Intelligence, Robotics and Autonomous Systems, Singapore, 2003.
6 Blackburn M., Nguyen H.:Learning in robot vision directed reaching: A comparison of methods, **Proc**. of the ARPA Image Understanding Workshop, Moterey, CA(1994)
7 Vapnik V., The nature of statistical learning theory, 2nd., Springer Heidelberg(2000)
8 Kecman V., Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models, The MIT Press, Cambridge Massachusetts (2001).
9 Potkonjak V., Vukobratović M., Babković K., Borovac B.:General Model of Dynamics of Human and Humanoid Motion: Feasibility, Potentials and Verification, Int. Jour. of Humanoid Robotics, vol. 3, no. 2, 21—48 (2006)