AN ANALYSIS OF THE ROBOT COLLISION AVOIDANCE USING THE PROGRAMMING THROUGH IMITATION

Aurel FRATU[1,*], **Michel DAMBRINE**[2, 3]

[1)] Prof., PhD, Dept. of Automatics, Electronics and Computers, „Transilvania” University of Braşov, Brasov, Romania
[2)] Prof., PhD, Univ Lille Nord de France, F-59000 Lille, France,
[3)] UVHC, LAMIH, F-59313 Valenciennes, France

***Abstract**: This paper presents an analysis of the collision avoidance of the cooperative robots using the programming through imitation. Each physical robot acts fully independently, communicating with corresponding virtual prototype and imitating her behavior. Each physical robot reproduces the motion of her virtual prototype. The estimation of the collision-free actions of the virtual cooperative robots and the transfer of the virtual joint trajectories to the physical robots who imitate there virtual prototypes, are the original ideas. We tested the present strategy on several simulation scenarios, involving two virtual robots and estimating collision-free actions, during of the cooperative tasks.*

***Key words:** virtual robots, cooperative robots, collision avoidance, motion imitation.*

1. INTRODUCTION

Important advancements were produced in the last years in fields of intelligent robots. From this point of view, promising applications were developed in robot-robot cooperation.

Cooperative robots are permanently in danger to be in collision. Therefore installations with cooperative robots in real world, require collision avoidance methods, which take into account the mutual constraints of the robots.

A key requirement for cooperative efficient operation is good coordination and reciprocal collision avoidance.

The contact of the robot with an obstacle must be detected and it will cause the robot to stop quickly and thereafter back off to reduce forces between the robot and environment. The problem of the contact with obstacle imposes the null velocity in the moment of the impact and to obtain the zero-velocity points on the pathway. The collision detection simply determines if two geometric objects are intersecting or not. The intersecting of two objects is possible in the virtual world, where the virtual objects can be intersected and there no exist the risk to be destroyed.

Using this strategy one detects collisions in all directions, protecting not only the physical end-effectors but also the work pieces and the physical robot itself.

The ability of predicting of the behavior of cooperative robots is important in design; the designers want to know whether the robot will be able to perform a typical task in a given time frame into a space with constraints.

The control engineer cannot risk a valuable piece of equipment by exposing it to untested control strategies. Therefore, a facile strategy for contact detection and collision avoidance, capable of predicting the behavior of a robotic manipulators, becomes imperative.

When the robots need to interact with their surrounding, it is important that the computer can simulate the interactions of the cooperative participants, with the passive or active changing environment in the graphics field, using virtual prototyping.

In this paper, we propose a fast method that simultaneously determines actions for two virtual robots that each must cooperate with other.

The actions for the cooperative tasks are computed for each virtual robot and are transferred, with a central coordination to corresponding physical robot which must imitate her virtual homonym. Thus, we will prove that our method guarantees the collision-free motion for each of the cooperative robots.

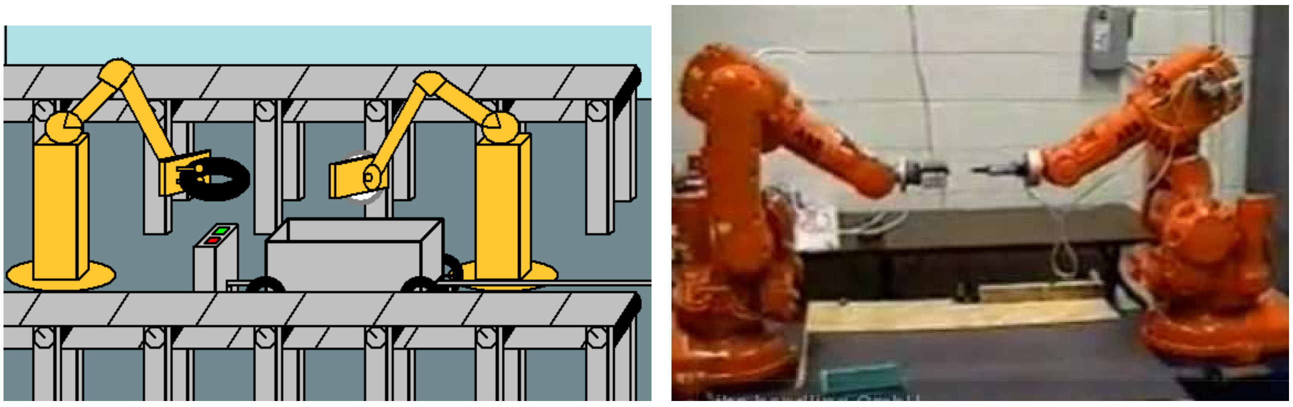
In this paper we develop a formally analyze of a new collision avoidance strategy for a group of two cooperative robots. We assume that our strategy is able to deduce the exact shape, position and velocity of the virtual obstacles and of the virtual robots, in the virtual environment. We transfer the behavior of the virtual cooperative robots, in the real world, to the physical cooperative robots.

This paper is focused on the collision avoidance through transfer the motion mapping from virtual space, in 3-D dimensional real space.

2. OVERVIEW OF IMITATION

Imitation is an important learning mechanism in many intelligent systems including robots. It is easy to recuperate kinematic information from virtual robot motion, using for example motion capture. Imitating the motion with stable robot dynamics is a challenging research problem [7].

[*] Corresponding author: Str. Mihai Viteazu nr. 5, Corp V, et III Cod. 500174, Jud. Braşov, Romania
Tel.: +40 268 418 836
E-mail addresses: *fratu@unitbv.ro* (A. Fratu), *michel.dambrine@univ-valenciennes.fr* (M. Dambrine)



a b

Fig, 1. A framework for robot learning by imitation. *a* − Virtual cooperative robots; b − Real cooperative robots.

In this paper, we propose a control framework for physical cooperative robots that uses capture data from their virtual prototypes and imitate them to track the motion in the real space avoiding the collision.

We focus on tracking joint angle trajectories, although some cooperative tasks may require tracking other quantities such as end-effectors trajectories which will be addressed in future work.

We present a model for which robust controller can be easily designed. A typical example is a linear quadratic regulator (LQR) [4], which we will use for our examples.

The tracking controller tries to make the joints follow the reference trajectory specified by the motion capture data from the virtual robots.

Joint trajectory tracking is enabled by commanding desired joint accelerations based on joint angle and velocity errors as well as supply forward joint accelerations. The tracking controller then solves an optimization problem with a quadratic cost function including errors from desired inputs and joint accelerations [1].

We will demonstrate the tracking ability of the proposed controller with dynamics simulation that takes into account joint velocity and torque limits. We apply the controller to tracking motion capture clips of two cooperative robots who accomplish a collaborative task. The resulting robot motion clearly preserves the original behavior of each virtual robot.

In addition, the controller does not require intensive pre-processing of motion capture data, which makes it potentially applicable to real time applications.

In this paper, we propose an approach to achieving pathway acquisition in robots programming, using imitation strategy.

The framework for our method is shown in Fig. 1.

First, a motion capture system transforms Cartesian position of virtual robot structure to virtual joint angles based on kinematic model. Then, the joint angles are converted in binary words and transferred to real robot joint controllers via intelligent interface. After this we employ the control loops structure to establish relationships between the virtual and real robot control systems.

We employed dimensionality reduction to represent posture information in a virtual low-dimensional space [3].

Optimization of the real robots behavior is performed in the low dimensional virtual space using the virtual robots.

In particular, for reciprocal correspondence, sensory feedback data are recorded from the real robot during motion [2]. A causal relationship between actions in the low dimensional virtual space and the expected sensory feedback is learned. This learned sensory motor mapping allows virtual motion dynamics to be optimized.

As well an inverse mapping from the real joint space, back to the original virtual joint space is then used to generate optimized motion. We present results demonstrating that the proposed approach allows a real robot to learn move based exclusively on virtual robot motion capture without the need for a detailed physical model of the robot.

3. MOTION CAPTURE AND KINEMATIC MAPPING

A. *Motion Capture Data Processing*

We transfer via intelligent interface the joint angle data from a motion capture system to a kinematic model for an anthropomorphic robot. To generate the desired motion sequence for the real robot, we capture the motions from a virtual robot model and map these to the joint settings of the physical robot.

Initially, a set of virtual postures is created to the virtual robot and the pictures’ positions are recorded for each posture, during motion. These recorded pictures’ positions provide a set of Cartesian points in the 3D capture volume for each posture.

To obtain the final robot posture, the virtual pictures’ positions are assigned as positional constraints on the physical robot. To derive the joint angles one use standard inverse kinematics (IK) routines.

The IK routine then directly generates the desired joint angles on the robot for each posture.

We assume to use the virtual robot prototypes and the motion capture systems to obtain the reference motion data, which typically consist of a set of trajectories in the Cartesian space.

The data is obtained using a motion capture channel taking into account the joint motion range. Due to the joint limits and the difference between the kinematics of the virtual robot and real robot, the joint angle data are pre-processed.

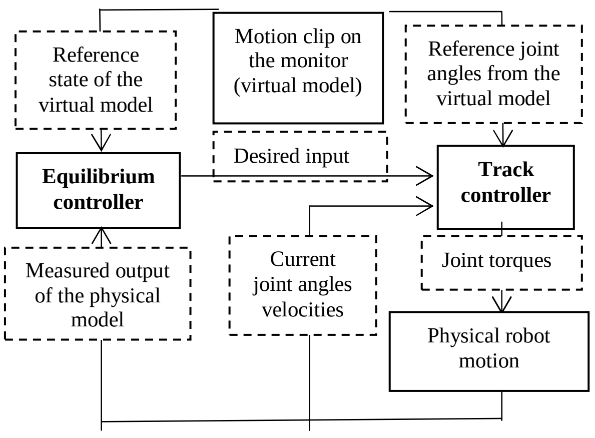

Reference state of the virtual model
Motion clip on the monitor (virtual model)
Reference joint angles from the virtual model
Equilibrium controller
Desired input
Track controller
Measured output of the physical model
Current joint angles velocities
Joint torques
Physical robot motion

Fig. 2. Overview of the controllers.

In our pre-processing, we assume that both virtual and physical robots are on the scene at the same time and estimate the correct arms position and orientation.

We then compute the inverse kinematics for new posture to obtain the cleaned joint angles and retain the difference from original joint angles.

At each frame during control, we add the difference to the original data to obtain the cleaned reference joint angles. This correction is extremely simple and our controller does not require supplementary cleanup.

B, Controller

Figure 2 shows the overview of the controller. The two main components are an equilibrium controller and a tracking controller. The equilibrium controller is responsible for keeping the whole physical structure in equilibrium, usually using a controller designed for a simplified dynamics model, such as Linear Quadratic Regulator (LQR). The output of the equilibrium controller is the desired input to the track controller.

The tracking controller is responsible for making every joint track the desired trajectory. It solves an optimization problem that respects both joint tracking and desired inputs to the simplified model and obtains the joint torques to be commanded to the real robot.

4. CONTROLLER COMPOSITION

A. Notations and Basic Equations. We denote the number of actuated joints of the robot by N_j. The total degrees of freedom (DOF) of the robot are then N_j including the DOF of type translation and rotation of the virtual joints. The robot configuration is uniquely defined by the generalized coordinate q. We also denote the generalized force by τ_J.

We suppose that virtual robots usually move with some of their links in contact with the virtual environment. More of that, one can intersect their virtual structure without risk to be destroyed.

Let N_c denote the number of links in contact with the environment. We represent the linear and angular velocities of the i-th contact link by a 6-dimensional vector $\dot{X}_{ci}$. The relationship between the generalized velocity $\dot{q}$ and $\dot{X}_{ci}$ is written as:

$$\dot{X}_{ci} = J_{ci}\,\dot{q} \tag{1}$$

where J_{ci} is the Jacobian matrix of the i-th contact link's position and orientation with respect to the generalized coordinates. Differentiating Eq. (1), we obtain the relationship of the accelerations:

$$\ddot{X}_{ci} = J_{ci}\,\ddot{q} + \dot{J}_{ci}\,\dot{q} \tag{2}$$

We define the compound contact Jacobian matrix $\boldsymbol{J}_c$ by:

$$J_c = \begin{pmatrix} J_{c1} \\ J_{c2} \\ J_{c3} \\ \\ J_{cNc} \end{pmatrix} \tag{3}$$

Because the source joint is not actuated, we can only control the joint torque vector τ_J. In addition, each of the N_c links in contact with the environment receives contact force f_{ci} and moment around the link local frame n_{ci} (i = 1, 2, …, N_c). We also define the compound contact force/moment vector by:

$$f_c \;=\; f_{c1}^T\, n_{c1}^T \;\cdots\; f_{cNc}^T\, n_{cNc}^T \tag{4}$$

The equation of motion of the robot is written as

$$M\ddot{q} + c = N^T\tau_J + J_c^T f_c \tag{5}$$

where M is the joint-space inertia matrix and C is the sum of Coriolis, centrifugal and gravity forces. Matrix $\boldsymbol{N}$ is used to map the joint torques into the generalized forces and has the form:

$$N = \begin{pmatrix} 0_{N_j \times 6} & 1_{N_j \times N_j} \end{pmatrix} \tag{6}$$

where 0* and 1^* are zero and identity matrices of the sizes indicated by their subscripts respectively.

Figure 3 shows the structure of the equilibrium controller. The equilibrium controller consists of two main components: a regulator to compute the input to the simplified model to keep it in equilibrium, and an observer to estimate the current state of the virtual robot prototype based on measurements.

We can use any simplified model as long as it represents the dynamics of the virtual robot and an equilibrium controller can be designed. A typical example is a linear system, for which a regulator can be easily designed by optimal control.

Details. Let us assume that the simplified model is linear and represented by the following state-space differential equation:

$$\dot{x} = Ax + Bu, \tag{7}$$

$$y = Cx, \tag{8}$$

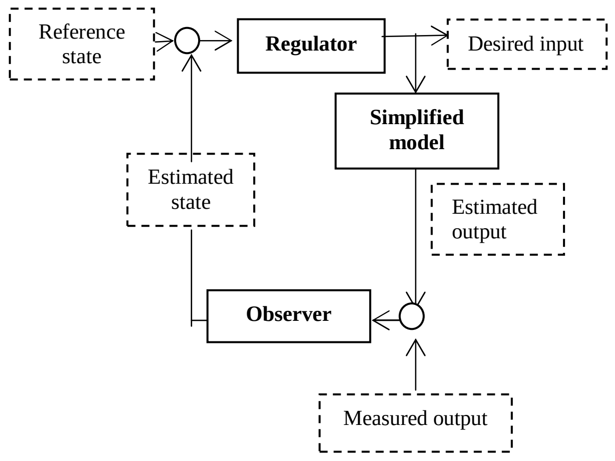


Fig. 3. Overview of the equilibrium controller.

where $\boldsymbol{x}$ is the state vector, u is the input, and y is the output of the simplified model. Also assume that we have designed a state feedback controller for equilibrium:

$$u = K(x_{ref} - x) \tag{9}$$

where K is a constant gain matrix and x_{ref} is a reference state, typically computed from the reference motion.

The observer compares the estimated and actual outputs to update the state estimate $\hat{x}$ as:

$$\dot{\hat{x}} = A\hat{x} + Bu + F(\hat{y} - y) \tag{10}$$

where F is the observer gain and $\hat{y} = C\hat{x}$ is the estimated output. Because we do not have access to real state, we replace the state $\boldsymbol{x}$ with its estimate $\hat{x}$ in Eq. (9):

$$u = K(x_{ref} - \hat{x}) \tag{11}$$

Using Eqs. (7), (8), (10) and (11), we obtain the following system of the estimated state and new input:

$$u_b = (x_{ref}^T y^T)^T ;$$

$$\dot{\hat{x}} = A_b\hat{x} + B_b u_b , \tag{12}$$

where:

$$A_b = A - BK - FC ,$$
$$B_b = B - F .$$

Equation (12) describes how to estimate the current state of the simplified model based on a reference state and measured output.

The estimated state and input to the simplified model computed by Eq. (11) will be used as the input to the tracking controller.

The system of tracking controller for each joint consists of two local controllers and a joint torque optimization.

B. Local Controllers. The local controllers compute the desired accelerations of joint and contact links based on the reference and current position and velocity as well as the reference accelerations [5].

In the joint controller, the desired acceleration $\hat{\ddot{\boldsymbol{q}}}$ is computed as follows at each joint:

$$\hat{\ddot{q}} = \ddot{q}_{ref} + k_d(\dot{q}_{ref} - \dot{q}) + k_p(q_{ref} - q) \tag{13}$$

where q is the current joint position, q_{ref} is the reference joint position in the captured data, and k_p and k_d are constant position and velocity gains that may be different for each joint.

We assume that the position and orientation of the virtual joints is available by computing the kinematics. We can therefore compute the desired linear and angular accelerations of the virtual joints, and combine them with all desired joint accelerations to form the desired acceleration vector $\hat{\ddot{q}}$.

Control law (13) is the same as the one used in resolved acceleration control except that the virtual joint is not actuated and the desired acceleration may be altered by the optimization part described later.

C. Optimizer. The task of the optimizer is to compute the control inputs based on the information obtained so far $\hat{\ddot{q}}$, $\hat{\ddot{X}}_c$ and the desired input to the simplified model obtained by the equilibrium controller. In most cases, however, these conditions conflict with each other. We therefore perform an optimization to compute a set of joint torques that respects all of these quantities.

The unknowns of the optimization are the joint torques τ_J and contact forces f_c.

The cost function to be minimized is:

$$Z = Z_q + Z_c + Z_\tau + Z_f \tag{14}$$

and each of the five terms will be described in detail in the following paragraphs.

The term Z_s addresses the error from the desired input to the simplified model. We consider a mapping from the simplified model the torque of a representative joint.

The term Z_q denotes the error from the desired joint accelerations, i.e.,

$$Z_q = \frac{1}{2}(\hat{\ddot{q}} - \ddot{q})^T W_q (\hat{\ddot{q}} - \ddot{q}) \tag{15}$$

The term Z_c denotes the error from the desired contact link accelerations, i.e.,

$$Z_c = \frac{1}{2}(\hat{\ddot{X}}_c - \ddot{X}_c)^T W_c (\hat{\ddot{X}}_c - \ddot{X}_c) \tag{6}$$

The term Z_τ is written as:

$$Z_\tau = \frac{1}{2}(\hat{\tau}_J - \tau_J)^T W_\tau (\hat{\tau}_J - \tau_J) , \tag{17}$$

where $\hat{\tau}_J$ is a reference joint torque, which is typically set to a zero vector and hence Z_τ acts as a damping term for the joint torque.

The term Z_f has a similar role for the contact force, i.e.,

$$Z_f = \frac{1}{2}(\hat{f}_c - f_c)^T W_f (\hat{f}_c - f_c), \quad (18)$$

where $\hat{f}_c$ is a reference contact force, which is also typically set to the zero vector.

Using Eqs. (2) and (5), the cost function can be converted to the following quadratic form:

$$Z = \frac{1}{2} y^T A y + y^T b + c, \quad (19)$$

where $y = (\tau_J^T \; f_c^T)^T$ is the unknown vector.

The optimization problem has an analytical solution:

$$y = -A^{-1} b. \quad (20)$$

D. Considering Contact Force and Hardware Limits. We have so far assumed that any contact force is available. Real hardware has limitations in joint angles, velocities and torques. We could add inequality constraints to enforce these constraints, but solving the optimization problem would take significantly longer than simply using Eq. (20).

We deal with these limitations by adjusting the parameters in the optimization instead of adding constraints, hence without changing the solution (20). The drawback is that the limitations are not always met, but the expectation is that the balance controller can compensate for the difference between approximate and exact solutions.

For the contact force limitations, we set larger values for elements of $\boldsymbol{W}_f$ corresponding to the frictions and moments.

To address the joint torque limit, we utilize the reference joint torque used in Eq. (17). If any of the joint torques exceeds its limit at a sampling time, we set the corresponding reference torque to the limit in the next sampling time and increase the influence. We can therefore expect that the excess torque would be relatively small and thus having little effect even if the torque is saturated by the limit.

5. SIMULATION FRAMEWORK

The basic essence of our framework is to describe each rigid object in the planning scene as a dynamical system, which is characterized by its state variables (i.e. position, orientation, linear and angular velocity).

In this framework, a robot arm can be a collection of rigid bodies, subject to the influence of various forces in the workspace, and restricted by various motion constraints.

This transforms a motion planning problem into a problem of defining suitable constraints, and then simulating the rigid body dynamics of the scene with each constraint acting as a virtual force on the objects such as the collision will be avoided.

This current method only deals with static data sets. Even, if this is not an issue for off-line computations, it prevents many uses in real-world applications since very small time steps are required to ensure stability. For real-world applications various ways to overcome this problem have been used [6].

We used a dynamics simulator called Robot Imitating Platform (RIP) with rigid-body contact model, developed at University of Brasov [8], who's precision has been demonstrated in some simulation settings for cooperative work using Virtual Reality.

The RIP is an architecture that provides libraries and tools to help software developers in programming. RIP is focused on 3D simulation of the dynamics systems and on a control and planning interface that provides primitives for motion planning by imitation.

Also is a object-oriented infrastructure for the integration of controllers, as well as the integration of planners with controllers to achieve feedback based planning, In particular, RIP is used to provide concrete implementations of motion planners. The proposed infrastructure, however, allows the definition of more complex problems such as planning among moving obstacles. The joint kinematics and inertial parameters are derived from the CAD model.

We used experimentally-verified joint motion range and joint torque limit information as well as the design specification for the joint velocity limit.

The joint motion range constraint is enforced during the inverse kinematics computation, but we did not consider the joint motion range in simulation assuming that the joints track the reference trajectory well enough. If a joint velocity comes close to the limit, we add a strong damping torque to reduce the speed.

If the optimized joint torque exceeds the limit, it is reset to the maximum value before the simulator computes the joint acceleration

The second contribution of this work involves the development of a platform for composing and evaluating controllers and motion planners.

This platform, called RIP utilizes a framework which allows both high level and low level controllers to be composed in a way that provides complex interactions between many robots.

6. CONCLUSIONS

Collision detection strategy is based on identifying the zero-velocity impact points between virtual objects in the moment of the impact. The null velocity in the moment of the impact requires a highly accurate model of robot dynamics and the environment in order to achieve the collision avoidance.

So, the problem of the contact detection is better analyzed on the virtual prototypes in the virtual environment where one may predict there behavior. The problem of the contact detection in the virtual environment on the virtual robots is important for the reason that this built-in function is proven superior to other collision detection devices.

The contact of two objects is possible in the virtual world, where the virtual objects can be intersected and there no exist the risk to be destroyed.

Learning approach such as learning by imitation is more flexible and can adapt to environmental change. This method is typically directly applicable to cooperative robots due the possibility to transfer the virtual joint trajectories from virtual space to the real space of the physical robots.

Programming real robots, especially to perform the behavior of the virtual robots is accomplished by imitation using virtual robots motion data capture.

The real (physical) robot will imitate her virtual prototype; it has no supplementary devices for collision avoidance, and gives it higher reliability and more cost efficiency. Also, since there is no device attached to the real robot tool, one not extends the tool offset distance, which allows bigger maximum tool weight and better reorientation performance.

REFERENCES

[1] V. Zordan and J. Hodgins, *Motion Capture-Driven Simulations that Hit and React*, Proceedings of ACM SIGGRAPH Symposium on Computer Animation, San Antonio, TX, July 2002, pp. 89–96.

[2] D. Silver, *Cooperative path finding*, The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'05), pp. 23–28, 2005.

[3] J. Pettre, J. Ondrej, A.-H. Olivier, A. Cretual, and S. Donikian, *Experiment based modeling, simulation and validation of interactions between virtual walkers*, In Symposium on Computer Animation. ACM, 2009.

[4] P. Reist and R. Tedrake, *Simulation-based LQR-trees with input and state constraints*, IEEE International Conference on Robotics and Automation (ICRA), pp, 5504–5510, 2010.

[5] K. Gold, *An information pipeline model of human-robot interaction*, Proceedings of the 4th ACM/IEEE international conference on Human robot interaction, pp. 85−92, New York, USA, 2009. ACM, doi:http://doi.acm.org/10.1145/1514095.

[6] A. Powers, S. Kiesler, S. Fussell, and C. Torrey, *Comparing a computer agent with a humanoid robot*, Proceedings of the ACM/IEEE international conference on Human-robot interaction (HRI '07). ACM, New York, USA, pp. 145−152, 2007.

[7] B. Price and C. Boutilier, *Accelerating reinforcement learning through implicit imitation*, Journal of Artificial Intelligence Research, vol. 19, 2003, pp. 569–629.

[8] A. Fratu, *Collision Prevention Method and Platform for a Dynamic Group of Cooperative Robots Who Communicate Wirelessly*, Revue RECENT, Vol. 12 (2011), No. 2 (32), pp. 131−134.