Multi-agent Cooperative Manipulation with Uncertainty: A Neural Net-Based Game Theoretic Approach

Qingguo Li and Shahram Payandeh
Experimental Robotics Laboratory(ERL)
School of Engineering Science
Simon Fraser University, 8888 University Drive
Burnaby, B.C., Canada V5A 1S6
qlib@cs.sfu.ca shahram@cs.sfu.ca

Abstract— **This paper proposes a novel planning method for multi-agent dynamic manipulation in the plane. The objective of planning is to find optimal forces exerted on the object by agents with which the object can follow given trajectory. The main contributions of the proposed approach is: First, through integrating of noncooperative game and neural-net approximation, the planner can deal with unknown pressure distribution effectively. Second, by introducing cooperative game between agents, the forces exerted by agents distributed optimally. Based on the dynamic model of the pushed object, the planing problem is solved in two levels in hierarchical manner. In the lower control level, generalized force inputs are designed by using minimax technique to achieve the tracking performance. The design procedure is divided into three steps. At first, a linear nominal control design is obtained via full-state linearization with desired eigenvalue assignment. Next, neural network systems are constructed to approximate and are tuned to eliminate the uncertainties related to pressure distribution. Finally, a minimax control scheme is specified to optimally attenuate the worst-case effect of the uncertainties due to residual of approximation and external disturbance to achieve a minimax tracking performance. In the coordination level, cooperative game is formulated between agents to distribute the generalized force, and the objective of the game is to minimize the worst case interaction force between agents and object. Simulations are carried out for three-agent cooperative manipulation, results demonstrate the effectiveness of the proposed planning method.**

I. Introduction

In robot manipulation, one of the basic task is to move object from an initial configuration to a goal configuration. One possible method is to grasp objects rigidly and then move them. The other is to move objects by using nonprehensile techniques[11]. Dynamic manipulation includes hopping, juggling, tapping, and batting, and impulse planar manipulation. Impulse planar manipulation were studied in [5]. Most work in dynamic manipulation were considering to use single agent to manipulate the objects, and the planning are carried out under the assumption of known friction and pressure distribution.

Asides, owning to the industrial needs and desire to understand human cooperative behavior, there has been much concern on multi-agent manipulation. In [4], manipulation protocols for a small team of mobile robots are developed to push large boxes. In [12], a coordinated manipulation method is developed for reorienting polygonal object in a plane. In [1], distributed cooperative strategies are proposed for a group of cooperative behavior-based mobile robots to handle an object. The common assumption they made is the motion of the object under pushing is quasi-static, and none of them discussed the problem of optimality of cooperation between robotic agents. In order to address the dynamic cooperative behavior, in [6], a nonprehensile cooperative dynamic manipulation model is proposed, in this model, two robotic agents make point contacts with the object, also relaxing the quasi-static assumption. The motion of the object under two agents pushing was modeled as a nonlinear system. Here we assume the finger agents are position/force controllable. The objective of cooperative planning is to find suitable interaction forces between agents and object, and the object will follow a given trajectory while it is pushed by agents with planned forces. The interest of the paper is to find an optimal solution such that the planned forces satisfy given criterion. The implementation of the planned forces will not discussed in this paper, which is a position/force control problem, and it has been studied extensively in the literature[10]. One centralized planning approach has been proposed in [8], backstepping techniques and quadratic programming are integrated to solve the planning problem, the assumption is that there is no uncertainty on pressure distribution. However, in practice the pressure distribution of the pushed object can not be known exactly as *a priori*, this will introduce uncertain frictional forces to the system. Hence, the introduction of an alternative approach to treat the planning problem with uncertainty is interesting. The main objective of this paper is to deal with the dynamic multi-agent manipulation planning problem under unknown pressure distribution. And the uncertainties due to pressure distribution will be approximated by using neural networks.

By introducing the cooperative manipulation into control framework, the coordination problem can be formulated as a control theoretic problem. The development of zero-sum differential games have inspired important applications in a special class of worst-case controller design problems-H^∞ optimal control[2], and it has been widely discussed for robustness and its capability of disturbance attenuation in linear and nonlinear control systems[2]. Neural networks possess a number of useful properties, such as the universal approximation capability, performance improvement through on- and offline learning, and distributed processing, which motivates its applications in control and signal processing. In order to deal with the effect of approximation error on the tracking error, several neural network-based H^∞ control schemes for nonlinear systems have been proposed [9][3]. Motivated by these factors, neural net-based H^∞ technique is utilized to cope with the uncertainty of pressure distribution in the cooperative planning problem.

In the dynamic pushing model, the forces exerted by agents need to obey some constraints. First, in order to avoid sliding between fingers and object, a constraint imposed on the force vector is that it must lie inside the friction cone of the contact surface. The other constraint is that in pushing action, the force vector can only direct inside the object. Owning to these constraints, it is difficult to solve the planning problem and find optimal forces directly based on the pushing model. Instead, by introducing generalized forces(wrench) into the model, the overall planning process is divided hierarchically into lower control level and higher coordination level. In the lower level, we need to find generalized forces applied to the object, such that the object follows the given trajectory and obtain desired velocities, this is called a tracking problem in control literature. The design procedure is divided into two steps. At first, a linear nominal control design is obtained via full-state linearization with desired eigenvalue assignment. Next, neural network systems are constructed to approximate and eliminate the uncertainties related to pressure distribution. Finally, a minimax control scheme is specified to optimally attenuate the worst-case effect of the uncertainties due to residual due to approximation to achieve a minimax tracking performance.

In the coordination level, the task is to distribute the generalized forces among agents optimally. The formulation of coordination depends on the choice of performance index. Here each robot agent try to minimize the worst-case interaction force between itself and the object, each agent can make its decision and has its own objective, thus the coordination problem can be casted as a cooperative game be tween agents. Since all of the constraints are convex, there will exist a Pareto optimal solution to the cooperative game, and the Pareto optimal solution is a set of optimal decision points. In order to single out a compromise Pareto solution and guarantee the worst case performance, the cooperative game can be transfered as a minimax problem, the goal is to minimize the maximal interaction forces. Through a simple transformation, the minimax problem will be solved by using linear programming.

Having introducing the problem and review the literature, we outline the remainder of the paper. In section 2, we formulate the multi-agent pushing as a nonlinear system with uncertainty, and outline the planning problem. Section 3 presents the results on planning of the optimal force inputs using neural net-based H^∞ technique and cooperative game. In section 4, simulations are carried out on three-agent manipulation to demonstrate the proposed planning method. The final section concludes with a summary of our results and suggestions for future work. The detail derivation of the dynamic model for multi-agent manipulation can be found in appendix.

II. Problem Formulation

In this paper, we consider the cooperative manipulation using multiple robotic agents pushing an object, and each robotic agent make frictional point contact with the object. Through the analysis of the motion of the object under pushing, the pushing process is modeled as a nonlinear system with uncertainty, and the planning problem is formulated as a nonlinear tracking control and minimax optimization problem.

A. Model of Dynamic Multi-agent Manipulation

Consider the planar motion of the object under three-agent pushing, XOY is the global coordinate, xoy is the local coordinate associated with the object, and assign o at the center of mass. The configuration space of the object is defined as (X_o, Y_o, θ). Where (X_o, Y_o) gives the position of the local coordinate, while θ denotes the orientation of the object. Denote the state of the system as $\overline{x} = (x_1, x_2, \cdots, x_6)^T = (X_o, \dot{X}_o, Y_o, \dot{Y}_o, \theta, \dot{\theta})^T$, we can derive following affine nonlinear system for three-agent dynamic manipulation

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \\ \dot{x}_5 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix} \\ \begin{bmatrix} \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} \frac{F_X}{m} \\ \frac{F_Y}{m} \\ \frac{\eta}{I_0} \end{bmatrix} + Mt \begin{bmatrix} \cos x_5 & -\sin x_5 & 0 \\ \sin x_5 & \cos x_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \\ \begin{bmatrix} \mathbf{W_1} & \mathbf{W_2} & \mathbf{W_3} \end{bmatrix} \begin{bmatrix} F_{1n} \\ F_{1t} \\ F_{2n} \\ F_{2t} \\ F_{3n} \\ F_{3t} \end{bmatrix} \end{cases} \tag{1}$$

where

$$Mt = \begin{bmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{I} \end{bmatrix}$$

$F_{in}, F_{it}, i = 1, 2, 3$ are the normal and tangential components of the contact force generated by agents. $\mathbf{W_i}$ is the wrench matrix for the agent that transfers the pushing forces into global coordinate. It transforms the contact force into local coordinate. F_X, F_Y are the friction forces in global coordinate, and T is the torque respect to the center of mass generated by the friction force. Due to the uncertainty of pressure distribution, the friction forces and torque are always unknown. However, they are functions of the linear and rotational velocities. And in this paper, we will use neural network as universal approximator to estimate the functions, and cancel the effect of the uncertainties. (please refer to [8] for the derivation of the model)

B. Cooperative Manipulation Planning Problem

The dynamical cooperative manipulation planning problem can be described as follows: Given an object on a plane at a known initial configuration, find the forces exerted by each finger (or agent), such that the object move to a specified goal configuration with specific velocities. The inertial forces are not negligible compared to quasi-static manipulation, the planning and cooperation of the fingers must be developed based on the dynamic model of motion described by (1). For manipulation, there also exist constraints on the inputs, for example, the pushing direction must lay inside the friction cone, otherwise there will be sliding between the fingers and object; the agent can only push the object, without pulling. The cooperative manipulation planning problem is formulated as follows:

Given a task of a continously differentiable and uniformly bounded trajectory $\mathbf{q_d} = (\mathbf{X_{or}}, \mathbf{Y_{or}}, \theta_\mathbf{r})$ and corresponding velocities $\dot{\mathbf{q}}_d = (\dot{X}_{or}, \dot{Y}_{or}, \dot{\theta}_r)$, design the state feedback control input $\mathbf{u(t)} = (\mathbf{F_{1n}}, \mathbf{F_{1t}}, \mathbf{F_{2n}}, \mathbf{F_{2t}}, \mathbf{F_{3n}}, \mathbf{F_{3t}})^\mathbf{T}$ such that the state of the system (1) follows the desired trajectory and minimize worst-case normal interaction forces at any instant time t. The objective function $M(\mathbf{u})$ is chosen as

$$M(\mathbf{u})(\mathbf{t}) = \min_{\mathbf{u(t)}} \max_{\mathbf{i}} \{\mathbf{F_{1n}}, \mathbf{F_{2n}}, \mathbf{F_{3n}}\} \tag{2}$$

III. Planning for Cooperative Manipulation

In this section, we will solve the cooperative manipulation planning problem and compute the forces F_{in}, F_{it} generated by each agent such that the object follows the given trajectory and obtains desired velocities.

The object on a plane under pushing have three degrees of freedom, and it will need three independent control inputs to fully control the object to follow any given trajectory. Here we introduce F_{OX}, F_{OY} and T_O as three generalized control inputs to the system, and let $\overline{\mathbf{u}} = (\overline{u}_1, \overline{u}_2, \overline{u}_3)^T = (F_{OX}, F_{OY}, T_O)^T$, and rearrange the system (1) as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \\ \dot{x}_5 \\ \dot{x}_2 \\ \dot{x}_4 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \\ \frac{F_X}{m} \\ \frac{F_Y}{m} \\ \frac{T}{I_0} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{T} \end{bmatrix} \overline{\mathbf{u}} \tag{3}$$

and with these control inputs, the system is fully actuated.

From (1), we know that the generalized control $\overline{\mathbf{u}}$ and original force inputs satisfy following relations

$$\overline{\mathbf{u}} = \begin{bmatrix} \cos x_5 & -\sin x_5 & 0 \\ \sin x_5 & \cos x_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{W_1} & \mathbf{W_2} & \mathbf{W_3} \end{bmatrix} \begin{bmatrix} F_{1n} \\ F_{1t} \\ F_{2n} \\ F_{2t} \\ F_{3n} \\ F_{3t} \end{bmatrix} \tag{4}$$

Thus the planning of the force inputs is divided into two phases hierarchically. First, generalized force and torque controller $\overline{\mathbf{u}}$ for the uncertain system (3) will be developed using neural net-based H^∞ tracking control design in order to cope with the uncertainty of pressure distribution. Then the coordination will be carried out on (4) to distribute the generalized control into forces generated by each finger agent. By considering the constraints, a cooperative game will be formulated to solve the problem.

A. Neural Net-based H^∞ Design for The Generalized Controller

Denote the configuration of the object as $\mathbf{q_1} = (\mathbf{X_o}, \mathbf{Y_o}, \theta)$, and $\mathbf{q_2} = (\dot{\mathbf{X}}_\mathbf{o}, \dot{\mathbf{Y}}_\mathbf{o}, \dot{\theta})$. By introducing the state vector

$$\mathbf{q} = \begin{bmatrix} \mathbf{q_1} \\ \mathbf{q_2} \end{bmatrix} \tag{5}$$

By considering the fact that the friction forces and torque are only functions of the velocities $\mathbf{q_2}$, the equation (3) is transferred into the following standard form:

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{q_2} \\ f(\mathbf{q_2}) \end{bmatrix} + \begin{bmatrix} 0 \\ Mt \end{bmatrix} \overline{\mathbf{u}} \tag{6}$$

where $f(\mathbf{q_2}) = (\frac{\mathbf{F_X(q_2)}}{\mathbf{m}} \quad \frac{\mathbf{F_Y(q_2)}}{\mathbf{m}} \quad \frac{\mathbf{T(q_2)}}{\mathbf{I_0}})^\mathbf{T}$, and I is the identity matrix.

Therefore we can define

$$\tilde{\mathbf{q}}_1 = \mathbf{q_1} - \mathbf{q_d} \tag{7}$$

as the tracking error that we would like to drive to zero. Since the nonlinearities appear together with the control in equation (6), feedback linearization is easy to design to cancel the nonlinearities. However, the approach is possible only while all the nonlinearities are well known. In oder to design the controller under unknown uncertainties, we need to estimate the uncertainties, here neural networks are introduced to approximate the nonlinearities $f(\mathbf{q_2})$ as $\hat{f}(\mathbf{q_2}, \mathbf{\Theta})$. Based

on the approximation, the state feedback control law can be designed as

$$\overline{\mathbf{u}} = M^{-1}(\ddot{\mathbf{q}}_d - K_1\tilde{\mathbf{q}}_1 - K_2\dot{\tilde{\mathbf{q}}}_1 + u_0 - \hat{f}(\mathbf{q_2}, \mathbf{\Theta})) \quad (8)$$

where $\ddot{\mathbf{q}}_d$ denotes the second order derivative of $\mathbf{q_d}$, Θ is a vector containing the tunable network parameters, and K_1, K_2 are 3×3 matrix to be designed and $\mathbf{u_0}$ is an auxiliary control signal yet to be specified.

Let

$$\hat{f}(\mathbf{q_2}, \mathbf{\Theta}) = \begin{bmatrix} \hat{f}_1(\mathbf{q_2}, \theta_1) \\ \hat{f}_2(\mathbf{q_2}, \theta_2) \\ \hat{f}_3(\mathbf{q_2}, \theta_3) \end{bmatrix} \quad (9)$$

be the neural network to approximate the nonlinearities $f(\mathbf{q_2})$, and the neural networks $\hat{f}_k(\mathbf{q_2}, \theta_\mathbf{k}), (\mathbf{k} = \mathbf{1, 2, 3})$ are composed of nonlinear neurons in hidden layers and linear neurons in the input and output layers, i.e,

$$\hat{f}_k(\mathbf{q_2}, \theta_\mathbf{k}) = \sum_{\mathbf{i=1}}^{\mathbf{P_k}} \theta_{\mathbf{ki}} \mathbf{H}(\sum_{\mathbf{j=1}}^{\mathbf{N}} \omega_{\mathbf{ij}}^{\mathbf{k}} \mathbf{q_{2j}} + \mathbf{m_i^k}), \quad \mathbf{k = 1, 2, 3} \quad (10)$$

For simplicity, only consider θ_k as the adjustable weighting parameters. denote

$$\theta_k = \begin{bmatrix} \theta_{k1} \\ \vdots \\ \theta_{kp_k} \end{bmatrix} \quad \text{and} \quad \xi_k = \begin{bmatrix} H(\sum_{j=1}^{N} \omega_{1j}^k \mathbf{q_{2j}} + \mathbf{m_i^k}) \\ \vdots \\ H(\sum_{j=1}^{N} \omega_{p_k j}^k \mathbf{q_{2j}} + \mathbf{m_{P_k}^k}) \end{bmatrix} \quad (11)$$

Here we choose the following hyperbolic tangent function as nonlinear neurons functions

$$H(\mathbf{q_2}) = \frac{\mathbf{e}^{\delta(\mathbf{q_2})} - \mathbf{e}^{-\delta(\mathbf{q_2})}}{\mathbf{e}^{\delta(\mathbf{q_2})} + \mathbf{e}^{-\delta(\mathbf{q_2})}} \quad (12)$$

where $\delta(.)$ is a function of $\mathbf{q_2}$. The weights ω_{ij}^k and the biases m_i^k for $1 \le i \le p_k, 1 \le j \le 3$, and $1 \le k \le 3$, are specified beforehand in this application, where p_k is the number of nonlinear neurons in the neural network system $\hat{f}(\mathbf{q_2}, \theta_\mathbf{k})$. The neural-network system $\hat{f}(\mathbf{q_2}, \mathbf{\Theta})$ can be denoted as

$$\hat{f}(\mathbf{q_2}, \mathbf{\Theta}) = \begin{bmatrix} \xi_1^T & 0 & 0 \\ 0 & \xi_2^T & 0 \\ 0 & 0 & \xi_3^T \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \mathbf{\Xi\Theta} \quad (13)$$

Now define the optimal parameter estimates Θ^* as follows

$$\Theta^* = arg \min_{\Theta \in \Omega_\theta} \max_{\mathbf{q_2} \in \mathbf{\Omega}} \|\hat{f}(\mathbf{q_2}, \mathbf{\Theta}) - \mathbf{f}(\mathbf{q_2})\| \quad (14)$$

where $\|.\|$ denotes the Euclidean norm, Ω_θ and Ω denote the largest compact sets that Θ and $\mathbf{q_2}$ belong to, respectively.

Substituting (8) and (7) into (6), yields,

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{q_2} \\ \dot{\mathbf{q}}_2 &= \mathbf{q_d^{(2)}} - \mathbf{K_1}\tilde{\mathbf{q}}_1 - \mathbf{K_2}\dot{\tilde{\mathbf{q}}}_1 \\ &\quad + u_0 + f(\mathbf{q_2}) - \hat{\mathbf{f}}(\mathbf{q_2}) \\ &= \mathbf{q_d^{(2)}} - \mathbf{K_1}\tilde{\mathbf{q}}_1 - \mathbf{K_2}\dot{\tilde{\mathbf{q}}}_1 \\ &\quad + u_0 + \Xi\Theta^* - \Xi\Theta + (f(\mathbf{q_2}) - \mathbf{\Xi\Theta}^*) \end{aligned} \quad (15)$$

Let us denote

$$\dot{e} = \begin{bmatrix} \tilde{\mathbf{q}}_1 \\ \dot{\tilde{\mathbf{q}}}_1 \end{bmatrix} = \begin{bmatrix} \mathbf{q_1 - q_d} \\ \mathbf{q_2 - \dot{q}_d} \end{bmatrix} \quad (16)$$

as the state error vector. And let

$$d(t) = f(\mathbf{q_2}) - \mathbf{\Xi\Theta}^* \quad (17)$$

be the optimal approximation error, and $\tilde{\Theta}(t)$ is defined as

$$\tilde{\Theta}(t) = \Theta^* - \Theta(t) \quad (18)$$

From (7), (15) and (16), we obtain

$$\begin{bmatrix} \dot{\tilde{\mathbf{q}}}_1 \\ \ddot{\tilde{\mathbf{q}}}_1 \end{bmatrix} = \begin{bmatrix} \dot{\tilde{\mathbf{q}}}_1 \\ -K_1\tilde{\mathbf{q}}_1 - K_2\dot{\tilde{\mathbf{q}}}_1 + u_0 + (\Xi\Theta^* - \Xi\Theta) + d \end{bmatrix} \quad (19)$$

the system in (19) represents the tracking error dynamics, it is a nominal linear uncertain system, the nonlinear uncertainties are modeled by d. A more convenient form of (19) could be

$$\dot{e} = Ae + Bu_0 + B(\Xi\Theta^* - \Xi\Theta) + Bd \quad (20)$$

where

$$A = \begin{bmatrix} 0 & I \\ -K_1 & -K_2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \quad (21)$$

The control parameters K_1, K_2 are selected such that A is Hurwitz and has desired eigenvalues such that the tracking dynamic (20) has a desired response if the system is free of uncertainty.

If the system is free of uncertainties due to the neural network approximation, by using control law in (8) with choice of linear control $u_0(t)$, the tracking error (19) will converge to zero asymptotically. However, in the case of uncertainty, the tracking performance will be deteriorated, and the system may be instable in worst case. In order to eliminate the effect of uncertainties, robust controller must be employed. The H^∞ control is one of the most efficient approach to attenuating the worst-case effect of uncertainties $d(t)$ on $e(t)$ in (20). Therefore, in the second step, the control signal $u_0(t)$ should be specified such that the worst-case effect of $d(t)$ on the tracking error $e(t)$ must be attenuated below a prescribed level ρ. The H^∞ tracking performance index is defined by[13]

$$\min_{u_0(t)} \max_{d(t)} \frac{\int_0^{t_f} [\|e(t)\|_Q^2 + \|u_0\|_R^2] \, dt}{\int_0^{t_f} \|d(t)\|^2 \, dt} \le \rho^2 \quad \forall d(t) \in L_2[0, t_f] \quad (22)$$

where $Q = Q^T > 0$ and $R = R^T > 0$ are the weighting matrices. However, in the case $e(0) \neq 0$, the performance must be modified as following

$$\begin{aligned} &\min_{u_0(t)} \max_{d(t)} \int_0^{t_f} (e^T(t)Qe(t) + u_0^T(t)Ru_0(t) \\ &-\rho^2 d^T(t)d(t))dt \le e^T(0)Pe(0) + \tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) \end{aligned} \quad (23)$$

where where η is the adaption gain, $P = P^T > 0$ is positive definite weighting matrices to be specified.

Subject to the tracking error dynamic equation in (20), let us define the cost function

$$\begin{aligned} J(e,u_0,d) &= \int_0^{t_f} e^T(t)Qe(t) + u_0^T(t)Ru_0(t) \\ &\quad -\rho^2 d^T(t)d(t)dt \end{aligned} \tag{24}$$

After some rearrangement, we obtain,

$$\begin{aligned} &J(e,u_0,d) = e^T(0)Pe(0) - e^T(t_f)Pe(t_f) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) - \tfrac{1}{\eta}\tilde{\Theta}^T(t_f)\tilde{\Theta}(t_f) \\ &+\int_0^{t_f}[e^T(t)Qe(t) + u_0^T(t)Ru_0(t) - \rho^2 d^T(t)d(t) \\ &+\tfrac{d}{dt}(e^T(t)Pe(t)) + \tfrac{1}{\eta}\tfrac{d}{dt}(\tilde{\Theta}^T(t)\tilde{\Theta}(t))]dt \\ &= e^T(0)Pe(0) - e^T(t_f)Pe(t_f) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) - \tfrac{1}{\eta}\tilde{\Theta}^T(t_f)\tilde{\Theta}(t_f) \\ &+\int_0^{t_f}[e^T(t)Qe(t) + u_0^T(t)Ru_0(t) - \rho^2 d^T(t)d(t) \\ &+\dot{e}^T(t)Pe(t) + e^T(t)P\dot{e}(t) \\ &+\tfrac{1}{\eta}\dot{\tilde{\Theta}}^T(t)\tilde{\Theta}(t) + \tfrac{1}{\eta}\tilde{\Theta}^T(t)\dot{\tilde{\Theta}}(t)]dt \end{aligned} \tag{25}$$

substituting (20) into the above equation, yields,

$$\begin{aligned} &J(e,u_0,d) = e^T(0)Pe(0) - e^T(t_f)Pe(t_f) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) - \tfrac{1}{\eta}\tilde{\Theta}^T(t_f)\tilde{\Theta}(t_f) \\ &+\int_0^{t_f}[e^T(t)(A^TP + PA + Q)e(t) + u_0^T(t)Ru_0(t) \\ &-\rho^2 d^T(t)d(t) + u_0^T(t)B^TPe(t) + e^T(t)PBu_0(t) \\ &+e^T(t)PB\Xi\tilde{\Theta}(t) + \tilde{\Theta}^T\Xi^TB^TPe(t) + \tfrac{1}{\eta}\dot{\tilde{\Theta}}^T(t)\tilde{\Theta}(t) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(t)\dot{\tilde{\Theta}}(t) + e^T(t)PBd(t) + d^T(t)B^TPe(t)]dt \end{aligned} \tag{26}$$

then we get the following result:

Theorem 1: For the system (3), if the control $\overline{\mathbf{u}}(t)$ is chosen as

$$\overline{\mathbf{u}} = Mt^{-1}(\mathbf{q_d^{(2)}} - \mathbf{K_1}\tilde{\mathbf{q}}_\mathbf{1} - \mathbf{K_2}\dot{\tilde{\mathbf{q}}}_\mathbf{1} + \mathbf{u_0} - \hat{\mathbf{f}}(\mathbf{q_2}, \mathbf{\Theta})) \tag{27}$$

with

$$\dot{\Theta} = \eta\Xi^TB^TPe(t) \tag{28}$$

$$u_0 = -R^{-1}B^TPe(t) \tag{29}$$

where $R = R^T > 0$ is a weighting matrix and $P = P^T > 0$ is the solution of the following algebraic Riccati-like equation

$$PA + A^TP + Q - PB(R^{-1} - \frac{1}{\rho^2}I)B^TP = 0 \tag{30}$$

Then the minimax tracking in (23) is guaranteed for a prescribed ρ and the corresponding worst-case $d^*(t)$ is of the form

$$d^*(t) = \frac{1}{\rho^2}B^TPe(t) \tag{31}$$

Proof: Introducing (30) to (26) we get

$$\begin{aligned} &J(e,u_0,d) = e^T(0)Pe(0) - e^T(t_f)Pe(t_f) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) - \tfrac{1}{\eta}\tilde{\Theta}^T(t_f)\tilde{\Theta}(t_f) \\ &+\int_0^{t_f}[e^T(t)(PA + A^TP + Q \\ &-PB(R^{-1} - \tfrac{1}{\rho^2}I)B^TP)e(t) + u_0^T(t)Ru_0(t) \\ &+u_0^T(t)B^TPe(t) + e^T(t)PBu_0(t) \\ &+e^T(t)PBR^{-1}B^TPe(t) \\ &-\rho^2 d^T(t)d(t) + d^T(t)B^TPe(t) \\ &+e^T(t)PBd(t) - \tfrac{1}{\rho^2}e^T(t)PBB^TPe(t)]dt \end{aligned} \tag{32}$$

From (27)-(30), and using the technique of completion of the squares we get

$$\begin{aligned} &J(e,u_0,d) = e^T(0)Pe(0) - e^T(t_f)Pe(t_f) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) - \tfrac{1}{\eta}\tilde{\Theta}^T(t_f)\tilde{\Theta}(t_f) \\ &+\int_0^{t_f}[Ru_0(t) + B^TPe(t)]^TR^{-1}[Ru_0(t) + B^TPe(t)] \\ &-(\rho d(t) - \tfrac{1}{\rho}B^TPe(t))^T(\rho d(t) - \tfrac{1}{\rho}B^TPe(t))]dt \end{aligned} \tag{33}$$

From the dynamic game theory[2], we know that the control $u_0(t)$ try to minimize $J(e_0,u_0,d)$, while the disturbance $d(t)$ wants to maximize the performance index $J(e_0,u_0,d)$, from (33), we obtain the optimal control as (27) and the worst-case $d^*(t)$ as (31), then

$$\begin{aligned} &\min_{u_0(t)}\max_{d(t)} J(e,u_0,d) = e^T(0)Pe(0) \\ &+\tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) - e^T(t_f)Pe(t_f) - \tfrac{1}{\eta}\tilde{\Theta}^T(t_f)\tilde{\Theta}(t_f) \\ &\le e^T(0)Pe(0) + \tfrac{1}{\eta}\tilde{\Theta}^T(0)\tilde{\Theta}(0) \end{aligned} \tag{34}$$

The above inequality holds when $P = P^T > 0$, and $R = R^T > 0$.

Remark 1: Generally, we choose $\rho < 0$ to attenuate $d(t)$ in order to achieve the robust minimax tracking performance. In the case $\rho \to \infty$, the H^∞ design is reduced to a H_2 optimal tracking control where no attenuation of $d(t)$ is considered. And in order to guarantee the positive definite solution of P, the Riccati-like equation (30) must satisfies following condition[13]

$$\rho^2 I \ge R \tag{35}$$

This tells us there is a tradeoff between attenuation level ρ and control input $u_0(t)$, i.e., If ρ is extremely small, It requires large $u_0(t)$.

From above analysis, a design procedure for the H^∞ controller can be summarized as follows.

Design Procedure for H^∞ controller

1. Specify K_1, K_2, determine A with desired eigenvalues.
2. choose the desired attenuation level ρ, selecting positive-definite matrix Q and R with $R \le \rho^2 I$.
3. Solve the positive-definite matrix P from the Riccati-like equation (30).
4. Choose the neural-network functions $\xi_k = [\xi_{k1}, \cdots, \xi_{kp_k}]$ for $k = 1, \cdots, 3$. in (11).
5. Compute the H^∞ control law (29) and $\overline{\mathbf{u}}$ as (27), and the parameter update law in (28).

Till now, we have designed the generalized controller for minimax tracking performance, Our next task is to redistribute it to the agents.

B. Coordination as A Minimax Problem

For a generalized controller (wrench) (27) obtained from H^∞ design, how to distribute the wrench between agents is the objective of the higher level coordination. The coordination between agents can be considered as game between agents. Each agent has an objective, i.e., minimizing the normal interaction force between itself and object, and can make its own decision through choosing the interaction forces component F_{in} and F_{it} for agent i. The problem can be written as

$$\begin{array}{l}\min_{\mathbf{u}}\{F_{1n}, \quad F_{2n}, \quad F_{3n}\} \\ \text{Subject to} \quad \mu_i F_{in} - |F_{it}| \geq 0, \quad i = 1,2,3 \\ F_{in} \geq 0, \qquad\qquad i = 1,2,3\end{array}$$

$$\begin{bmatrix} \cos x_5 & -\sin x_5 & 0 \\ \sin x_5 & \cos x_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{W_1} & \mathbf{W_2} & \mathbf{W_3} \end{bmatrix} \begin{bmatrix} F_{1n} \\ F_{1t} \\ F_{2n} \\ F_{2t} \\ F_{3n} \\ F_{3t} \end{bmatrix} = \begin{bmatrix} F_{OX} \\ F_{OY} \\ T_O \end{bmatrix} \tag{36}$$

where $\mathbf{u(t)} = (\mathbf{F_{1n}, F_{1t}, F_{2n}, F_{2t}, F_{3n}, F_{3t}})^{\mathbf{T}}$.

Since all the constraints in problem (36) are linear and convex, if the problem is strictly feasible, there will be a Pareto solution to the game. The definition of Pareto optimality is as follows: For an objective function $f = [f_1(\mathbf{x}), \mathbf{f_2}(\mathbf{x}), \cdots, \mathbf{f_m}(\mathbf{x})]$, a design variable vector $\mathbf{x}^* \in \mathbf{\Omega}$ is Pareto optimal if and only if there is no vector $\mathbf{x} \in \mathbf{\Omega}$, with the characteristics

$$\begin{array}{l} f_i(\mathbf{x}) \leq \mathbf{f_i}(\mathbf{x}^*) \quad \text{for all} \quad \mathbf{i = 1, \cdots, m} \\ \text{and} \quad f_i(\mathbf{x}) < \mathbf{f_i}(\mathbf{x}^*) \quad \text{for at least on} \quad \mathbf{i, 1 \leq i \leq m} \end{array}$$

From the definition, there may exists a set of Pareto optimal points. In order to obtain a single best compromise Pareto solution to the game, we consider the worst case interaction force between agents and object, the cooperative game can be transfered as a minimax optimization problem

$$\begin{array}{l}\min_{\mathbf{u}} \max_i\{F_{1n}, \quad F_{2n}, \quad F_{3n}\} \\ \text{Subject to} \quad \mu_i F_{in} - |F_{it}| \geq 0, \quad i = 1,2,3 \\ F_{in} \geq 0, \qquad\qquad i = 1,2,3\end{array}$$

$$\begin{bmatrix} \cos x_5 & -\sin x_5 & 0 \\ \sin x_5 & \cos x_5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{W_1} & \mathbf{W_2} & \mathbf{W_3} \end{bmatrix} \begin{bmatrix} F_{1n} \\ F_{1t} \\ F_{2n} \\ F_{2t} \\ F_{3n} \\ F_{3t} \end{bmatrix} = \begin{bmatrix} F_{OX} \\ F_{OY} \\ T_O \end{bmatrix} \tag{37}$$

This problem can be solved by using standrad multi-objective optimization routines or solved by linear programming through introducing slack variable.

IV. Simulation Results

In previous section, we proposed a new planning method for dynamic multi-agent manipulation. Here simulations will be carried out by using a triangular object under two-agent or three agent pushing, the object is shown in Fig. 1. where $a = 0.2m, b = 0.17m$, angle $A = \pi/6$, we assume the uniform distribution of mass, the mass density $\rho = 200kg/m^2$, and mass is $m = 1.7kg$. using the approach proposed in [7], the mass moment of inertia is $I_0 = 0.0037$. The friction coefficient between the object and surface is $\mu = 0.5$, friction coefficient between the fingers and object is $\mu_i = 0.8$. The local coordinate is located at the center of mass and fixed on the object, and assign the global coordinate at the same location, but fixed on the supporting plane.

Consider agent 1 pushing at edge AC, agent 2 pushing at BC, agent 3 pushing at AB, assign contact coordinate to each finger, the associated contact normal and tangential vectors are written as

$$\begin{array}{ll} \mathbf{n_1} = (\frac{\mathbf{1}}{\mathbf{2}}, -\frac{\sqrt{\mathbf{3}}}{\mathbf{2}})^{\mathbf{T}} & \mathbf{t_1} = (\frac{\sqrt{\mathbf{3}}}{\mathbf{2}}, \frac{\mathbf{1}}{\mathbf{2}})^{\mathbf{T}} \\ \mathbf{n_2} = (-\frac{\sqrt{\mathbf{3}}}{\mathbf{2}}, -\frac{\mathbf{1}}{\mathbf{2}})^{\mathbf{T}} & \mathbf{t_2} = (\frac{\mathbf{1}}{\mathbf{2}}, -\frac{\sqrt{\mathbf{3}}}{\mathbf{2}})^{\mathbf{T}} \\ \mathbf{n_3} = (\mathbf{0}, \mathbf{1})^{\mathbf{T}} & \mathbf{t_3} = (\mathbf{-1}, \mathbf{0})^{\mathbf{T}} \end{array} \tag{38}$$

the contact points between fingers and object are given as

$$\begin{array}{l} L_1 = (-0.0167 \quad 0.0289) \\ L_2 = (0.0500 \quad 0.00289) \\ L_3 = (-0.0400 \quad -0.0283) \end{array} \tag{39}$$

computing the wrench matrices, we get

$$\mathbf{W_1} = \begin{bmatrix} 0.5000 & 0.8660 \\ -0.8660 & 0.5000 \\ 0.0476 & -0.0333 \end{bmatrix} \tag{40}$$

$$\mathbf{W_2} = \begin{bmatrix} -0.8660 & 0.5000 \\ -0.5000 & -0.8660 \\ -0.0278 & 0.0481 \end{bmatrix} \tag{41}$$

$$\mathbf{W_3} = \begin{bmatrix} 0 & -1.000 \\ 1.000 & 0 \\ -0.0400 & -0.0283 \end{bmatrix} \tag{42}$$

A. Planning under Three-agent Manipulation

When we consider the translation and rotation of the object, three agents are needed to manipulate the object following predefined trajectory. The trajectories are given as

$$\begin{array}{lll} X_{or}(t) = \sin(t) & \dot{X}_{or}(t) = \cos(t) & \ddot{X}_{or}(t) = -\sin(t) \\ Y_{or}(t) = \cos(t) & \dot{Y}_{or}(t) = -\sin(t) & \ddot{Y}_{or}(t) = -\cos(t) \\ \theta_{or}(t) = \frac{1}{2}\sin(t) & \dot{\theta}_{or}(t) = \frac{1}{2}\cos(t) & \ddot{\theta}_{or}(t) = -\frac{1}{2}\sin(t) \end{array} \tag{43}$$

And we consider the uncertainties F_x, F_y, T associated with pressure distribution entering the system (1) randomly, and $F_x \sim [-0.5F_{xn}, 0.5F_{xn}]$,

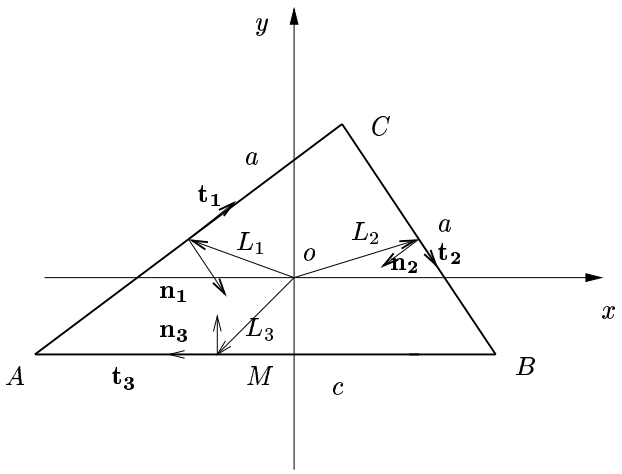

Fig. 1. Manipulation of triangular object

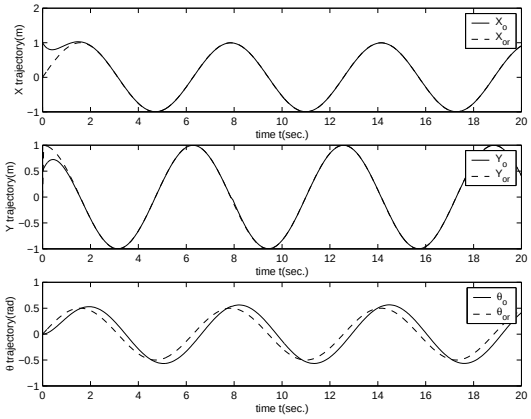

Fig. 2. H^∞ tracking results of positions ($\rho = 0.1$)

$F_y \sim [-0.5F_{yn}, 0.5F_{yn}]$, $T \sim [-0.5T_n, 0.5T_n]$, where F_{xn}, F_{yn}, T_n are the friction forces and torque generated under the assumption with known pressure distribution that is given above. The numerical computation procedure for F_{xn}, F_{yn}, T_n is described in [8]. And the initial condition $X_o(0) = 1, \dot{X}_o = 0, Y_o(0) = 0.5, \dot{Y}_o = 0, \theta_o(0) = 0.2, \dot{\theta}_o = 0$.

First, following the *design procedure* in above section, the H^∞ robust controller design is give by following steps.

1. Specify

$$K_1 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 8 \end{bmatrix} \qquad K_2 = \begin{bmatrix} 9 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

such that the eigenvalues of the nominal system are $-0.3467, -8.6533, -2.5 + 2.7839i, -2.5 - 2.7839i, -6.7016, -0.2984$.

2. select the attenuation level $\rho = 0.3$, and $\rho = 0,05$, respectively. And $Q = diag[100I_3, \quad 100I_3]$, and $R = \rho^2 I$.

3. solve the Riccati-like equation (30) using MATLAB function "are", we get,

$$P = \begin{bmatrix} 172.2222 & 0 & 0 & 16.6667 & 0 & 0 \\ 0 & 196.4286 & 0 & 0 & 25. & 0 \\ 0 & 0 & 121.25 & 0 & 0 & 6.25 \\ 16.6667 & 0 & 0 & 7.4074 & 0 & 0 \\ 0 & 25 & 0 & 0 & 10.7143 & 0 \\ 0 & 0 & 6.25 & 0 & 0 & 11.25 \end{bmatrix}$$

4. select neural network structure as (11). And for simplicity reason, we choose $\omega_{ij}^k = 1$ for all i, j, k. i.e.,

$$\omega = [1 \quad 1 \quad 1]$$

The neural network systems are selected to be $\xi_i = [\xi_{i,1}, \cdots, \xi_{i,17}]^T$ for $i = 1, 2, 3$, where $\xi_{i,j}$ is nonlinear neural function and has following form,

$$\xi_{i,j} = \frac{e^{\omega^T \mathbf{q_2 - 4 + 0.5(j-1)}} - e^{\omega^T \mathbf{q_2 + 4 - 0.5(j-1)}}}{e^{\omega^T \mathbf{q_2 - 4 + 0.5(j-1)}} + e^{\omega^T \mathbf{q_2 + 4 - 0.5(j-1)}}}$$

for $i = 1, 2, 3, \quad j = 1, \cdots, 17$. And

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}, \quad \Xi = \begin{bmatrix} \xi_1^T & 0 & 0 \\ 0 & \xi_2^T & 0 \\ 0 & 0 & \xi_3^T \end{bmatrix}$$

where $\theta_i = [\theta_{i,1}, \cdots, \theta_{i,17}]^T$ for $i = 1, 2, 3$. The initial guess of $\Theta(0) = 0$.

5. Compute the H^∞ controller according (27) and (29). and update the weights of neural networks according to (28).

Fig. 2- 7 present the simulation results of neural network-based H^∞ tracking. In Fig. 2- 3 and Fig. 5- 6, simulation results demonstrate the minimax tracking performance for attenuation level of $\rho = 0.05$ and $\rho = 0.1$, respectively. Fig. 4 and Fig. 7 show the generalized forces under different attenuation levels. According to the simulation results, it is clear that a small attenuation level ρ may yield a better tracking performance.

After obtaining the generalized forces, now a minimax optimization problem can be set up as (37), Here $\mu = 0.5$ is the friction coefficient between the finger agents and object, and the wrench matrix is as (42). The results are shown in Fig. 8 and Fig. 9 for different ρ. The resulting forces satisfy the performance index (2).

V. Conclusions and Future Works

This paper proposed a planning method dynamic cooperative multi-agent manipulation with unknown pressure distribution. The coordination between agents is performed hierarchically. In lower level, the generalized force inputs are designed by integrating linearization with neural network-based H^∞ technique, and H^∞ tracking performance is achieved. The H^∞ performance consists of the desired attenuation property and the boundedness property of all the argument variables. Through applying the generalized force to the object, the object will track the given trajectory, and obtain desired velocities. In higher coordination level, a cooperative game is solved to distribute the

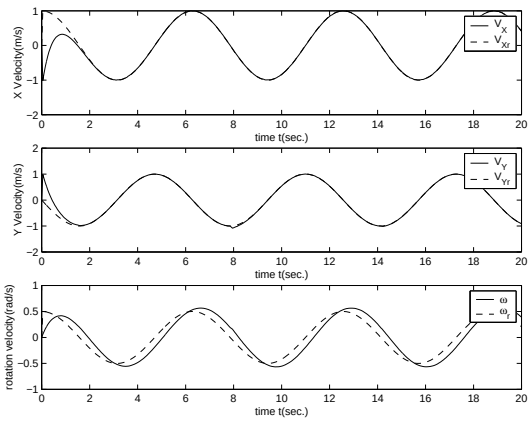

Fig. 3. H^∞ tracking results of velocities ($\rho = 0.1$)

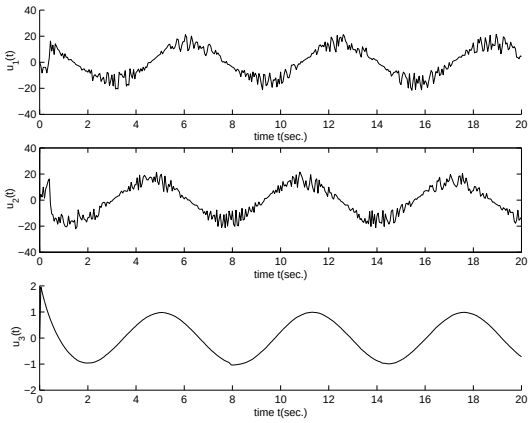

Fig. 4. Generalized forces $\overline{\mathbf{u}}(t)(\rho = 0.1)$

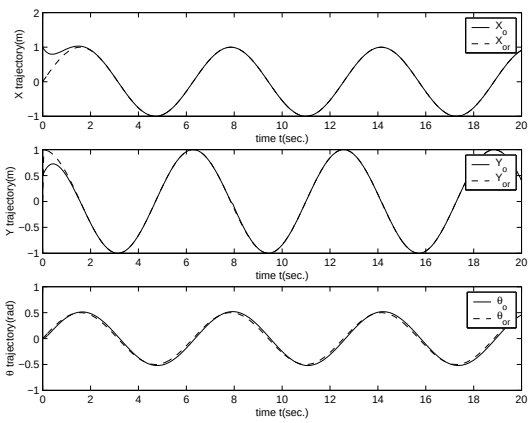

Fig. 5. H^∞ tracking results of positions ($\rho = 0.05$)

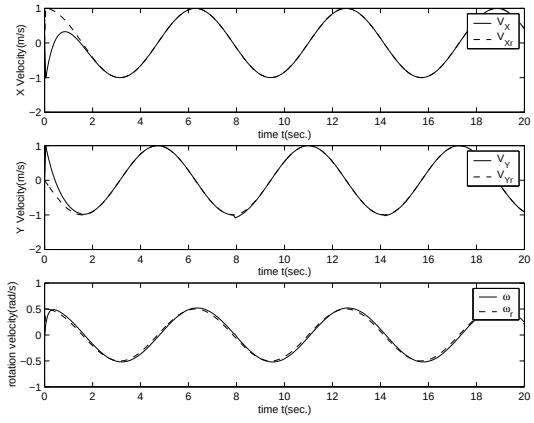

Fig. 6. H^∞ tracking results of velocities ($\rho = 0.05$)

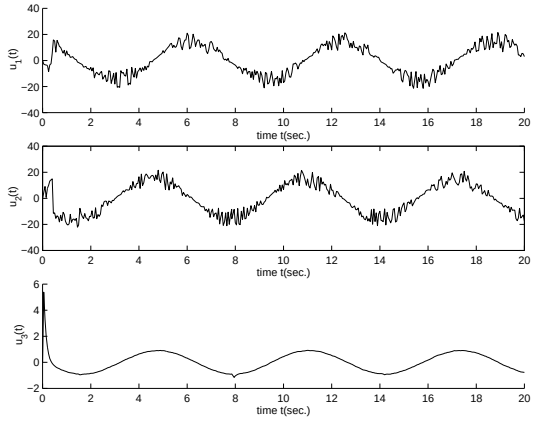

Fig. 7. Generalized forces $\overline{\mathbf{u}}(t)(\rho = 0.05)$

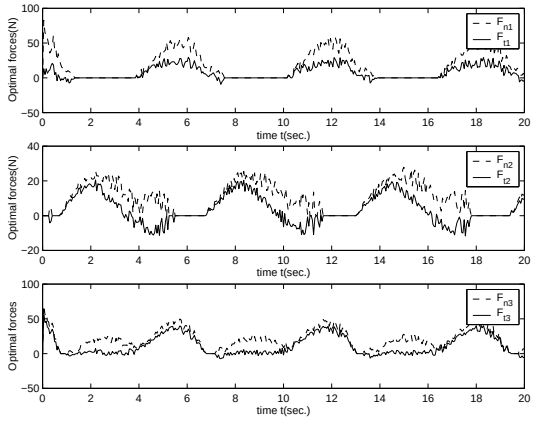

Fig. 8. Force coordination under three-agent manipulation ($\rho = 0.1$)

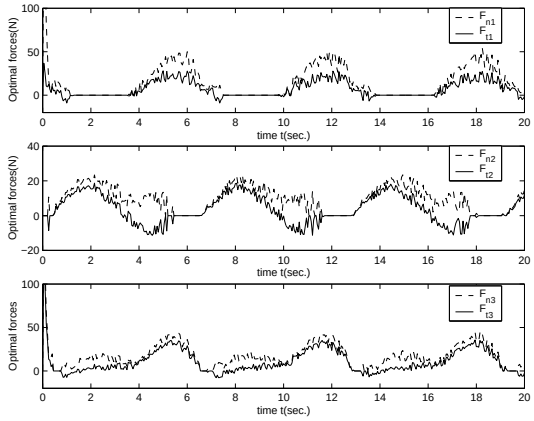

Fig. 9. Force coordination under three-agent manipulation ($\rho = 0.05$)

generalized force between agents, and obtain minimax performance. The coordination under two-agent and three-agent manipulation are studied. The hierarchical structure for coordination is flexible, when more agents are added for manipulation, the redesign of lower level generalized force controller is not needed. Simulation results demonstrate the proposed coordination method.The proposed approach does not consider the dynamics of the agents, the integration of the dynamics of agents and object in planning will be addressed and how to plan contact points optimally between agent and object is also very interesting.

References

[1] M.N. Ahmadabadi and E Nakano. A constrain and move approach to distributed object manipulation. *IEEE Trans. on Robotics and Automation*, 17(2):157–172, 2001.

[2] T. Basar and P. Bernhard. *H^∞-optimal control and related minimax design problems-a dynamic game approach.* Birkhauser, Boston, second edition edition, 1995.

[3] Y. Chen and B. Chen. A nonlinear adaptive $\mathbf{H}^\infty$ tracking control design in robotic systems via neural networks. *IEEE Trans. on Control Systems Technology*, 5(1):13–29, 1997.

[4] D.R. Donald, J. Jennings, and D. Rus. Information invariant for distributed manipulation. *Intl. J. of Robotics Research*, 16(5):673–702, 1997.

[5] W.H. Huang and M.T. Mason. Mechanics, planning, and control for tapping. *Intl. J. of Robotics Research*, 19(4):883–894, 2000.

[6] Q. Li and S. Payandeh. Modeling and analysis of dynamic planar multi-agent manipulation. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 200–205, July 2001.

[7] Q. Li and S. Payandeh. Planning velocities of free sliding objects as free boundary value problem. Technical report, ERL, Simon Fraser University, 2001.

[8] Q. Li and S. Payandeh. Centralized cooperative planning for dynamic multi-agent planar manipulation. In *Proc. 2002 IEEE Intl. Conf. on Decision and Control*, Dec 2002.

[9] C. Lin and T. Lin. An $\mathbf{H}^\infty$ design approach for neural net-based control schemes. *IEEE Trans. on Automatic Control*, 46(10):1599–1605, 2001.

[10] J.Y.S. Luh and Y.F. Zhang. Constrained relations between two coordinated industrial robots for motion control. *Intl. J. of Robotics Research*, 6(3):60–70, 1987.

[11] M.T. Mason. Progress in nonprehensile manipulation. *Intl. J. of Robotics Research*, 18(11):1129–1141, 1999.

[12] D. Rus. Coordinated manipulation of objects. *Algorithmica*, 19(1):129–147, 1997.

[13] A. Stoorvogel. *The H^∞ control problem: A State Space Approach.* Prentice-Hall, Upper Saddle River, NJ, 1992.