Advanced controller design using neural networks for nonlinear dynamic systems with application to micro/nano robotics

Qinmin Yang

ADVANCED CONTROLLER DESIGN USING NEURAL NETWORKS FOR NONLINEAR DYNAMIC SYSTEMS WITH APPLICATION TO MICRO/NANO ROBOTICS

by

QINMIN YANG

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

2007

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following five articles that have been submitted for publication as follows:

Pages 5-43 are under review to the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B.

Pages 44-68 are accepted for publication in the IEEE TRANSACTION ON NANOTECHNOLOGY.

Pages 69-94 are published in the INTERNATIONAL JOURNAL ON NANOTECHNOLOGY.

Pages 95-128 are under review to AUTOMATICA.

Pages 129-168 are intended for submission to the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.

ABSTRACT

The dissertation focuses on neural network (NN) control designs for nonlinear systems with application to micro/nano robotic. Critical problems in nano scale including thermal drift are also addressed. This dissertation is given in the form of several papers.

To start with, a suite of novel controllers is developed in the first paper for the manipulation of microscale objects in a micro-electromechanical system (MEMS). The proposed robust and the adaptive neural network controllers overcome the unknown contact dynamics and ensure their performance in the presence of actuator constraints.

Next, in the second paper, thermal drift, as the major source of uncertainty in nano scale, is discussed and compensated by using block based phase-correlation method. This consideration is needed to realize a truly automatic manipulation of nano objects.

Subsequently, the third paper uses the drift compensator from the second paper to develop a NN-based adaptive force design for nanomanipulation to accommodate the unknown dynamics, while maintaining a constant force applied on the nano sample.

In order to address the optimality in terms of a standard quadratic cost function, the fourth paper introduces a reinforcement learning-based controller for the nanoscale manipulation by considering the Bellman equation. This controller consists of an action network and a critic network. Both of the networks are trained in an online fashion with the updating algorithms derived from dynamic programming (DP).

To make our scheme applicable to a more general class of affine systems with immeasurable states, an output feedback design with an extra NN observer is introduced in the final paper while relaxing the separation principle. By using the Lyapunov approach, the stability of the above mentioned controller designs are demonstrated.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to Professor Sarangapani Jagannathan, my advisor, for his support, understanding and guidance during my doctoral study. I appreciate his wide knowledge and expertise in various areas and his assistance in writing my publications and dissertation. His leadership, logical way of thinking, hard work and scholarship have set an example I hope to match some day.

I wish to express my warm and sincere thanks to Professor Eric Bohannan, who directed me into the research of nanotechnology. His kind support and guidance have been of great value in this work.

I would like to extend my gratitude to the other members of my committee Professor Kelvin Erickson, Professor Chang-Soo Kim, Dr. Shoukat Ali and Professor Cihan Dagli, who supported me during my education and offered constructive comments and suggestions regarding my research and papers.

I am grateful to my fellow colleagues, Pingan He, James Fonda, Jonathan Vance, Hao Wu and Dr. Maciej Zawodniok for their academic support and friendships. They each helped make my time in the Ph.D. program more fun and interesting.

My best regards goes to the Embedded Control Systems and Networking Lab at UMR for financial support throughout my studies and research.

Finally, I owe my loving thanks to my parents Zhenliu Yang and Shengfang Qin, for their understanding, encouragement and sacrifice during many years of my education abroad. My loving thanks are also due to the Edwards family. They let me own a happy “family” in U.S.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

Various control methodologies are designed for application environments ranging from outer space to deep sea. Besides these applications, there is a new emerging application area called Micro/Nanorobotics. This new application area is concerned with the design and fabrication of nanorobots or apparatus such as an Atomic Force Microscope (AFM) that are capable of manipulating objects at nanoscale. One of the key challenges in the application area is the automatic assembly of micro/nano-scale devices either by manipulation with macro/micro devices or by self-assembly on programmed templates or scaffolds. More recent applications in the nanoworld have become possible because of the developing trends in imaging, manipulation and fabrication technologies using the micro/nano mechatronics and MEMS/NEMS technologies. By the invention of Scanning Tunneling Microscope (STM) and Scanning Probe Microscope (SPM), topographic images with at omic resolution became possible. Further, by utilizing the cantilever from the scanning probe microscope, manipulation and fabrication in the nano scale was realized successfully in 1990, which opened the door for new applications. However, the micro/nano scale manipulation is still in its infancy. Therefore, issues that are commonly encountered at the nanoscale during manipulation of objects are addressed in this study.

Micro or Nano-manipulation, which aims at manipulating objects with micrometer or nanometer size, is a precursor for nanomanufacturing. By accurately controlling atoms, molecules, or nano scale objects, numerous applications of nanotechnology can be cited in the area of molecular biology and genetics, solid-state physics, chemistry, material science, computer industry and medicine. By reducing the object size from micro to nano meter, new sensors, tera-byte capacity memories, DNA computers, man-made materials, etc., would be possible within the near future. Today, manipulation of particles with the order of 10nm in diameter using AFM is being investigated by many researchers both at UMR and elsewhere. However, for future new nanotechnology products in industry, there are still many challenges to be solved.

From macro to micro/nano world, any noise even though it may be small can become a major hurdle. Without suitable mechanism or process to compensate for the noise, automatic real-time controller designs will become impractical. As a consequence, when operated in ambient conditions without stringent environmental controls, nanomanipulation typically requires extensive user intervention to compensate for the spatial uncertainties associated with the microscope and its piezoelectric drive mechanism, such as hysteresis, creep, and thermal drift. Some of the uncertainties related to the piezo system usually are compensated by AFM or SPM vendor software. However, the thermal drift will increase with time which makes it harder for compensation. Mainly the AFM tip drifts along with time by thermal effects at a speed of about one atomic diameter per second, even when the voltage inputs for controlling the tip and stage position are held constant. In our study, a block-based phase correlation theme is implemented to estimate and compensate the drift during imaging and manipulation so that tasks are carried out as if drift does not exist.

Automation is a prerequisite for modern manufacturing. To realize micro/nano technology at a reasonable cost, automatic assembly of MEMS/NEMS using advanced control techniques are highly desirable in the future. However, designing controllers for the manipulation and handling of micro/nano-scale objects poses a much greater challenge, partly due to the immaturity of the micro/nano physics. Modeling of such micro/nano-scale forces during manipulation is a whole lot different than in a macro scale system. The dominant forces acting on a MEMS/NEMS system are mainly van der Waals, capillary and electrostatic forces, while the forces due to gravity are negligible. Typically, these forces vary much with environment and could not be precisely measured. Furthermore, there are a lot of uncertainties, for instance fabrication imperfections and complex device nonlinearities, which make the actuation and manipulation of such devices difficult. To accommodate these uncertainties and nonlinearities, robust and robust adaptive neural network (NN) controllers with the ability to learn online are designed for these applications. Mathematically, the stability analysis is demonstrated using standard Lyapunov method.

In the literature, there are many approaches proposed for designing stable controllers for nonlinear systems. However, stability is only a bare requirement for the controller design. A further consideration is the optimality based on a specific cost function which is used to determine the performance of the system. In other words, a controller scheme should not only achieve the stability of the closed-loop system, but also to keep the cost function as small as possible. Of the available methods, dynamic programming (DP) has been extensively applied to generate optimal control action for nonlinear dynamic systems. One of the drawbacks of DP is the computation cost which increases with the dimension of the nonlinear system, referred to as the "curse of dimensionality". Additionally, most of their implementations are done in an offline fashion or require the dynamics of the nonlinear systems to be known a priori. As an alternate approach, several appealing online learning neural controller designs were introduced. They are also referred to as forward dynamic programming (FDP) or adaptive critic designs (ACD). In our study, we are considering NN-based online learning adaptive critic designs for both nanomanipulation tasks and more general nonlinear discrete systems with a standard quadratic-performance index as the cost function. Our scheme is also extended to output feedback counterpart to be applicable for systems with certain unavailable staes for measurement. The requirement of separation principle is also relaxed, which is normally employed for linear systems but unapplicable for nonlinear systems.

In the dissertation, in Paper 1, modeling and corresponding robust controller designs for micromanipulation are introduced. Subsequently, drift compensation algorithm is developed for the task of nanomanipulation in Paper 2. With the compensator, the mechanism of nanomanipulation is described and NN-based adaptive force controller is designed in Paper 3. To increase the performance of the closed-loop system in terms of a standard quadratic cost function, an online reinforcement learning-based control algorithm using neural network for nanomanipulation task is explained in Paper 4. The expansion of the reinforcement learning design for general affine nonlinear systems and its output feedback version are discussed in Paper 5.

PAPER 1

A Suite of Robust Controllers for the Manipulation of Micro-Scale Objects

Qinmin Yang and S. Jagannathan

Department of Electrical and Computer Engineering

University of Missouri – Rolla,

Rolla, Missouri, U.S.A 65409

Email: qyy74@umr.edu, sarangap@umr.edu

ABSTRACT

A suite of novel robust controllers is introduced for the pick-up operation of micro-scale objects in a micro-electromechanical system (MEMS). In MEMS, adhesive, surface tension, friction and van der Waals forces are dominant. Moreover, these forces are typically unknown. The proposed robust controller overcomes the unknown contact dynamics and ensures its performance in the presence of actuator constraints by assuming that the upper bounds on these forces are known. On the other hand, for the robust adaptive critic-based neural network controller, the unknown forces are estimated online. It consists of an action NN for compensating the unknown system dynamics, and a critic NN to approximate certain strategic utility function and to tune the action NN weights. By using the Lyapunov approach, the uniformly ultimate boundedness (UUB) of the closed-loop manipulation error is shown for all the controllers for the pick-up task. To imitate the real practical system, a few of the system states are considered measurable and the measurement noise is also introduced. An output feedback version of the adaptive NN controller is proposed by taking advantage of the separation principle through a high gain observer design. The problem of measurement noise is also overcome by constructing a reference system. Simulation results with various controllers are presented and compared to substantiate the theoretical conclusions.

Keywords

Micromanipulation, robust controller, adaptive neural network, reinforcement learning

I. INTRODUCTION

Micro Electro-Mechanical System (MEMS) is a relatively new technology involving miniaturization of systems and components to create complex machines that are of micron size in nature. These are used in a variety of applications involving sensing, actuation and communication. The MEMS has revolutionized a major part of the sensor and actuator industry. Typical MEMS products include inkjet printer heads and accelerometers for airbags [1]. Although these products require little or no assembly, automatic assembly of hybrid MEMS devices are desirable. Much effort has been put forth for the micro-assembly, or micro-manipulation [1]-[2], [4]-[6], [11]-[14]. Among them, in [1] the pick up and release tasks with van der Waals force are analyzed whereas in [6] manipulation using SEM is introduced. Research effort in [4] proposed a manipulation system in open air and fulfilled manipulations with a gold coated piezoresistive silicon cantilever.

Modeling of such micro-scale devices for actuation is much different than in macro scale system. At micro scale, surface forces are predominant while volumnic forces are negligible [1]. The dominant forces acting on a MEMS system are mainly van der Waals, capillary and electrostatic forces, while the forces due to gravity are negligible. Typically, these forces vary with environment and can not be precisely measured. Furthermore, uncertainties, for instance fabrication imperfections and complex system nonlinearities, make the actuation and manipulation of such devices difficult.

At the same time, modeling and simulation are critical and fundamental for designing proper handling techniques. Work on modeling adhesive forces and the utilization of the models in micromanipulation has been carried out by many researchers. Arai et al [5] studied the effects of attractive forces and handling strategies in micromanipulation, Rollot et al [28] studied various modes in micromanipulation by combining analytical micro force models and Newton-Euler dynamics whereas Sitti and Hashimoto [29] built the model for manipulation of nano particles, and Feddema et al [30] introduced a computational model of van der Waals forces and electrostatic forces for interactions between a micro sphere and a micro cube.

Designing controllers for the manipulation and handling of micro-scale objects poses a much greater challenge in terms of accommodating the nonlinearities in the system. Hence, these forces have to be modeled in order to design a controller for the MEMS. To confront some of the issues of nonlinearities and uncertainties in such MEMS, a robust controller is designed. The robust controller requires the upper bound on the uncertainties and nonlinearities.

Moreover, in practical control problems, the amplitude of the control signal is subject to prescribed actuator constraints due to saturation problem. Ignoring these constraints may lead to unsatisfactory performance or even instability of the closed-loop system. In adaptive control systems, the saturation constraint problem becomes particularly critical because of the parameter adaptation transients which may introduce large control signals [25]. However, the research activity devoted to the problem of controlling a nonlinear system in the presence of saturation is still relatively limited [23], [24]. Thus, in this paper, actuator constraints have been incorporated into the robust controller design in contrast to other works [7] where no explicit magnitude constraints are treated.

On the other hand, in the case of robust adaptive critic-based neural network (NN) controller, reinforcement learning (RLNN) feature [11] is utilized to approximate the uncertainties online. The RLNN structure consists of two NNs: an action NN for compensating the uncertain nonlinear system dynamics, and a critic NN for tuning the weights of the action NN. A novel utility function, which is viewed as the system performance index over time, was defined as the critic NN input. The critic signal approximates the long term performance measure and provides an additional corrective action based on current and past long-term system performance in contrast with the standard adaptive dynamic programming scheme [9], [16]-[18], where the critic signal alone is used to tune the action NN weights and in standard adaptive NN control literature where a short term performance is normally utilized [7]. The critic NN output along with the filtered tracking error is used to tune the action NN.

Providing tracking error information to the action NN will make the proposed controller similar to the other adaptive controllers [7] and therefore it is avoided. Moreover, a Lyapunov approach is used to show the stability of the closed-loop system in contrast with the existing schemes in adaptive dynamic programming based critic NN control schemes [16]-[18]. The proposed NN structure has an advantage over supervised learning NN-based controllers, where desired system outputs are not required. In our scenario, the desired outputs are the probe location and the contact dynamics, which are typically unknown. An offline learning phase is not required in this approach in contrast with adaptive dynamic programming-based critic control schemes [16]-[18].

Finally, in many practical problems, not all state variables are measurable due to technical or economic reasons [33]. For instance, in the micromanipulation system, a laser measuring instrument [32] or a scanning electron microscope (SEM) [13] is employed to obtain the position of the micro objects whereas the velocities are not measured. Nevertheless, the obtained information is usually contaminated with measurement noise. Therefore, an output feedback controller is designed by implementing a high-gain observer which is used to estimate the actual system states which includes velocities. The bounded measurement noise is integrated into a new reference system and overcome. Theoretical and simulation results indicate that the output feedback adaptive NN controller is able to perform the task successfully.

Therefore in this paper, both a robust and adaptive critic-based NN controllers, and their output feedback version are proposed for pick up task in a micromanipulation system. These two controllers are contrasted based on their performance. The main contributions of this work can be summarized as: 1) A computation model for pick-up task is formulated considering the unknown micro adhesive forces including van der Waals, surface tension and electrostatic forces; 2) A robust controller is designed to accommodate the unknown interactive micro forces for the task of picking up the micro particles; 3) An adaptive critic-based NN scheme is introduced to achieve a better response – a cost function is utilized to evaluate the system performance. The NNs are updated in an online fashion without offline training phase and the persistent excitation

(PE) condition requirement is overcome; 4) To overcome the unmeasured states in the presence of measurement noise, a high-gain observer is added with the NN controller.

A brief background on NNs and stability of nonlinear system are given in Section II. The interactive force analysis of the pick-up task and associated dynamic models are presented in Section III and Section IV, respectively. Next, the robust and adaptive NN controller designs are given in Section V. Finally, Section VI shows the simulation results to substantiate our theoretical conclusions.

II. BACKGROUND

A. NN Background

A general function $f(x) = C^{(s)}$ can be approximated using a neural network with at least two layers of appropriated weights given by

$$f(x) = W^T \sigma(V^T x) + \zeta \tag{2.1}$$

where W and V are constant-weight matrices of the NN (the first column of these matrices include the bias vectors so that tuning the weight matrices results in tuning the biases as well), x is the input vector, $\sigma(V^T x)$ is the vector of hidden-layer activation functions, and ζ is the error in approximation. If the input to the hidden-layer weight matrix V is selected randomly and kept constant, and the vector of hidden-layer activation functions is selected as a basis function, whereas the output layer weights are only tuned provided sufficiently large number of nodes in the hidden layer is chosen, then a one-layer NN will result [10]. For simplicity, define the net output for a one-layer NN as

$$y = W^T \sigma(x) + \zeta \tag{2.2}$$

For suitable approximation, $\sigma(x)$ must form a basis to the function that is being approximated. Since it is already known that (2.2) can approximate any continuous function over a compact set and a set of target weights exist, then the control objective is to tune the actual weights such that they approach their targets.

Neural network controller designs have relied upon the function approximation property (2.1) [15]. Thus, the performance of the controller mainly depends on the learning algorithm as suggested in [16]. Among various NN controller structures, adaptive critic designs [16] utilize reinforcement learning for NN weight tuning. These designs address the general problem of how to optimize a measure of utility or goal function in an unknown, noisy, and nonlinear system.

In a typical adaptive critic NN architecture, the critic NN evaluates the system performance index and tunes the action generating NN, which in turn provides the control input signal to the plant to be controlled. There are too many papers dealing with control using adaptive critic NN architecture to be mentioned here. For details, readers can refer to [9], [16]-[18]. However, very few papers [16] present the closed-loop stability analysis with performance guarantee. This paper overcomes these limitations by using Lyapunov approach for control applications. Next the following definition is required.

B. Stability of Closed-loop Systems

Considering a nonlinear system given by

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x) \end{aligned} \tag{2.3}$$

where x is the state vector, u is the input vector, and y is the output vector [6]. For a control input u, the closed-loop system (2.3) is uniformly ultimately bounded (UUB), if for all $x(t_0) = x_0$, there exist a $\varepsilon > 0$ and a constant $T = T(x_0, \varepsilon)$ such that $\|x(t)\| < \varepsilon$ for all $t \geq t_0 + T$.

III. INTERACTION FORCES MODEL

Manipulation and handling of micro-scale objects are required for the assembly and maintenance of micro machines and their parts. In this study, we consider the manipulation of micro-sized sphere shaped objects or microparticles 50 μm in diameter. When manipulating objects in the micro domain, the pickup should be understood using micro-physics [2], [3]. Modeling is necessary for picking up and placing micro-spheres laying on a planar substrate. In the manipulation process, the micro-sphere is to be picked up and to be placed at another location for assembly. As a brief description, the probe, which is treated as the end-effector and manipulator, is lowered to make contact with the micro-sphere. Once contact has been established, the micro-object has to be picked up by retracting the probe as a result of adhesive forces [4]. Next, the probe will be moved with the micro objects to a desired target position. After that, the object will be placed on the substrate by creating a repulsive force.

However, the process of placing the micro-object is also an intricate process and different from that of pick-up. Generally, by selecting proper system parameters, the spheres can be picked up by the probe due to the attractive force between them [1]. On the other hand, the job of releasing spheres need totally different techniques. Various placing methods have been introduced in [1], [13], [14] and [31]. For instance in [31] electrostatic interaction is utilized. In this paper, we will concentrate our work on the pick-up task of micro-spheres.

For the purpose of designing a controller for the object-handling task, we shall restrict ourselves with the intricacies of the physics during pick up process as shown in Fig. 1. The adhesion forces are dominant in the system. Actually, adhesive forces are considered to play an important role in the manipulation process. These are given by:

- Van der Waals forces,
- Surface tension (or capillary) and
- Electrostatic (or coulomb) forces.

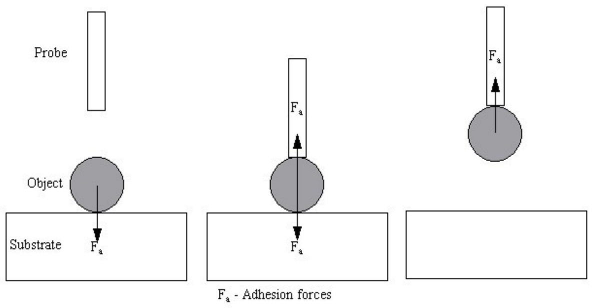


Fig 1. Object Handling Task

A. Van der Waals Force

Van der Waals force acts between atoms resulting from interaction between electrons in the outermost bands rotating around the nucleus of the atoms. An overview of it is given in [19]. Van der Waals forces are present in every environmental condition. Depending on the object geometry, material type, the van der Waals force can be calculated based on the interaction energy between atoms or molecules. For ideal geometries, the van der Waals forces are given by

$$F_{bp}^{VdW} = \frac{A_{bp}^{w} R_b}{6D_{bp}^2},\ F_{bs}^{VdW} = \frac{A_{bs}^{w} R_b}{6D_{bs}^2},\ \text{and}\ F_{bb}^{VdW} = \frac{A_{bb}^{w} R_b}{6D_{bb}^2} \tag{3.1}$$

respectively, for ball-probe, ball-substrate and for ball-ball interaction. Here R_b is the sphere radius, A_{ij}^{w} is the Hamaker constant of “i-water-j” interface, and D_{ij} is the separation distance. Furthermore, van der Waals forces are greatly influenced by the surface roughness [2]. It has been shown that increasing the surface roughness decreases the van der Waals forces [4]. Thus, taking the surface roughness into consideration as shown in Fig. 2, the van der Waals force is expressed as [6]

$$F_{vdwb} = \left(\frac{z}{z + b/2} \right)^2 F_{vdw} \tag{3.2}$$

where z is the distance, b is the height of the surface irregularities, and F_{vdw} is the van der Waals forces between the plane plate and the sphere.

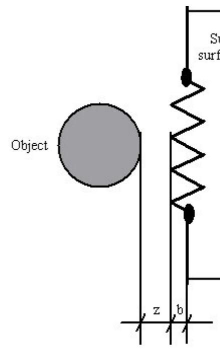


Fig 2. Rough Plate and Plane Sphere

B. Surface Tension Force

In ambient operational environment, water layer is present on the surface of the sphere and the substrate. A liquid bridge occurs between them at close contact as shown in Fig. 3.

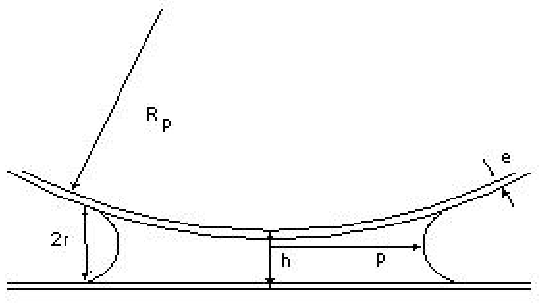


Fig 3. Capillary Force Parameters during a Sphere and Flat Surface Contact

In [20], the macroscopic theory of capillarity is proven to be applicable for curvature radius in the order of molecular size. Assuming that (i) $r << p << R_p$, (ii) the surfaces are coated with a film of constant thickness e, (iii) the contact angle is 0, which should be the true in our case, and (iv) the surface attraction through the liquid phase is negligible, the capillary force can be written as [21]

$$F^{cap} = 4\pi\gamma R_p(1 - \frac{h-2e}{2r}) \tag{3.3}$$

where γ is the liquid (water) surface energy, e is the thickness of the water layer, and r is the radius of curvature of the meniscus as shown in Fig. 3. Moreover, the volume of liquid condensed in the bridge and the film thickness distribution can also influence the capillary force, but it can be ignored in our case [21]. The capillary forces for probe-ball and ball-substrate can be calculated from (3.3). It is important to notice that by baking the sample before manipulation process can reduce the capillary forces greatly [21].

C. Electrostatic Force

For the electrostatic force, Coulomb forces are considered only. Using the point charge assumption, the electrostatic force between an uncharged metal wall and a charged sphere is given by

$$F^{elec} = \varepsilon_0 \pi d^2 \left(\frac{3\varepsilon_1}{\varepsilon_1 + 2} \right)^2 E^2 \tag{3.4}$$

where ε_0 and ε_1 are the dielectric constants of free-space and the material, respectively. The parameter d, is the diameter obtained as $d = d_1 d_2 / (d_1 + d_2)$, where, d_1 and d_2 are the diameters of the two micro-spheres under consideration. The parameter E, is the voltage between the probe and the substrate. It has also been shown that the electrostatic forces can be minimized by applying an external voltage.

IV. DYNAMIC MODEL

A dynamic model of the micro-scale object handling system is formulated considering all the forces mentioned above [4], [7]. The objects considered in this work include micro-spheres of diameter 50 to 200 μm (radius R_b varies from 25 μm to 100 μm). In particular, we will also assume a rectangular block shaped probe.

When the system is shown as Fig. 4, the dynamic model for the object handling task can be written as [4]

$$m_p \ddot{Y}_p = F_{ext} \sin(\pi/2 - \theta) - F_{bp}^{VdW} \cos\theta - F_{bp}^{cap} \cos\theta - F_{bp}^{elec} \cos\theta - m_p g \tag{4.1}$$

$$m_b \ddot{D}_1 = (F_{bp}^{VdW} + F_{bp}^{cap} + F_{bp}^{elec}) \cos\theta - F_{bs}^{VdW} - F_{bs}^{cap} - F_{bs}^{elec} - m_b g \tag{4.2}$$

$$Y_p = D_1 + R_b + (R_b + D_2)\cos\theta \tag{4.3}$$

where $\ddot{Y}_p$ is the instantaneous acceleration of the probe, F_{ext} is the external force applied on the probe, θ is the angle of inclination of the probe with the vertical axis, F_{ij}^{VdW} is the van der Waals forces, F_{ij}^{cap} is the capillary forces and F_{ij}^{elec} is the electrostatic forces for the ball-probe (bp) and the ball-substrate (bs) interfaces presented in (3.1) through (3.4), respectively. Here m_p is the mass of the probe, and m_b denotes the mass of the micro sphere. There are two constraints for this model [22]

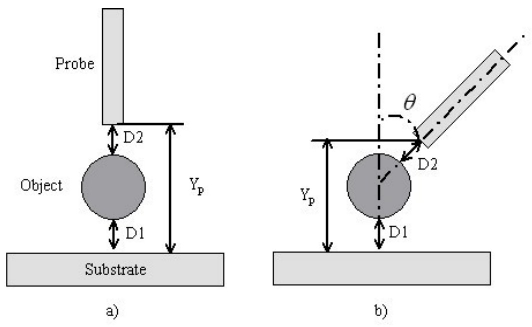


Fig 4. Intersurface Distances Notation

• A condition imposed by the substrate reaction (when ball contacts the substrate at $D_1 = D_0 = 0.4nm$):

$$D_1 = 0.4nm \Rightarrow \ddot{D}_1 \geq 0 \tag{4.4}$$

• A detachment constraint expressed by:

$$F_{ext} > 2R_b \pi W_{ball-water-substrate} \tag{4.5}$$

where $W_{ball-water-substrate}$ is the surface work of adhesion.

Practically, the manipulation time has to be small. Further, the applied force has to be appropriate to prevent ball or substrate deformation. The object and the substrate are sometimes fragile and will be damaged under improper applied force due to controller design.

From (4.1) through (4.3), we can find that the dynamic model for the manipulation and handling of micro-scale objects are quite nonlinear and unknown. For instance, the water surface energy, thickness of the water layer, Hamaker constant, electric charge density, diameter of the object, height of immersion and many others are typically unknown. Under these circumstances, one has to apply advanced control schemes in order to manipulate such micro-scale objects. The control scheme must guarantee object manipulation in the event of such unknown uncertainties without damaging samples.

V. CONTROLLER DESIGN

The suite of controller designs proposed in this paper is based on the filtered tracking error formulation [7]. In this paper, by using the filtered tracking error system formulation, the robust and robust adaptive neural network controllers are given in detail. For the purpose of controller design, θ is considered to be zero, which is a valid approach to pick up micro particles [4], [26]. Similar analysis could be performed for different values of θ as well.

A. Filtered Tracking Error Dynamics Formulation

As stated above, placing the objects on a substrate requires other intricate processes and will not be discussed in this paper. The pick-up of the sphere can be viewed with increased D_1 while making $D_2 = D_0$ (atomic contact distance) when the probe is retracting. For detailed illustration, initially the sphere is resting on the surface of the substrate and the probe is parked exactly above the sphere. After the force is applied on the probe, it will move downwards and make contact with the sphere. Due to the presence of adhesive forces between the probe and sphere, the micro object will be picked up when the probe is retracted. To accomplish this task, a fundamental condition to be fulfilled [1] will be

$$F_{bp} > F_{bs} + F_g \tag{5.1}$$

which means that the adhesive force between the ball and probe F_{bp} should be greater than the force of surface attraction F_{bs} plus the gravitational force F_g. This condition is critical for material selection.

Hence, for pick-up micro object, the control objective is suitably chosen as mentioned above. Differentiating (4.3) to get

$$\dot{Y}_p = \dot{D}_1 + \dot{D}_2 \tag{5.2}$$

and

$$\ddot{Y}_p = \ddot{D}_1 + \ddot{D}_2 \tag{5.3}$$

or

$$\ddot{D}_2 = \ddot{Y}_p - \ddot{D}_1 \tag{5.4}$$

Let the error between the desired and the target position be defined as

$$e = D_2 - D_0 \tag{5.5}$$

Then, when the error becomes zero $D_2 = D_0$. If D_1 keeps increasing, this implies that the probe has picked up the micro-sphere. Differentiating (5.5) to get

$$\dot{e} = \dot{D}_2 \tag{5.6}$$

and further

$$\ddot{e} = \ddot{D}_2 = \ddot{Y}_p - \ddot{D}_1 \tag{5.7}$$

Let r be the filtered tracking error which is defined as,

$$r = \dot{e} + \Lambda e \tag{5.8}$$

where $\Lambda \in R$ is a positive design parameter. Further, differentiating (5.8) yields

$$\dot{r} = \ddot{e} + \Lambda \dot{e} \tag{5.9}$$

Substituting for $\ddot{e}$ and $\dot{e}$ from (5.6) and (5.7) results in

$$\dot{r} = \ddot{Y}_p - \ddot{D}_1 + \Lambda \dot{D}_2 = (F_1(Y_p) - F_2(D_1)) + \Lambda \dot{D}_2 + v \tag{5.10}$$

where

$$F_1(Y_p) = \frac{1}{m_p}(-F_{bp}^{VdW} - F_{bp}^{cap} - F_{bp}^{elec} - m_p g) \tag{5.11}$$

and

$$F_2(D_1) = \frac{1}{m_b}(F_{bp}^{VdW} + F_{bp}^{cap} + F_{bp}^{elec}) - \frac{1}{m_b}\left(F_{bs}^{VdW} + F_{bs}^{cap} + F_{bs}^{elec} + m_b g\right) \tag{5.12}$$

and v is the control input given by

$$v = \frac{1}{m_p} F_{ext} \tag{5.13}$$

Thus the tracking error dynamics can be rewritten as

$$\dot{r} = F(X) + \Lambda \dot{D}_2 + v \tag{5.14}$$

where $F(X) = F_1(Y_p) - F_2(D_1)$ is an unknown nonlinear function and $X = \left[Y_p, D_2\right]^T \in R^2$.

B. Robust Controller Design

Robust controllers have been widely implemented in dynamic systems with unknown or slowly-varying uncertain parameters. In our system, a typical robust saturation controller can be selected as

$$\tau = -\hat{F}\left(X\right) - \Lambda \dot{D}_2 - k_v r - v_1 \tag{5.15}$$

where $k_v \in R$ is the feedback gain and the auxiliary feedback signal v_1 is chosen later with $\hat{F}(X)$ is an estimate for $F(X)$ that is not updated online.

Assumption 1: Let $F_M(X)$ is a known scalar function that bounds the uncertainties $\tilde{F}(X) = F(X) - \hat{F}(X)$ so that

$$\left\|\tilde{F}(X)\right\| \le F_M(X) \tag{5.16}$$

The intent is that $F_M(X)$ is a simplified function that can be computed using the bounding properties of the forces that act upon the micro sphere. The assumption is standard in robust control literature such as sliding mode and others [7], [33]. Observing the micro-forces in Section III, it can be seen that the forces are upper bounded.

Regardless of the saturation constraint, let $v = \tau$ and apply (5.15) in (5.14) to get

$$\dot{r} = -k_v r + \left(F(X) - \hat{F}(X)\right) + v_1 \tag{5.17}$$

or

$$\dot{r} = -k_v r + \tilde{F}(X) + v_1 \tag{5.18}$$

where $\hat{F}(X)$ is an accurate estimate of $F(X)$ and in the presence of no auxiliary signal, then $\tilde{F}(X) \to 0$ and (5.18) becomes

$$\dot{r} = -k_v r \tag{5.19}$$

If k_v is properly selected as a positive constant, then from (5.19) and (5.9), one can readily see that $e \to 0$ with $t \to \infty$. Thus, $D_2 = D_0$ and the sphere is said to be manipulated (pick-up task).

However, MEMS and other typical actuators have magnitude constraints and, as a result, the closed-loop stability analysis is more involved since the magnitude constraints of the actuator are treated as saturation nonlinearity. Assuming $v_{\max}$ is the upper limit for the actuator, in order to incorporate the magnitude constraints with the controller, now select the control input as

$$v = \begin{cases} \tau(t), & if \quad |\tau(t)| \le v_{\max} \\ v_{\max}\operatorname{sgn}(\tau(t)), & if \quad |\tau(t)| > v_{\max} \end{cases} \tag{5.20}$$

where v is the actual control input and τ is the desired applied force, which is selected to be equal to (5.15). Hence, we define $\Delta u = v - \tau$ or $v = \tau + \Delta u$. Using (5.20) in (5.14) now results in $\dot{r} = -k_v r + \tilde{F}(X) + v_1 + \Delta u$ where Δu can be regarded as a disturbance. In order to combat the disturbance, define $\dot{e}_\Delta$ as

$$\dot{e}_\Delta = -k_v e_\Delta + \Delta u \tag{5.21}$$

Now define the error as

$$e_u = r - e_\Delta \tag{5.22}$$

Differentiating (5.22) and substituting (5.21) in (5.22) to get

$$\dot{e}_u(t) = -k_v e_u(t) + \tilde{F}(X) + v_1 \tag{5.23}$$

Select the auxiliary input in (5.15) as [7]

$$v_1 = \begin{cases} -e_u \dfrac{F_M(X)}{|e_u|}, & if\ |\tau(t)| \le v_{\max}\ |e_u| \ge \beta \\ -e_u \dfrac{F_M(X)}{\beta}, & if\ |\tau(t)| \le v_{\max}, |e_u| < \beta \\ 0, & if\ \ |\tau(t)| > v_{\max} \end{cases} \tag{5.24}$$

In computing the robust control term v_1, β is chosen as a small design parameter.

Theorem 1: Consider the system given in (4.1) - (4.3), and take the Assumption 1. Then using the robust controller (5.20), the error, $|e_u|$, $|r|$ and $|e|$ is eventually bounded to the neighborhood of β.

Proof: We will take the case when $|\tau(t)| \le v_{\max}$. Select the Lyapunov function candidate

$$L = \frac{1}{2}\ \mathrm{e}_\mathrm{u}^2 \tag{5.25}$$

Differentiate the above equation and substituting error dynamics (5.23) to get

$$\dot{L} = -k_v e_u^2 + e_u \tilde{F}(X) + e_u v_1 \quad \le -k_v e_u{}^2 + |e_u| F_M(X) + e_u v_1 \tag{5.26}$$

There are now two cases to consider – $|e_u| \ge \beta$ and $|e_u| < \beta$.

Case 1: $|e_u| \ge \beta$. In this case, according to the definition of the robust control term (5.24), one has

$$\dot{\mathrm{L}} \le -k_v e_\mathrm{u}{}^2 + |e_u| F_M(X) - e_u{}^2 F_M(X)/|e_u| \le -k_v e_\mathrm{u}{}^2 \tag{5.27}$$

Therefore, $\dot{L}$ is negative in terms of $|e_u|$. Hence L is decreasing in this region and $|e_u|$ decreases towards β.

Case 2: $|e_u| < \beta$. In this case, according to the definition of the robust control term (5.24), one has

$$\dot{L} \le -k_v e_u^2 + |e_u| F_M(X) - e_u^2 F_M(X)/\beta \le -k_v e_u^2 + |e_u| F_M(X)(1 - |e_u|/\beta) \tag{5.28}$$

The last term is generally positive in this region, so nothing can be said about whether L is increasing or decreasing. In general L may be increasing in this region so that $|e_u|$ increases towards β. Given the boundedness of $|e_u|$ and using (5.22), one can conclude $|r|$ is bounded. Using (5.8), $|e|$ is bounded.

Similarly the proof can be shown when $|\tau(t)| > v_{\max}$.

C. Adaptive Neural Network Controller Design

In the above section, a robust controller with input magnitude constraints is presented wherein the unknown dynamics of the manipulation system is overcome by assuming a bounded known function. In this subsection, an adaptive neural network (NN) [11] is utilized where the unknown manipulation dynamics are approximated online.

First, an action NN is employed to approximate this unknown system dynamics. According to [12], a single layer NN can be used to approximate any nonlinear continuous function over the compact set when the input layer weights are selected at random and held constant whereas the output layer weights are only tuned provided sufficiently large number of nodes in the hidden-layer is chosen. Therefore, a single layer NN is employed here whose output is defined as $\hat{w}_1^T \varphi\left(v_1^T X\right)$, where $\hat{w}_1 \in R^{n_1}$ and $v_1 \in R^{2 \times n_1}$ are the output and input layer weights, n_1 is the number of the hidden layer nodes, $\varphi(\cdot)$ is the activation function vector, and $X = \left[Y_p, D_2\right]^T \in R^2$ is the action neural network input. For simplicity, the action NN output is expressed as

$$\hat{F}(X) = \hat{w}_1^T \varphi(X) \tag{5.29}$$

Thus, the adaptive neural network control input can be selected as

$$v = -\hat{F}(X) - \Lambda \dot{D}_2 - k_v r \tag{5.30}$$

where $k_v \in R$ is the feedback gain selected to be positive constant.

Applying (5.30) in (5.14) to get

$$\dot{r} = -k_v r + \left(F(X) - \hat{F}(X)\right) \tag{5.31}$$

or

$$\dot{r} = -k_v r + \tilde{F}(X) \tag{5.32}$$

where $\tilde{F}(X) = F(X) - \hat{F}(X)$ is the function approximation error. When the neural network is properly trained and $\hat{F}(X)$ is an accurate estimate of $F(X)$, then $\tilde{F}(X) \to 0$ and (5.32) becomes

$$\dot{r} = -k_v r \tag{5.33}$$

If k_v is properly selected as a positive constant, then from (5.33) and (5.8) one can see that $e \to 0$ with $t \to \infty$. Thus, $D_2 = D_0$ and the sphere is said to be manipulated (pick-up task) with D_1 keeps increasing.

The unknown function $F(X)$ can be approximated by the action NN as

$$F(X) = w_1^T \varphi\left(v_1^T X\right) + \varepsilon(X) = w_1^T \varphi(X) + \varepsilon(X) \tag{5.34}$$

where $w_1 \in R^{n_1}$ is the target output layer weight, and $\varepsilon(X)$ is the NN approximation error. Define the weight estimation error $\tilde{w}_1 \in R^{n_1}$ by

$$\tilde{w}_1 = w_1 - \hat{w}_1 \tag{5.35}$$

Thus (5.31) becomes

$$\dot{r} = -k_v r + \tilde{w}_1^T \varphi(X) + \varepsilon(X) \tag{5.36}$$

At the same time, a critic NN is implemented to evaluate the system performance index and tunes the action generating NN. The input to the critic NN is chosen as [11]

$$z(t) = \int_0^t r^2(\tau)\, d\tau \tag{5.37}$$

A choice of the critic NN signal is given by

$$R(t) = \hat{w}_2^T \sigma\left(v_2^T z(t)\right) = \hat{w}_2^T \sigma\left(z(t)\right) \tag{5.38}$$

where $\hat{w}_2 \in R^{n_2}$ and $v_2 \in R^{n_2}$ are the output and input layer weights, n_2 is the number of the hidden layer nodes, $\sigma(\cdot)$ is the hidden layer activation function vector, and $z(t) \in R$ is the input to the neural network. The critic NN input defines the long term system performance over time. The critic signal, $R(t)$, provides an additional corrective action based on current and past performance. This information along with filtered tracking error is used to tune the action NN. The critic signal can also be viewed as a look-ahead factor, which is determined based on past performance. The proposed reinforcement learning-based NN controller structure is depicted in Fig. 5. An inner action generating NN loop eliminates the nonlinear dynamics of the system, while the adaptive NN critic design is modular so that existing industrial controller can be easily updated to obtain the proposed one by simply adding the inner NNs. This modular design avoids the need for the redesign of the industrial control systems [15].

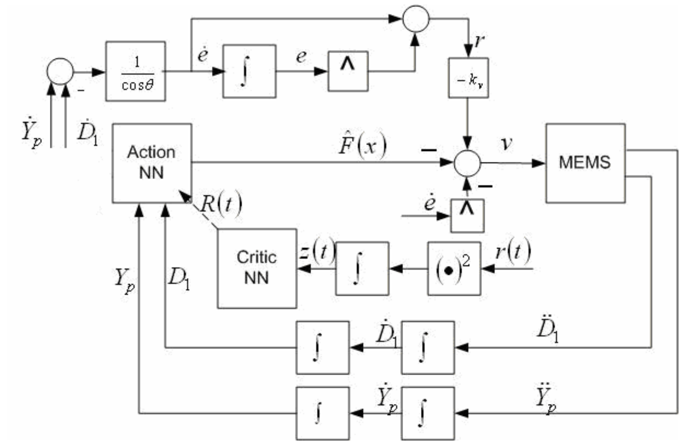


Fig 5. NN Controller Architecture

The next step is to determine the weight updates so that the performance of the closed-loop tracking error dynamics is guaranteed.

Assumption 2: The desired trajectory D_0 is bounded so that $|D_0| < D_B$ with D_B a known scalar bound. In fact, D_0 becomes the inter-atomic distance.

Assumption 3: The NN approximation error $\varepsilon(X)$ is bounded above by $|\varepsilon(X)| < \varepsilon_N$ over the compact set.

Assumption 4: Both the ideal weights and the activation functions for all NNs are bounded by known positive values so that

$$\|w_1\| \le w_{1\max}, \ \|w_2\| \le w_{2\max} \tag{5.39}$$

$$\|\sigma(\cdot)\| \le \sigma_{\max}, \ \|\varphi(\cdot)\| \le \varphi_{\max} \tag{5.40}$$

Theorem 2: Consider the system given in (4.1) through (4.3), and take the Assumptions 2 through 4. Let the action NN weights tuning algorithm be given by

$$\dot{\hat{w}}_1 = \varphi(X)\left(r - \hat{w}_1^T\varphi(X) + k_1 R(t)\right) \tag{5.41}$$

where k_1 is a design parameter and $R(t)$ is the critic signal, which is given by the critic NN in (5.38). The critic NN weights be tuned by

$$\dot{\hat{w}}_2 = -\sigma\left(z(t)\right)\left(r + R\left(t\right)\right) \tag{5.42}$$

with the control signal selected by (5.30). Then the filtered tracking error r and the NN weights estimates, $\hat{w}_1$ and $\hat{w}_2$, are UUB provided

(1) $k_v > 1/2$ (5.43)

(2) $0 < k_1 < 1$ (5.44)

Proof: Since $\dot{\tilde{w}}_1 = -\dot{\hat{w}}_1$, the updating rules (5.41) can be rewritten as

$$\begin{aligned}\dot{\tilde{w}}_1 &= \varphi\left(X\right)\left(-r + \hat{w}_1^T\varphi\left(X\right) - k_1 R\left(t\right)\right)\\ &= \varphi\left(X\right)\left(-r - \overline{e}_1 + w_1^T\varphi\left(X\right) + k_1\tilde{w}_2^T\sigma\left(z\left(t\right)\right) - k_1 w_2^T\sigma\left(z\left(t\right)\right)\right)\\ &= \varphi\left(X\right)\left(-r - \overline{e}_1 + k_1\overline{e}_2 + \eta_1 - k_1\eta_2\right)\end{aligned} \tag{5.45}$$

where

$$\overline{e}_1 = \tilde{w}_1^T\varphi\left(X\right),\ \overline{e}_2 = \tilde{w}_2^T\sigma\left(z\left(t\right)\right),\ \eta_1 = w_1^T\varphi\left(X\right),\ \eta_2 = w_2^T\sigma\left(z\left(t\right)\right) \tag{5.46}$$

Similarly, (5.42) can be rewritten as

$$\dot{\tilde{w}}_2 = \sigma\left(z\left(t\right)\right)\left(r - \overline{e}_2 + \eta_2\right) \tag{5.47}$$

The Lyapunov function candidate is defined as

$$V = \frac{1}{2}\left(r^2 + \tilde{w}_1^T\tilde{w}_1 + \tilde{w}_2^T\tilde{w}_2\right) \tag{5.48}$$

Differentiating (5.48) to get

$$\dot{V} = r\dot{r} + \tilde{w}_1^T\dot{\tilde{w}}_1 + \tilde{w}_2^T\dot{\tilde{w}}_2 \tag{5.49}$$

Substitution of (5.36), (5.45) and (5.47) into (5.49)

$$\begin{aligned}\dot{V} &= r(-k_v r + \tilde{w}_1^T \varphi(X) + \varepsilon(X)) + \tilde{w}_1^T \varphi(X)(-r - \overline{e}_1 + k_1 \overline{e}_2 + \eta_1 - k_1 \eta_2) \\ &\quad + \tilde{w}_2^T \sigma(z(t))(r - \overline{e}_2 + \eta_2) \\ &\le (-k_v r^2 + r\varepsilon(X) + \frac{1}{2}(r^2 + \overline{e}_2^2)) + (-\overline{e}_2^2 + \overline{e}_2 \eta_2) \\ &\quad + (-\overline{e}_1^2 + \frac{1}{2} k_1 (\overline{e}_1^2 + \overline{e}_2^2) + \overline{e}_1(\eta_1 - k_1 \eta_2))\end{aligned} \tag{5.50}$$

Simplify (5.50) to get

$$\dot{V} \le -\frac{1}{2}\left(2k_v - 1\right) r^2 + r\varepsilon\left(X\right) - \frac{1}{2}\left(2 - k_1\right)\overline{e}_1^2 + \overline{e}_1\left(\eta_1 - k_1\eta_2\right) - \frac{1}{2}\left(1 - k_1\right)\overline{e}_2^2 + \overline{e}_2\eta_2 \tag{5.51}$$

Complete the square to get

$$\begin{aligned}\dot{V} &\le -\frac{1}{2}(2k_v - 1)\left(r - \frac{\varepsilon(X)}{(2k_v - 1)}\right)^2 - \frac{1}{2}(2 - k_1)\left(\overline{e}_1 - \frac{\left(\eta_1 - k_1\eta_2\right)}{(2 - k_1)}\right)^2 \\ &\quad - \frac{1}{2}(1 - k_1)\left(\overline{e}_2 - \frac{\eta_2}{(1 - k_1)}\right)^2 + D^2\end{aligned} \tag{5.52}$$

where

$$D^2 \le D_{\max}^2 = \frac{1}{2}\left(\frac{\varepsilon_N^2}{\left(2k_v - 1\right)} + \frac{2\left(w_{1\max}^2 \varphi_{\max}^2 + w_{2\max}^2 \sigma_{\max}^2\right)}{2 - k_1} + \frac{w_{2\max}^2 \sigma_{\max}^2}{1 - k_1}\right) \tag{5.53}$$

This further implies that the $\dot{V} < 0$ as long as (5.43) and (5.44) hold and

$$|r| > \frac{\varepsilon_N}{\left(2k_V - 1\right)} + \frac{\sqrt{2} D_{\max}}{\sqrt{2k_v - 1}} \tag{5.54}$$

or

$$|\overline{e}_1| > \frac{w_{1\max}\varphi_{\max} + k_1 w_{2\max}\sigma_{\max}}{(2 - k_1)} + \frac{\sqrt{2} D_{\max}}{\sqrt{2 - k_1}} \tag{5.55}$$

or

$$|\overline{e}_2| > \frac{w_{2\max}\sigma_{\max}}{\left(1 - k_1\right)} + \frac{\sqrt{2} D_{\max}}{\sqrt{1 - k_1}} \tag{5.56}$$

According to a standard Lyapunov extension theorem [7], this demonstrates that the filtered tracking error and the error in weight estimates are UUB. The boundedness of $|\bar{e}_1|$ and $|\bar{e}_2|$ implies that $\|\tilde{w}_1\|$ and $\|\tilde{w}_2\|$ are bounded, and this further implies that the weight estimates $\hat{w}_1$ and $\hat{w}_2$ are bounded.

D. Adaptive NN Controller with High-Gain Observer

In the above subsections, the robust and the adaptive NN controllers are proposed based on state feedback. However, in practical applications, Y_p and D_1 are usually observed by a laser measuring system [32] or a scanning electron microscope (SEM) [13], which has measurement noise making the measurements inaccurate. In this regard, we extend our adaptive NN controller to an output feedback version by implementing a high-gain observer. Similar extensions can be done for the robust controller.

Consider the system dynamics stated in (4.1) through (4.3), the separation principle can be applied to separate the state feedback controller scheme with the high gain observer design [33].

By assuming that the outputs are y_1 and y_2 corresponding to Y_p and D_1, respectively, but with measurement noise, a high-gain observer is designated as

$$\begin{aligned}
&\dot{x}_1 = x_2 + (2/\varepsilon_1)(y_1 - x_1) \\
&\dot{x}_2 = -F_{bp}^{VdW}/m_b + v/m_p + (1/\varepsilon_1^2)(y_1 - x_1) \\
&y_1 = Y_p + \rho_1 \\
&\dot{x}_3 = x_4 + (2/\varepsilon_2)(y_2 - x_3) \\
&\dot{x}_4 = (F_{bp}^{VdW} - F_{bs}^{VdW})/m_b + (1/\varepsilon_2^2)(y_2 - x_3) \\
&y_2 = D_1 + \rho_2
\end{aligned} \tag{5.57}$$

where x_1 and x_2 are the estimates of Y_p and its velocity, while x_3 and x_4 are the estimates of D_1 and its velocity with ε_1, ε_2 are small design constants. Here, we introduce ρ_1 and

ρ_2 as the measurement noise. Further, the terms of $-F_{bp}^{VdW}/m_b$ and $(F_{bp}^{VdW}-F_{bs}^{VdW})/m_b$ in the second and fifth equations are the nominal model of the observer, which is a simplified version of the model discussed in the above section considering the fact that van der Waals forces are the dominant adhesive forces [1] and $m_p \ll m_b$.

Assumption 5: The measurement noise and their derivatives up to the second order are bounded [34] by $|\rho_i| \le \rho_{iN}$, $|\dot{\rho}_i| \le \rho'_{iN}$, $|\ddot{\rho}_i| \le \rho''_{iN}$ for i = 1, 2.

Assumption 6: The derivatives of function $F_1(Y_p)$ and $F_2(D_1)$ over the compact set are bounded by $|\dot{F}_1| \le F'_{1M}$ and $|\dot{F}_2| \le F'_{2M}$.

Assumption 6 is a mild assumption from micro-scale physics implying that there will be no change in the force by infinite magnitude.

By applying separation principle, an output feedback adaptive NN controller is obtained by replacing the states Y_p and D_1 by their estimate x_1 and x_3 provided by the high-gain observer in (5.57), respectively. In other words, the control input now is selected as

$$v = -\hat{F}(\hat{X}) - \Lambda \dot{\hat{D}}_2 - k_v \hat{r} \tag{5.58}$$

where $\hat{X} = \left[x_1, \hat{D}_2\right]^T \in R^2$, $\hat{D}_2 = x_1 - x_3 - 2R_b$, $\dot{\hat{D}}_2 = x_2 - x_4$ and $\hat{r} = \dot{\hat{D}}_2 + \Lambda(\hat{D}_2 - D_0)$.

The updating laws for NNs are also changed to

$$\dot{\hat{w}}_1 = \varphi\left(\hat{X}\right)\left(\hat{r} - \hat{w}_1^T \varphi\left(\hat{X}\right) + k_1 \hat{R}\left(t\right)\right) \tag{5.59}$$

$$\dot{\hat{w}}_2 = -\sigma\left(\hat{z}(t)\right)\left(\hat{r} + \hat{R}\left(t\right)\right) \tag{5.60}$$

where $\hat{R}(t) = \hat{w}_2^T \sigma\left(\hat{z}(t)\right)$ and $\hat{z}(t) = \int_0^t \hat{r}^2(\tau)d\tau$.

Hence, we have following theorem.

Theorem 3: Consider the system given in (4.1) through (4.3) and the output feedback controller (5.58) with updating law (5.59) – (5.60). Let Assumptions 5 and 6 hold. Consider the original state feedback controller (5.30) and the updating law (5.41) and (5.42), the filtered tracking error and the NN weights estimates are UUB. Then, there exists ε_{1M}, ε_{2M}, such that, for every $0<\varepsilon_1<\varepsilon_{1M}$, $0<\varepsilon_2<\varepsilon_{2M}$, the filtered tracking error and the NN weights estimates of the closed-loop system with the output feedback controller (5.58) is UUB.

Proof: The proof is divided into two steps. First step is to take care of the measurement noise. And the second step is to prove the UUB of the closed-loop system.

In the first step, consider the observer for D$_1$. Let $z_1 = Y_p + \rho_1$, $z_2 = \dot{Y}_p + \dot{\rho}_1$. The original system (4.2) and the output can be rewritten as

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= F_1(z_1 - \rho_1) + v + \ddot{\rho}_1 \\ y_1 &= z_1 \end{aligned} \tag{5.61}$$

Further, by using Mean Value Theorem, one can rewrite (5.61) as

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= F_1(z_1) + v - \dot{F}_1(\zeta_1)\rho_1 + \ddot{\rho}_1 \\ y_1 &= z_1 \end{aligned} \tag{5.62}$$

where $\zeta_1 \in [0 \;\; \rho_1]$ or $\zeta_1 \in [\rho_1 \;\; 0]$. Thus $-\dot{F}_1(\zeta_1)\rho_1 + \ddot{\rho}_1$ can be considered as a disturbance, which appears to be bounded from Assumptions 5 and 6. In other words, a new reference system without measurement noise can be constructed. Same analysis applies for D$_1$. Such a high-gain observer design based system is thoroughly discussed in

[33]. Moreover, one can readily assert that the adaptive NN controller based on state feedback can be translated to system (5.62) with UUB stability.

Thereafter, the second step is similar to the proof in [33] and thus omitted in this paper. As a result, the tracking error in terms of z_1, z_2 and the NN weights estimates are UUB. Due to boundedness of the measurement noise, one can conclude that the filtered tracking error and the NN weights estimates of the original closed-loop system are UUB.

VI. SIMULATION RESULTS

To substantiate our methods, simulation results are shown in this section. The purpose of the controller is to provide a control force for the probe to pick up the micro object. Initially, it is assumed that the object is in contact with the substrate before it is picked up by the probe. The controller provides the force to cause the actual capture and to retain the micro-sphere at the tip of the probe. Once the capture occurs, and the external force to be applied through the probe is determined and maintained to keep the micro-sphere attached to the probe.

The dynamics of the system are expressed as (4.1) through (4.3), with $m_p = 1.0 \times 10^{-5} kg$ the mass of the probe, $m_b = 1.0 \times 10^{-7} kg$ the mass of the micro sphere, and $R_b = 50 \mu m$ is the radius of the micro sphere. Initially, the probe is assumed to park right above the object at a height of $100 \mu m$, which means the approaching angle $\theta = 0^\circ$. That is also the typical way to approach micro objects for capturing [4], [26]. The surface roughness is assumed to be $1.0 \times 10^{-10} m$ [27]. The humidity is arbitrarily set to 50% [28]. To testify the controller designs, model uncertainties and environmental noise are added in the system (4.1) and (4.2) as Gaussian form with zero mean and $\sigma^2 = 1.0 \times 10^{-9} N^2$.

For comparison, a traditional PD controller is first designed based on the filtered tracking error with the control input selected as $v = -\Lambda\dot{D}_2 - k_v r$, where $k_v = 5, \Lambda = 10^{-3}$. Fig. 6 shows the trajectories of the probe and micro object, while Fig. 7 shows the control input. In Fig. 6, the trajectories of D_1 and D_2-D_0 are depicted. The goal of the controller is drive the probe to adhere the particle, which means that D_1 should increase while maintaining D_2-D_0 to be zero at the same time. Although the PD controller is easy for implementation and capable of picking up the micro sphere, it was found that the applied force appears to be highly oscillatory as depicted in Fig. 7. These oscillations might damage the fragile sample or even the probe.

Meanwhile, Fig. 8 displays the trajectories, while Fig. 9 depicts the applied force on the probe by using a robust controller. The controller parameters are also chosen as $k_v = 5, \Lambda = 10^{-3}$ in (5.15). In estimating $F(X)$, we set $\hat{F}(X) = \frac{1}{m_b}(F_{bs}^{VdW} - F_{bp}^{VdW}) = \frac{A_{bs}^{w} R_b}{6m_b}(\frac{1}{D_1} - \frac{1}{6D_2})$, since usually van der Waals force is the dominant adhesive force [1] and $m_p \ll m_b$. Further, $F_M(X) = \hat{F}(X)/10$ and $\beta = 0.1\mu m$ in (5.24). The results show that the robust controller could avoid the large scale force oscillation before grabbing the object successfully. However, due to the model uncertainties and other unknown parameters, the controller output still demonstrates a small fluctuation.

Fig. 10 shows the distances and Fig. 11 shows the control input resulting from using a reinforcement learning-based controller with $k_v = 5, \Lambda = 10^{-3}, k_1 = 0.8$. In both the action and critic NNs, hyperbolic tangent sigmoid transfer function is used. The hidden layer of the action NN consists of 10 nodes, while the critic NN consists of 5 nodes.

Simulation results show that the NN controller can approximate the unknown system dynamics and avoids the oscillation phenomenon. Furthermore, because of the learning ability of NN, the influence of the unknown uncertainties is greatly reduced.

For quantifying the comparison results, we utilize a cost function to measure the performance of each controller, which is widely used for comparing control designs [17]-[18]. In this paper, we define a standard quadratic cost function as following

$$J(t_0,t_f)=\int_{t_0}^{t_f}(D_2(t)-D_0)^T Q(D_2(t)-D_0)+v^T(t)Rv(t)dt \tag{6.1}$$

where R and Q are positive definite matrices (they are scalar in our case). t_0 is the initial time while t_f is the final time of the simulation. One can see that (6.1) represents the amount of effort the controller yields and a measure of the system response. In our work, the parameters are set as $Q=10^4$, $R=10^6$, $t_0=0ms$, $t_f=3.2ms$, respectively. As a result, the performance index for each controller is shown in Table 1.

Table 1 Performance Comparison

Controller type	$J(t_0,t_f)$
PD controller	539.45
Robust controller	231.63
NN controller	168.75

Mainly due to the additional robust auxiliary input, the robust controller design is able to produce a more stable control signal, while achieving a much better outcome than the PD design in terms of the cost. From the table, we can find the PD controller requires more than double effort as its robust coordinate. Moreover, since a critic NN is introduced to evaluate the system performance, the adaptive NN controller succeeds in obtaining the best cost function.

Moreover, to testify the feasibility of our output feedback adaptive NN controller, the simulation is carried on with parameters $\varepsilon_1 = \varepsilon_2 = 0.01$ in (5.57). The measurement noise is also added in the simulation as dual-tone form [34] $\rho_{1,2} = (\sin(t) + 0.5\sin(3.33t)) \times 10^{-6} m$ for both D_1 and Y_p. The system response and the actual applied force are plotted in Figs. 12 and 13, where one can find that although there exists a big variation of control input at the beginning due to the measurement noise and the converging of the observer, the control input becomes steady soon indicating that the observer approximates the actual states as well.

Meanwhile, it can be seen that capture occurs around 10^{-3} s for the robust and the robust NN controllers, which can be observed by the stabilizing of the applied force and trajectory of D_2-D_0. By contrast, it takes longer to capture the micro-sphere by using the PD controller.

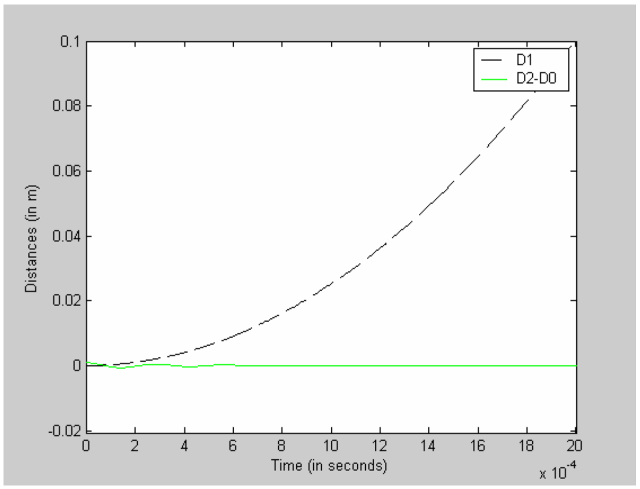


Fig 6. Displacement Using a Conventional PD Controller

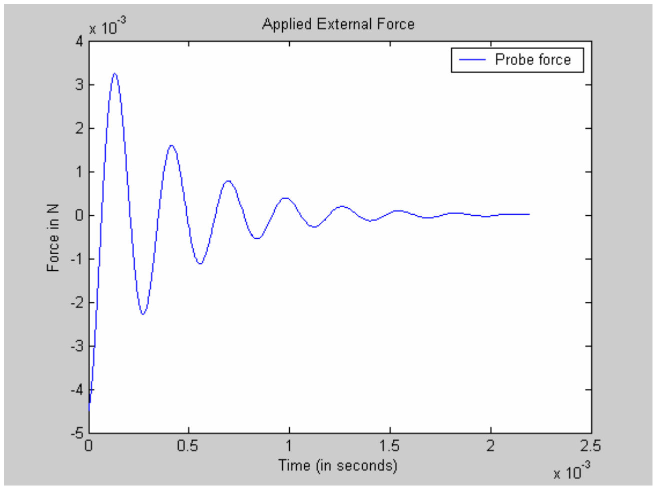


Fig 7. Applied External Force Using a Conventional PD Controller

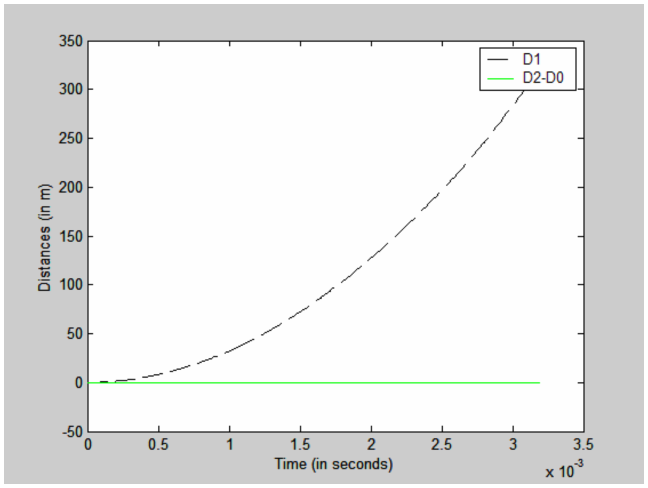


Fig 8. Displacement Using Robust Controller

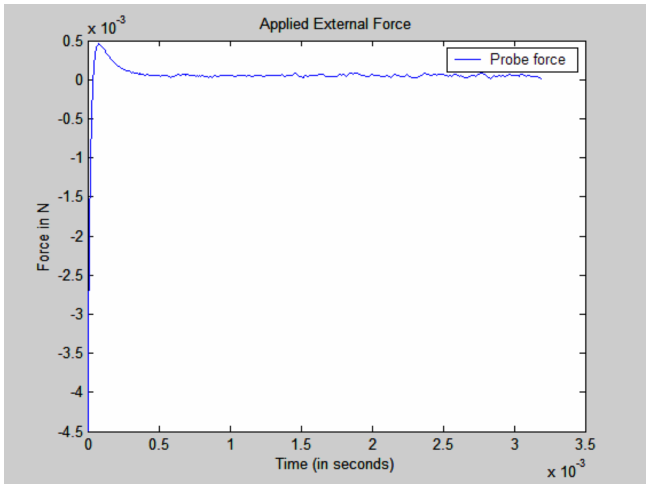


Fig 9. Applied External Force Using Robust Controller

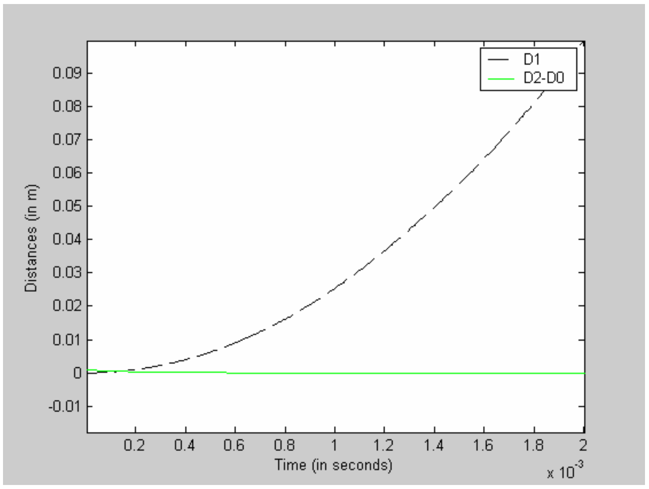


Fig 10. Displacement Using an Adaptive Critic NN Controller

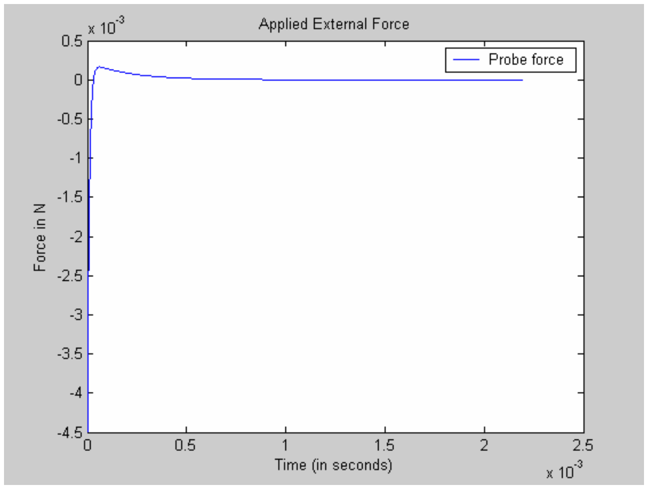


Fig 11. Applied External Force with an Adaptive Critic NN Controller

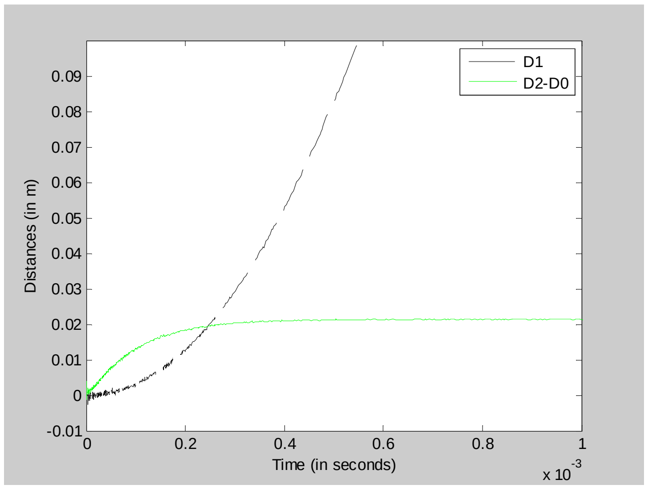


Fig 12. Displacement Using an Output Feedback Adaptive Critic NN Controller

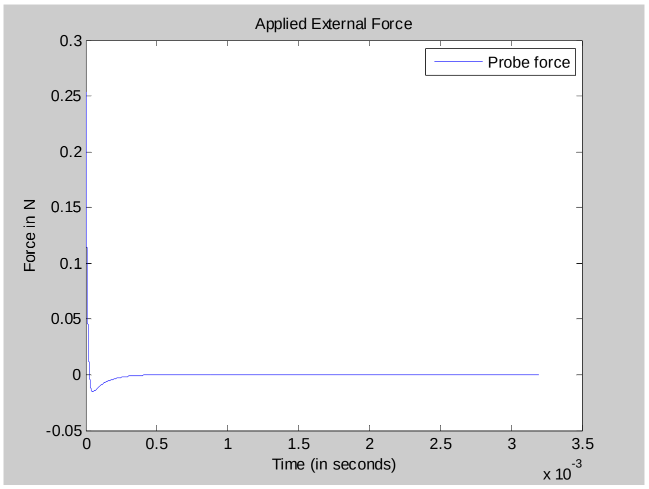


Fig 13. Applied External Force Using Output Feedback Adaptive Critic NN Controller

VII. CONCLUSIONS

In this paper, a suite of robust manipulation controllers was presented for pick-up task of a micro sphere. Closed-loop stability is demonstrated using a robust controller by assuming that the upper bound on the unknown dynamics of the contact forces is known. Then, a reinforcement learning-based adaptive NN controller was presented for the task of picking up a micro-sphere from a substrate wherein the need to know an upper bound on the unknown dynamics is relaxed. The controllers have been proved to have guaranteed stability and the task of manipulation was possible even when the nonlinearities and uncertainties are not modeled for. Simulation results indicate that the robust controller and the NN controller outperform a conventional PD controller in terms of the response time and applied force during the object manipulation. Furthermore, the NN controller has advantage over the robust controller with regard to tolerating model uncertainties and noise. The comparison is strengthened by using a standard quadratic cost function. To overcome the lack of feedback of certain states and the presence of measurement noise, an output feedback adaptive critic-based NN controller with high-gain observer is proposed and verified in a simulation environment.

In the future, experiments need be carried out to substantiate our theoretical conclusions. A better model should be built based on the experiment data and how to obtain a satisfactory estimate of $F(X)$ for the robust controller is also a part of future work.

VIII. ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments. They also would like to thank Pingan He and V. Janardhan for initiating this research topic and their contributions.

IX. REFERENCES

1. J. T. Feddema, P. Xavier, and R. Brown "Micro-assembly planning with van der Waals force," Proc. of the IEEE Int. Symposium on Assembly and Task Planning, pp. 32 – 38, 1999.
2. F. Arai, D. Andou, and T. Fukuda, "Adhesion forces reduction for micro manipulation based on micro physics," Micro Electro Mechanical Systems, 1996, MEMS '96, Proc. of the Ninth Annual International Workshop on an Investigation of Micro Structures, Sensors, Actuators, Machines and Systems, 11-15, pp.354 - 359, 1996.
3. M. Sitti, "Survey of nanomanipulation systems," in Proc. of the IEEE-Nanotechnology Conference, pp. 75-80, Maui, USA, Nov. 2001.
4. Y. Rollot, S. Regnier, S. Haliyo, L. Buchaillot, J. C. Guinot and P. Bidaud, "Experiments on micromanipulation using adhesion forces in unconstrained environment," Proc. of the 2000 IEEE/RS Int. Conf. on Intelligent Robots and Systems, vol. 1, pp. 653 – 658, 2000.

5. F. Arai, D. Andou, Y. Nonoda, T. Fukuda, H. Iwata and K. Itoigawa, “Integrated micro-end effector for micromanipulation,” IEEE/ASME Trans. on Mechatronics, vol. 3, pp. 17 – 23, 1998.

6. S. Saito, H. Miyazaki, and T. Sato, “Pick and place operation of a micro-object with high reliability and precision based on micro-physics under SEM,” Proc. of the IEEE Int. Conf. on Robotics and Automation, vol. 4, pp. 2736 – 2743, 1999.

7. F. L. Lewis, S. Jagannathan and A. Yesilderik, “Neural Network Control of Robot Manipulators and Nonlinear Systems,” Taylor and Francis, 1999.

8. P. J. Werbos, “Neurocontrol and supervised learning: An overview and evaluation,” Handbook of Intelligent Control, edited by David A. White and Donald A. Sofge, Van Nostrand Reinhold, pp. 65-90, 1992.

9. A. G. Barto, “Reinforcement learning and adaptive critic methods,” Handbook of Intelligent Control, edited by David A. White and Donald A. Sofge, Van Nostrand Reinhold, pp. 469-492, 1992.

10. B. Igelnik and Y. H. Pao, “Stochastic choice of basis functions in adaptive function approximation and the functional-link net,” IEEE Trans. Neural Networks, vol. 6, pp. 1320-1329, 1995.

11. V. Janardhan, P. He and S. Jagannathan, “Neural network controller for the manipulation of microscale objects,” Proc. of the IEEE Symposium on Intelligent Control, pp. 55-60, 2004.

12. M. P. Boukallel and J., Abadie, “Micromanipulation tasks using passive levitated force sensing manipulator,” Proc. Of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 529-534, Oct. 2003

13. K. Koyano and T. Sato, “Micro object handling system with concentrated visual fields and new handling skills,” Proc. of the IEEE Conf. on Robotics and Automation, pp.2541-2548, 1996.

14. H. Miyazaki and T. Sato, “Pick and Place Shape Forming of Three-Dimensional Micro Structures from Fine Particles,” Proc. 1996 IEEE Int. Conf. on Robotics and Automation, pp.2535-2540, April 1996.

15. S. Jagannathan and G. Galan, “Adaptive critic neural network-based object grasping control using a three-finger gripper,” IEEE Trans. Neural Networks, vol. 15, no. 2, pp. 395-407, 2004.

16. P. J. Werbos, “New directions in ACDS: Keys to intelligent control and understanding the brain,” in Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks, vol. 3, pp. 61–66, 2000.

17. D. Prokhrov and D. C. Wunch, “Adaptive critic designs,” IEEE Trans. Neural Networks, vol. 8, pp. 997–1007, Sept. 1997.

18. D. Han and S. N. Balakrishnan, “State-constrained agile missile control with adaptive-critic-based neural networks,” IEEE Trans. Contr. Syst. Technol., vol. 10, pp. 481–489, July 2000.

19. J.N Israelachvili, “The Nature of van der Waals Forces,” Contemporary Physics, Vol. 15, No. 2, pp. 159-177, 1974.

20. P.A Thompson and M.O Robbins, “Simulations of contact-line motion: slip and the dynamic contact angle,” Phys. Rev. Lett. 63, pp. 766-769, 1989.

21. J. Crassous, E. Charlaix, H. Gayvallet, and J. Loubert, “Experimental study of a nanometric liquid bridge with a surface force apparatus,” Langmuir, vol. 9, no. 8, pp. 1995–1998, 1993.

22. S. Jagannathan and Qinmin Yang, “A Robust Controller for the Manipulation of Micro-Scale Objects,” Proc. of American Control Conference (ACC '05), pp. 4154-4159, 2005.

23. F. Ohkawa and Y. Yonegawa, “A discrete model reference adaptive system for a plant with input amplitude constraints,” Int. J. Contr., vol. 36, pp. 747–753, 1982.

24. S. P. Karason and A. M. Annaswamy, “Adaptive control in the presence of input constraints,” IEEE Trans. Automat. Contr., vol. 39, pp. 2325–2330, 1994.

25. F. Z. Chaoui, F. Giri, J. M. Dion, M. M-Saad, and L. Dugard, “Direct adaptive control subject to input amplitude constraint,” IEEE Trans. Automat. Contr., vol. 45, pp. 485–490, 2000.

26. H. Miyazaki and T. Sato, “Pick and Place Shape Forming of Three-Dimensional Micro Structures from Fine Particles,” Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, ICRA'96, pp. 2535 - 2540, 1996.

27. Q. Zhou, P. Kallio, F. Arai, T. Fukuda, and H. Koivo, “A Model for Operating Spherical Micro Objects,” Proc. of the 1999 Int. Symposium on Micromechatronics and Human Science, MHS'99, pp. 79 - 85, 1999.

28. Y. Rollot, S. Regnier and J. C. Guinot, “Dynamical model for the micromanipulation by adhesion: experimental validations for determined conditions,” Int. J. of Micromechatronics, vol.1, no.4, pp.273-297, 2002.

29. M. Sitti and H. Hashimoto, “Tele-Nanorobotics Using Atomic Force Microscope,” Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'98, pp. 1739 - 1746, 1998.

30. J.T. Feddema, P. Xavier and R. Brown, “Assembly Planning at the Micro Scale,” Prof. of the IEEE Int. Conf. on Robotics and Automation, Leuven, Belgium, pp. 16-21, 1998.

31. S. Saito, H. Himeno and K. Takahashi, “Electrostatic detachment of an adhering particle from a micromanipulated probe,” J. of Applied Physics, vol. 93, no.4, pp. 2219-2224, February 2003.

32. A. Buerkle and S. Fatikow: “Laser measuring system for a flexible microrobot-based micromanipulation station,” Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 799-804, 2000

33. H.K. Khalil, “Nonlinear Systems,” Prentice Hall, 2002.

34. H.S. Sane, D.S. Bernstein, H.J. Sussmann, “Lyapunov-based output-feedback adaptive stabilization of minimum phase second-order systems,” Proc. of the IEEE Conf. on Decision and Control, vol. 2, pp. 1181 – 1186, 2001.

PAPER 2

Automatic Drift Compensation Using Phase-Correlation Method for Nanomanipulation

Qinmin Yang[1]*, S. Jagannathan*[1] *and E. W. Bohannan*[2]

1. Department of Electrical and Computer Engineering

2. Materials Research Center

University of Missouri – Rolla,

Rolla, Missouri, U.S.A 65409

Email: qyy74@umr.edu, sarangap@umr.edu, bohannan@umr.edu

ABSTRACT

Nanomanipulation and nanofabrication with an Atomic Force Microscope (AFM) or other Scanning Probe Microscope (SPM) is a precursor for nanomanufacturing. It is still a challenging task to accomplish nanomainpulation automatically. In ambient conditions without stringent environmental controls, the task of nanomanipulation requires extensive human intervention to compensate for the spatial uncertainties of the SPM. Among these uncertainties, thermal drift, which affects spatial resolution, is especially hard to solve because it tends to increase with time and cannot be compensated simultaneously by feedback from the instrument.

In this paper, a novel automatic compensation scheme is introduced to measure and estimate drift one-step ahead. The scheme can be subsequently utilized to compensate for the thermal drift so that a real-time controller for nanomanipulation can be designed as if drift did not exist. Experimental results show that the proposed compensation scheme can predict drift with a small error, and therefore can be embedded in the controller for manipulation tasks.

Keywords

Nanomanipulation, Scanning Probe Microscope, thermal drift, phase-correlation method, neural network

I. INTRODUCTION

Nanomanipulation, which aims at manipulating nanometer size objects with nanometer precision, has become possible since 1990 [1] after the invention of scanning tunneling microscopes (STM), atomic force microscopes (AFM) and other types of scanning probe microscope (SPMs). By accurately controlling atoms, molecules, or nano scale objects, numerous applications of nanotechnology can be cited in the area of molecular biology and genetics, solid-state physics, chemistry, material science, computer industry and medicine. By reducing the object size from micro meter to nano meter, new sensors, tera-byte capacity memories, DNA computers, man-made materials, etc., would be possible within the near future [2].

Today, manipulation of particles with the order of 10nm in diameter using Atomic Force Microscopes (AFMs) is being investigated by many researchers [3], [10]-[13], [17]. Preliminary controller designs for nanomanipulation systems were introduced in [14] and [15]. Besides using AFM or other SPM as imaging tools, they are also employed as teleoperated manipulators at the nano scale. However, for future new nanotechnology products, there are still many challenges to be addressed. From macro to nano world, any nonlinearity such as thermal noise even if it is small will cause major hurdles during manipulation with the microscope. Without treating these uncertainties, real-time controller designs will be impractical. Therefore at present, nanomanipulation requires extensive user intervention to compensate for the spatial uncertainties associated with the microscope and its piezoelectric drive mechanism, such as hysteresis, creep, and thermal drift [3], [16]-[17] when operated in ambient conditions without stringent environmental controls.

Among the uncertainties that AFM encounters, hysteresis can be reduced by scanning in the same direction always, while creep effects almost vanish by waiting a few minutes after a large scanning motion [3]. Alternatively, in [16] a comprehensive study is presented on techniques that are being developed to compensate them. Usually, these solutions are normally embedded into AFM software for compensation although they slow down the manipulation tasks.

Nevertheless, unlike other uncertainties, the effect of drift will increase with time and it cannot be compensated automatically by the instrument. In other words, due to temperature change in the ambient environment, the AFM tip drifts with time at a speed of about one atomic diameter per second, even when the voltage inputs for controlling the tip position are held constant. Although drift can be greatly reduced by placing the microscope in a temperature-controlled and ultrahigh vacuum (UHV) environment, this will be expensive and difficult, and therefore limits its applications in industry. At the same time, other uncertainties such as calibration error and instrument noise will be introduced during the manipulation processes and their effects are similar to that of the thermal drift which renders gross manipulation inaccuracies. As a result, it will typically take hours for an experienced operator to construct a pattern with several nano particles using AFM. To efficiently and successfully accomplish such tasks or even more complex ones, automated manipulation is desirable. For an automated nanomanipulation, drift compensation is the first step.

Several researchers have addressed the problem of drift and proposed solutions in [3] through [8] and [17]. However, most of them [4]-[8] are assuming that the drift being held at a constant value. Additionally, [4]-[8] are computing the drift by considering the entire image data although during manipulation part of topography of the sample is going to be changed.. To overcome this problem, in this paper, a block-based phase correlation method is employed to divide the entire image into blocks, using which drift for each block is estimated individually. Thereafter, the drift value of the entire image is computed based on the drift calculation for each block.

Further, to make this method suitable for future real-time controller design, both a neural network (NN) and signal reconstruction technique are also necessary and proposed here. As a matter of fact, given diverse working conditions during manipulation, an artificial neural network (NN) is utilized for predicting drift at the next sampling interval for relaxing the need for drift models. Using signal reconstruction techniques, drift can be expressed as a continuous function of time for any real-time controller design.

The paper is organized as follows. In Section II, the problem of drift is introduced whereas Section III presents the detailed compensation methodology for the drift problem. The system implementation and experimental results are included in Section IV before conclusions.

II. PROBLEM STATEMENT

Mainly due to thermal expansion and contraction of the microscope components and the sample in ambient conditions, drift usually appears in successive AFM scans even when all of the scanning parameters are not altered. In the *x-y* plane (or the horizontal plane), drift can be observed as a translation between different images, as shown in Fig. 1. The drift velocities on the *x-y* plane are reported to vary from 0.01~0.1 nm/s [3]. However during our experiments, the problem due to drift appears to be worse at times. As observed from Fig. 1, the graphite sample is drifting to the left at a speed of around 0.5nm/s. So the drift between any two images taken at 256 sec interval can be as much as 128nm, which is larger than the diameter of the nano particles themselves which are normally manipulated. Meanwhile, from the height data of the sample, it can be observed that drift along the *z* direction is approximately 0.005nm/s during our experiments.

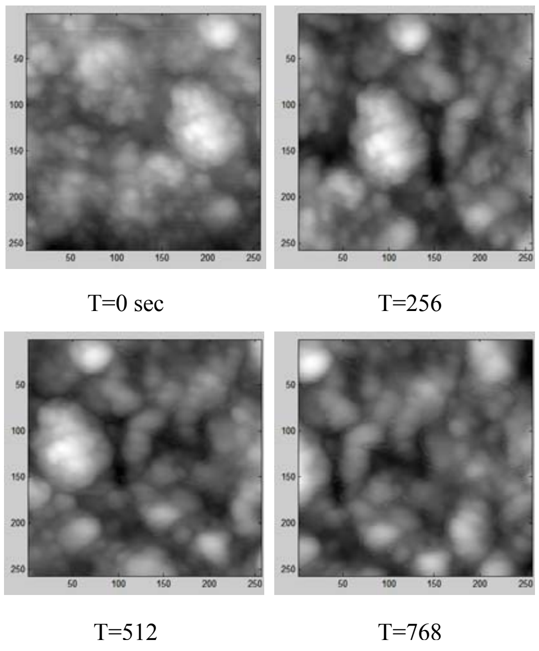


Fig 1. Image Sequences of a Graphite Sample Taken at 256 Sec Intervals by AFM Showing Drift on the *x*, *y* Plane due to Ambient Conditions. The Scanned Area is 512 by 512nm^2.

Unfortunately, measuring drift in the *z* direction precisely will be difficult or impossible because the topographic data provided by an SPM are essentially relative height information in terms of discrete points on the sample surface. Fortunately, considering the vertical drift is comparatively small and has little impact on the controller [3], there is no need to estimate its exact value. Thus, it is normally sufficient for a drift compensation scheme to estimate and compensate the drift along the *x* and *y* directions and under the reduced influence of the noise from the *z* axis, so that automated nanomanipulation can be performed as if drift does not exist.

Past experiments show that the drift along x and y directions can be observed as a translational movement and not rotation [3], [17]. Furthermore, there is negligible correlation between the two directions [3]. Hence, ideally, the height data between the two consecutive collections along with the drift can be written as

$$h_{k+1}(x,y) = h_k(x+\Delta x_k, y+\Delta y_k) + \Delta z_k \tag{1}$$

where Δx_k, Δy_k and Δz_k denote drift in the *x*, *y* and *z* axes, respectively, between time instants k and $k+1$. Here, we assume that the drift along *z* direction for the overall imaging area of the sample is constant, which appears to be a reasonable assumption.

Although several methods [4] - [8] to compensate for the drift in the horizontal plane have been proposed, these techniques fail to provide accurate compensation when the drift velocity changes, as illustrated by the experimental results in Fig. 1. A novel Kalman filter based estimator [3] and compensator is introduced. However, the user still has to select a tracking window and the appropriate model parameters for every experiment, which will be very difficult for automation. Moreover, the techniques in [3]-[8] are based on comparing successive images of an unmodified sample or unmodified part of the sample. Unfortunately, the topography in the scanning region is usually changed during manipulation or fabrication processes. For instance, as shown in Fig. 2, there are two particles being pushed by the operator, and it is not too difficult to notice that there drift exists between the two images. Under this condition, the methods reported in [4]-[8] render inaccurate results. The tracking window technique can solve this problem but it is not a true automatic approach.

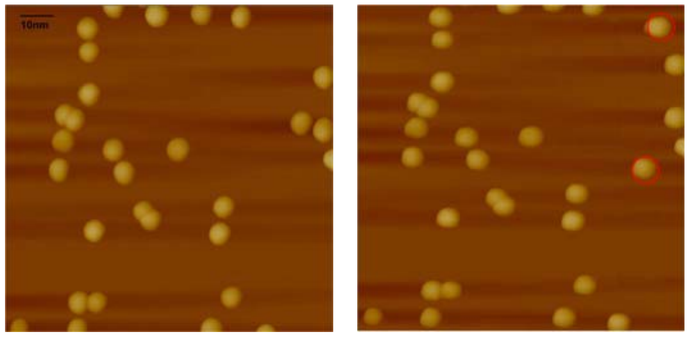


Fig 2. Image Sequences of Gold Particles on Mica Substrate with Nano Manipulation under AFM. Note: There Are Two Particles at the Right Top Corner Moved by the Operator. Drift also Presents Towards Upside.

In this paper, drift will be measured and processed using block-based phase correlation method in a totally automatic manner, and without human intervention and even when some areas of the sample have been altered due to manipulation.

III. COMPENSATION METHODOLOGY

The block diagram of the proposed compensation methodology is depicted in Fig. 3. The entire system will operate in a recursive fashion with a constant sampling interval, where images of the sample are secured by the microscope. Our drift compensator is updated even when the manipulation is carried out elsewhere on the sample during the inter-imaging period, which is the case discussed in [4]-[8].

The entire manipulation scheme will be executed by the following procedure. Once the microscope acquires a most up-to-date image data, the block-based phase correlation algorithm starts and computes a measurement of current drift value. As soon as this computation is done, it delivers the measurement to the neural network (NN) predictor. The NN predictor estimates the drift for the next imaging instant, which in turn is employed by the signal reconstruction block to form a continuous variation of drift as a function of time between current imaging instant and the next one. With the drift information expressed as a continuous function of time, the task of nanomanipulation can be accomplished automatically by the controller as the drift estimate can be explicitly utilized during nanomanipulation.

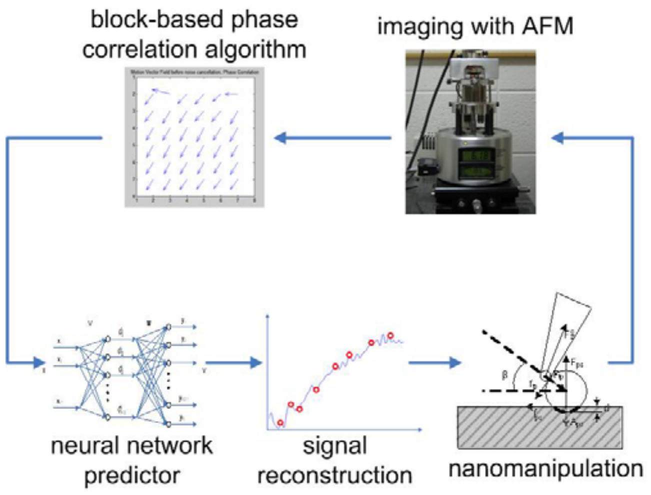


Fig 3. Block Diagram of the Overall Proposed Drift Compensation System

As stated above, there is no correlation between drift along *x* and *y* axes [3]. Therefore, for simplicity, only the drift in the *x* direction is discussed in the following subsections and drift in the *y*-direction can be obtained similarly and therefore omitted.

A. Gradient Imaging

In principle, AFM operates by measuring attractive or repulsive forces between the tip and the sample surface. As a raster-scan drags the tip over the sample, some sort of detection apparatus (e.g. laser) tracks the forces by monitoring the vertical deflection of the AFM cantilever, which indicates the height of the sample locally. Thus, the images provided by AFM are essentially the height data of the sample locally.

It is important to note that drift in the *z* direction depends upon the measurement errors from the *x* and *y* directions. Although it is comparatively small and has little impact on nanomanipulation, it could still influence the accuracy of our drift algorithm. Therefore, to minimize this error, gradient information will be used for measuring drift, which is defined as

$$g_k(x, y) = h_k(x, y) - h_k(x-1, y) \tag{2}$$

From (2), one can find that a new image is formed by using the gradient information and by just taking the height difference between each pixel and the corresponding horizontal neighboring pixel in the original image. This is also called horizontal gradient image. Vertical gradient image can be defined similarly and it is also applicable for our approach. In this paper, only the horizontal gradient is discussed.

Considering the drift factor and substituting (1) into (2) yields

$$\begin{aligned} g_{k+1}(x, y) &= h_{k+1}(x, y) - h_{k+1}(x-1, y) \\ &= h_k(x+\Delta x_k, y+\Delta y_k) + \Delta z_k - (h_k(x+\Delta x_k - 1, y+\Delta y_k) + \Delta z_k) \\ &= g_k(x+\Delta x_k, y+\Delta y_k) \end{aligned} \tag{3}$$

where the effect of *z*-axis drift is eliminated. Moreover, as observed in our experiments, results using gradient-based images will yield accurate results than using the original height data due to the presence of drift along *z* axis. Once the microscope finishes the sample imaging and the gradient calculation, the gradient data is forwarded to the phase correlation module.

B. Block-based Phase Correlation Method

The drift measurement problem is similar to the motion estimation (ME) and compensation (MC) issue in the area of signal processing. Among various techniques, phase correlation technique measures the motion directly from the image, so that it can give a more accurate and robust estimate of the motion vector and a motion field with much lower entropy [9]. Additionally, phase correlation method is computationally very efficient, which will allow more time for manipulation operations between imaging instants. In particular this method shows a better performance on translational and large-scale motion and these are the characteristics that are normally observed in AFM drift.

On the other hand, as we argued in the former section, existing methods [3]-[7] will produce inaccurate results in the presence of topography changes of the sample surface resulting from the manipulation or they need human intervention to mark them manually [8]. In our algorithm, to distinguish the drift from other user-defined operations and further eliminate the drift automatically, the image is divided into blocks, and the drift calculation is performed for each block separately. As a matter of fact, block-based motion estimation and compensation schemes are quite popular in practice due to their robust performance and they do not require object identification. Moreover, they allow some objects in the image to be moved while not influencing the motion estimation of other blocks. This feature makes it easier to estimate the drift of the overall image even when some parts of the sample surface have been altered by operators, which is usually the case in the nanomanipulation environments.

The schematic diagram of our block-based phase correlation method and the parameters used in our experiments are depicted in Fig. 4. Other settings may also be possible for different experimental conditions.

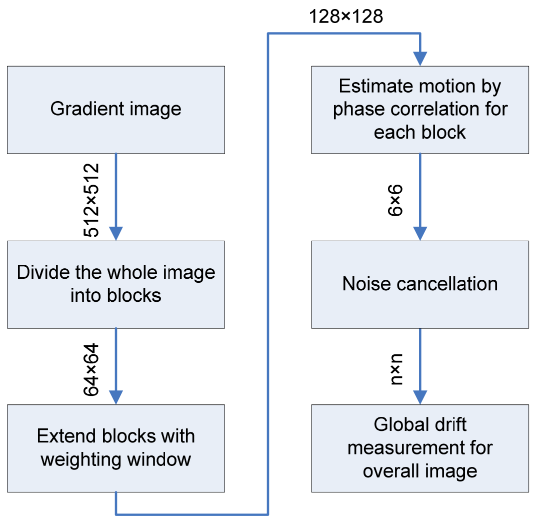

Gradient image
512×512
Divide the whole image into blocks
64×64
Extend blocks with weighting window
128×128
Estimate motion by phase correlation for each block
6×6
Noise cancellation
n×n
Global drift measurement for overall image

Fig 4. Schematic Diagram of Block-based Phase Correlation Method

In the proposed scheme, assuming a new gradient frame with 512 by 512 pixels is received from AFM, it is first divided into 64 by 64 pixel blocks. By using a straightforward calculation, there will be 64 blocks from a 512 by 512 image. The objective of the first step is to estimate the drift value for each block by comparing the new image with the previous one. For more accurately estimating the cross correlation of corresponding block pairs in respective image frames, we extend the blocks to 128 by 128 pixel in size, centered around the formerly defined 64 by 64 pixel blocks for calculation. It can be readily found that, with bigger blocks, the overlapping area between the block pair is larger. Therefore, their correlation can still be kept to be high even with a large amount of drift.

Subsequently, a two-dimensional raised cosine weighting window is applied to each 128 by 128 extended block to enforce more weight on our formerly defined 64 by 64 region. The two-dimensional raised cosine window is defined as follows

$$\begin{aligned} w(x, y) &= w_x \cdot w_y \\ &= \frac{1}{4}\left[1-\cos\left(\frac{2\pi(x+1/2)}{M}\right)\right]\cdot\left[1-\cos\left(\frac{2\pi(y+1/2)}{M}\right)\right], \text{ for x, y} = 1,2, ..., \text{M} \end{aligned} \tag{4}$$

where M is the size of the image, which is equal to 128 for our system. The raised cosine function is also illustrated in Fig. 5, which clearly demonstrates that the pixels in the center are given higher emphasis.

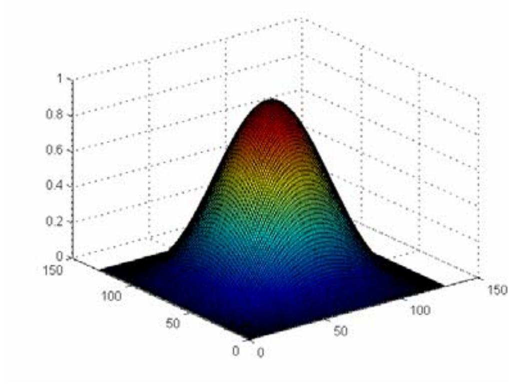


Fig 5. Two-dimensional Raised Cosine Window Dunction

Thereafter, the phase correlation method measures the movement between two blocks directly from their phase values. The basic principle is briefly discussed next.

Assume that there exists a translational shift exists between frames k and $k+1$. In this paper, the same relationship stands for consecutive gradient images, which can be rewritten from (3) as

$$g_{k+1}(x, y) = g_k(x+\Delta x, y+\Delta y) \tag{5}$$

Taking 2-D Fourier transform of (5) yields

$$G_{k+1}(f_x, f_y) = G_k(f_x, f_y) \cdot \exp[j2\pi(\Delta x f_x + \Delta y f_y)] \tag{6}$$

Therefore, displacement in the spatial-domain is reflected as a phase change in the frequency spectrum domain. Further, the cross-correlation between any two frames can be written as

$$c_{k,k+1}(x, y) = g_{k+1}(x, y) \cdot g_k^*(-x, -y) \tag{7}$$

whose Fourier transform is given by

$$C_{k,k+1}(f_x, f_y) = G_{k+1}(f_x, f_y) \cdot G_k^*(f_x, f_y) \tag{8}$$

After normalizing the cross-power spectrum by its magnitude and eliminating the luminance variation influence during our phase analysis, we obtain its phase as

$$\Phi[C_{k,k+1}(f_x, f_y)] = \frac{G_{k+1}(f_x, f_y) G_k^*(f_x, f_y)}{\left|G_{k+1}(f_x, f_y) G_k^*(f_x, f_y)\right|} \tag{9}$$

By substituting (6) into (9), we have

$$\Phi[C_{k,k+1}(f_x, f_y)] = \exp[-j2\pi(\Delta x \cdot f_x + \Delta y \cdot f_y)] \tag{10}$$

whose 2-D inverse Fourier transform is given by

$$c_{k,k+1}(x,\ y) = \delta(x - \Delta x,\ y - \Delta y) \tag{11}$$

where δ is an impulse function on the *x-y* plane. As a result, the displacement in the spatial-domain corresponds to an impulse in the correlation domain. Therefore, by finding the location of the impulse in (11), we are able to obtain an estimate of the displacement, which is represented by a motion vector. In our system, the phase correlation for each block pair in consecutive frames is calculated using 128 by 128 fast Fourier transform (FFT).

In practice, since the motion between any two blocks can not be both pure translational and noise free, usually we have a phase correlation map similar to what is depicted in Fig. 6. Although there is an obvious peak appearing, there are other peaks also and some with noise.

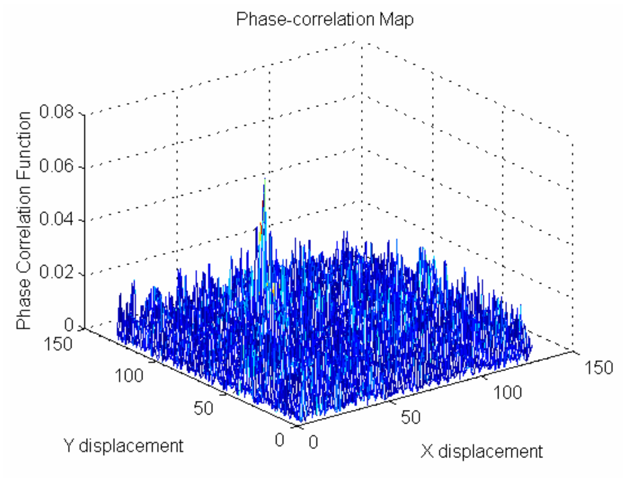


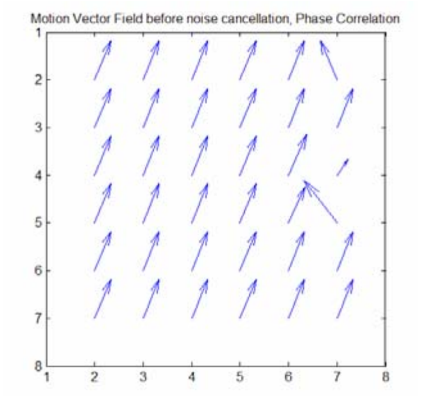
Fig 6. Map of a Typical Phase Correlation Function between Two Blocks

In other words, in ideal situations where there is only a spatial shift between images due to drift, it should be reflected as a single spike after the application of phase correlation technique. Therefore, the highest peak in the phase correlation map usually corresponds to the actual drift value. Even if the images are contaminated with noise, the highest peak still provides the best estimate of drift between two frames. However, for our 128 by 128 pixel extended blocks, due to nontranslational movement and other unexpected noise, several peaks with height closer to one another might be appearing in the correlation map. In this case, several candidates will be selected first instead of just choosing the highest peak. Thereafter, the peak that best represents the displacement vector for the object block has to be found by examining the peaks using image correlation in terms of the mean squared error (MSE) criterion. The candidate possessing the highest image correlation is then identified, and its corresponding drift displacement is accepted as the motion vector for the object block. Note that a maximum drift of +/- 64 pixels is assumed in order to ensure that there exists an overlapping area with enough size between corresponding block pair. In case the drift exceeds the assumption, one may increase the block size to 128 by 128 to solve this problem.

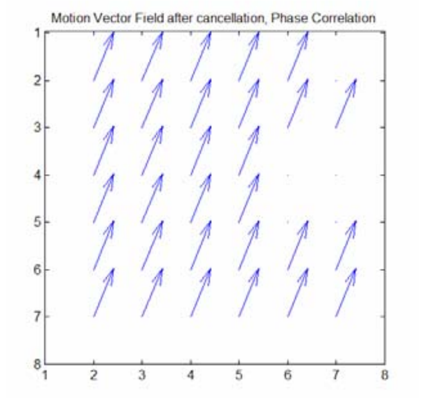
Finally, after the motion vectors for all blocks are computed, we could obtain a motion vector as a field map shown in Fig. 7, which is the result of applying our algorithm on the experiment data shown in Fig. 2. Ideally, if translational drift is the only reason for the motion vectors within the image sequence, a satisfactory drift measurement of the whole frame can be produced by simply calculating the mean of all motion vectors. However, as we have stated before, to fulfill nanomanipulation, some particles or some parts of the sample surface are designed to be altered. Additionally, image data are usually corrupted by noise and other uncertainties at the nano scale. As a result, some of the blocks will have considerably different motion vector values from others as shown in Fig. 7(a). Thus, a specific noise cancellation mechanism is required to pick the "contaminated" blocks out and restrain them from being involved into the final calculation.

By assuming that only a limited area of the sample is altered during a short interval, a simple but effective approach to remove this noise is to first compute the mean of all motion vectors, pick some blocks which are farthest from the mean, and set their values invalid. By this way, a more accurate drift measurement can be obtained by computing the mean of motion vectors of the left blocks.

In practice, a constant threshold value of ε is used in our scheme for noise cancellation. In particular, after getting the mean, all the blocks whose distance to this mean value larger than ε are not considered in the final calculation of the drift measurement for the overall image. Usually, the choosing of ε depends on how large area of the sample is being manipulated, which should be known *a priori*.



(a)



(b)

Fig 7. (a) The Motion Vector Field Corresponding to Fig. 2 before Noise Cancellation. (b) The Motion Vector Field after Noise Cancellation

C. Time Series Prediction with Neural Networks

After obtaining the drift measurement at the current time instant k, the drift behavior in the next time step $k+1$ must be estimated for compensation purpose. In [3], Kalman filter based estimator is introduced for this purpose. Although Kalman filter can provide the best estimation based on maximum likelihood optimization, the model and parameters used in the filter have to be identified beforehand, where a general model and parameter settings are still impossible under multifarious sample materials and varying ambient conditions.

As an alternative, in this paper, a two-layer neural network (NN) is employed for predicting drift in the subsequent time instant, as shown in Fig. 8. The matrices V and W are the hidden layer and output layer weights. Moreover, as noticed from Fig. 8, the number of nodes in input, hidden and output layer is $N+1$, N_2 and 1, respectively, where N denotes the history data utilized in the calculations. It is well known that NN have excellent approximation capability for any nonlinear temporal mapping. Assuming that the environmental conditions will not change much in a short time period, NN can learn the statistical nature of the drift from historical data and other information. In our system, not only the previous drift measurements are forwarded to the NN predictor, but also the temperature fluctuations are measured and taken as an additional input to the NN. The weights of the NN are updated in a supervised training mode along with drift measurement feedback from the phase correlation algorithm.

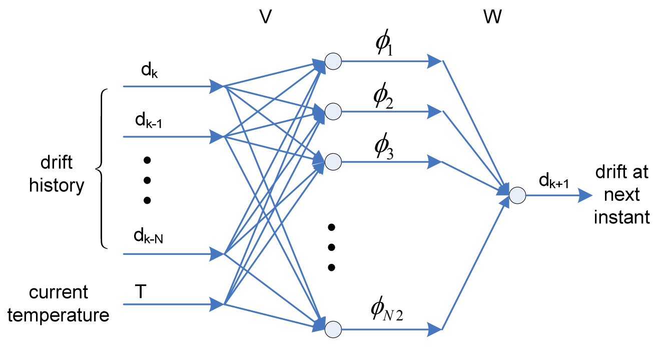


Fig 8. Architecture of the Two-layer NN Predictor

D. Signal Reconstruction Using Sinc Function

For a real-time controller design, it is necessary to obtain a drift description as a continuous function of time from the discrete measured points. Considering that the power spectra of the time series for drift exhibits a bandwidth of the order of 0.001 Hz [3], it is possible to get proper reconstruction results using sinc function, as long as the sampling interval between images are small enough. In our applications, the samples are imaged every 256 sec or the sampling frequency is about 0.004 Hz. Therefore, it is reasonable to use sinc function to reconstruct the drift signal without much loss of information.

Therefore, it follows that

$$d(t) = \sum_{i=0}^{k+1} d_i \cdot \mathrm{sinc}\left(\frac{\pi(t-t_i)}{\Delta t}\right), \text{ for } t \in (t_k, t_{k+1}) \tag{12}$$

where $d(t)$ is the continuous drift function at time between current sampling instant to next one, d_i is the drift measurement ($i = 0,\ 1 \ldots k$) or prediction ($i = k+1$) on sampling time t_i, and Δt is the sampling interval.

IV. IMPLEMENTATION AND EXPERIMENT RESULTS

To verify our proposed work, the drift compensator is implemented on a multimode scanning probe microscope (SPM) with NanoScope IIIa controller (Veeco Instruments) at UMR's Materials Research Center. The laboratory has air conditioning but the ambient temperature is not tightly controlled. Additionally, no humidity control is also provided in the laboratory. The AFM is forced to operate in tapping mode.

In our experiments, the imaging frequency is set at 0.004 Hz, which implies that each recursive loop of our system takes about 256 sec. Meanwhile, the samples are imaged at a scan rate of 4 Hz. At each scanning, a 512 by 512 pixel height image representing 1μm^2 area is obtained. This means it takes the AFM about 128 seconds to finish one imaging routine. Thus, almost half of the loop time can be used for algorithm computation, manipulation, fabrication and other tasks on the sample. Typically, the computing time of our algorithm is about 15 seconds on a Pentium M 1.86 GHz computer with 1.00 GB RAM. This means that most of the time can be allocated for manipulation and or fabrication operations.

For the NN-based drift predictor, a fixed time window of past eight drift measurement values were fed into the input layer of the NN, ($N = 8$). Laboratory temperature information is also collected by a thermal sensor attaching to the head of the microscope and fed as an additional NN input. The two-layer NN consists of $N_2 = 50$ neurons in the hidden layer. The initial weights of all layers are selected at random between [0, 1]. The activation functions of the first layer are selected as hyperbolic tangent sigmoid functions and that of the second layer are taken as pure linear functions. Initially, the first 20 sets of drift measurement data will be used for offline training by using Levenberg-Marquardt backpropagation algorithm. After that, along with accumulating new measurement data from the phase correlation algorithm block, online learning is utilized using the training set with the most recent 50 data points.

First, we applied our system on the same sample depicted in Fig. 1. In this experiment, no manipulation work is executed on the sample surface. With the compensator, the AFM is able to focus on the same location along the x direction as shown in Fig. 9.

The second experiment is taken with a sample of Au on mica substrate for the duration of 8 hours in order to test the feasibility of the NN predictor and signal reconstruction block. Fig. 10 displays the errors between the measured and predicted drift values along the *x* direction, which average at 1.62 nm with a peak of 6.88 nm. In Fig. 11, we can see the continuous function of drift after the signal reconstruction process.

In the end, to evaluate the effectiveness of our algorithm under the influence of manipulation, which is one of the major contributions of our paper, the compensator is implemented for an automatic manipulation task. Fig. 12 depicts the results of manipulating gold particles with 30 nm of diameter on a mica substrate. One of the particles is manipulated to form a line with the other two. The measurement and compensation are finished before starting manipulation routines.

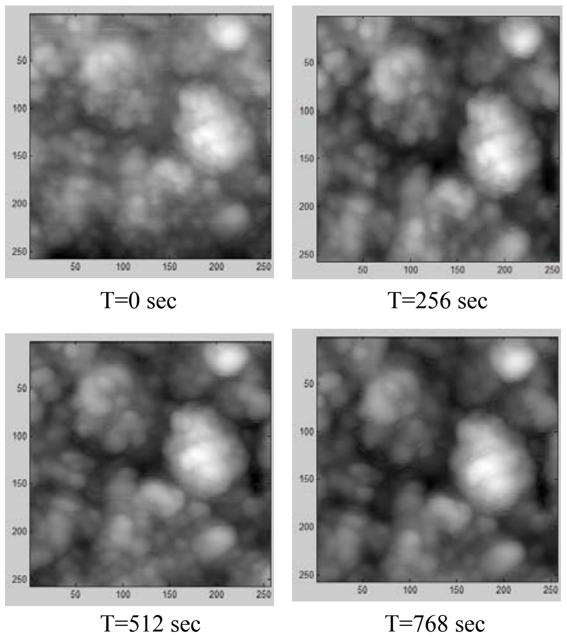


Fig 9. Image Sequences of a Graphite Sample by AFM Tapping Mode Taken at 256 Sec Intervals with Drift Compensation. The Scanned Area is 512×512nm^2. Note: for the Purpose of Comparison, Only Compensation of Drift on *x* Axis is Shown.

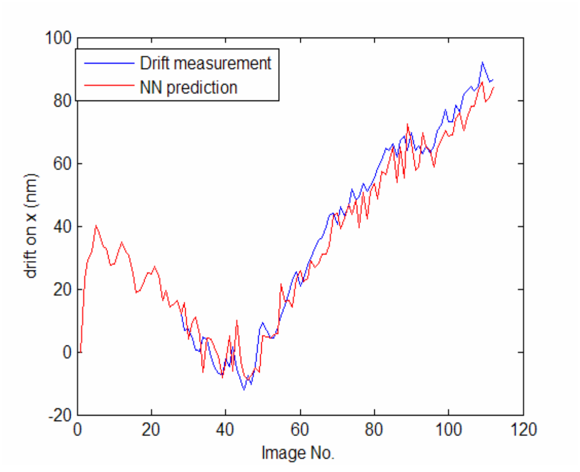


Fig 10. Measured Drift Value from Phase Correlation Algorithm and Predicted Value from NN

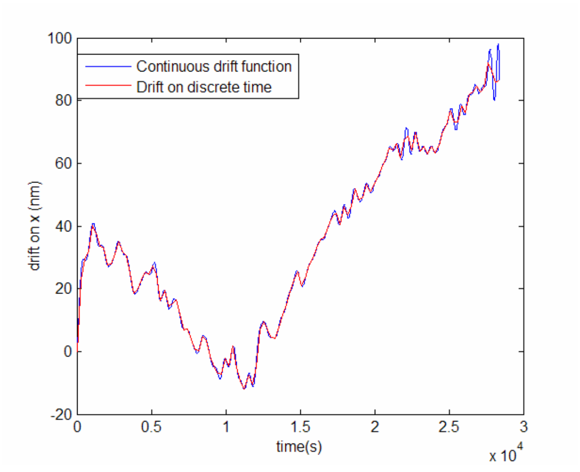
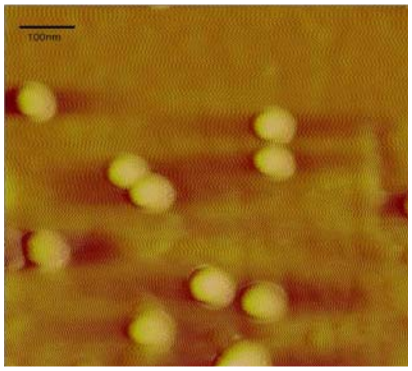
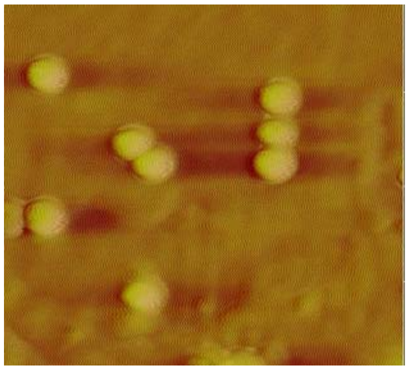


Fig 11. Continuous Drift Function after Signal Reconstruction Compared with the Discrete Drift Measurement



a



b

Fig 12. Manipulation of 30nm Gold Particles Using the Block-based Phase Correlation Compensator

V. CONCLUSIONS AND FUTURE WORK

To realize full automated nanomanipulation and nanofabrication, effects of nonlinearity and spatial uncertainties of AFMs have to be compensated in order to minimize user intervention. This paper describes a novel compensation system for drift, which is a major cause of spatial uncertainty. The compensating scheme can be subsequently used in designing a real-time controller for nanomanipulation. Experimental results show that the proposed scheme is able to predict drift which can be successfully utilized for compensation during nanomanipulation.

As part of future work, similar to the drift compensation, more efficient tools must be developed for other uncertainties, such as creep, hysteresis and so on. Trying other prediction methodologies to lower the tracking errors is also our future work. Moreover, since the microscope undergoes drift as well when capturing images, fundamentally speaking, any images obtained from AFM are drift "contaminated". To eliminate the drift error within an image, possible solutions in the future include: 1) Updating the current image by using the force feedback from the microscope during manipulation; 2) Scanning a smaller local area instead of the whole image for calculating drift; 3) Using multiple tips and conduct manipulation and drift compensation simultaneously in a parallel way.

VI. ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments.

VII. REFERENCES

1. D. M. Eigler and E. K. Schweitzer, "Positioning single atoms with a Scanning Electron Microscope," Nature, pp. 524 – 526, 1990.
2. M. Sitti, "Survey of nanomanipulation systems," Proc. 1st IEEE Conf. Nanotechnology, pp. 75–80, 2001.
3. B. Mokaberi and A. A. G. Requicha, "Towards automatic nanomanipulation: drift compensation in scanning probe microscopes," IEEE Int. Conf. on Robotics and Automation, 2004.
4. V. Y. Yurov and A. N. Klimov, "Scanning tunneling microscope calibration and reconstruction of real image: Drift and slope elimination," Rev. Sci. Inst., vol. 65, no. 5, pp. 1551-1557, 1994.

5. R. Staub, D. Alliata and C. Nicolini, “Drift elimination in the calibration of scanning probe microscopes,” Rev. Sci. Inst. vol. 66, no. 3, pp. 2513-2516, 1995.

6. J. T. Woodward and D. K. Schwartz, “Removing drift from scanning probe microscope images of periodic samples,” J. Vac. Sci. Technol. B, vol. 16, no. 1, pp. 51-53, 1998.

7. S. H. Huerth and H. D. Hallen, “Quantitative method of image analysis when drift is present in a scanning probe microscope,” J. Vac. Sci Technol. B, vol. 21, no. 2, pp. 714-718, 2003.

8. K. J. Ito, Y. Uehara, S. Ushioda and K. Ito, “Servomechanism for locking scanning tunneling microscope tip over surface nanostructures,” Rev. of Sci. Inst., vol. 71, no. 2, pp. 420-423, 2000.

9. C. Stiller, J. Konrad, “Estimating Motion in Image Sequences,” IEEE Signal Processing Magazine, pp.70-91, 1999.

10. Guangyong Li, Ning Xi, Heping Chen, C. Pomeroy and M. Prokos, “‘Videolized’ atomic force microscopy for interactive nanomanipulation and nanoassembly,” IEEE Trans. on Nanotechnology, vol. 4, Issue 5, pp. 605 – 615, 2005.

11. W. Vogl, B. Ma, and M. Sitti, “Augmented Reality User Interface for an Atomic Force Microscope based Nanorobotic System,” IEEE Trans. on Nanotechnology, vol. 5, no. 4, pp. 397-406, 2006.

12. M. Sitti and H. Hashimoto, “Teleoperated Touch Feedback of Surfaces at the Nanoscale: Modeling and Experiments,” IEEE/ASME Trans. on Mechatronics, vol. 8, no. 2, 2003.

13. A. A. G. Requicha, “Nanorobots, NEMS and Nanoassembly,” Proc. IEEE, vol. 91, No. 11, pp. 1922-1933, 2003.

14. Qinmin Yang and S. Jagannathan, “Atomic force microscope-based nanomanipulation with drift compensation,” Int. J. Nanotechnology, vol. 3, no. 4, pp.527–544, 2006.

15. M. Sitti, and H. Hashimoto, “Controlled pushing of nanoparticles: modeling and experiments,” IEEE-ASME Trans. Mechatronics, vol. 5, no. 2, pp.199–211, 2000.

16. J. E. Griffith and D. A. Grigg, “Dimensional metrology with scanning probe microscopes,” J. Appl. Phys., Vol. 74, pp. R83-R109, 1993.

17. B. Mokaberi and A. A. G. Requicha, “Drift compensation for automatic nanomanipulation with scanning probe microscopes,” IEEE Trans. on Automation Science & Engineering, Vol. 3, No. 3, pp. 199-207, July 2006.

PAPER 3

Atomic Force Microscope-based Nanomanipulation with Drift Compensation

Qinmin Yang and S. Jagannathan

Department of Electrical and Computer Engineering

University of Missouri – Rolla,

Rolla, Missouri, U.S.A 65409

Email: qyy74@umr.edu, sarangap@umr.edu

ABSTRACT

Automating the task of nanomanipulation is extremely important since it is tedious for humans. This paper proposes an atomic force microscope (AFM) based force controller to push nano particles on the substrates. A block phase correlation-based algorithm is embedded into the controller for the compensation of the thermal drift which is considered as the main external uncertainty during nanomanipulation. Then, the interactive forces and dynamics between the tip and the particle, particle and the substrate including the roughness effect of the substrate are modelled and analysed. Further, a neural network (NN) is employed to approximate the unknown nanoparticle and substrate contact dynamics. Using the NN-based adaptive force controller the task of pushing nano particles is demonstrated. Finally, using the Lyapunov-based stability analysis, the uniform ultimate boundedness (UUB) of the closed-loop tracking error, NN weight estimates and force errors are shown.

Keywords

Nanomanipulation, Atomic Force Microscope, neural network, drift compensation

I. INTRODUCTION

Nanomanipulation, which aims at manipulating and handling nanometer size objects and structures with nanometer precision, has become a recent topic of research [9]. Nanomanipulation is also a first and critical step for achieving any complex functional nano devices. Applications of the nanotechnology can be found in several fields like biotechnologies (ADN and protein study) and data storage or material science (nanotube or surface film characterization).

However, for manufacturing nanotechnology products, the challenges in nanomanipulation and handling of particles in nano scale require cross-disciplinary approaches. Typically, assemblies of small nano structures built by nanomanipulation today consist of ten to twenty particles, and may take an experienced user a whole day to construct using Atomic Force Microscope (AFM) as the manipulator. To efficiently accomplish such tasks or even more complex ones, the manipulation process should be more automated and it requires less human intervention.

When in ambient conditions, i.e., at room temperature and humidity, in air or in liquid, and without stringent environmental controls, nanomanipulation encounters multiple external disturbances. Among these uncertainties, the thermal drift is the most important one, which can be observed as a horizontal translation during the manipulation process. Research presented in [14] provides a satisfactory real-time drift compensation algorithm, based on which, some controllers can be designed without considering the influence of the thermal drift.

The research on nanomanipulation is still immature because the physical and chemical phenomenon at this scale has not been well understood. A significant amount of work on modeling interactive forces during manipulation was introduced in [1], [15-16]. Nevertheless, in these works, the substrate was assumed to be ideally flat and the roughness effects were ignored. The surface roughness of the substrate can be one of the major hurdles during the manipulation task. In this paper, a novel mathematical model of the nonlinear particle-substrate contact dynamics incorporating the roughness effects is introduced.

Additionally, some of the experimental samples used in the nanomanipulation can be fragile. Improper applied force could damage these nano objects or even the AFM tip. Thus, designing controllers for the manipulation and handling of nano scale objects poses a much greater challenge in terms of accommodating the nonlinearities and uncertainties in the system. In this paper, a NN based controller is proposed where the unknown part of the system dynamics is approximated by using a one-layer NN with an additional force control loop guaranteeing the applied force to be close to a desired value. The controller also compensates the effect of thermal drift as presented next.

This paper is organized as following: the thermal drift compensation algorithm is firstly presented in Section II. The system interaction forces model and the dynamic model are given in Section III and Section IV, respectively. Next, the NN controller is designed in Section V. Section VI describes how the drift compensator is embedded into the controller. Finally, Section VII shows the simulation results to substantiate our theoretical conclusions.

II. THERMAL DRIFT COMPENSATION

Due to thermal changes in ambient conditions, drift usually appears in successive AFM scans of a sample even when the scanning parameters are not altered. In the *x-y* plane, drift can be observed as a translation between different images, which depends on thermal changes and other unclear factors. From the height data of the sample, it can be observed that drift is present even in the *z* direction. The drift velocities on the *x*-*y* plane are reported to vary from 0.01~0.1 nm/s [17]. So the drift between two images taken at 256 sec interval can be as much as 25.6 nm, which is larger than the diameter of the particles that are normally manipulated. In our experiments, drift in the *z*-direction is about 0.005nm/s [14], which is considered negligible in our pushing task. Due to thermal drift, the nanomanipulation task can fail unless it is properly compensated.

Our first goal is to develop a drift compensation scheme to estimate and compensate for the drift in the *x* and *y* directions under the influence of the noise from the *z* axis so that nanomanipulation can be performed as if drift does not exist. Fortunately, experiments show that the drift in *x* and *y* directions can be seen as a translational movement, not rotation. In addition, there is negligible correlation between the two. The block diagram of the proposed compensation system is depicted in Fig. 1. For simplicity, only the drift in the *x* direction is shown [14].

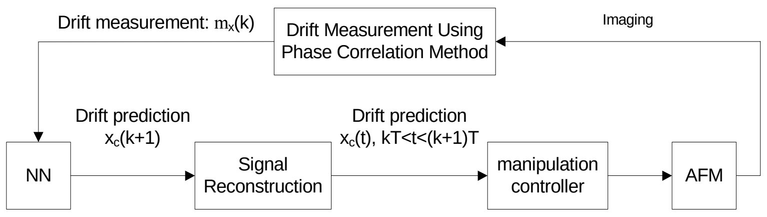

Drift measurement: $m_x(k)$
Drift Measurement Using Phase Correlation Method
Imaging
Drift prediction $x_c(k+1)$
Drift prediction $x_c(t)$, $kT<t<(k+1)T$
NN
Signal Reconstruction
manipulation controller
AFM

Fig 1. Block Diagram of the Proposed Drift Compensator

Due to the working principles of AFM, the topographic data of the sample cannot be collected during the pushing procedure. So that the solution is stated as follows: 1) the sample is scanned at a constant frequency; 2) at each iteration, after obtaining the scanning data, the drift $x_c(t)$ and $y_c(t)$ is estimated and predicted; 3) during the subsequent time interval before the next scanning time, the pushing task can be performed by compensating drift.

In the proposed scheme, drift is measured by using a block phase correlation-based algorithm at each sampling time. Based on current and previous data, drift for the subsequent sampling time instant can be predicted by using a neural network. Further, signal reconstruction technique is applied to obtain the drift in continuous time. For more details, refer to [14]. With the compensator, controller can be designed as if the drift does not exist.

III. INTERACTION FORCES

In our work, the nano particles on the substrate will be manipulated by the AFM tip. The AFM tip apex is assumed to be a spherical ball with radius R_t = 30 nm, and the particle radius is denoted as R_p. β is the pushing angle, which is the angle between the pushing direction and the horizontal plane. Interactive forces among the AFM tip, particle, and substrate after the tip contacts the particle can be seen in Fig. 2. A_{ps} is the adhesion forces between particle and substrate. F_{ps} and F_{tp} denote the particle-substrate and tip-particle attractive/repulsive interaction force, while f_{ps} and f_{tp} correspond to the frictional forces for the particle-substrate and tip-particle, respectively. Elastic deformation of the particle is possible and here only the elastic deformation between the particle and the substrate is considered. The indentation is denoted as d.

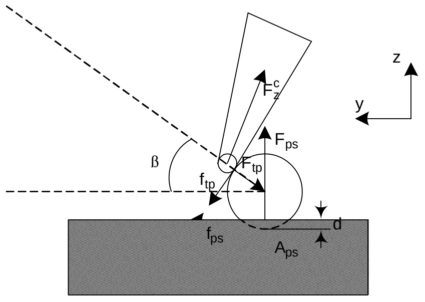


Fig 2. The Interacting Forces between AFM Tip, Nanoparticle and Substrate

Gravitational forces are relatively very small in the nano scale and, therefore, are neglected. The main components of the adhesion forces are van der Waals, capillary, and electrostatic forces [1]. Therefore, the adhesion force between particle and substrate is given by $A_{ps} = A_{ps}^{vdw} + A_{ps}^{cap} + A_{ps}^{es}$. The analysis of these contact forces and frictional forces is very important for modelling the nano manipulation process.

A. Van der Waals Forces

Van der Waals force is force acting between atoms, which is caused by a momentary dipole moment between atoms resulting from interaction between electrons in the outermost bands rotating around the nucleus. An overview is given in [2]. Depending on the object geometry and the material type, van der Waals force between atoms or molecules is proportional to the inverse of the sixth power of distance between the molecules [3]. The van der Waals force between the particle and substrate can be expressed as

$$A_{ps}^{vdw} = \frac{2HR_p^3}{3h^2(h+2R_p)^2} \tag{3.1}$$

where H is Hamaker constant, and h is the particle-substrate distance.

Since we are interested on the nano manipulation task carried out in an ambient environment, there will be always a liquid layer on the surface of the sample. Therefore $H = \sqrt{(H_{tip} - H_{liquid})(H_{particle} - H_{liquid})}$ [4]. After taking the surface roughness into consideration, van der Waals force becomes [5]

$$A_{ps}^{vdw} = \frac{2HR_p^3}{3h^2(h+2R_p)^2}\left(\frac{h}{h+b/2}\right)^2 \tag{3.2}$$

where b is the peak to peak height of the surface irregularity.

B. Capillary Forces

As stated above, in ambient operation environment, due to the presence of the water layer on the surfaces of particle and substrate, a liquid bridge is created between them at close contact as shown in Fig. 3.

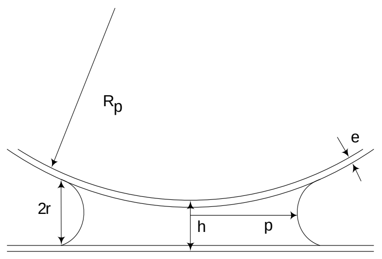


Fig 3. Capillary Force Parameters during a Sphere and Flat Surface Contact

In early work [6], molecular dynamic simulations have shown that the macroscopic theory of capillarity should hold down to radius of curvature of the order of some molecular size. By assuming that (i) $r << p << R_p$, (ii) the surfaces are coated with a film of constant thickness e, (iii) the contact angle is zero, which should be the true in our case, and (iv) the surfaces attraction through the liquid phase is negligible, the capillary force can be written as [7]

$$A_{ps}^{cap} = 4\pi\gamma R_p(1 - \frac{h-2e}{2r}) \tag{3.3}$$

where γ is the liquid (water) surface energy, e is the thickness of the water layer, and r is the radius of curvature of the meniscus as shown in Fig. 3. Moreover, the volume of liquid condensed in the bridge and the film thickness distribution can also influence the capillary force, but as stated in [7], these variations can be ignored in our case.

C. Electrostatic Forces

In the case of non-conducting particles, there are charges trapped around the perimeter of the particles, and during pushing or contact, triboelectrification process introduces local charges. For general cases, a model for the electrostatic forces is desirable. However, by grounding a (semi) conducting substrate such as Si, Au, or HOPG, the electrostatic forces can be greatly reduced [1]. Moreover, the nonconducting particles can be coated with Au, and all the substrate and particles can be grounded. It was reported that the electrostatic forces A_{ps}^{es} is less than one percent of the capillary force and therefore could become ignored [8].

D. Frictional Forces

During pushing, the friction on the particle-substrate and the particle-tip plays an important role. Similar to the case of macro domain, when the particle is sliding smoothly on the substrate, the frictional force at the micro/nano scale can be given as

$$f_{ps} = \mu_{ps} F_{ps} \tag{3.4}$$

where μ_{ps} is the particle–substrate sliding friction coefficient. Also, frictional force exists between the tip and particle that needs to be taken into account.

IV. DYNAMIC MODEL

When manipulating objects in the nano domain, the micro-physics of the problem must be taken into account [7-8]. Modeling is necessary for pushing nano-spheres laying on a planar substrate. A dynamic model of the pushing system is formulated considering all of the forces mentioned above. The objects considered in this work include nano-particles of diameter 30 to 500 nm, which is required to be pushed from point A to B, as shown in Fig. 4. The angle between y axis and the pushing direction is denoted as γ.

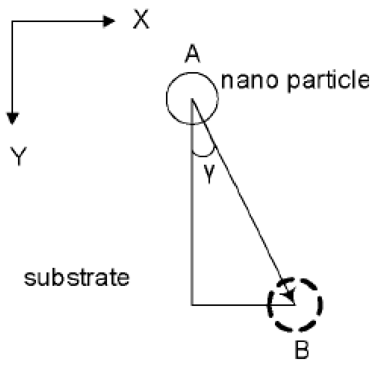


Fig 4. Coordinate Frames of the 2-D AFM Image Graphics Display during the Particle Pushing

A. Elastic Deformation of the Particle

Since the contact area between the particle and AFM tip is very small, only the vertical deformation between the particle and substrate will be considered [1]. In the contact area, only the elastic deformation is to be modeled. In the nano scale, the elastic deformation of the surface caused by the adhesive forces is large compared to their effective range of action. Therefore, the JKR (Johnson-Kendall-Roberts) model analysis will be valid [10]. The indentation of the particle is given as

$$\begin{aligned} a^3 &= R_p \cdot (F_{ps} + 3\pi R_p w + \sqrt{6\pi R_p w F_{ps} + (3\pi R_p w)^2}) / K \\ d &= a^2 / R_p - \frac{2}{3}\sqrt{3\pi w a / K} \end{aligned} \quad (4.1)$$

where a is the contact radius and d is the indentation [11].

B. Interacting Force Analysis

The interacting forces occurring between the tip, particle, and substrate are shown in Fig. 2. The deflection forces applied on the cantilever along the *x*, *y*, and *z* axes are denoted as F_x^c, F_y^c, F_z^c. Since the tip is very small compared to the diameter of the particle, the AFM cantilever can be seen as a point object at the apex of the tip. A point mass model of the interaction forces during pushing is derived in [1] where k_x, k_y, k_z and b_x, b_y, b_z represents the elastic and damping coefficients of the cantilever along the *x*, *y*, and *z* axes, respectively. The term $m_c = k_z \big/ \left(4\pi^2 f_r\right)$ is the cantilever effective mass, while f_r is the cantilever resonant frequency. Let us assume initially that the surface of the substrate is smooth in order to arrive at the dynamics. Later this assumption is relaxed.

The particle will be acted upon static friction and kinetic friction during the pushing operation [1]. At the beginning, the tip is approaching the particle under the stage movement. After the tip contacts the particle, due to the static friction force between the particle and the substrate, the particle and tip will be together and follow the stage motion until the applied cantilever load exceeds the static friction. In this phase, assuming the stage is moving slowly, the following equations can be derived

$$\begin{aligned}
F_x^c &= k_x x \cos\alpha = k_x x_s \cos\alpha \\
F_y^c &= k_y y \cos\alpha = k_y y_s \cos\alpha \\
F_z^c &= k_z \sqrt{x^2+y^2}\sin\alpha = k_z \sqrt{x_s^2+y_s^2}\sin\alpha \\
F_{ps} &= A_{ps} + F_x^c \,//\, F_y^c \cdot \sin\alpha + F_z^c \cos\alpha \\
f_{ps} &= F_x^c \,//\, F_y^c \cdot \cos\alpha - F_z^c \sin\alpha
\end{aligned} \tag{4.2}$$

After the particle is detached from the substrate, assume the particle is pushed in a constant speed, *V*, and is purely sliding. If the speed is very slow, we can obtain equilibrium equations as

$$\begin{aligned}
F_{tp}\sin\beta + A_{ps} + f_{tp}\cos\beta &= F_{ps} \\
F_{tp}\cos\beta &= f_{tp}\sin\beta + f_{ps} \\
f_{ps} &= \mu_{ps} F_{ps} \\
f_{tp}\cdot R_p &= f_{ps}\cdot (R_p - d)
\end{aligned} \tag{4.3}$$

and further

$$\begin{aligned}
F_{ps} &= \frac{A_{ps}\cos\beta}{\cos\beta - (1 - d/R_p + \sin\beta)\mu_{ps}} \\
F_{tp} &= \frac{(1-d/R_p)\sin\beta + 1}{\cos\beta}\mu_{ps} F_{ps}
\end{aligned} \tag{4.4}$$

For building cantilever dynamics, denoting the deflections of the probe along the *x*, *y* and *z* axes as ζ_x, ζ_y and ζ_z. In AFM-based manipulation systems, only ζ_z can be measured, which satisfies

$$\zeta_z = F_z^c / k_z + F_y^c / k_{yz} \approx F_z^c / k_z \tag{4.5}$$

$$F_z^c = f_{tp}\cos\phi + F_{tp}\sin\phi \tag{4.6}$$

where $\phi = \beta - \alpha$, and α is the cantilever tilt angle from the base guaranteeing the point contact of the particle with the substrate. Equation (4.5) shows that ζ_z can be seen as a direct measure of the force F_z^c.

Additionally, for positioning in atomic domain, piezoelectric actuators are utilized in AFM systems. By denoting the sample position along *x*, *y*, and *z* directions as x_s, y_s, z_s, respectively, the dynamics of the stage along each axis are given by [1]

$$\begin{aligned}
&\frac{1}{w_x^2}\ddot{x}_s + \frac{1}{w_x Q_x}\dot{x}_s + x_s = \tau_x - F_x = \tau_x - f_{ps}\cos\gamma \\
&\frac{1}{w_y^2}\ddot{y}_s + \frac{1}{w_y Q_y}\dot{y}_s + y_s = \tau_y - F_y = \tau_y - f_{ps}\sin\gamma \\
&\frac{1}{w_z^2}\ddot{z}_s + \frac{1}{w_z Q_z}\dot{z}_s + z_s = \tau_z - F_{ps} + A_{ps}
\end{aligned} \tag{4.7}$$

where w is the resonant frequency, Q is the amplification factor and τ is the stage driving forces.

C. Substrate Roughness

During pushing, the surface cannot be smooth, especially at the nano scale. Further, the movement of the base in the z-direction and alignment errors will make the above assumption unreasonable.

Hence, the displacement of the particle on vertical direction can be seen as a disturbance on the pushing angle β which is given by

$$\begin{aligned}
\beta &= \sin^{-1}\frac{h_{set} - z_s - z_{sub} + d + \zeta_z\cos\alpha}{R_p} \\
&\approx \sin^{-1}\frac{h_{set} - z_s - z_{sub} + d}{R_p} = \beta(z_s)
\end{aligned} \tag{4.8}$$

where h_{set} is the predetermined parking height of the tip, and z_{sub} is the sample surface height displacement, or the roughness of the surface, which is reasonably assumed to be bounded. Therefore, the system dynamics can be expressed as

$$\begin{aligned}
&\frac{1}{w_x^2}\ddot{x}_s + \frac{1}{w_x Q_x}\dot{x}_s + x_s + \cos\gamma f_{ps}(z_s, z_{sub}) = \tau_x \\
&\frac{1}{w_y^2}\ddot{y}_s + \frac{1}{w_y Q_y}\dot{y}_s + y_s + \sin\gamma f_{ps}(z_s, z_{sub}) = \tau_y \\
&\frac{1}{w_z^2}\ddot{z}_s + \frac{1}{w_z Q_z}\dot{z}_s + z_s + F_{ps}(z_s, z_{sub}) = \tau_z + A_{ps}
\end{aligned} \tag{4.9}$$

Two pushing strategies are introduced in [1]. Because the pushing operation can be affected greatly by the substrate surface topographic changes, in our work, we will design a controller for the constant contact force control algorithm. That is, on the basis of the system dynamics in (4.8), the controller will be designed such that it will change the horizontal position $(x_s,\ y_s)$ of the stage from A to B, while keeping $F_z^c = F_z^c(z_s, z_{sub})$ at a desired value. This will ensure that the tip contacts the particle with almost the same height away from the substrate during the pushing, so that the chance for the tip to lose contact with the particle is minimized. This requirement also guarantees that a proper force will be applied on the sample without damaging it.

V. CONTROLLER DESIGN

Our goal is to design a control input that guarantees a desired stage motion and applied force on the cantilever. In this section, let us first assume that the drift does not exist. As can be seen from (4.5), F_z^c lateral force has a direct effect on ζ_z, so that the control goal can be translated as keeping a desired constant ζ_z during pushing. Hence, the system state is defined as $s = \begin{bmatrix} x_s & y_s & z_s \end{bmatrix}^T$. Given a desired trajectory $s_d = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T$ for the stage, the filtered tracking error can be defined as

$$r = \dot{e} + \Delta e \tag{5.1}$$

where $\Delta \in \mathrm{R}^{3\times 3}$ is a designed diagonal matrix selected through pole placement with positive entries. $e = s - s_d \in \mathrm{R}^3$, $\dot{e} = \dot{s} - \dot{s}_d$ represent the trajectory error and the velocity error, respectively. x_d, y_d can be readily derived from Fig. 4. Moreover, since we can obtain the topographic information of the substrate by using the image mode of AFM, and the resolution of AFM could reach as low as 1nm, a good estimation of z_{sub} can be done in advance. Based on the estimate value of $\hat{z}_{sub}$, the desired trajectory of the stage on *z*-axis can be defined such that $\hat{z}_{sub} + z_d = 0$, which is necessary for the cantilever to maintain contact with the particle. This selection will ensure that when the filtered tracking error converges to zero, the trajectory error $e(t)$ eventually converges to zero, too. It can be also easily found that when the controller guarantees that the filtered tracking error $r(t)$ is bounded, $e(t)$ is also bounded.

In the presence of bounded disturbances (e.g. the estimation error of the surface height $z_e = z_{sub} - \hat{z}_{sub}$) and modeling uncertainties, the system dynamics can be expressed from (4.9) in matrix form as

$$\Omega^{-2}\ddot{s} + \Omega^{-1}Q^{-1}\dot{s} + f(s) + d = \tau \tag{5.2}$$

where the matrices in this equation are defined as

$$\Omega = \begin{bmatrix} w_x & 0 & 0 \\ 0 & w_y & 0 \\ 0 & 0 & w_z \end{bmatrix} \tag{5.3a}$$

$$Q = \begin{bmatrix} Q_x & 0 & 0 \\ 0 & Q_y & 0 \\ 0 & 0 & Q_z \end{bmatrix} \tag{5.3b}$$

$$f(s) = \begin{bmatrix} x_s + \cos\gamma f_{ps}(z_s) \\ y_s + \sin\gamma f_{ps}(z_s) \\ z_s + F_{ps}(z_s) - A_{ps} \end{bmatrix} \tag{5.3c}$$

$\tau = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^T$ is the control input vector and d stands for the disturbance vector. It can be reasonably assumed that d is bounded by $d \le d_N$. Thus, differentiating (5.1) yields

$$\dot{r} = \ddot{s} - \ddot{s}_d + \Delta\dot{e} \tag{5.4}$$

Substituting (5.2) into (5.4) yields the filtered tracking error system

$$\begin{aligned} \dot{r} &= \Omega^2\tau - \Omega Q^{-1}\dot{s} - \Omega^2 f(s) - \Omega^2 d - \ddot{s}_d + \Delta\dot{e} \\ &= \Omega^2\tau - \Omega^2 f(s) - \Omega^2 d - g(s) \end{aligned} \tag{5.5}$$

where $g(s) = \Omega Q^{-1}\dot{s} + \ddot{s}_d - \Delta\dot{e}$.

Given a smooth trajectory and when the parameter matrices Ω and Q are accurately known, the control input can be selected as

$$\tau = f(s) + \Omega^{-2} g(s) - K_v r + K_f \zeta_e \tag{5.6}$$

with $f(s)$ known accurately and $K_v \in \mathrm{R}^{3\times3}$ is the diagonal gain matrix. The outer loop $K_f \zeta_e$ is a proportional loop with $K_f \in \mathrm{R}^3$ being the gain matrix for the force controller loop and $\zeta_e = \zeta_z - \zeta_d$ represents the deflection error between the actual to a desired value in deflection. In the absence of disturbances and model uncertainties, applying (5.6) in (5.5) could yield an asymptotically stable filtered tracking error system. However, the dynamic model for the manipulation of a nanoparticle is quite nonlinear and unknown. For example, Hamaker constant, the thickness of the water layer, the surface roughness and so on are typically unknown. In other words, $f(s)$ in (5.6) is not known beforehand. Consequently, a novel learning controller scheme is necessary for this task.

In this paper, a one-layer neural network (NN) is implemented to approximate the unknown system dynamics, or $f(s)$. Further, by using Lyapunov-based stability analysis, appropriate NN weight updating scheme can be derived to guarantee the stability of the closed-loop system.

Select the control input as

$$\tau = \hat{f}(s) + \Omega^{-2} g(s) - K_v r + K_f \zeta_e \tag{5.7}$$

the closed-loop filtered tracking error dynamics (5.6) become

$$\begin{aligned} \Omega^{-2}\dot{r} &= \hat{f}(s) - f(s) - K_v r + K_f \zeta_e - d \\ &= -\tilde{f}(s) - K_v r + K_f \zeta_e - d \end{aligned} \tag{5.8}$$

where $\hat{f}(s) \in \mathrm{R}^3$ is the approximated value of $f(s) \in \mathrm{R}^3$, and $\tilde{f}(s) = f(s) - \hat{f}(s) \in \mathrm{R}^3$ is the approximation error. From (5.8), it is clear that the closed-loop filtered tracking error system is driven by the functional approximation error. The stability of the system should be shown in the presence of this error.

According to [12], a single layer NN can be used to approximate any nonlinear continuous function over the compact set when the input layer weights are selected at random and held constant whereas the output layer weights are only tuned provided sufficiently large number of nodes in the hidden-layer is chosen. Therefore, a single-layer NN is used here. Assume that there exist target weights $W \in \mathrm{R}^{n\times 3}$ such that the nonlinear dynamics can be written as

$$f(s) = W^T \phi(V^T s) + \varepsilon \tag{5.9}$$

where $V \in \mathrm{R}^{3\times n}$ is the input layer weight which will not be tuned, n is the number of the hidden layer nodes, and $\phi(\cdot)$ is the activation function vector. Let the approximation error $\varepsilon \in \mathrm{R}^3$ satisfies $\|\varepsilon\| \le \varepsilon_N$ with the bound ε_N known. For simplicity, the output of the NN is expressed as $f(s) = W^T\phi(s) + \varepsilon$, and the NN output is defined by

$$\hat{f}(s) = \hat{W}^T\phi(s) \tag{5.10}$$

where $\hat{W} \in \mathrm{R}^{n\times 3}$ is a matrix of actual weights. Then the next step is to determine the weight updates so that the performance of the closed-loop filtered tracking error dynamics of the manipulation system is guaranteed.

Let W be a matrix of unknown target weights required for the approximation and assume they are bounded by known values such that

$$\|W\| \le W_{\max} \tag{5.11}$$

The error in weights during estimation is defined as

$$\tilde{W} = W - \hat{W} \tag{5.12}$$

Therefore, the control input is selected as

$$\tau = \hat{W}^T\phi(s) + \Omega^{-2}g(s) - K_v r + K_f\zeta_e \tag{5.13}$$

Substituting (5.9), (5.10), (5.13) into (5.8) yields

$$\begin{aligned}\Omega^{-2}\dot{r} &= \hat{W}^T\phi(s) - W^T\phi(s) - \varepsilon - K_v r + K_f\zeta_e \\ &= -\tilde{W}^T\phi(s) - \varepsilon - d - K_v r + K_f\zeta_e\end{aligned} \tag{5.14}$$

The structure of the proposed NN controller is depicted as Fig. 5. An inner action-generating NN loop eliminates the nonlinear dynamics of the manipulation process and contact dynamics. The outer PD tracking loop designed via Lyapunov analysis guarantees the stability of the closed-loop system in tracking a desired trajectory for pushing the nano particle. The proportional force controller loop ensures that the cantilever will apply a desired force on the particle and will not lose contact with it. The embedded drift compensation scheme will ensure as if drift does not exist.

The next step will be to determine an appropriate weight updating algorithm for the NN so that the closed-loop stability of the grasping controller can be demonstrated.

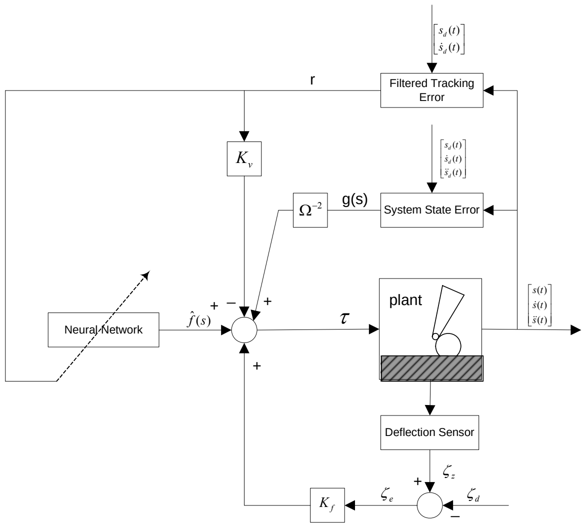

$\begin{bmatrix} s_d(t) \\ \dot{s}_d(t) \end{bmatrix}$
r
Filtered Tracking Error
K_v
$\begin{bmatrix} s_d(t) \\ \dot{s}_d(t) \\ \ddot{s}_d(t) \end{bmatrix}$
Ω^{-2}
g(s)
System State Error
Neural Network
$\hat{f}(s)$
τ
plant
$\begin{bmatrix} s(t) \\ \dot{s}(t) \\ \ddot{s}(t) \end{bmatrix}$
Deflection Sensor
ζ_z
ζ_e
ζ_d
K_f

Fig 5. NN Controller Architecture

Theorem: Assume that the desired trajectory for the stage, the unknown disturbances, and the approximation errors are bounded, respectively, by the known constants s_N, d_N, ε_N. Select the NN weight tuning update as

$$\dot{\hat{W}} = -F\phi(s)r^T \tag{5.15}$$

where $F \in \mathrm{R}^{n \times n}$ is a diagonal constant learning rate matrix with positive entries. Then the tracking error $r(t)$ and the weight estimation errors $\tilde{W}$ are UUB. Further, the force error is also UUB.

Proof: First, not taking into consideration the force control loop, the closed-loop system is expressed as

$$\Omega^{-2}\dot{r} = -\tilde{W}^T\phi(s) - \varepsilon - d - K_v r \tag{5.16}$$

Select the Lyapunov function candidate $V \in \mathrm{R}$ as

$$V = \frac{1}{2}\Omega^{-2}r^T r + \frac{1}{2}tr\left\{\tilde{W}F^{-1}\tilde{W}\right\} \tag{5.17}$$

and we evaluate the first derivative of V along the system trajectories to get

$$\dot{V} = \Omega^{-2}r^T\dot{r} + tr\left\{\tilde{W}F^{-1}\dot{\tilde{W}}\right\} \tag{5.18}$$

Substituting (5.12), (5.15), and (5.16) into (5.18) yields

$$\begin{aligned} \dot{V} &= r^T(-\tilde{W}^T\phi(s) - \varepsilon - d - K_v r) + tr\left\{\tilde{W}F^{-1}F\phi(s)r^T\right\} \\ &= -r^T K_v r - r^T(\varepsilon + d) \\ &\le -K_{v\min}\|r\|^2 + (\varepsilon_N + d_N)\|r\| \end{aligned} \tag{5.19}$$

with $K_{v\min}$ the minimum singular value of K_v. Since ε_N is constant, $\dot{V} \le 0$ as long as

$$\|r\| > (\varepsilon_N + d_N)/K_{v\min} \tag{5.20}$$

In other words, $\dot{V}$ is negative outside a compact set. According to a standard Lyapunov theorem [13], it can be concluded that the tracking error $r(t)$ and the NN weights estimates error $\tilde{W}$ are UUB. Furthermore, the tracking error bound can be made as small as desired by increasing the smallest eigenvalue $K_{v\min}$.

To show the bound on the force tracking error or the deflection tracking error ζ_e, we use an approach that can be compared to Barbalat's extension [13]. Thus, note first that in part of the proof we have shown that all quantities on the right-hand side of (5.16) are bounded. Therefore, from the invertibility of Ω, it follows that $\dot{r}$ is bounded. Hence, the tracking error dynamics are expressed as

$$\Omega^{-2}\dot{r} = -\tilde{W}^T\phi(s) - \varepsilon - K_v r + K_f\zeta_e \tag{5.21}$$

or

$$\begin{aligned} K_f\zeta_e &= \Omega^{-2}\dot{r} + \tilde{W}^T\phi(s) + \varepsilon + K_v r \\ &\equiv B(r,\dot{r},s,\tilde{W},\varepsilon) \end{aligned} \tag{5.22}$$

where all quantities at the right-hand side are bounded. Therefore, we obtain

$$\zeta_e = K_f^{-1}B(r,\dot{r},s,\tilde{W},\varepsilon) \tag{5.23}$$

which shows that the force tracking error ζ_e is bounded. Moreover, it can be found that the force tracking error bound can be made as small as desired by increasing the force tracking error gain K_f.

VI. DRIFT COMPENSATION

The previous section describes the surface roughness effects and dynamic model development for pushing operation. In order to accommodate the effects of drift, the system dynamics will be given by

$$\Omega^{-2}\ddot{s}_r + \Omega^{-1}Q^{-1}\dot{s}_r + f(s_r) + d = \tau \tag{6.1}$$

where $s_r = s + s_c$ is the real position of the particle on the stage coordinates, and $s_c = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ is the drift value at the current time instant. The amount of drift in the *x-y* plane, x_c, y_c, can be estimated satisfactorily by using our drift compensator scheme from [14], while z_c is negligible. Fortunately, $f(s_r)$ is just the function of z_r, thus the compensator results can be easily combined into the controller by adjusting the desired system trajectories as $s_d = \begin{bmatrix} x_d + x_c & y_d + y_c & z_d \end{bmatrix}^T$. In other words, the relative position of the particle with respect to the stage can be obtained and thus be controlled to track the desired trajectory.

VII. SIMULATION RESULTS

In this section, the NN controller is simulated and the control objective is to guide the stage movement to follow a desired trajectory with desired force applied on the cantilever.

As shown in Fig. 4, the task is to push the particle from point A to B with a constant speed. Thus, the desired trajectory of the stage along x and y axes will be ramps with slope as the desired speed. In our simulation, $R_p = 30nm$, $\gamma = 30^\circ$, and the desired speed $v = 1000nm/s$. Moreover, the desired reflection $\zeta_z = 12.77nm$, which is selected by experiments [1]. To test the robust of our method under the roughness effects, the substrate surface is set as a sinusoid function with the amplitude of 1nm.

First, assume that the drift does not exist. Simulation results are shown in Figs 6 through 8 by using PD and NN controllers for comparison purposes. The dashed lines are denoted as the desired trajectories and the solid ones are actual system outputs. Simulation results depicted in Fig. 8 demonstrate that the NN controller can approximate the unknown system dynamics and the tracking errors converge in less than 0.4seconds even with the presence of the roughness effect. Further, because of the outer force control loop, the force applied on the cantilever also converges to the desired value quickly and is not disturbed much by the surface roughness effects of the sample. By contrast, although the traditional PD controller can obtain a satisfactory performance without surface roughness, it fails to achieve acceptable results as shown in Fig. 7 due to the unexpected surface roughness effect. In all these results, it is assumed that drift does not exist.

To demonstrate the importance of the proposed drift, the NN controller performance is depicted in Fig. 9 without the drift compensation. Here drift is assumed to be constant at 0.1nm/s. From the result, the performance of the NN controller is greatly deteriorated. The position of the particle is moving away. If this continues with time, the tip will lose contact with the particle unless a suitable compensation is added.

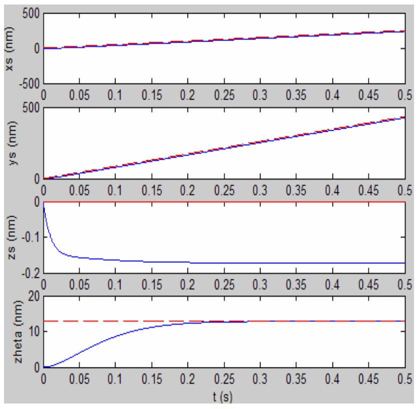


Fig 6. Performance of the PD Controller without Surface Roughness Effects

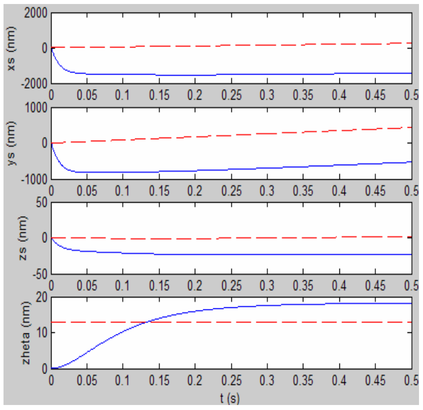


Fig 7. Performance of the PD Controller in the Presence of Surface Roughness Effects of the Substrate

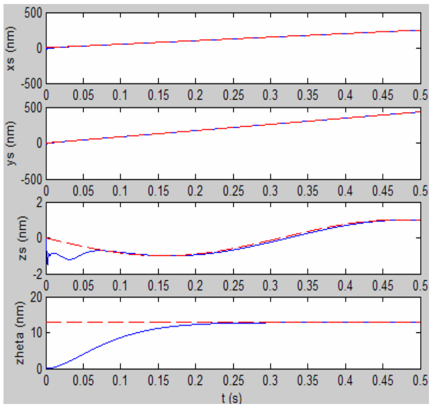


Fig 8. Performance of the Proposed NN Controller with Surface Roughness Effects

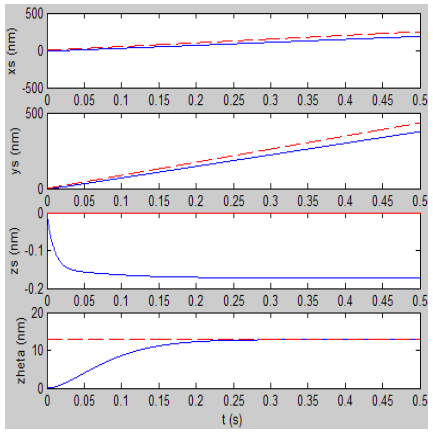


Fig 9. Performance of the Proposed NN Controller without Drift Compensator

VIII. CONCLUSIONS

The task of manipulating nano objects is complex and requires a sophisticated controller to compensate for the nonlinear cantilever and contact dynamics. In this paper, a novel controller scheme was presented for guiding the stage so that the position of the nano particle follows a predefined trajectory. The controller includes an embedded drift compensator, a NN to approximate the unknown dynamics, and an additional force feedback loop. The tuning of the NN weights was performed online and the controller offers guaranteed tracking performance under the influence of the surface roughness effect. The pushing task was accomplished when the tip makes contact with the object and the stage is driven to a desired position when a suitable force is applied. The simulation result demonstrates that the proposed controller was able to perform the pushing task successfully in terms of tracking and force errors.

IX. REFERENCES

1. Sitti, M. and Hashimoto, H., (2000) ‘Controlled Pushing of Nanoparticles: Modeling and Experiments,’ IEEE/ASME Trans. on Mechatronics, Vol. 5, Issue 2, pp. 199-211.
2. Israelachvili, J.N, (1974) ‘The Nature of van der Waals Forces,’ Contemporary Physics, Vol. 15, No. 2, pp. 159-177.
3. Hamaker, H.C., (1937) ‘The London-van der Waals Attraction Between Spherical Particles,’ Physica IV, No. 10, pp. 1058-1072.
4. Israelachvili, J., (1992) ‘Intermolecular and Surface Forces,’ 2nd ed. London, U.K.: Academic.
5. Saito, S., Miyazaki, H., and Sato, T., (1999) ‘Pick and place operation of a micro-object with high reliability and precision based on micro-physics under SEM,’ IEEE International Conference on Robotics and Automation, Vol. 4, pp. 2736 – 2743.
6. Thompson, P.A. and Robbins, M.O., (1989) ‘Simulations of contact-line motion:slip and the dynamic contact angle,’ Phys. Rev. Lett. 63, 766.
7. Crassous, J., Charlaix, E., Gayvallet, H., and Loubert, J., (1993) ‘Experimental study of a nanometric liquid bridge with a surface force apparatus,’ Langmuir, Vol. 9, No. 8, pp. 1995–1998.

8. Arai, F., Ando, D., and Fukuda, T., (1995) ‘Micro manipulation based on micro physics: Strategy based on attractive force reduction and stress measurement,’ Proc. IEEE Int. Conf. Robotics and Automation, Vol. 2, pp. 236–241.

9. Sitti, M., (2001) ‘Survey of nanomanipulation systems,’ IEEE Proceedings of the Nanotechnology, pp. 75 – 80.

10. Johnson, K. L., and Greenwood, J. A., (1997) ‘An adhesion map for the contact of elastic spheres,’ J. Colloid. Interface Sci. Vol. 192, pp. 326-333.

11. Sarid, D., Hunt, J. P., and Workman, R. K. et al., (1998) ‘The role of adhesion in tapping-mode atomic force microscopy,’ Appl. Phys. A, Vol. 66, pp. 283–286.

12. Igelnik, B., and Pao, Y. H., (1995) ‘Stochastic choice of basis functions in adaptive function approximation and the functional-link net,’ IEEE Trans. Neural Networks, Vol. 6, pp. 1320-1329.

13. Lewis, F. L., Jagannathan, S., and Yesildirek, A., (1999) ‘Neural Network Control of Robot Manipulators and Nonlinear Systems,’ London, U.K.: Taylor & Francis.

14. Yang, Qinmin, Jagannathan, S. and Bohannan, E. W., (2005) ‘Block Phase Correlation-based Automatic Drift Compensation for Atomic Force Microscopes,’ Proc. of 5th IEEE Conf. on Nanotechnology, Vol. 1, pp. 370-373.

15. Li, Guangyong, Xi, Ning, Yu, Mengmeng, Fung, Wai Keung, (2003) ‘Modeling of 3-D interactive forces in nanomanipulation,’ Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Vol. 3, pp. 27-31.

16. Li, G. Y., Xi, N., Yu, M., and Fung, W. K., (2003) ‘Augmented reality system for real-time nanomanipulation,’ Proc. of 4th IEEE Int. Conf. Nanotechnology, Vol. 2, pp. 64-67.

17. Mokaberi, B. and Requicha, A. A. G. (2004) ‘Towards automatic nanomanipulation: drift compensation in scanning probe microscopes,’ IEEE Int. Conf. on Robotics and Automation, Vol. 1, pp. 416-421.

PAPER 4

Online Reinforcement Learning Neural Network Controller Design with Drift Compensation for Nanomanipulation

Qinmin Yang and S. Jagannathan

Department of Electrical and Computer Engineering

University of Missouri – Rolla,

Rolla, Missouri, U.S.A 65409

Email: qyy74@umr.edu, sarangap@umr.edu

ABSTRACT

Nanomanipulation implies manipulating objects in nanometer size with nanometer precision. Typically, it takes operators several hours to perform a simple task which is the major hurdle for manufacturing these devices. Automating the task of nanomanipulation is the prerequisite for the manufacturing of nano devices in the future. To accomplish the task automatically and quickly, the proposed novel scheme consists of a block-based phase correlation scheme to mitigate thermal drift, and a novel reinforcement learning neural network (NN)-based controller, referred to adaptive critic controller for nanomanipulation. In the online NN reinforcement learning controller design, one NN is designated as the critic NN, which approximates the long-term cost function. Meanwhile, an action NN is employed to derive an optimal control signal to track a desired system trajectory for the stage while minimizing the cost function. Online updating weight tuning schemes for these two NNs are also derived. Furthermore, by using the standard Lyapunov approach, the uniformly ultimate boundedness (UUB) of the tracking error and weight estimates is shown. The proposed scheme is evaluated and verified in the simulation environment.

Keywords

Nanomanipulation, neural networks, drift compensation, phase correlation method, adaptive critic design, online learning, Lyapunov method

I. INTRODUCTION

Nanomanipulation (Sitti, 2001) aims at manipulating and handling nanometer size objects and structures with nanometer precision. It is also a first and critical step for achieving any complex functional nano devices. In the past decade, applications of nanoscale research can be found in several fields such as biotechnologies (Lu, 2004), data storage (Requicha, 1999) and prototyping devices (Fukuda and Arai, 2000). However, typically, assembly of small nano structures built by nanomanipulation today consist of ten to twenty particles, and may take an experienced user an entire day to construct using Scanning Probe Microscope (SPM) as the manipulator under tightly controlled conditions. To efficiently accomplish such tasks or even more complex ones, the manipulation process should be automated in order to minimize human intervention.

First of all, it is highly desirable to manipulate the nano objects in ambient conditions, which will lower the cost and complexity greatly for industrial manufacturing. However, in ambient conditions, nanomanipulation encounters multiple external disturbances, which are nonlinear and can result in major problems. Among these uncertainties, thermal drift is the most important one. Research presented in (Yang, Jagannathan, & Bohannan, 2005) provides a satisfactory real-time drift compensation algorithm, based on which, controllers can be designed without considering the influence of the thermal drift.

On the other hand, a number of methods have been introduced for the optimal control of nonlinear systems in the literature aiming at obtaining the best performance of the system. Of the available methods, dynamic programming (DP) has been extensively applied to generate optimal or suboptimal control for nonlinear dynamic systems (Bellman & Dreyfus, 1962; Rekasius, 1964; Leake & Liu, 1967; Kirk, 1970; Werbos, 1977]. However, one of the major drawbacks for conventional DP is the computation cost with the increasing dimension of the nonlinear system, which is referred to as the “curse of dimensionality” (Kirk, 1970). Therefore, adaptive DP schemes (Luus, 2000; Murray, Cox, Lendaris, & Saeks, 2002) have been developed recently. Nevertheless, most of them are implemented either in an offline fashion using iterative schemes or require the dynamics of the nonlinear systems to be known *a priori*. Unfortunately, these requirements are often not practical for real-world applications, since the exact model of the nonlinear is usually not available. Additionally, stability of the closed-loop system using adaptive dynamic programming is not discussed.

Reinforcement learning was originated from the research on animal behavior and its interactions with the environment. Differing from the traditional supervised learning in neural network (NN), there is no desired behavior or training examples employed within reinforcement learning schemes. The learner is not told which action to take, but instead must discover the action that yields the best reward for a given condition by interacting with the environment. Nevertheless, it is common to apply reinforcement learning for optimal controller design, since the cost function or the performance index can be directly seen as a form of reinforcement signal. Of the available reinforcement learning schemes, the temporal difference (TD) learning method (Barto, Sutton, & Anderson, 1983; Sutton, 1988; Watkins & Dayan, 1992; Sutton & Barto, 1998) has found many applications in the engineering area. The advantage of reinforcement learning in general is that the knowledge of the system dynamics is not required even though an iterative approach is typically utilized. To obtain a satisfactory reinforcement signal for each action and system state pair, the approach must visit each system state and apply action often enough (Boone, 1997), which in turn requires the system to be time-invariant, or stationary in the case of stochastic system.

To overcome the iterative offline methodology for real-time applications, several appealing online neural controller design methods were introduced in (Si, Barto, Powell, & Wunsch, 2004; Prokhorov & Wunsch 1997; Miller, Sutton, & Werbos, 1990; Werbos, 1977; Werbos, 1987). They are also referred to as forward dynamic programming (FDP) or adaptive critic designs (ACD). The central theme of this approach is that the optimal control law and cost function are approximated by parametric structures, such as neural networks (NNs), polynomials or splines (Tsitsiklis & Van Roy, 1997), which are trained over time along with the feedback information. In other words, in ACD methods, instead of finding the exact minimum, a parametric structure is employed to approximate the Bellman equation defined as

$$J(x(k)) = \min_{u(k)} \{ J(x(k+1)) + U(x(k), x(k+1)) \}$$

where $x(k)$ is the state and $u(k)$ is the control at time step k. The strategic utility function or cost function $J(x(k))$ represents the minimum cost or performance measure associated with going from k to final step N, $U(x(k), x(k+1))$ is the utility function denoting the cost incurred in going from k to $k+1$ step using control $u(k)$, and $J(k+1)$ is the minimum cost or performance measure associated in going from state $k+1$ to the final step N. In the ACD literature, NNs are widely used for approximation.

In (Si & Wang, 2001), a new NN learning algorithm based on gradient descent rule is introduced. However, no proof of the convergence or stability of the system was given. By contrast, Lyapunov analysis was derived in (He & Jagannathan 2005) and (Kim & Lewis 2000). However, the approach presented in (Kim & Lewis 2000) is specifically designed for robotic systems whose dynamics are introduced in continuous-time. On the other hand, (Si & Wang 2001) and (He & Jagannathan 2005) only employ a simplified binary reward or cost function which is a simplified variant of the standard Bellman equation. To the best of our knowledge, there is no published work presenting the convergence of the closed-loop system with standard Bellman equation.

In this paper, we are considering NNs as the parametric structure to approximate optimal control law and cost function for nonlinear discrete systems with quadratic-performance index as the cost function. The entire system consists of two NNs: an action NN to derive the optimal (or near optimal) control signal to track not only the desired system output but also to minimize the long-term cost function; an adaptive critic NN to approximate the long-term cost function $J(x(k))$ and to tune the action NN weights.

Before practically applying the control design on nanomanipulation system, a satisfactory model is required to verify these methods using simulation. A significant amount of work on modeling interactive forces with surface roughness effect during manipulation was introduced in (Yang & Jagannathan, 2006; Sitti & Hashimoto, 2000). Based on that model, real-time controllers can be designed to automate the nanomanipulation process. However, due to extremely complex and dynamic environmental conditions during nanomanipulation tasks, it will be extremely hard to implement any iteration based optimal controllers. As a matter of fact, for every single manipulation attempt, the environmental conditions or the system dynamics are different from another trial making the supervised learning not an option. Consequently, in this paper, the online learning controller design is proposed on nanomanipulation system and simulation results show its effectiveness.

The paper is organized as follows. In Section II, we first introduce the background of adaptive critic designs and assumptions of the system. Principles of nanomanipulation system are also included in Section 2. In Section 3, drift compensator algorithm is briefly introduced. Section 4 presents the detailed controller design methodology with learning algorithm for the action and critic NNs. Theoretic results are proposed in Section 5 and simulation results on Atomic Force Microscope (AFM) based nanomanipulation system are demonstrated in Section 6.

II. BACKGROUND

A. Optimal Control

In this paper, we consider the following stabilizable nonlinear affine system, given in the form

$$\begin{aligned} x\left(k+1\right) &= f_0(x(k),u(k)) \\ &= f\left(x\left(k\right)\right)+g\left(x\left(k\right)\right)u\left(k\right)+d\left(k\right) \end{aligned} \tag{1}$$

with the state $x(k)=[x_1(k),x_2(k),\cdots,x_n(k)]^T \in R^n$ at time instant *k*. $f(x(k)) \in R^n$ is a unknown nonlinear function vector, and $g(x(k)) \in R^{n\times n}$ is a matrix of unknown nonlinear functions, $u(k) \in R^n$ is the control input vector and $d(k) \in R^n$ is the unknown but bounded disturbance vector, whose bound is assumed to be a known constant, $\|d(k)\| \le d_m$. Here $\|\cdot\|$ stands for the Frobenius norm (Lewis, 1999), which will be used through out this paper. It is also assumed that the state vector $x(k)$ is available at the kth step.

Assumption 1: Let the diagonal matrix $g(x(k)) \in R^{n \times n}$ be a positive definite matrix for each $x(k) \in R^n$, with $g_{\min} \in R^+$ and $g_{\max} \in R^+$ represent the minimum and maximum eigenvalues of the matrix $g(x(k))$, respectively, such that $0 < g_{\min} \le g_{\max}$.

Further, the long-term cost function is defined as

$$\begin{aligned} J(k) = J(x(k),u) &= \sum_{i=t_0}^{\infty} \gamma^i r(k+i) \\ &= \sum_{i=t_0}^{\infty} \gamma^i [q(x(k+i)) + u^T(k+i)Ru(k+i)] \end{aligned} \tag{2}$$

where $J(k)$ stands for $J(x(k),u)$ for simplicity, and u is a control policy. $r(k)$ is the immediate cost function or Lagrangian and γ $(0 \le \gamma \le 1)$ is the discount factor for the infinite-horizon problem. As observed from (2), the long-term cost function is the discounted sum of the immediate cost, which is defined as

$$\begin{aligned} r(k) &= q(x(k)) + u^T(k)Ru(k) \\ &= (x(k) - x_d(k))^T Q(x(k) - x_d(k)) + u^T(k)Ru(k) \end{aligned} \tag{3}$$

where R and Q are positive definite matrices. In this paper, we are using a widely used standard quadratic cost function defined based on the control effort and the tracking error $e(k)$, which will be defined later in contrast with (Si & Wang, 2001; He & Jagannathan, 2005). The immediate cost function $r(k)$ can be viewed as the cost associated with the current step.

The basic idea in adaptive critic or reinforcement learning design is to approximate the long-term cost function $J(k)$ (or its derivative, or both), and generate the control signal minimizing the cost. By using learning, the online approximator will converge to the optimal cost function and the controller will converge to the optimal controller correspondingly. As a matter of fact, for a state feedback optimal control law, which can be expressed as $u^*(k) = u^*(x(k))$, the optimal long-term cost function can be written alternatively as $J^*(k) = J^*(x(k), u^*(x(k))) = J^*(x(k))$, which is just a function of the current state (Bertsekas, 2000). Next, one can state the following assumption.

Assumption 2: The optimal cost function $J^*(k)$ is finite and bounded over the compact set $S \subset R^n$ by J_m.

This assumption is mild and therefore acceptable. By taking the system as stabilizable, the optimal controller is able to achieve a finite cost function, which is the lowest, compared with all other control laws.

B. Nanomanipulation

Nowadays, assemblies of small nano structures built by nanomanipulation are typically realized by using an Atomic Force Microscope (AFM) as the manipulator, which is a special type of SPM. Initially used as the imaging tool, now the AFM tip is utilized as robotic hand to precisely position nano objects and assemble them. However, due to the lack of understanding of nano physics and chemistry, intelligent automatic manipulation yet precise strategies have not been developed for specific applications (Sitti, & Hashimoto, 2000). Thus, the purpose of this paper is to introduce a pushing mechanism for nano particles in the order of 10 nm.

The simplified geometrical relationship between AFM tip, nano sphere and substrate (stage) is shown in Fig. 1. Briefly, the objective of nanomanipulation is to drive the AFM tip to mechanically push nano particles along a desired track. An alternative way is to drive the stage instead of the tip to accomplish the pushing task. In our experiments, the latter approach is selected.

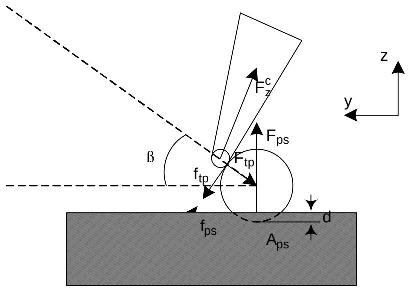


Fig 1. Geometry and the Interacting Forces between AFM Tip, Nano Particle and Stage during Pushing Process

Before designing any control scheme, the model analysis is undertaken involving the adhesion forces between AFM tip, substrate and nano particle to be pushed. In the nano world, gravitational forces are relatively very small and, therefore, are neglected. The main components of the adhesion forces are van der Waals, capillary, and electrostatic forces (Sitti & Hashimoto, 2000).

After taking all those adhesion forces and friction forces into consideration along with the surface roughness effect, a satisfactory model is built in (Yang & Jagannathan, 2006), which will be also adopted in this paper. Since we are driving the stage instead of the tip to accomplish the task, the equation governing the system is (Hicks & Atherton, 1997)

$$\begin{aligned}
&\frac{1}{w_x^2}\ddot{x}_s+\frac{1}{w_xQ_x}\dot{x}_s+x_s+\cos\theta f_{ps}(z_s,z_{sub})=\tau_x\\
&\frac{1}{w_y^2}\ddot{y}_s+\frac{1}{w_yQ_y}\dot{y}_s+y_s+\sin\theta f_{ps}(z_s,z_{sub})=\tau_y\\
&\frac{1}{w_z^2}\ddot{z}_s+\frac{1}{w_zQ_z}\dot{z}_s+z_s+F_{ps}(z_s,z_{sub})=\tau_z+A_{ps}
\end{aligned} \tag{4}$$

where $(x_s,\ y_s,\ z_s)$ is the position of the stage on x, y, and z axis, respectively. $(w_x,\ w_y,\ w_z)$ is the resonant frequency and $(Q_x,\ Q_y,\ Q_z)$ is the amplification factor for the stage. $(\tau_x,\ \tau_y,\ \tau_z)$ is the stage driving force which is seen as the control input signal. The term θ is the angle between y axis and the pushing direction, and z_{sub} is the substrate surface height displacement, or the roughness of the surface, which is simulated to be a sinusoid function in this paper for simplification. Now f_{ps} is the friction force and F_{ps} is the attractive/repulsive interaction force between the particle and substrate, which is a complex function of the pushing environment. For more details, please refer to (Yang & Jagannathan, 2006) and (Sitti & Hashimoto, 2000). Equation (4) indicates that the manipulation system can be viewed as an affine nonlinear system of second order.

To fulfill Assumption 1, we define the tracking error of nanomanipulation system as

$$e_s=\begin{bmatrix}x_s\\y_s\\z_s\end{bmatrix}-\begin{bmatrix}x_d\\y_d\\z_d\end{bmatrix} \tag{5}$$

where $(x_d,\ y_d,\ z_d)$ is the desired movement of the stage. Based on that, filtered tracking error can be defined as $s=\dot{e}_s+\Lambda e_s$, with Λ a positive definite design parameter matrix.

Common usage is to select Λ diagonal with large positive entries. Therefore, the system dynamics can be rewritten in term of the filtered tracking error as follows

$$
\begin{aligned}
\dot{s} &= \ddot{e}_s + \Lambda \dot{e}_s \\
&= \begin{bmatrix} \ddot{x}_s \\ \ddot{y}_s \\ \ddot{z}_s \end{bmatrix} - \begin{bmatrix} \ddot{x}_d \\ \ddot{y}_d \\ \ddot{z}_d \end{bmatrix} + \Lambda \begin{bmatrix} \dot{x}_s \\ \dot{y}_s \\ \dot{z}_s \end{bmatrix} - \Lambda \begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{z}_d \end{bmatrix} \\
&= \begin{bmatrix} \left(\Lambda - \frac{w_x}{Q_x}\right)\dot{x}_s - w_x^2 x_s - \ddot{x}_d - \Lambda \dot{x}_d - w_x^2 \cos\theta f_{ps} \\ \left(\Lambda - \frac{w_y}{Q_y}\right)\dot{y}_s - w_y^2 y_s - \ddot{y}_d - \Lambda \dot{y}_d - w_y^2 \sin\theta f_{ps} \\ \left(\Lambda - \frac{w_z}{Q_z}\right)\dot{z}_s - w_z^2 z_s - \ddot{z}_d - \Lambda \dot{z}_d - w_z^2 (F_{ps} - A_{ps}) \end{bmatrix} + \begin{bmatrix} w_x^2 & 0 & 0 \\ 0 & w_y^2 & 0 \\ 0 & 0 & w_z^2 \end{bmatrix} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \\
&= f_s(s) + w \cdot \tau
\end{aligned}
\tag{6}
$$

As long as the controller guarantees that the filtered tracking error s is bounded, the tracking error e_s is bounded. In order to apply our nonlinear discrete-time controller, the system dynamics (6) need to be discretized by using the standard zero-order-hold techniques to obtain an affine nonlinear discrete-time system (Borgers & Sarin, 1997) which is given by

$$s(k+1) = T\left(F_s(s(k), s(k-1), \ldots) + w \cdot \tau\right) + s(k) \tag{7}$$

where T is the sampling time and F_s is the corresponding nonlinear function of f_s in discrete form. By rearranging (7), one can get an affine nonlinear discrete-time system (1), with the filtered tracking error as the new system state.

III. DRIFT COMPENSATION IMPLEMENTATION

Due to thermal changes and the dynamic environment, under ambient conditions, thermal drift, usually appears in successive SPM scans of a sample even when the scanning parameters are not altered. In the *x-y* plane, drift can be observed as a translation between different images, which depends on thermal changes and other unclear factors. From the height information of the sample, drift can be noticed even in the *z* direction. The drift velocities on the *x*-*y* plane are reported to vary from 0.01~0.1 nm/s (Mokaberi & Requicha, 2004). So the drift between two images taken at 256 sec interval can be as much as 25.6 nm, which is comparable to the diameter of the particles that are normally manipulated. In our experiments, drift in the *z*-direction is about 0.005nm/s (Yang, & Jagannathan, 2005), which is considered negligible. As a result, due to unexpected thermal drift, the nano-manipulation task can fail unless it is compensated.

Our first goal is to develop a drift compensation scheme to estimate and in turn compensate the drift along the *x* and *y* directions so that nanomanipulation can be performed as if drift does not exist. Fortunately, experiments show that the drift in *x* and *y* directions can be seen as a translational movement, not rotation (Mokaberi & Requicha, 2004). In addition, there is negligible correlation between the two directions. Therefore, the compensator can be designed for *x* and *y* directions separately. The block diagram of the proposed compensation system is depicted in Fig. 2. For simplicity, only the drift in the *x* direction is shown (Yang & Jagannathan, 2005). Due to the working principles of AFM, the sample topographic information is not available during the pushing procedure. So the overall solution is stated as follows: 1) the sample is scanned at a constant frequency; 2) at each iteration, after obtaining the scanning data, the drift value, $x_c(t)$ and $y_c(t)$, is estimated and predicted; 3) during the subsequent time interval before the next scanning, the pushing task can be performed by using a control scheme that compensates the drift.

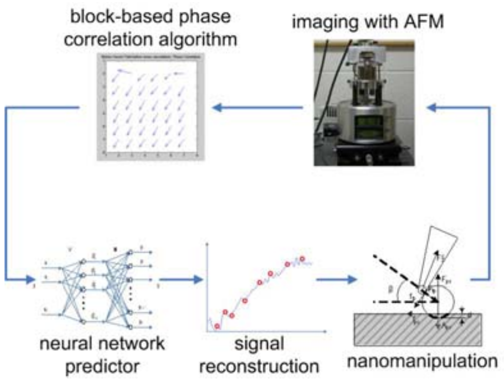


Fig 2. Block Diagram of the Whole Pushing Mechanism with Drift Compensator

In the proposed scheme, the drift is measured by using a block phase correlation algorithm at each sampling time. Based on current and previous data, drift for the subsequent sampling instant can be predicted. Further, signal reconstruction technique is applied to obtain a drift function in continuous time, which is directly applicable to controller design. For more details, refer to (Yang and Jagannathan, 2005). Consequently, the controller can be designed as if the drift does not exist.

Define the states as $s = [x_s \quad y_s \quad z_s \quad \dot{x}_s \quad \dot{y}_s \quad \dot{z}_s]^T$. In order to accommodate the effects of drift, the system states will change to $s_r = s + s_c$ where s_r is the actual position of the particle on the stage coordinates, and $s_c = [x_c \quad y_c \quad z_c \quad \dot{x}_c \quad \dot{y}_c \quad \dot{z}_c]^T$ is the drift value at the current time instant. The amount of drift in the *x*-*y* plane, x_c, y_c, can be estimated satisfactorily by using our drift compensator scheme from (Yang and Jagannathan, 2005), while z_c is negligible. Therefore, the compensator development can be easily combined into the controller by adjusting the desired system trajectories as $s_d = [x_d + x_c \quad y_d + y_c \quad z_d \quad \dot{x}_d + \dot{x}_c \quad \dot{y}_d + \dot{y}_c \quad \dot{z}_d]^T$. In other words, the relative position of the particle with respect to the stage can be calculated and thus be controlled to track the desired trajectory. Next, we will propose our intelligent control algorithms regardless of the drift.

IV. ONLINE REINFORCEMENT LEARNING CONTROLLER DESIGN

For the purpose of this paper, our objective is to design an online reinforcement learning NN controller for the system such that, 1) all the signals in the closed-loop system remain UUB; 2) the state $x(k)$ follows a desired trajectory $x_d(k) \in R^n$; and 3) the long-term cost function (2) is minimized so that a near optimal control input can be generated. Here, the "online" means the learning of the controller takes place "in real-time" by interacting with the plant, instead of in an offline manner.

The block diagram of the proposed controller is shown in Fig. 3, where the action NN is designed to provide an optimal/near-optimal control signal to the plant while the critic NN approximates the long-term cost function. The learning of the two NNs is performed online without any offline learning phase. The learning algorithms are provided later.

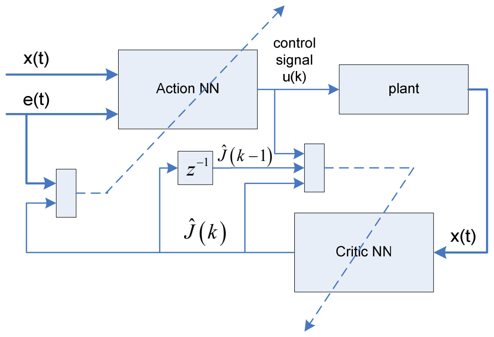


Fig 3. Online Neural Dynamic Programming Based Controller Structure

In our controller architecture, we consider the action and the critic NN having two layers, as shown in Fig. 4. The output of the NN can be given by $Y = W^T \phi(V^T X)$, where

V and W are the hidden layer and output layer weights, respectively. The number of nodes in input, hidden and output layer is denoted as N_1, N_2 and N_3, respectively.

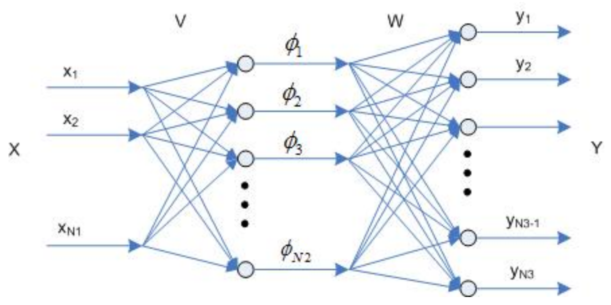


Fig 4. Two Layer Neural Network Structure

One of the interesting features of NN is that they have the property to act as universal function approximators. In other words, a general function $f(x) \in C^{N_3}(S)$ can be written as

$$f(x) = W^T \phi(V^T x) + \varepsilon(x) \tag{8}$$

with $\varepsilon(x)$ a NN functional reconstruction error vector. In our design, V is selected initially at random and held fixed during entire learning process. It is demonstrated in (Igelnik and Pao 1995) that if the hidden layer weights, V, are chosen initially at random and kept constant and if N_2 is sufficiently large, the NN approximation error $\varepsilon(x)$ can be made arbitrarily small since the activation function vector forms a basis.

Furthermore, in this paper, a novel tuning algorithm is proposed to make the NN weights robust so that the Persistency of Excitation (PE) condition is not needed, which will be discussed in the later subsection. Next we present the controller design. Before we proceed, the following mild assumption is needed.

Assumption 3: The desired trajectory of the system states, $x_d(k)$, is a smooth bounded function of time *k* over the compact subset of R^n. For our nanomanipulation system, the desired value is zero, which means we want the filtered tracking error to be zero.

A. The Action NN Design

Considering the general system (1), the tracking error at instant *k* is defined as

$$e(k) = x(k) - x_d(k) \tag{9}$$

Then future value of the tracking error using system dynamics from (1) can be rewritten as

$$e(k+1) = f(x(k)) + g(x(k))u(k) + d(k) - x_d(k+1) \tag{10}$$

To eliminate the tracking error, a desired control signal is given by

$$u_d(k) = g^{-1}(x(k))(-f(x(k) + x_d(k+1) + l_1 e(k)) \tag{11}$$

where $l_1 \in R^{n \times n}$ is a design matrix selected such that the tracking error, $e(k)$, is converging to zero.

Since both $f(x(k))$ and $g(x(k))$ are unknown smooth nonlinear functions, the desired feedback control $u_d(k)$ cannot be implemented directly. Instead, in this paper, an action NN is employed to generate the control signal. From (11) and considering Assumptions 1 and 2, the desired control signal can be approximated as

$$u_d(k) = w_a^T \phi_a(v_a^T s(k)) + \varepsilon_a(s(k)) = w_a^T \phi_a(s(k)) + \varepsilon_a(s(k)) \tag{12}$$

where $s(k) = \left[x^T(k), e^T(k)\right]^T \in R^{2n}$ is the action NN input vector. As stated above, the action NN consists of two layers, and $w_a \in R^{n_a \times n}$ and $v_a \in R^{2n \times n_a}$ denote the desired weights of the output and hidden layer, respectively, with $\varepsilon_a(s(k))$ is the action NN approximation error, and n_a is the number of the neurons in the hidden layer. Since v_a is fixed, for simplicity purpose, the hidden layer activation function vector $\phi_a(v_a^T s(k)) \in R^{n_2}$ is denoted as $\phi_a(s(k))$.

Considering the fact that the desired weights of the action NN are unknown, the actual NN weights have to be trained online and its actual output at time *k* can be expressed as

$$u(k) = \hat{w}_a^T(k)\phi_a(v_a^T s(k)) = \hat{w}_a^T(k)\phi_a(s(k)) \tag{13}$$

where $\hat{w}_a(k) \in R^{n_a \times n}$ is the actual weight matrix of the output layer at instant *k*.

Using the action NN output as the control signal, and substituting (12) and (13) into (10) yields

$$\begin{aligned} e(k+1) &= f(x(k)) + g(x(k))v(k) + d(k) - x_d(k+1) \\ &= l_1 e(k) + g(x(k))\left(v(k) - v_d(k)\right) + d(k) \\ &= l_1 e(k) + g(x(k))\left(\tilde{w}_a^T(k)\phi_a(s(k)) - \varepsilon_a(s(k))\right) + d(k) \\ &= l_1 e(k) + g(x(k))\zeta_a(k) + d_a(k) \end{aligned} \tag{14}$$

where

$$\tilde{w}_a(k) = \hat{w}_a(k) - w_a \tag{15}$$

$$\zeta_a(k) = \tilde{w}_a^T(k)\phi_a(s(k)) \tag{16}$$

$$d_a(k) = -g(x(k))\varepsilon_a(s(k)) + d(k) \tag{17}$$

Thus, the closed-loop tracking error dynamics is expressed as

$$e(k+1) = l_1 e(k) + g(x(k))\zeta_a(k) + d_a(k) \tag{18}$$

Next the critic NN design follows.

B. The Critic NN Design

As stated above, a near optimal controller should be able to stabilize the closed-loop system by minimizing the cost function. In this paper, a critic NN is employed to approximate the long-term cost function $J(k)$. Since the actual $J(k)$ is unavailable for us at the *k*th time instant in an online learning framework, the critic NN is tuned online in order to converge to the actual $J(k)$.

First, the prediction error generated by the critic or the Bellman error (Si and Wang, 2001) is defined as

$$e_c(k) = \gamma \hat{J}(k) - [\hat{J}(k-1) - r(k)] \tag{19}$$

where the subscript "c" stands for the "critic" and

$$\hat{J}(k) = \hat{w}_c^T(k)\phi_c(v_c^T x(k)) = \hat{w}_c^T(k)\phi_c(x(k)) \tag{20}$$

where $\hat{J}(k) \in R$ is the critic NN output which is an approximation of $J(k)$. In our design, the critic NN is also a two-layer NN, while $\hat{w}_c(k) \in R^{n_c \times 1}$ and $v_c \in R^{n \times n_c}$ represent its actual weight matrix of the output and hidden layer, respectively. The term n_c denotes the number of the neurons in the hidden layer. As we claimed above, the cost function is function of the states for given control laws. Thus, similar to HDP, only the system states $x(k) \in R^n$ are selected as the critic NN input in this paper. The activation function vector of the hidden layer $\phi_c(v_c^T x(k)) \in R^{n_c}$ is denoted as $\phi_c(x(k))$ for simplicity. Provided that enough number of the neurons in the hidden layer, the optimal long-term cost function $J^*(k)$ can be approximated by the critic NN with arbitrarily small approximation error $\varepsilon_c(x(k))$,

$$J^*(k) = w_c^T \phi_c(v_c^T x(k)) + \varepsilon_c(x(k)) = w_c^T \phi_c(x(k)) + \varepsilon_c(x(k)) \tag{21}$$

Similarly, the critic NN weight estimation error can be defined as

$$\tilde{w}_c(k) = \hat{w}_c(k) - w_c \tag{22}$$

while the approximation error is given by

$$\zeta_c(k) = \tilde{w}_c^T(k)\phi_c(x(k)) \tag{23}$$

Thus, we

$$\begin{aligned} e_c(k) &= \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \\ &= \gamma\zeta_c(k) + \gamma J^*(k) - \zeta_c(k-1) - J^*(k-1) + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1) \end{aligned} \tag{24}$$

Next we discuss the weight tuning algorithms for both of critic and action NNs.

C. Weight Updating for the Critic NN

Following the discussion from the last section, the objective function to be minimized by the critic NN can be defined as a quadratic function of the Bellman error as

$$E_c(k) = \frac{1}{2} e_c^T(k) e_c(k) = \frac{1}{2} e_c^2(k) \tag{25}$$

Using a standard gradient-based adaptation method, the weight updating algorithm for the critic NN is given by

$$\hat{w}_c(k+1) = \hat{w}_c(k) + \Delta\hat{w}_c(k) \tag{26}$$

where

$$\Delta\hat{w}_c(k) = \alpha_c \left[-\frac{\partial E_c(k)}{\partial \hat{w}_c(k)} \right] \tag{27}$$

with $\alpha_c \in R^+$ is the adaptation gain, which is a positive constant.

Combining (19), (20), (25) with (27), the critic NN weight updating rule can be obtained by using the chain rule as

$$\Delta\hat{w}_c(k) = -\alpha_c \frac{\partial E_c(k)}{\partial \hat{w}_c(k)} = -\alpha_c \frac{\partial E_c(k)}{\partial e_c(k)} \frac{\partial e_c(k)}{\partial \hat{J}(k)} \frac{\partial \hat{J}(k)}{\partial \hat{w}_c(k)} = -\alpha_c \gamma \phi_c(x(k)) e_c(k) \tag{28}$$

Thus, the critic NN weight updating algorithm is obtained as

$$\hat{w}_c(k+1) = \hat{w}_c(k) - \alpha_c \gamma \phi_c(x(k))(\gamma \hat{J}(k) + r(k) - \hat{J}(k-1)) \tag{29}$$

D. Weight Updating for the Action NN

The basis for adapting the action NN is to track the desired trajectory and to lower the cost function. Therefore, the error for the action NN can be formed by using the functional estimation error $\zeta_a(k)$, and the error between the nominal desired long-term cost function $J_d(k) \in R$ and the critic signal $\hat{J}(k)$. Now we define the cost function vector as $\bar{J}(k) = \left[\hat{J}(k) \quad \hat{J}(k) \quad ... \quad \hat{J}(k)\right]^T \in R^{n\times 1}$. Let

$$\begin{aligned} e_a(k) &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1}\left(\bar{J}(k) - J_d(k)\right) \\ &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1}\bar{J}(k) \end{aligned} \tag{30}$$

where $\zeta_a(k)$ is defined in (16). Given Assumption 1, we define $\sqrt{g(x(k))} \in R^{n\times n}$ as the principle square root of the diagonal positive definite matrix $g(x(k))$, i.e., $\sqrt{g(x(k))} \times \sqrt{g(x(k))} = g(x(k))$, and $\left(\sqrt{g(x(k))}\right)^T = \sqrt{g(x(k))}$ (He and Jagannathan, 2005). The desired long-term cost function $J_d(k)$ is nominally defined and is considered to be zero ("0"), which means as low as possible.

Hence, the weights of the action NN $\hat{w}_a(k)$ are tuned to minimize the error

$$E_a(k) = \frac{1}{2} e_a^T(k) e_a(k) \tag{31}$$

Combining (14), (16), (18), (30), (31) and using the chain rule yields

$$\begin{aligned}\Delta\hat{w}_a(k) &= -\alpha_a \frac{\partial E_a(k)}{\partial \hat{w}_a(k)} = -\alpha_a \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \zeta_a(k)} \frac{\partial \zeta_a(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_a \phi_a(s(k))\left(g(x(k))\zeta_a(k) + \bar{J}(k)\right)^T \\ &= -\alpha_a \phi_a(s(k))\left(e(k+1) - l_1 e(k) - d_a(k) + \bar{J}(k)\right)^T \end{aligned} \tag{32}$$

where $\alpha_a \in R^+$ is the positive adaptation gain of the action NN. However, $d_a(k)$ is typically unavailable for us, so as in the ideal case, we assume $d(k)$ and the mean value of $\varepsilon_a(s(k))$ over the compact subset of R^{2n} to be zero and obtain the weight updating algorithm for the action NN as

$$\hat{w}_a(k+1) = \hat{w}_a(k) - \alpha_a \phi_a(s(k))\left(e(k+1) - l_1 e(k) + \bar{J}(k)\right)^T \tag{33}$$

V. MAIN THEORETIC RESULT

Assumption 4: Let w_a and w_c be the unknown output layer target weights for the action and critic NNs, respectively, and assume that they are upper bounded such that

$$\|w_a\| \le w_{am}, \text{ and } \|w_c\| \le w_{cm} \tag{34}$$

where $w_{am} \in R^+$ and $w_{cm} \in R^+$ represent the bounds on the unknown target weights.

Fact 1: The activation functions for the action and critic NNs are bounded by known positive values, such that

$$\|\phi_a(k)\| \le \phi_{am},\ \|\phi_c(k)\| \le \phi_{cm} \tag{35}$$

where $\phi_{am}, \phi_{cm} \in R^+$ is the upper bound for the activation functions. In this paper, a hyperbolic tangent sigmoid function is employed as the transfer function, which satisfies $\|\phi_a(k)\| \le n_a$, $\|\phi_c(k)\| \le n_c$.

Assumption 5: The NN approximation errors $\varepsilon_a(s(k))$ and $\varepsilon_c(x(k))$ are bounded above over the compact set $S \subset R^n$ by ε_{am} and ε_{cm}.

Assumption 5 is a mild assumption since it always holds for continuous functions (Lewis, Yesildirek, & Liu, 1996; Jagannathan, 2006).

Fact 2: With the Assumption 1, 4, the term $d_a(k)$ in (17) is bounded over the compact set $S \subset R^n$ by

$$\left\| d_a(k) \right\| \le d_{am} = g_{\max}\varepsilon_{am} + d_m \tag{36}$$

Combining Assumption 1, 3, and 4 and Facts 1, and 2, the main result of this paper is introduced in the following theorem.

Theorem 1: Consider the system given by (1). Let the Assumptions 1 through 4 hold with the disturbance bound d_m a known constant. Let the control input be provided by the action NN (13), with the critic NN (20). Further, let the weights of the action NN and the critic NN be tuned by (29) and (33), respectively. Then the tracking error $e(k)$, and the NN weight estimates of the action and critic NNs, $\hat{w}_a(k)$ and $\hat{w}_c(k)$ are UUB, with the bounds specifically given by (A.9) through (A.11) in Appendix A, provided the controller design parameters are selected as

(a) $0 < \alpha_a \left\| \phi_a(k) \right\|^2 < g_{\min} / g_{\max}^2$ (37)

(b) $0 < \alpha_c \left\| \phi_c(x(k)) \right\|^2 < 1$ (38)

(c) $0 < l_{\max} < \sqrt{3}/3$ (39)

(d) $\gamma > 1/2$ (40)

where α_a and α_c are NN adaptation gains, and α is employed to define the strategic utility function.

Proof: See Appendix A.

VI. SIMULATION RESULTS

To demonstrate the feasibility of the theoretic results, the on-line learning controller design is implemented on a nanomanipulation system by simulation. In the implementation, the system dynamics are discretized as an affine nonlinear discrete-time system with standard zero-order-hold discretization techniques explained in (Lewis, 1992). The parameters used in this simulation are set as follows:

Table 1 Parameters Used in Simulation for Nanomanipulation

Parameter	w_x	w_y	w_z
Value	1570 rad/s	1570 rad/s	117.6 rad/s
Parameter	Q_x, Q_y, Q_z	θ	R
Value	20	30°	0.1
Parameter	F	γ	Λ
Value	0.1	0.5	100
Parameter	α_a	α_c	l_1
Value	1×10^{-8}	1×10^{-8}	0.1
Parameter	n_a	n_c	d_m
Value	20	20	1.0×10^{-10}

The simulation is run with time step of 1×10^{-5} s. The radius of the pushed nano particle is 15nm. The objective or the desired trajectory is to realize the movement of the particle along the sample surface with a constant speed, which is set to be v = 1000 nm/s.

The direction of desired trajectory is at an angle of 30° with respect to y axis and 60° with respect to *x* axis. A proper force on the nano particle will indicate that the particle is being pushed by the tip, which could be observed by the stability of the stage on *z* axis. In this paper, the surface roughness is modeled by a 2-dimensional sinusoid function with form as $z_{sub} = \sin(0.1 * x_{sub} + 0.1 * y_{sub})$ (unit: nm), where (x_{sub}, y_{sub}) is the position on substrate coordinates.

Our online learning controller is first applied on the system, with the results shown as in Fig. 5. The reason why $z_s - z_{sub}$ is shown in Fig. 5 instead of z_s is to eliminate the effect of surface roughness and make it more convenient to verify the stability of the stage along *z* axis.

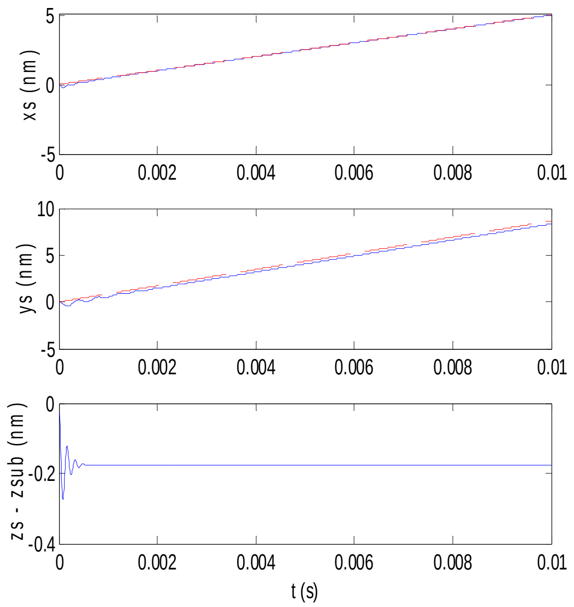


Fig 5. Simulation Results of the Online Learning Controller on Nanomanipulation System. Solid line: Trajectories of the Actual Movement of the Stage; Dashed Line: Desired Movement of the Stage. Note: There is no Desired Trajectory in the *z* Axis.

To compare the performance, the system is also simulated with a traditional NN controller, which is the discretized version of the one constructed in (Yang, & Jagannathan, 2006). The results are shown at Fig. 6. From the results, we can find that the online learning controller is better at stabilizing the stage along the *z* axis by exerting a more stable force on the particle. Meanwhile, the cost of the online learning controller is calculated to be 377.75, which is much better than that of the traditional NN controller (468.02). Although the critic design adds more complexity to the system implementation, it is able to generate better performance in terms of cost and force along the z-direction.

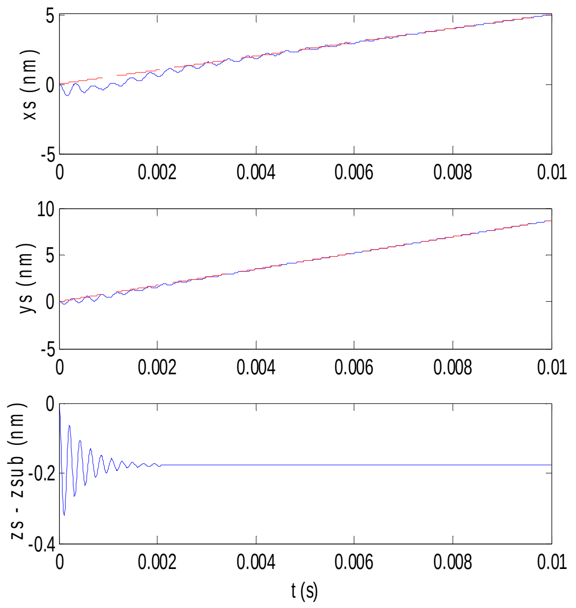


Fig 6. Simulation Results of Traditional NN Controller on Nanomanipulation System. Solid Line: Trajectories of the Actual Movement of the Stage; Dashed Line: Desired Movement of the Stage.

Furthermore, to demonstrate the importance of the proposed drift compensator, our real-time controller design is applied on the manipulation system without the drift compensation. Here assuming the drift is only 5 nm, the system response is depicted in Fig. 7. From the result, one can find that a very small drift will deteriorate the performance of the closed-loop system. If this continues with time, the system will approach instability unless a compensation mechanism is adopted.

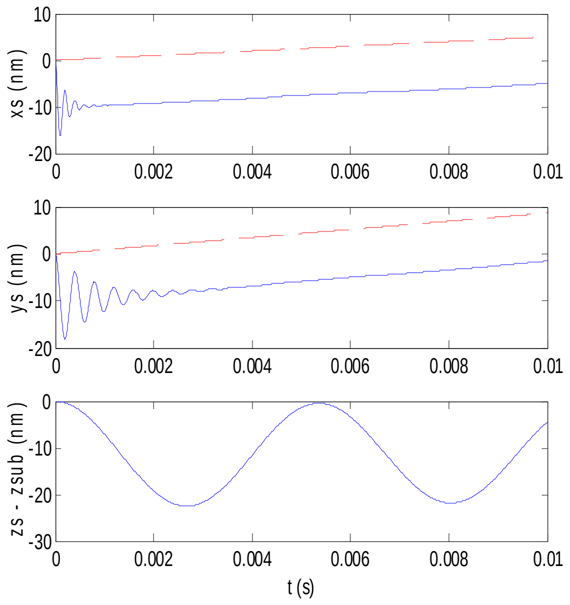


Fig 7. Simulation Results of the Online Learning Controller on Nanomanipulation System without Drift Compensation. Solid Line: Trajectories of the Actual Movement of the Stage; Dashed Line: Desired Movement of the Stage.

VII. CONCLUSIONS

A novel reinforcement learning-based online neural controller is designed for affine nonlinear systems to deliver a desired performance under bounded disturbance. The proposed NN controller optimizes the long-term cost function by introducing a critic NN. Unlike the many applications where the controller is trained offline, the control input is updated in an online fashion. Online learning control designs are especially useful for such complex systems whose dynamics are varying along with time and whose exact models are unreachable. To guarantee that a control system must be stable all of the time, the UUB of the closed-loop tracking errors and NN weight estimates is verified by using Lyapunov analysis in the presence of bounded disturbances and approximation errors. Nanomanipulation system is a promising application and demands that the task is made automatic. However, its "fragile" dynamics do not allow the implementation of iterative based control design. The feasibility of our advanced control method is also strengthened through the simulation results.

VIII. REFERENCES

1. Barto, A. G., Sutton, R. S. and Anderson, C. W. (1983). Neuron like adaptive elements that can solve difficult learning control problems. IEEE Trans. Syst., Man, Cybern., vol. 13, pp. 834–847.
2. Bellman, R. and Dreyfus S. (1962). Applied dynamic programming. Princeton, NJ: Princeton Univ. Press.
3. Bertsekas, D. P. (2000). Dynamic programming and optimal control. Belmont, MA: Athena Scientific.
4. Boone, G. (1997). Efficient reinforcement learning: Model-based acrobot control. Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 229 - 234.
5. Borgers, T., and Sarin, R. (1997). Learning through reinforcement and replicator dynamics. J. Economic Theory, vol. 77, no. 1, pp. 1–17.
6. Fukuda, T., & Arai, F. (2000). Prototyping design and automation of micro/nano manipulation system. Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 192-197.

7. He, P., and Jagannathan, S. (2005). Reinforcement learning-based output feedback control of nonlinear systems with input constraints. IEEE Trans. Syst., Man, Cybern., vol. 35, pp. 150–154.

8. Hicks, T. R., and Atherton, P. D. (1997). The Nano Positioning Book. Bracknell, U.K.: Queensgate Inst.

9. Igelnik, B., and Pao, Y. H. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans. Neural Network, vol. 6, no. 6, pp. 1320–1329.

10. Jagannathan, S. (2006). Neural network control of nonlinear discrete-time systems. Taylor and Francis (CRC Press), Boca Raton, FL.

11. Kim, Y. and Lewis, F. L. (2000). Optimal design of CMAC neural network controller for Robot Manipulators. IEEE Trans. Systems, Man, and Cybernetics, vol. 30, no. 1, pp. 22-31.

12. Kirk, D. (1970). Optimal control theory: an introduction. Englewood Cliffs, NJ: Prentice-Hall.

13. Leake, R. J., & Liu, Ruey-wen (1967). Construction of suboptimal control sequences. SIAM J. Control and Optimization vol. 5, No. 1, pp. 54-63.

14. Lewis, F. L. (1992). Applied Optimal Control and Estimation, Prentice-Hall, New Jersey.

15. Lewis, F. L., Yesildirek, A., & Liu, K. (1996). Multilayer neural-net robot controller with guaranteed tracking performance. IEEE Trans. Neural Networks, vol. 7, pp. 388–399.

16. Lewis, F. L., Jagannathan, S., & Yesildirek, A. (1999). Neural network control of robot manipulators and nonlinear systems. Taylor & Francis, PA.

17. Lu, J. H. (2004). Nanomanipulation of extended single-DNA molecules on modified mica surfaces using the atomic force microscopy. Colloids and Surf. B: Biointerfaces, vol. 39, no. 4, pp. 177-80.

18. Luus, R. (2000). Iterative dynamic programming. CRC Press, Boca Raton, FL.

19. Miller, W. T., Sutton, R. S., and Werbos, P. J., Eds. (1990). Neural networks for control. Cambridge, MA, MIT Press.

20. Mokaberi, B., and Requicha, A. A. G. (2004). Towards automatic nanomanipulation: drift compensation in scanning probe microscopes. IEEE Int. Conf. on Robotics and Automation, Vol. 1, pp. 416 - 421.

21. Murray, J. J., Cox, C. J., Lendaris, G. G., and Saeks, R. (2002). Adaptive dynamic programming. IEEE Transactions on Systems, Man, and Cybernetics, Part C 32(2), pp. 140-153.

22. Prokhorov, D., and Wunsch, D. (1997). Adaptive critic designs. IEEE Trans. Neural Networks, Vol. 8, No.5, p.997-1007.

23. Rekasius, Z. V. (1964). Suboptimal design of intentionally nonlinear controllers. IEEE Transactions on Automatic Control, vol. 9, no. 4, pp. 380--386.

24. Requicha, A. A. G. (1999). Massively Parallel Nanorobotics for Lithography and Data Storage. Int. Journal of Robotics Research, 18, pp. 344-350

25. Si, J., and Wang, Y. T. (2001). On-line learning control by association and reinforcement. IEEE Trans. Neural Networks, vol. 12, no. 2, pp. 264–276.

26. Si, J., Barto, A. G., Powell, W. B., & Wunsch, D., Eds. (2004). Handbook of learning and approximate dynamic programming. Wiley-IEEE Press.

27. Sitti, M., and Hashimoto, H. (2000). Controlled pushing of nanoparticles: modeling and experiments. IEEE/ASME Trans. on Mechatronics.

28. Sitti, M. (2001). Survey of nanomanipulation systems. IEEE Proceedings of the Nanotechnology, pp.75–80.

29. Sutton, R. S. (1988). Learning to predict by the methods of temporal difference. Machine Learning, vol. 3, pp. 9–44.

30. Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: an introduction. the MIT Press, Cambridge, MA.

31. Tsitsiklis, J. N., & Van Roy B. (1997). An analysis of temporal-difference learning with function approximation. IEEE Trans. Automatic Control, vol. 42, no. 5, pp. 674–690.

32. Watkins, C. J. C. H., and Dayan, P. (1992). Q-learning. Machine Learning, vol. 8, pp. 257–277.

33. Werbos, P. J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. General System Yearbook, vol. 22, pp. 25–38.

34. Werbos, P. J. (1987). Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. IEEE Transactions on Systems, Man, and Cybernetics, 17, pp. 7–20.

35. Yang, Qinmin, Jagannathan, S., and Bohannan, E. W. (2005). Block phase correlation-based automatic drift compensation for atomic force microscopes. Proc. of 5th IEEE Conf. on Nanotechnology, Vol. 1, pp. 370-373.

36. Yang, Qinmin, and Jagannathan, S. (2006). Atomic force microscope-based nanomanipulation with drift compensation. Int. Journal Nanotechnology, Vol. 3, No. 4, pp. 527-544.

Appendix

Proof of Theorem 1: Define the Lyapunov candidate as

$$\begin{aligned} L(k) &= \sum_{i=1}^{4} L_i \\ &= \frac{\gamma_1}{3}\|e(k)\|^2 + \frac{\gamma_2}{\alpha_a} tr\left(\tilde{W}_a^T(k)\tilde{W}_a(k)\right) + \frac{\gamma_3}{\alpha_c} tr\left(\tilde{W}_c^T(k)\tilde{W}_c(k)\right) + \gamma_4 \|\zeta_c(k-1)\|^2 \end{aligned} \tag{A.1}$$

where $\gamma_i \in R^+$, $i = 1,2,3,4$ are design parameters. Hence, the first difference of the Lyapunov function is given by

$$\begin{aligned} \Delta L_1 &= \frac{\gamma_1}{3}\left(\|e(k+1)\|^2 - \|e(k)\|^2\right) \\ &= \frac{\gamma_1}{3}\left(\|le(k) + g(x(k))\zeta_a(k) + d_a(k)\|^2 - \|e(k)\|^2\right) \\ &\le -\frac{\gamma_1}{3}\left(1 - 3l_{\max}^2\right)\|e(k)\|^2 + \gamma_1 g_{\max}^2 \|\zeta_a(k)\|^2 + \gamma_1 \|d_a(k)\|^2 \end{aligned} \tag{A.2}$$

$$\begin{aligned} \Delta L_2 &= \frac{\gamma_2}{\alpha_a} tr\left(\tilde{W}_a^T(k+1)\tilde{W}_a(k+1) - \tilde{W}_a^T(k)\tilde{W}_a(k)\right) \\ &= \frac{\gamma_2}{\alpha_a} tr\left(-2\tilde{W}_a^T(k)\alpha_a\phi_a\left(s(k)\right)\left(g(x(k))\zeta_a(k) + \bar{J}(k) + d_a(k)\right) + \Delta\hat{W}_a^T(k)\Delta\hat{W}_a(k)\right) \\ &= -2\gamma_2\zeta_a^T(k)g(x(k))\zeta_a(k) - 2\gamma_2\zeta_a^T(k)\left(\bar{J}(k) + d_a(k)\right) \\ &\quad + \gamma_2\alpha_a\|\phi_a\left(s(k)\right)\|^2 \|g(x(k))\zeta_a(k) + \bar{J}(k) + d_a(k)\|^2 \\ &\le -2\gamma_2 g_{\min}\|\zeta_a(k)\|^2 - 2\gamma_2\zeta_a^T(k)\left(\bar{J}(k) + d_a(k)\right) \\ &\quad + \gamma_2\alpha_a\|\phi_a\left(s(k)\right)\|^2 g_{\max}^2\|\zeta_a(k)\|^2 + \gamma_2\alpha_a\|\phi_a\left(s(k)\right)\|^2 \\ &\quad \times\left(\|\bar{J}(k) + d_a(k)\|^2 + 2\gamma_2\alpha_a\left(\bar{J}(k) + d_a(k)\right)^T g(x(k))\zeta_a(k)\right) \end{aligned}$$

$$= \gamma_2 \Big\{ -g_{\min} \|\zeta_a(k)\|^2 - \left(g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2\right)\|\zeta_a(k)\|^2$$

$$-2\zeta_a^T(k)\left(I - \alpha_a \|\phi_a(s(k))\|^2 g(x(k))\right)\left(\bar{J}(k) + d_a(k)\right) + \alpha_a \|\phi_a(s(k))\|^2 \|\bar{J}(k) + d_a(k)\|^2 \Big\}$$

$$= \gamma_2 \Big\{ -g_{\min} \|\zeta_a(k)\|^2 - \left(g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2\right) \times \left\| \zeta_a(k) + \frac{\left(I - \alpha_a \|\phi_a(s(k))\|^2 g(x(k))\right)}{g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2} \right\|^2 \quad \text{(A.3)}$$

$$+ \frac{1 - \alpha_a \|\phi_a(s(k))\|^2 g_{\min}}{g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2} \|\bar{J}(k) + d_a(k)\|^2 \Big\}$$

Set $\gamma_2 = \gamma_2' \gamma_2''$, where

$$\gamma_2'' \frac{1 - \alpha_a \|\phi_a(s(k))\|^2 g_{\min}}{g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2} \le \frac{1}{2},$$

therefore,

$$\begin{aligned} \Delta L_2 \le & -\gamma_2 g_{\min} \|\zeta_a(k)\|^2 - \gamma_2 \left(g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2\right) \\ & \times \left\| \zeta_a(k) + \frac{\left(I - \alpha_a \|\phi_a(s(k))\|^2 g(x(k))\right)}{g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2} \right\|^2 + \gamma_2' \frac{\|\bar{J}(k) + d_a(k)\|^2}{2} \\ \le & -\gamma_2 g_{\min} \|\zeta_a(k)\|^2 - \gamma_2 \left(g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2\right) \\ & \times \left\| \zeta_a(k) + \frac{\left(I - \alpha_a \|\phi_a(s(k))\|^2 g(x(k))\right)}{g_{\min} - \alpha_a \|\phi_a(s(k))\|^2 g_{\max}^2} \right\|^2 + \gamma_2' n \|\zeta_c(k)\|^2 + \gamma_2' n \|J^*(k) + d_a(k)\|^2 \end{aligned} \quad \text{(A.4)}$$

At the same time,

$$\begin{aligned} \Delta L_3 &= \frac{\gamma_3}{\alpha_c} tr\left(\tilde{W}_c^T(k+1)\tilde{W}_c(k+1) - \tilde{W}_c^T(k)\tilde{W}_c(k)\right) \\ &= \frac{\gamma_3}{\alpha_c} tr\left(-2\tilde{W}_c^T(k)\alpha_c \gamma \phi_c(x(k)) e_c(k)\right) + \frac{\gamma_3}{\alpha_c} tr\left(\Delta \hat{W}_c^T(k) \Delta \hat{W}_c(k)\right) \\ &= -2\gamma_3 e_c(k)\left(e_c(k) - \gamma J^*(k) + \zeta_c(k-1) + J^*(k-1) - r(k) + \varepsilon_c(k) - \varepsilon_c(k-1)\right) \\ &\quad + \gamma_3 \alpha_c \gamma^2 e_c^2(k) \|\phi_c(k)\|^2 \end{aligned}$$

$$
\begin{aligned}
&\le -\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c\left(k\right)\right\|^2\right)e_c^2\left(k\right)-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2\left(k-1\right)+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2\\
&+\frac{\gamma_3}{4}r(k)+\frac{\gamma_3}{4}\left(\varepsilon_c(k)-\varepsilon_c(k-1)\right)^2\\
&\le -\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c\left(k\right)\right\|^2\right)e_c^2\left(k\right)-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2\left(k-1\right)+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2\\
&+\frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2+\frac{\gamma_3}{4}\left(\zeta_a(k)+w_a{}^T\phi_a\left(k\right)\right)^T R\left(\zeta_a(k)+w_a{}^T\phi_a\left(k\right)\right)+\gamma_3\varepsilon_{cm}{}^2\\
&\le -\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c\left(k\right)\right\|^2\right)e_c^2\left(k\right)-\gamma_3\gamma^2\zeta_c^2(k)+\frac{\gamma_3}{4}\zeta_c^2\left(k-1\right)+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2\\
&+\frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2+\frac{\gamma_3}{8}R_{\max}\left\|\zeta_a(k)\right\|^2+\frac{\gamma_3}{8}R_{\max}\left\|w_a{}^T\phi_a\left(k\right)\right\|^2+\gamma_3\varepsilon_{cm}{}^2
\end{aligned}
\tag{A.5}
$$

where $Q_{\max}$ and $R_{\max}$ are the maximum eigenvalue of matrix Q and R, respectively, and

$$
\Delta L_4=\gamma_4\left(\left\|\zeta_c(k)\right\|^2-\left\|\zeta_c(k-1)\right\|^2\right)
\tag{A.6}
$$

Combining (A.1) - (A.6) yields

$$
\begin{aligned}
\Delta L(k) &\le -\frac{\gamma_1}{3}\left(1-3l_{\max}^2\right)\left\|e(k)\right\|^2+\gamma_1 g_{\max}^2\left\|\zeta_a(k)\right\|^2+\gamma_1\left\|d_a(k)\right\|^2-\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2\\
&-\gamma_2\left(g_{\min}-\alpha_a\left\|\phi_a\left(s(k)\right)\right\|^2 g_{\max}^2\right)\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\left\|\phi_a\left(s(k)\right)\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a\left(s(k)\right)\right\|^2 g_{\max}^2}\right\|^2\\
&+\gamma_2' n\left\|\zeta_c(k)\right\|^2+\gamma_2' n\left\|J^*(k)+d_a(k)\right\|^2-\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k)-\gamma_3\gamma^2\zeta_c^2(k)\\
&+\frac{\gamma_3}{4}\zeta_c^2(k-1)+\frac{\gamma_3}{4}\left(\gamma J^*(k)-J^*(k-1)\right)^2+\frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2+\frac{\gamma_3}{8}R_{\max}\left\|\zeta_a(k)\right\|^2\\
&+\frac{\gamma_3}{8}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2+\gamma_4\left(\left\|\zeta_c(k)\right\|^2-\left\|\zeta_c(k-1)\right\|^2\right)+\gamma_3\varepsilon_{cm}{}^2\\
&=-\left(\frac{\gamma_1}{3}(1-3l_{\max}^2)-\frac{\gamma_3}{4}Q_{\max}\right)\left\|e(k)\right\|^2-\left(\gamma_2 g_{\min}-\gamma_1 g_{\max}^2-\frac{\gamma_3}{8}R_{\max}\right)\left\|\zeta_a(k)\right\|^2\\
&-\left(\gamma_3\gamma^2-\gamma_2' n-\gamma_4\right)\left\|\zeta_c(k)\right\|^2-(\gamma_4-\frac{\gamma_3}{4})\left\|\zeta_c(k-1)\right\|^2-\gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k)\\
&-\gamma_2\left(g_{\min}-\alpha_a\left\|\phi_a\left(s(k)\right)\right\|^2 g_{\max}^2\right)\times\left\|\zeta_a(k)+\frac{\left(I-\alpha_a\left\|\phi_a\left(s(k)\right)\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a\left(s(k)\right)\right\|^2 g_{\max}^2}\right\|^2+D_M^2
\end{aligned}
\tag{A.7}
$$

where

$$D_M^2 = \frac{\gamma_2' n}{2} d_a^2(k) + \left(\frac{\gamma_3}{4} + \frac{\gamma_2' n}{2}\right) J_m^2 + \frac{\gamma_3}{6} R_{\max} \left\| w_a{}^T \phi_a(k) \right\|^2 + \gamma_1 \left\| d_a(k) \right\|^2 + \gamma_3 \varepsilon_{cm}{}^2 \tag{A.8}$$

For the standard Lyapunov analysis, equation (A.7) and (A.8) implies that $\Delta L \le 0$ as long as the conditions (37) – (40) are satisfied and following holds

$$\left\| e(k) \right\| \ge \frac{2\sqrt{3} D_M}{\sqrt{4\gamma_1 \left(1 - 3 l_{\max}^2\right) - 3\gamma_3 Q_{\max}}} \tag{A.9}$$

or

$$\left\| \zeta_a(k) \right\| \le \frac{2\sqrt{2} D_M}{\sqrt{8\gamma_2 g_{\min} - 8\gamma_1 g_{\max}^2 - \gamma_3 R_{\max}}} \tag{A.10}$$

or

$$\left\| \zeta_c(k) \right\| \le \frac{D_M}{\sqrt{\gamma_3 \gamma^2 - \gamma_2' n - \gamma_4}} \tag{A.11}$$

According to the standard Lyapunov extension theorem (Jagannathan, 2006), the analysis above demonstrates that the tracking error $\left\| e(k) \right\|$ and the weights of the estimation errors are UUB. Further, the boundedness of $\left\| \zeta_a(k) \right\|$ and $\left\| \zeta_c(k) \right\|$ implies that the weight estimations $\left\| \hat{w}_a(k) \right\|$ and $\left\| \hat{w}_c(k) \right\|$ are also bounded.

PAPER 5

Near Optimal Control of Affine Nonlinear Discrete-time Systems using Online Approximators

Qinmin Yang and S. Jagannathan

Department of Electrical and Computer Engineering

University of Missouri – Rolla,

Rolla, Missouri, U.S.A 65409

Email: qyy74@umr.edu, sarangap@umr.edu

ABSTRACT

In this paper, both state and output feedback reinforcement learning online approximator-based near optimal controller is proposed for general multi-input and multi-output affine unknown nonlinear discrete-time systems in the presence of bounded disturbances. Each of the controller designs contains two entities, an action network that is designed to produce optimal signal and a critic network that evaluates the performance of the action network. The critic is an optimal or near optimal estimator of the cost-to-go function that is tuned online using recursive equations derived from dynamic programming (DP). The critic is termed adaptive as it adapts itself to output the optimal cost-to-go function and the action network is adapted simultaneously based on the information provided by the critic to derive the control signal in order to track a desired system trajectory while minimizing the cost function. Here neural networks (NN) are used both for the action and critic whereas any online approximators such as radial basis functions (RBF), splines, fuzzy logic etc can be utilized. For the output feedback controller, an additional NN is designated as the observer to estimate the system states and separation principle is not required. The parameters of the online approximators or NN weight tuning rules for these two controller schemes are also derived while ensuring uniformly-ultimately-boundedness (UUB) of the closed-loop system using Lyapunov. Furthermore, the effectiveness of the two controllers is tested on a pendulum balancing system and a two-link robotic arm system in simulation.

Keywords

Online approximators, neural network, reinforcement learning, on-line learning, dynamic programming, Lyapunov method

I. INTRODUCTION

In the literature, there are many ways for designing stable controllers for nonlinear systems. However, stability is only a bare requirement for the controller design. A further consideration is the optimality based on a predefined cost function, which is used to determine the performance of the system. In other words, a controller scheme should not only achieve the stability of the closed-loop system, but also keep the cost as small as possible. A number of methods have been introduced for the optimal control of nonlinear systems.

Of the available methods, dynamic programming (DP) has been extensively applied to generate optimal control for nonlinear dynamic systems [1]-[4], [22] by utilizing Bellman's Principle of Optimality – "no matter how an intermediate point is reached in an optimal trajectory, the rest of the trajectory (from the intermediate point to the end) must be optimal." It can provide the truly optimal solutions for nonlinear dynamic systems, but one of its drawbacks is the computation cost with the increasing dimension of the nonlinear plant, which is referred to as the "curse of dimensionality"

[4]. Therefore, to confront this issue, adaptive DP methods (e.g., see [8], [29]) have been developed recently. However, most of them are implemented either by an offline using iterative schemes or require the dynamics of the plants to be known *a priori*. Unfortunately, these requirements are often not practical for real-world systems, since the exact dynamics of the plant is usually not available.

On the other hand, reinforcement learning is originated from animal behavior research and its interactions with the environment. It is based on the common sense reasoning that if an action is followed by a satisfactory outcome (reinforcement signal), then the tendency to repeat that action is strengthened, i.e., reinforced. Differing from the traditional supervised neural network (NN) learning, there is no desired behavior or training examples employed with reinforcement learning schemes. Nevertheless, it is common to apply reinforcement learning for optimal controller design, since the cost function can be directly seen as a form of reinforcement signal. Of the available reinforcement learning schemes, the temporal difference (TD) learning method [9]-[12] has found many applications in engineering since it does not require the knowledge of the system dynamics even though an iterative approach is typically utilized. However, to obtain a satisfactory reinforcement signal for each action, the approach must visit each system state by applying each action often enough [13], which requires that the system be time-invariant, or stationary in the case of stochastic system.

To overcome the iterative offline methodology for real-time applications, several appealing online approximators-based controller designs methods were introduced in [17], [19]-[22]. They are also referred to as forward dynamic programming (FDP) or adaptive critic designs (ACD). The central theme of this approach is that the optimal control law and cost function are approximated by online parametric structures, such as neural networks (NNs), which are trained over time along with the information that is fed back from the system response. Depending upon whether the NNs approximate the cost function or its derivative, or both, the ACDs are classified as: 1) heuristic dynamic programming (HDP); 2) dual heuristic dynamic programming (DHP); and 3) globalized dual heuristic dynamic programming (DHP). It is important to note that when the action is introduced as an additional input to the critic, then the ACD will be referred as action dependent (AD) version of the ACD.

It is important to note that, instead of finding the exact minimum, ACDs try to approximate the Bellman equation $J\left(x(k)\right) = \min_{u(k)}\left\{J\left(x(k+1)\right) + U\left(x(k), x(k+1)\right)\right\}$, where $x(k)$ is the state and $u(k)$ is the control at time step *k*, the strategic utility function $J\left(x(k)\right)$ represents the minimum cost or performance measure associated with going from *k* to final step *N*, $U\left(x(k), x(k+1)\right)$ is the utility function denoting the cost incurred in going from *k* to *k+1* using control $u(k)$, and $J(k+1)$ is the minimum cost or performance measure associated in going from state *k+1* to the final step *N*. The NNs are widely used for approximation in the ACD literature.

Numerous papers have appeared on ACDs using NN but stability is rarely studied. The ones where some sort of stability is discussed are briefly introduced next. A new NN learning algorithm based on gradient descent rule is introduced in [6] and the approach is evaluated on a single cart-pole balancing system and a pendulum and a triple-link inverted pendulum. However, no proof of the convergence or stability of the system was given. By contrast, Lyapunov analysis was derived in [14] and [15] using a variant of

Bellman equation. The approach presented in [15] is specific to robotic systems whose dynamics are introduced in continuous-time. On the other hand, [6] and [14] employ simplified binary reward or cost function which is a variant of the standard Bellman equation. By contrast, in this paper, a novel reinforcement learning-based controller is introduced for multi-input multi-output affine nonlinear discrete-time systems first by assuming state feedback using NN. However, the approach is generic enough that any online approximator based scheme such as RBFs, splines, CMAC can be utilized.

Meanwhile, an output feedback controller scheme is usually necessary when certain states of the plant are unavailable for measurement. It is important to note that, the separation principle, which is normally employed for linear systems, does not hold for nonlinear systems [28]. Therefore, for nonlinear systems, a state observer that estimates the true states online, in general, does not guarantee the stability of the entire closed-loop system when it is used in conjunction with a stabilizing controller. Further, design of an observer and a controller combination becomes more challenging if optimality has to be ensured. Therefore, an output feedback controller with reinforcement learning design is also proposed.

In this paper, we are considering online approximator-based methodology using NNs for the control of nonlinear discrete systems with quadratic-performance index as the cost function. The state feedback scheme consists of two online approximators; in this case two NNs: an action NN for the action network to derive the optimal (or near optimal) control signal to track not only the desired system output but also to minimize the long-term cost function; a critic NN for the critic network to approximate the long-term cost function $J\left(x(k)\right)$ and to tune the action NN weights. For the output feedback controller, an additional NN is employed as the observer to estimate the unavailable system states.

Since the control signal is not used in the critic NN as an additional input, the proposed approach could be seen as an online approximator-based HDP or neural HDP approach. Besides addressing the problem of optimization, contributions of this paper include: 1) the demonstration of the uniformly ultimately boundedness (UUB) of the overall system even in the presence of approximation errors and bounded unknown disturbances unlike in the existing adaptive critic works where the convergence is shown under ideal circumstances; 2) the online approximator parameters or NN weights are tuned online instead of offline training that is commonly employed in ACD; and 3) the linear-in-parameters (LIP) assumption is overcome along with the persistent excitation (PE) condition requirement. Finally, the proposed approach uses the standard Bellman equation and not a variant of the Bellman equation [14].

The paper is organized as follows. In Section II, the background of ACDs and assumptions of the system are introduced. Section III presents the detailed state feedback controller design methodology with learning algorithm for the action and critic NNs. Output feedback control scheme is proposed in Section IV and simulation results on two-link robotic arm and pendulum are demonstrated in Section V.

II. BACKGROUND

A. Adaptive Critic Design

In this paper, we consider the following stabilizable nonlinear affine system, given in the form as

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) \\ y(k) &= x_1(k) \end{aligned} \tag{1}$$

with the state $x(k) = [x_1(k), x_2(k), \cdots, x_n(k)]^T \in R^{nm}$ at time instant k and each $x_i(k) \in R^m$, $i = 1, \cdots, n$. The terms $f(x(k)) \in R^m$ is a unknown nonlinear vector field, $g(x(k)) \in R^{m\times m}$ is a diagonal matrix of unknown nonlinear vector fields, $u(k) \in R^m$ is the control input vector, $y(k) \in R^m$ is the output vector and $d(k) \in R^m$ is the unknown but bounded disturbance vector field, whose bound is assumed to be a known constant, $\|d(k)\| \le d_m$. Here $\|\square\|$ stands for the Frobenius norm [5], which will be used through out this paper. First, the state vector $x(k)$ is assumed available at the *k*th step for the state feedback controller.

Assumption 1: Let the matrix $g(x(k)) \in R^{m\times m}$ be a positive definite diagonal matrix for each $x(k) \in R^{nm}$, with $g_{\min} \in R^+$ and $g_{\max} \in R^+$ represent the minimum and maximum eigenvalues of the matrix $g(x(k))$, respectively, such that $0 < g_{\min} \le g_{\max}$.

For the purpose of this paper, our objective is to design an online reinforcement learning NN controller for the system (1) such that 1) all the signals in the closed-loop system remain uniformly ultimately bounded in the presence of bounded disturbances and approximation errors; 2) the state $x(k)$ follows a desired trajectory $x_d(k) = [x_{1d}(k), x_{2d}(k), \cdots, x_{nd}(k)]^T \in R^{nm}$, or equivalently speaking, the output $y(k)$ follows a desired trajectory $y_d(k) \in R^m$; and 3) the long-term cost function (2) is minimized so that a near optimal control input can be generated. Here, the "online" means the controller learning occurs "in real-time" through constant interaction with the plant, instead of an offline manner.

Assumption 2: The desired trajectory of the system states, $x_d(k) = \left[x_{1d}(k), x_{2d}(k), \cdots, x_{nd}(k)\right]^T$, satisfies $x_{id}(k+1) = x_{(i+1)d}(k)$, $i = 1, \cdots, n-1$, and $y_d(k)$ is a known smooth bounded function over the compact subset of R^m.

Note that, from assumption 2, one can derive that $x_{id}(k) = y_d(k+i-1)$, $i = 1, \cdots, n$.

Meanwhile, to introduce the issue of optimality into our design, the long-term cost function is defined as

$$\begin{aligned} J(k) = J(x(k), u) &= \sum_{i=t_0}^{\infty} \gamma^i r(k+i) \\ &= \sum_{i=t_0}^{\infty} \gamma^i [q(x(k+i)) + u^T(k+i) R u(k+i)] \end{aligned} \tag{2}$$

where $J(k)$ stands for $J(x(k), u)$ for simplicity, and u is a control policy, *R* is a positive definite matrix and $q(x(k))$ is a positive definite function of the states, while γ $(0 \le \gamma \le 1) \triangleright$ is the discount factor for the infinite-horizon problem. As observed from (2), the long-term cost is defined in terms of the discounted sum of the immediate cost or Lagrangian $r(k)$, which is given by

$$\begin{aligned} r(k) &= q(x(k)) + u^T(k) R u(k) \\ &= (x_1(k) - x_{1d}(k))^T Q (x_1(k) - x_{1d}(k)) + u^T(k) R u(k) \\ &= (y(k) - y_d(k))^T Q (y(k) - y_d(k)) + u^T(k) R u(k) \end{aligned} \tag{3}$$

where *Q* is a positive definite matrix. In this paper, we are using a widely used standard quadratic cost function defined based on the output tracking error $e(k) = y(k) - y_d(k)$, which will be contrasted with [6] and [14]. The immediate cost function $r(k)$ can be viewed as a system performance index for the current step.

The basic idea in adaptive critic or reinforcement learning design is to approximate the long-term cost function J (or its derivative, or both), and generate the control signal minimizing the cost. By using learning, the online approximator will converge to the near optimal cost and the controller will converge to the near optimal controller correspondingly. In fact, for an optimal control law, which can be expressed as $u^*(k) = u^*(x(k))$, the optimal long-term cost function can be written alternatively as $J^*(k) = J^*(x(k), u^*(x(k))) = J^*(x(k))$, which is just a function of the current state [16]. Next, one can state the following assumption.

Assumption 3: The optimal cost function $J^*(k)$ is finite and bounded over the compact set $S \subset R^n$ by J_m.

B. Two Layer NN

In our controller architecture, we consider the NNs having two layers, as shown in Fig. 1. The output of the NN can be given by $Y = W^T \phi(V^T X)$, where V and W are the hidden layer and output layer weights, respectively. The number of hidden layer nodes is denoted as N_2. A general function $f(x) \in C^{N_3}(S)$ can be written as [18]

$$f(x) = W^T \phi(V^T x) + \varepsilon(x) \tag{4}$$

with $\varepsilon(x)$ a NN functional reconstruction error vector. In our design, V is selected initially at random and held. It is demonstrated in [18] that if the hidden layer weights, V, are chosen initially at random and held while N_2 is sufficiently large, the NN approximation error $\varepsilon(x)$ can be made arbitrarily small since the activation function vector forms a basis vector. Additionally, one can relax this assumption of bounded approximation error by using a robust signal through an auxiliary control input, which is relegated as part of a future publication. Since NN are utilized here as an illustration of the online approximator, the rest of the paper uses NNs for the online approximator.

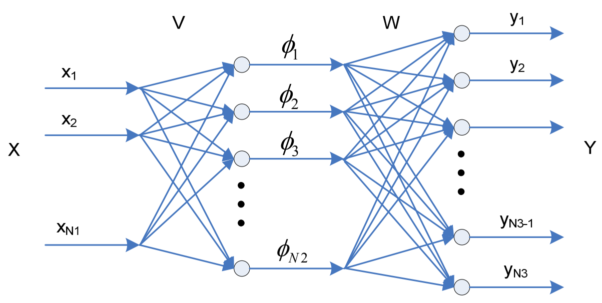


Fig 1. Two Layer Neural Network Structure

III. STATE FEEDBACK ONLINE REINFORCEMENT LEARNING CONTROLLER DESIGN

The block diagram of the proposed controller is shown in Fig. 2 where the action NN is providing a near optimal control signal to the plant while the critic NN approximates the long-term cost function. The learning of the two NNs is performed online without any offline learning phase.

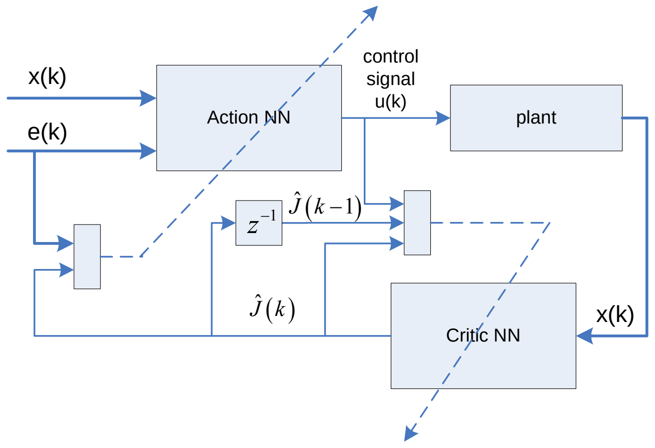


Fig 2. Online Neural Dynamic Programming Based Controller Structure

Furthermore, persistence of excitation (PE) condition is necessary in adaptive control literature which is also required to guarantee boundedness of the NN weight estimates. Unfortunately, it may be difficult to verify the PE condition of the output function $\phi(V^T x)$ of the NN hidden layer. In this paper, a novel tuning algorithm is proposed to make the NN weights robust so that PE is not needed.

A. The Action Network Design

The tracking error at instant k is defined as

$$e_i(k) = x_i(k) - x_{id}(k) = x_i(k) - y_d(k+i-1),\ i = 1,...,n \tag{5}$$

Then future value of the tracking error using system dynamics from (1) can be rewritten as

$$e_n(k+1) = f(x(k)) + g(x(k))u(k) + d(k) - y_d(k+n) \tag{6}$$

The desired control signal can be given by

$$u_d(k) = g^{-1}(x(k))(-f(x(k) + y_d(k+n) + l_1 e(k)) \tag{7}$$

where $l_1 \in R^{m\times m}$ is a design matrix selected such that the tracking error, $e_n(k)$, converges to zero.

Since both of $f(x(k))$ and $g(x(k))$ are unknown smooth nonlinear functions, the desired feedback control $u_d(k)$ cannot be implemented directly. Instead, an action NN is employed to generate the control signal. From (7) and considering Assumption 2, the desired control signal can be approximated as

$$u_d(k) = w_a^T \phi_a(v_a^T s(k)) + \varepsilon_a(s(k)) = w_a^T \phi_a(s(k)) + \varepsilon_a(s(k)) \tag{8}$$

where $s(k) = \left[x^T(k), y_d^T(k), y_d^T(k+n)\right]^T \in R^{(n+2)m}$ is the action NN input vector. The action NN consists of two layers, and $w_a \in R^{n_a \times m}$ and $v_a \in R^{(n+2)m\times n_a}$ denote the desired weights of the output and hidden layer, respectively, with $\varepsilon_a(s(k))$ is the action NN approximation error, and n_a is the number of the neurons in the hidden layer. Since v_a is fixed, for simplicity purpose, the hidden layer activation function vector $\phi_a(v_a^T s(k)) \in R^{n_2}$ is denoted as $\phi_a(s(k))$.

Considering the fact that the desired weights of the action NN are unknown, the actual NN weights have to be trained online and its actual output can be expressed as

$$u(k) = \hat{w}_a^T(k)\phi_a(v_a^T s(k)) = \hat{w}_a^T(k)\phi_a(s(k)) \tag{9}$$

where $\hat{w}_a(k) \in R^{n_a \times m}$ is the actual weight matrix of the output layer at instant *k*.

Using the action NN output as the control signal, and substituting (8) and (9) into (6) yields

$$\begin{aligned} e_n(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) - y_d(k+n) \\ &= l_1 e_n(k) + g(x(k))\big(u(k) - u_d(k)\big) + d(k) \\ &= l_1 e_n(k) + g(x(k))(\tilde{w}_a^T(k)\phi_a(s(k)) - \varepsilon_a(s(k))) + d(k) \\ &= l_1 e_n(k) + g(x(k))\zeta_a(k) + d_a(k) \end{aligned} \tag{10}$$

where

$$\tilde{w}_a(k) = \hat{w}_a(k) - w_a \tag{11}$$

$$\zeta_a(k) = \tilde{w}_a^T(k)\phi_a(s(k)) \tag{12}$$

$$d_a(k) = -g(x(k))\varepsilon_a(s(k)) + d(k) \tag{13}$$

Thus, the closed-loop tracking error dynamics is expressed as

$$e_n(k+1) = l_1 e_n(k) + g(x(k))\zeta_a(k) + d_a(k) \tag{14}$$

At the meantime, we can notice that $e_i(k) = x_i(k) - x_{id}(k) = e_n(k-n+i),\ i = 1,\ldots,n$. Next the critic NN design is introduced.

B. The Critic Network Design

As stated above, a near optimal controller should be able to stabilize the closed-loop system by minimizing the cost function. In this paper, a critic NN is employed to approximate the target long-term cost function $J(k)$. Since $J(k)$ is unavailable at the *k*th time instant in an online learning framework, the critic NN is tuned online in order to ensure that its output converges close to $J(k)$.

First, the prediction error for the critic or the Bellman error [6] is defined as

$$e_c(k) = \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \tag{15}$$

where the subscript "c" stands for the "critic" and

$$\hat{J}(k) = \hat{w}_c^T(k)\phi_c(v_c^T x(k)) = \hat{w}_c^T(k)\phi_c(x(k)) \tag{16}$$

where $\hat{J}(k) \in R$ is the critic NN output which is an approximation of $J(k)$. In our design, the critic NN is also a two-layer NN, while $\hat{w}_c(k) \in R^{n_c \times 1}$ and $v_c \in R^{nm \times n_c}$ represent its actual weight matrix of the output and hidden layer, respectively. The term n_c denotes the number of the neurons in the hidden layer. Similar to HDP, the system states $x(k) \in R^n$ are selected as the critic network input. The activation function vector of the hidden layer $\phi_c(v_c^T x(k)) \in R^{n_c}$ is denoted as $\phi_c(x(k))$ for simplicity. Provided that enough number of the neurons in the hidden layer, the optimal long-term cost function $J^*(k)$ can be approximated by the critic network with arbitrarily small approximation error $\varepsilon_c(k)$ as

$$J^*(k) = w_c^T \phi_c(v_c^T x(k)) + \varepsilon_c(x(k)) = w_c^T \phi_c(x(k)) + \varepsilon_c(x(k)) \tag{17}$$

Similarly, the critic network NN weight estimation error can be defined as

$$\tilde{w}_c(k) = \hat{w}_c(k) - w_c \tag{18}$$

where the approximation error is given by

$$\zeta_c(k) = \tilde{w}_c^T(k)\phi_c(x(k)) \tag{19}$$

Thus, we have

$$\begin{aligned} e_c(k) &= \gamma \hat{J}(k) - \hat{J}(k-1) + r(k) \\ &= \gamma\zeta_c(k) + \gamma J^*(k) - \zeta_c(k-1) - J^*(k-1) + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1) \end{aligned}$$

Next we discuss the weight tuning algorithms for critic and action NNs.

C. Weight Updating for the Critic Network

Following the discussion from the last section, the objective function to be minimized by the critic network is defined as a quadratic function of tracking errors as

$$E_c(k) = \frac{1}{2}e_c^T(k)e_c(k) = \frac{1}{2}e_c^2(k) \tag{20}$$

Using a standard gradient-based adaptation method, the weight updating algorithm for the critic network is given by

$$\hat{w}_c(k+1) = \hat{w}_c(k) + \Delta\hat{w}_c(k) \tag{21}$$

where

$$\Delta\hat{w}_c(k) = \alpha_c\left[-\frac{\partial E_c(k)}{\partial \hat{w}_c(k)}\right] \tag{22}$$

with $\alpha_c \in R$ is the adaptation gain.

Combining (15), (16), (20) with (22), the critic NN weight updating rule can be obtained by using the chain rule as

$$\Delta\hat{w}_c(k) = -\alpha_c\frac{\partial E_c(k)}{\partial \hat{w}_c(k)} = -\alpha_c\frac{\partial E_c(k)}{\partial e_c(k)}\frac{\partial e_c(k)}{\partial \hat{J}(k)}\frac{\partial \hat{J}(k)}{\partial \hat{w}_c(k)} = -\alpha_c\gamma\phi_c(x(k))e_c(k) \tag{23}$$

Thus, the critic NN weight updating algorithm is obtained as

$$\hat{w}_c(k+1) = \hat{w}_c(k) - \alpha_c \gamma \phi_c(x(k))(\gamma \hat{J}(k) + r(k) - \hat{J}(k-1)) \tag{24}$$

D. Weight Updating for the Action Network

The basis for adapting the action NN is to track the desired trajectory and to lower the cost function. Therefore, the error for the action NN can be formed by using the functional estimation error $\zeta_a(k)$, and the error between the nominal desired long-term cost function $J_d(k) \in R$ and the critic signal $\hat{J}(k)$. Now we define the cost function vector as $\bar{J}(k) = \begin{bmatrix} \hat{J}(k) & \hat{J}(k) & ... & \hat{J}(k) \end{bmatrix}^T \in R^{m \times 1}$. Let

$$\begin{aligned} e_a(k) &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1}(\bar{J}(k) - J_d(k)) \\ &= \sqrt{g(x(k))}\zeta_a(k) + \left(\sqrt{g(x(k))}\right)^{-1}\bar{J}(k) \end{aligned} \tag{25}$$

where $\zeta_a(k)$ is defined in (12). Given Assumption 1, we define $\sqrt{g(x(k))} \in R^{m \times m}$ as the principle square root of the diagonal positive definite matrix $g(x(k))$, i.e., $\sqrt{g(x(k))} \times \sqrt{g(x(k))} = g(x(k))$, and $\left(\sqrt{g(x(k))}\right)^T = \sqrt{g(x(k))}$ [14]. The desired long-term cost function $J_d(k)$ is nominally defined and is considered to be zero (“0”), which means as low as possible.

Hence, the weights of the action NN $\hat{w}_a(k)$ are tuned to minimize the error

$$E_a(k) = \frac{1}{2} e_a^T(k) e_a(k) \tag{26}$$

By combining (10), (12), (14), (25), (26) and utilizing the chain rule, one can obtain

$$\begin{aligned}\Delta\hat{w}_a(k) &= -\alpha_a \frac{\partial E_a(k)}{\partial \hat{w}_a(k)} = -\alpha_a \frac{\partial E_a(k)}{\partial e_a(k)} \frac{\partial e_a(k)}{\partial \zeta_a(k)} \frac{\partial \zeta_a(k)}{\partial \hat{w}_c(k)} \\ &= -\alpha_a \phi_a(s(k)) \left(g(x(k))\zeta_a(k) + \bar{J}(k) \right)^T \\ &= -\alpha_a \phi_a(s(k)) \left(e_n(k+1) - l_1 e_n(k) - d_a(k) + \bar{J}(k) \right)^T \end{aligned} \tag{27}$$

where $\alpha_a \in R^+$ is the adaptation gain of the action NN. However, $d_a(k)$ is typically unavailable, so as in the ideal case, we take it as zero and obtain the weight updating algorithm for the action NN as

$$\hat{w}_a(k+1) = \hat{w}_a(k) - \alpha_a \phi_a(s(k)) \left(e_n(k+1) - l_1 e_n(k) + \bar{J}(k) \right)^T \tag{28}$$

E. Theoretic Result

Assumption 4: Let w_a and w_c be the unknown output layer target weights for the action and critic NNs, respectively, and assume that they are upper bounded such that

$$\|w_a\| \le w_{am}, \text{ and } \|w_c\| \le w_{cm} \tag{29}$$

where $w_{am} \in R^+$ and $w_{cm} \in R^+$ represent the bounds on the unknown target weights.

Fact 1: The activation functions for the action and critic NNs are bounded by known positive values, such that

$$\|\phi_a(k)\| \le \phi_{am}, \ \|\phi_c(k)\| \le \phi_{cm} \tag{30}$$

where $\phi_{am}, \phi_{cm} \in R^+$ is the upper bound for the activation functions.

Assumption 5: The NN approximation errors $\varepsilon_a(s(k))$ and $\varepsilon_c(x(k))$ are bounded above over the compact set $S \subset R^m$ by ε_{am} and ε_{cm} [11].

Fact 2: With the Assumption 1 and 4, the term $d_a(k)$ in (13) is bounded over the compact set $S \subset R^m$ by

$$\|d_a(k)\| \le d_{am} = g_{\max}\varepsilon_{am} + d_m \tag{31}$$

Combining Assumption 1, 3, and 4 and Facts 1, and 2, following theorem is introduced.

Theorem 1: Consider the system given by (1) with all system states measurable. Let the Assumptions 1 through 5 hold with the disturbance bound d_m a known constant. Let the control input be provided by the action NN (9), with the critic NN (16). Further, let the weights of the action NN and the critic NN be tuned by (23) and (27), respectively. Then the tracking error $e(k)$, and the NN weight estimates of the action and critic NNs, $\hat{w}_a(k)$ and $\hat{w}_c(k)$ are uniformly-ultimately-bounded (UUB), provided that the controller design parameters satisfy

$$0 < \alpha_a \left\|\phi_a(k)\right\|^2 < g_{\min} / g_{\max}^2 \tag{32}$$

$$0 < \alpha_c \gamma^2 \left\|\phi_c(k)\right\|^2 < 1 \tag{33}$$

$$0 < l_{1\max} < \sqrt{3}/3 \tag{34}$$

$$\gamma > 1/2 \tag{35}$$

where α_a and α_c are NN adaptation gains, γ is employed to define the strategic utility function and $l_{1\max} \in R^+$ is the largest eigenvalue of square matrix l_1.

Proof: See Appendix A.

Remark 1: The proposed scheme results in a well-defined controller since a single NN is utilized to approximate two nonlinear functions.

Remark 2: The action and critic NN weights can be initialized at zero or random. This means that there is no explicit off-line learning phase needed.

Remark 3: It is important to note that persistency of excitation condition is not utilized and certainty equivalence principle is not employed, in contrast to standard work in discrete-time adaptive control [7]. In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption, and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function shown in the appendix weighs the tracking errors, $e(k)$, the weight estimation errors of all the NNs for the controller, $\tilde{W}(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected *a priori* in an ad hoc manner.

Remark 4: In this work, two-layer NNs are utilized as online approximators for action and critic network signals whereas any other online approximators such as CMAC, splines, fuzzy logic, and so on can be utilized instead. Lyapunov proof of the controller convergence still holds.

IV. OUTPUT FEEDBACK ONLINE REINFORCEMENT LEARNING CONTROLLER DESIGN

In the state feedback design, all states are assumed to be available for the controller. However, in this section, the output feedback version of our online reinforcement learning scheme is introduced when certain states of the plant are unavailable.

A. Observer Structure

Consider system (1), assuming that only the output vector $y(k) \in R^m$ is available at the k^{th} step. Therefore, to estimate other system states, a NN observer is first proposed.

For the system described by (1), we use the following NN-based state observer to estimate the actual state $x(k)$ as

$$\begin{aligned}
&\hat{x}_1(k) = \hat{x}_2(k-1) \\
&\qquad \vdots \\
&\hat{x}_n(k) = \hat{w}_o^T(k-1)\phi_o(v_o^T\hat{z}(k-1)) = \hat{w}_o^T(k-1)\phi_o\left(\hat{z}\left(k-1\right)\right)
\end{aligned} \tag{36}$$

where $\hat{x}_i\left(k\right) \in R^m,\ i = 1,\cdots,n$ is the estimated value of $x_i\left(k\right) \in R^m$, and $\hat{z}(k-1) = \left[\hat{x}_1^T(k-1),\ldots,\hat{x}_n^T(k-1),u^T(k-1)\right]^T \in R^{(n+1)m}$ is the input vector to the NN observer at the kth instant, $\hat{w}_o(k-1) \in R^{n_o \times m}$ and $v_o \in R^{(n+1)m \times n_o}$ denote the output and hidden layer weights of the NN observer, and n_o is the number of the hidden layer neurons. For simplicity purpose, the hidden layer activation function vector $\phi_o(v_o^T\hat{z}(k-1)) \in R^{n_o}$ is written as $\phi_o\left(\hat{z}\left(k-1\right)\right)$.

Define the state estimation error as

$$\tilde{x}_i(k) = \hat{x}_i(k) - x_i(k),\ i = 1,\ldots,n \tag{37}$$

where $\tilde{x}_i(k) \in R^m,\ i = 1,\ldots,n$ is the state estimation error. As a matter of fact, by comparing (1) and (36), one can find that the observer NN approximates the nonlinear function given by $f(x(k-1)) + g(x(k-1))u(k-1)$. Thus, ideally this nonlinear function can be expressed as

$$\begin{aligned}
f(x(k-1)) + g(x(k-1))u(k-1) &= w_o^T\phi_o(v_o^Tz(k-1)) + \varepsilon_o(z(k-1)) \\
&= w_o^T\phi_o(z(k-1)) + \varepsilon_o(z(k-1))
\end{aligned} \tag{38}$$

where $w_o \in R^{n_o \times m}$ is the target observer NN weight matrix, $\varepsilon_o(z(k-1))$ is the NN approximation error, and the NN input is given by

$z(k-1)=\left[x_1^T(k-1),\ldots,x_n^T(k-1),u^T(k-1)\right]^T \in R^{(n+1)m}$. Again, for convenience, the hidden layer activation function vector $\phi_o(v_o^T z(k-1)) \in R^{n_o}$ is written as $\phi_o(z(k-1))$.

Combining (36), (37) and (38), one obtains

$$\begin{aligned}
\tilde{x}_n(k) &= \hat{x}_n(k) - x_n(k) \\
&= \hat{x}_n(k) - f\left(x(k-1)\right) + g\left(x(k-1)\right)u(k-1) - d(k-1) \\
&= \left(\hat{w}_o^T(k-1) - w_o^T\right)\phi_o\left(\hat{z}(k-1)\right) + w_o^T\left(\phi_o\left(\hat{z}(k-1)\right) - \phi_o\left(z(k-1)\right)\right) \\
&\quad -\varepsilon_o\left(z(k-1)\right) - d\left(k-1\right) \\
&= \tilde{w}_o^T(k-1)\phi_o\left(\hat{z}(k-1)\right) + w_o^T\phi_o\left(\tilde{z}(k-1)\right) - \varepsilon_o\left(z(k-1)\right) - d(k-1) \\
&= \xi_o(k-1) + d_o(k-1)
\end{aligned} \tag{39}$$

where

$$\tilde{w}_o(k-1) = \hat{w}_o(k-1) - w_o \tag{40}$$

$$\xi_o(k-1) = \tilde{w}_o^T(k-1)\phi_o(\hat{z}(k-1)) \tag{41}$$

$$\tilde{\phi}_o(z(k-1)) = \phi_o(\hat{z}(k-1)) - \phi_o(z(k-1)) \tag{42}$$

$$d_o(k-1) = w_o^T\tilde{\phi}_o(z(k-1)) - \varepsilon_o(z(k-1)) + d(k-1) \tag{43}$$

Therefore, the dynamics of the estimation error is obtained using (37) and (39) as

$$\begin{aligned}
\tilde{x}_1(k) &= \tilde{x}_2(k-1) \\
&\vdots \\
\tilde{x}_n(k) &= \xi_o(k-1) + d_o(k-1)
\end{aligned} \tag{44}$$

B. Action and Critic Network Design

Since some of the actual system states are unavailable for the action and critic NNs, their input and updating rules have to be changed accordingly. The basic idea is to substitute the unavailable system states with the corresponding estimated values from the observer NN. Consequently, the action NN input is taken as

$\hat{s}(k) = \left[\hat{x}^T(k), y_d^T(k), y_d^T(k+n) \right]^T \in R^{(n+2)m}$, while the input to the critic NN is replaced by $\hat{x}(k)$. Thus, in our output feedback design, the control input to the plant is provided as

$$u(k) = \hat{w}_a^T(k)\phi_a(v_a^T \hat{s}(k)) = \hat{w}_a^T(k)\phi_a(\hat{s}(k)) \tag{45}$$

The long term cost function is approximated by

$$\hat{J}(k) = \hat{w}_c^T(k)\phi_c\left(v_c^T \hat{x}(k)\right) = \hat{w}_c^T(k)\phi_c\left(\hat{x}(k)\right) \tag{46}$$

Meanwhile, the training algorithms governing action and critic NNs are updated as

$$\hat{w}_a(k+1) = \hat{w}_a(k) - \alpha_a \phi_a\left(\hat{s}(k)\right)\left(\hat{e}_n(k+1) - l_1 \hat{e}_n(k) + \bar{J}(k)\right)^T \tag{47}$$

$$\hat{w}_c(k+1) = \hat{w}_c(k) - \alpha_c \gamma \phi_c\left(\hat{x}(k)\right)\left(\gamma \hat{J}(k) + r(k) - \hat{J}(k-1)\right) \tag{48}$$

where $\hat{e}_n(k) = \hat{x}_n(k) - x_{nd}(k)$.

Thus, (10) has to be rewritten as

$$\begin{aligned} e_n(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) - y_d(k+n) \\ &= l_1 e_n(k) + g(x(k))\left(u(k) - u_d(k)\right) + d(k) \\ &= l_1 e_n(k) + g(x(k))(\tilde{w}_a^T(k)\phi_a(\hat{s}(k)) - \varepsilon_a(\hat{s}(k))) + d(k) \\ &= l_1 e_n(k) + g(x(k))\hat{\zeta}_a(k) + \hat{d}_a(k) \end{aligned} \tag{49}$$

where

$$\hat{\zeta}_a(k) = \tilde{w}_a^T(k)\phi_a\left(\hat{s}(k)\right) \tag{50}$$

$$\hat{d}_a(k) = -g\left(x(k)\right)\varepsilon_a\left(\hat{s}(k)\right) + d(k) \tag{51}$$

C. Weight Updating for the Observer NN

The observer NN weight update is driven by the state estimation error $\tilde{x}_1(k) = \hat{x}_1(k) - y(k)$, i.e.,

$$\hat{w}_o(k+1) = \hat{w}_o(k) - \alpha_o \phi_o(\hat{z}(k)) \left(\hat{w}_o^T(k) \phi_o(\hat{z}(k)) + l_2 \tilde{x}_1(k) \right)^T \tag{52}$$

where $l_2 \in R^{m \times m}$ is a design matrix, and $\alpha_o \in R^+$ is the adaptation gain for the NN observer.

D. Theoretic Result

Theorem 2: Consider the system given by (1) with only the output available. Let the Assumptions 1 through 5 hold (Assumption 4 and 5 also hold for the observer NN) with the disturbance bound d_m a known constant. Let system states be estimated by observer NN (36), the control input be provided by the action NN (45), with the critic NN (46) tuning the action NN weights. Further, let the observer, action and critic NN weights be tuned by (52), (47) and (48), respectively. Then the tracking error $e(k)$, and the observer, action and critic NN weights, $\hat{w}_o(k)$, $\hat{w}_a(k)$ and $\hat{w}_c(k)$ are uniformly ultimately bounded, with the bounds specifically given by (A.26) through (A.30) provided (32)-(34) and following additional conditions

$$0 < \alpha_o \left\| \phi_o(k) \right\|^2 < 1 \tag{53}$$

$$\gamma > \sqrt{3}/3 \tag{54}$$

Proof: See Appendix B.

Remark 1: The proposed output feedback controller scheme results in a well-defined controller since a single NN is utilized to approximate two nonlinear functions.

Remark 2: The observer, action and critic NN weights can be initialized at zero or random. This means that there is no explicit off-line learning phase needed.

Remark 3: It is important to note that persistency of excitation condition is not utilized and certainty equivalence principle is not employed, in contrast to standard work in discrete-time adaptive control [7]. In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption, and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function shown in the appendix weighs the tracking errors, $e(k)$, the weight estimation errors of all the NNs, $\tilde{W}(k)$ including the observer. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected *a priori* in an ad hoc manner.

Remark 4: In this work, two-layer NNs are utilized as online approximators for observer, action and critic network signals whereas any other online approximators such as CMAC, splines, fuzzy logic, and so on can be utilized instead. Lyapunov proof of the controller convergence still holds.

Remark 5: Since separation principle does not hold for nonlinear systems, the proposed output feedback controller relaxes this strong assumption since Lyapunov proof includes observer estimation error and weight estimation error terms along with the action and critic network terms.

V. SIMULATION RESULTS

To demonstrate the feasibility of the theoretic results, the on-line learning controller design is implemented on a pendulum balancing system and a two-link robotic arm system by simulation.

A. Pendulum Balancing System

First, our approach is examined on a pendulum swing up and balancing task. The example under investigation is identical to that in [6] and [25]. The continuous-time dynamics of the pendulum can be written as follows,

$$\begin{aligned} \frac{d\omega}{dt} &= \frac{3}{4ml^2}(F + m\lg\sin\theta) \\ \frac{d\theta}{dt} &= \omega \end{aligned} \tag{55}$$

where $m = 1/3$ and $l = 3/2$ are the mass and length of the pendulum, respectively. The original system states include the angle θ and angular velocity ω . In the implementation, the system dynamics are discretized with standard zero-order-hold technique presented in [27]. The time step is taken to be 0.05.

The task requires that the controller swings up the bar and balances it at the vertical position. Initially, the pendulum starts at $\theta = \pi$, which means the bar is released loosely straight down. Further, a bounded uniformly distributed noise on [-0.02, 0.02] is introduced with bound $d_m = 0.02$. The other design and simulation parameters are set as following

Table 1 Summary of Parameters Used in Simulation of Pendulum

Parameter	R	Q	γ	l_1	l_2	α_o
Value	0.1	0.1	0.5	0.1	0.5	0.8
Parameter	α_c	α_a	n_o	n_a	n_c	d_m
Value	0.01	0.1	5	10	10	0.02

In our study 100 consecutive trials for both state feedback and output feedback designs are attempted and the task is successfully accomplished for every trial. Fig. 3

shows the simulation results of state feedback controller for one of the trials and Fig. 4 shows that of output feedback controller in terms of θ and ω. Also these symbols should be expressed in the plots.

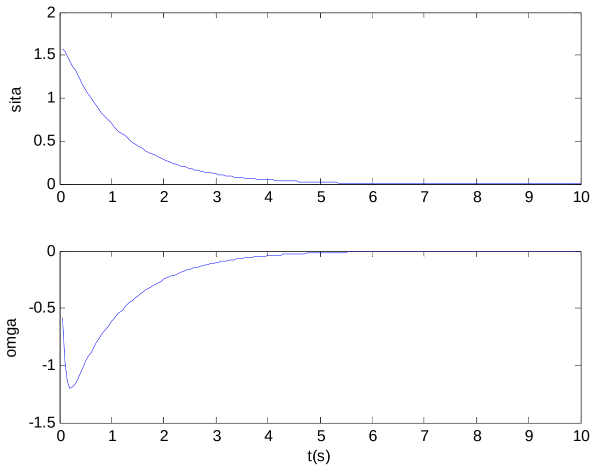


Fig 3. Simulation Results of the State Feedback Online Learning Controller on Pendulum Balancing System

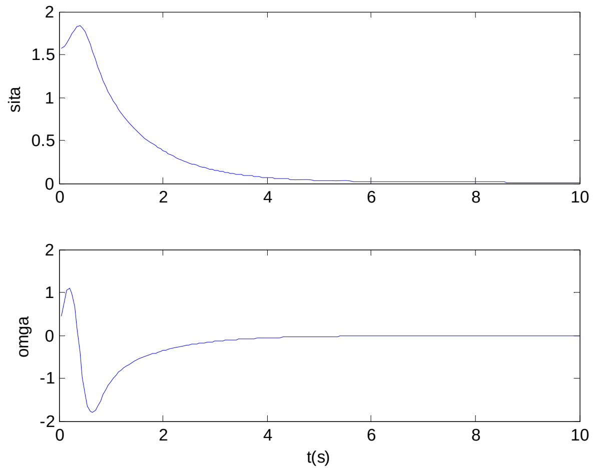


Fig 4. Simulation Results of the Output Feedback Online Learning Controller on Pendulum Balancing System

B. 2-link Planar Robot Arm System

In the second implementation, the 2-link planar robot arm system shown in Fig. 5 and discussed in [5] [24] is considered.

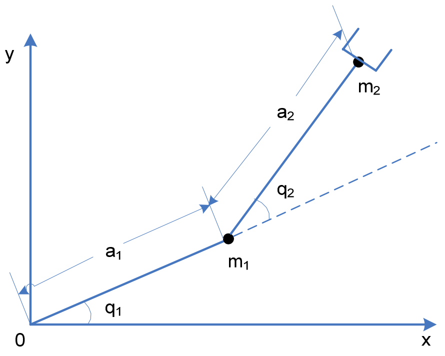


Fig 5. Geometry of a Two-link Planar Robot Arm

The continuous-time manipulator dynamics is as follows [5]

$$\begin{bmatrix} \alpha+\beta+2\eta\cos q_2 & \beta+\eta\cos q_2 \\ \beta+\eta\cos q_2 & \beta \end{bmatrix}\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}+\begin{bmatrix} -\eta(2\dot{q}_1\dot{q}_2+\dot{q}_2^2)\sin q_2 \\ \eta\dot{q}_1^2\sin q_2 \end{bmatrix}$$
$$+\begin{bmatrix} \alpha e_1\cos q_1+\eta e_1\cos(q_1+q_2) \\ \eta e_1\cos(q_1+q_2) \end{bmatrix}=\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{56}$$

where $\alpha=(m_1+m_2)a_1^2$, $\beta=m_2a_2^2$, $\eta=m_2a_1a_2$, $e_1=g/a_1$.

g	$9.8\,m/s^2$, the acceleration of gravity;
m_1, m_2	point mass of the links at distal end;
a_1, a_2	length of the links;
q_1, q_2	rotational angle of the joints;
τ_1, τ_2	torque applied on the joints.

In most of the controller designs, joint angles q_1 and q_2 are the states while τ_1 and τ_2 are the control input. After using the same technique as that of the pendulum example for discretization, the system dynamics in discrete-time can be written in an affine form as (1).

The simulation parameters used in this simulation are tabulated as below:

Table 2 Summary of Parameters Used in Simulation of 2-link Robot Arm

Parameter	m_1	m_2	a_1	a_2	R	Q	γ	d_m
Value	0.8	2.3	1	1	2	1	0.5	0.1
Parameter	l_1	l_2	n_o	n_a	n_c	α_o	α_c	α_a
Value	0.1	0.5	5	10	10	0.01	1×10^{-4}	1×10^{-5}

In the simulation, the time step is set as 1 ms. To be more realistic, the system is also added with a bounded random disturbance (give more details). The initial states of the system are set at $q_1(0) = q_2(0) = 10^\circ$. Our goal is to manipulate the robot arm back to zero with the lowest cost and simulation will be stopped when the rotational angles converge to zero.

A typical system response using state feedback online learning controller is shown in Fig. 6, while Fig. 7 depicts the system response with output feedback version. From the simulation results, we can find that both the designs are capable of accomplishing the control targets.

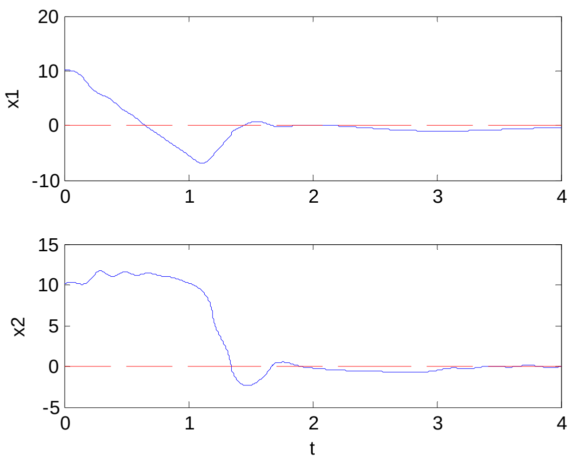


Fig 6. Simulation Results of the State Feedback Online Learning Controller on 2-link Planar Robot Arm. Solid Line: Trajectories of the Rotational Angles; Dashed Line: Desired Final Values of the Angles.

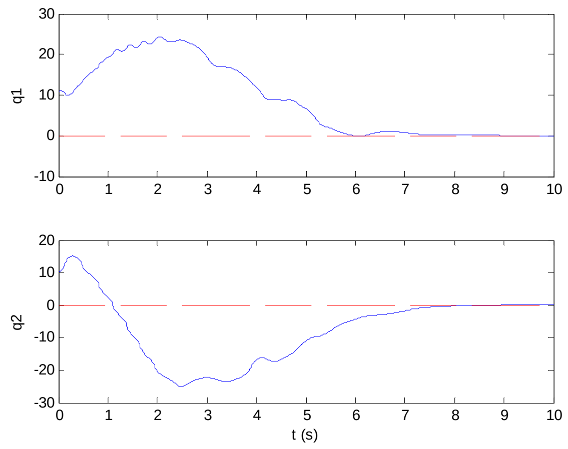


Fig 7. Simulation Results of the Output Feedback Online Learning Controller on 2-link Planar Robot Arm. Solid Line: Trajectories of the Rotational Angles; Dashed Line: Desired Final Values of the Angles.

VI. CONCLUSIONS

A novel reinforcement learning-based online neural controller is designed for affine nonlinear systems to deliver a desired performance under bounded disturbance.

Depending on the availability of system states, both state feedback and output feedback version are introduced in this paper. The proposed NN controller optimizes the long-term cost function by introducing a critic NN. Unlike the many applications where the controller is trained offline, the control input is updated in an online fashion. To guarantee that a control system must be stable all of the time, the boundedness of the closed-loop tracking errors and NN weight estimates is verified by using Lyapunov analysis in the presence of bounded disturbances and approximation errors. The observer estimates unavailable system states in the output feedback design. Persistency of excitation condition, certainty equivalence and separation principles are not required. The feasibility of the two methods is also strengthened through the controller implementations on pendulum and 2-link robotic arm.

VII. REFERENCES

1. R. Bellman and S. Dreyfus, “Applied Dynamic Programming,” Princeton, NJ: Princeton Univ. Press, 1962.
2. Z. V. Rekasius, “Suboptimal design of intentionally nonlinear controllers,” IEEE Transactions on Automatic Control, vol. 9, no. 4, pp. 380--386, 1964.
3. R. J. Leake and Ruey-wen Liu, “Construction of suboptimal control sequences,” SIAM J. Control and Optimization vol. 5, No. 1, pp. 54-63, 1967.
4. D. Kirk, “Optimal Control Theory: An Introduction,” Englewood Cliffs, NJ: Prentice-Hall, 1970.
5. F. L. Lewis, S. Jagannathan, and A. Yesildirek, “Neural Network Control of Robot Manipulators and Nonlinear Systems,” Taylor & Francis, PA, 1999.

6. J. Si and Y. T. Wang, “On-line learning control by association and reinforcement,” IEEE Trans. Neural Networks, vol. 12, no. 2, pp. 264–276, Mar.2001.

7. K. J. Astrom, and B. Wittenmark, “Adaptive Control,” Addison Wesley, Reading, Massachusetts, 2nd ed. 1994

8. R. Luus, “Iterative Dynamic Programming,” CRC Press, Boca Raton, FL, 2000.

9. A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuron like adaptive elements that can solve difficult learning control problems,” IEEE Trans. Syst., Man, Cybern., vol. 13, pp. 834–847, 1983.

10. R. S. Sutton, “Learning to predict by the methods of temporal difference,” Machine Learning, vol. 3, pp. 9–44, 1988.

11. C. J. C. H. Watkins and P. Dayan, “Q-learning,” Machine Learning, vol. 8, pp. 257–277, 1992.

12. R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” the MIT Press, Cambridge, MA, 1998.

13. G. Boone, “Efficient reinforcement learning: Model-based acrobot control,” in Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 229 - 234, Albuquerque, NM, 1997.

14. P. He and S. Jagannathan, “Reinforcement learning-based output feedback control of nonlinear systems with input constraints,” IEEE Trans. Syst., Man, Cybern., vol. 35, pp. 150–154, 2005.

15. Y. Kim and F.L. Lewis, “Optimal design of CMAC neural network controller for Robot Manipulators,” IEEE Trans. Systems, Man, and Cybernetics, vol. 30, no. 1, pp. 22-31, Feb 2000.

16. D. P. Bertsekas, "Dynamic Programming and Optimal Control. Belmont," MA: Athena Scientific, 2000.

17. J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds., "Handbook of Learning and Approximate Dynamic Programming," Wiley-IEEE Press, 2004.

18. B. Igelnik and Y. H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," IEEE Trans. Neural Network, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

19. D. Prokhorov and D. Wunsch, "Adaptive critic designs," IEEE Trans. Neural Networks, Vol. 8, No.5, p.997-1007.

20. W. T. Miller, R. S. Sutton, and P. J. Werbos, Eds., "Neural Networks for Control," Cambridge, MA, MIT Press, 1990.

21. P. J. Werbos, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," IEEE Transactions on Systems, Man, and Cybernetics 17, pp. 7–20, 1987.

22. P. Werbos, "Advanced forecasting methods for global crisis warning and models of intelligence," General System Yearbook, vol. 22, pp. 25–38, 1977.

23. T. Borgers and R. Sarin, "Learning through reinforcement and replicator dynamics," J. Economic Theory, vol. 77, no. 1, pp. 1–17, 1997.

24. F. L. Lewis, C. T. Abdallah, and D. M. Dawson, "Control of Robot Manipulators," New York, McMillan, 1993.

25. "COINS Tech. Rep.," Univ. Mass., Amherst, 96–88, Dec. 1996.

26. S. Jagannathan, "Neural Network Control of Nonlinear Discrete-time Systems," Taylor and Francis (CRC Press), Boca Raton, FL 2006.

27. F. L. Lewis, “Applied Optimal Control and Estimation,” Prentice-Hall, New Jersey, 1992.

28. M. Krstic, I. Kanellakopoulos, and P. Kokotovic, “Nonlinear and Adaptive Control Design,” New York: Wiley, 1995.

29. J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, “Adaptive dynamic programming,” IEEE Transactions on. Systems, Man, and Cybernetics–Part C, Vol. 32, pp. 140-153, 2002

Appendix A: Proof of Theorem 1

Proof: Consider the following Lyapunov candidate

$$\begin{aligned} L_s(k) &= \sum_{i=1}^{4} L_{si} \\ &= \frac{\gamma_1}{3}\left\|e_n(k)\right\|^2 + \frac{\gamma_2}{\alpha_a} tr\left(\tilde{W}_a^T(k)\tilde{W}_a(k)\right) + \frac{\gamma_3}{\alpha_c} tr\left(\tilde{W}_c^T(k)\tilde{W}_c(k)\right) + \gamma_4\left\|\zeta_c(k-1)\right\|^2 \end{aligned} \tag{A.1}$$

where $\gamma_i \in R^+$, $i = 1,2,3,4$ are design parameters. Hence, the first difference of the Lyapunov function is given by

$$\begin{aligned} \Delta L_{s1} &= \frac{\gamma_1}{3}\left(\left\|e_n(k+1)\right\|^2 - \left\|e_n(k)\right\|^2\right) \\ &= \frac{\gamma_1}{3}\left(\left\|le_n(k) + g(x(k))\zeta_a(k) + d_a(k)\right\|^2 - \left\|e_n(k)\right\|^2\right) \\ &\le -\frac{\gamma_1}{3}\left(1 - 3l_{\max}^2\right)\left\|e_n(k)\right\|^2 + \gamma_1 g_{\max}^2\left\|\zeta_a(k)\right\|^2 + \gamma_1\left\|d_a(k)\right\|^2 \end{aligned} \tag{A.2}$$

$$\begin{aligned}
\Delta L_{s2} &= \frac{\gamma_2}{\alpha_a} tr\left(\tilde{W}_a^T(k+1)\tilde{W}_a(k+1) - \tilde{W}_a^T(k)\tilde{W}_a(k)\right) \\
&= \frac{\gamma_2}{\alpha_a} tr\left(-2\tilde{W}_a^T(k)\alpha_a\phi_a(s(k))(g(x(k))\zeta_a(k) + \bar{J}(k) + d_a(k)) + \Delta\hat{W}_a^T(k)\Delta\hat{W}_a(k)\right) \\
&\le -2\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2 - 2\gamma_2\zeta_a^T(k)(\bar{J}(k) + d_a(k)) + \gamma_2\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\left\|\zeta_a(k)\right\|^2 \\
&\quad +\gamma_2\alpha_a\left\|\phi_a(s(k))\right\|^2 \times\left(\left\|\bar{J}(k) + d_a(k)\right\|^2 + 2\gamma_2\alpha_a\left(\bar{J}(k) + d_a(k)\right)^T g(x(k))\zeta_a(k)\right) \qquad \text{(A.3)} \\
&= \gamma_2\left\{-g_{\min}\left\|\zeta_a(k)\right\|^2 - \left\|\zeta_a(k) + \frac{(I - \alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k)))}{g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2 \times\right. \\
&\quad \left.(g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2) + \frac{1 - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\min}}{g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\left\|\bar{J}(k) + d_a(k)\right\|^2\right\}
\end{aligned}$$

Set $\gamma_2 = \gamma_2'\gamma_2''$, where $\gamma_2'' \dfrac{1 - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\min}}{g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2} \le \dfrac{1}{2}$, therefore,

$$\begin{aligned}
\Delta L_{s2} &\le -\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2 - \gamma_2\left\|\zeta_a(k) + \frac{(I - \alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k)))}{g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2 \\
&\quad \times\left(g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right) + \gamma_2'\frac{\left\|\bar{J}(k) + d_a(k)\right\|^2}{2} \qquad \text{(A.4)} \\
&\le -\gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2 - \gamma_2\left\|\zeta_a(k) + \frac{(I - \alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k)))}{g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2 \\
&\quad \times\left(g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right) + \gamma_2' n\left\|\zeta_c(k)\right\|^2 + \gamma_2' n\left\|J^*(k) + d_a(k)\right\|^2
\end{aligned}$$

At the same time,

$$
\begin{aligned}
\Delta L_{s3} &= \frac{\gamma_3}{\alpha_c} tr\left(\tilde{W}_c^T(k+1)\tilde{W}_c(k+1) - \tilde{W}_c^T(k)\tilde{W}_c(k)\right) \\
&= \frac{\gamma_3}{\alpha_c} tr\left(-2\tilde{W}_c^T(k)\alpha_c\gamma\phi_c(x(k))e_c(k)\right) + \frac{\gamma_3}{\alpha_c} tr\left(\Delta\hat{W}_c^T(k)\Delta\hat{W}_c(k)\right) \\
&= -2\gamma_3 e_c(k)\left(e_c(k) - \gamma J^*(k) + \zeta_c(k-1) + J^*(k-1) - r(k) + \varepsilon_c(k) - \varepsilon_c(k-1)\right) \\
&\quad + \gamma_3\alpha_c\gamma^2 e_c^2(k)\left\|\phi_c(k)\right\|^2 \\
&= -\gamma_3\left(1 - \alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k) - \gamma_3\gamma^2\zeta_c^2(k) \\
&\quad + \gamma_3\left(\gamma J^*(k) - \zeta_c(k-1) - J^*(k-1) + r(k) - \varepsilon_c(k) + \varepsilon_c(k-1)\right)^2 \\
&\le -\gamma_3\left(1 - \alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k) - \gamma_3\gamma^2\zeta_c^2(k) + \frac{\gamma_3}{4}\zeta_c^2(k-1) \\
&\quad + \frac{\gamma_3}{4}\left(\gamma J^*(k) - J^*(k-1)\right)^2 + \frac{\gamma_3}{4}r(k) + \frac{\gamma_3}{4}\left(\varepsilon_c(k) - \varepsilon_c(k-1)\right)^2 \\
&\le -\gamma_3(1 - \alpha_c\gamma^2\left\|\phi_c(k)\right\|^2)e_c^2(k) - \gamma_3\gamma^2\zeta_c^2(k) + \frac{\gamma_3}{4}\zeta_c^2(k-1) + \frac{\gamma_3}{4}(\gamma J^*(k) - J^*(k-1))^2 \\
&\quad + \frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2 + \frac{\gamma_3}{4}\left(\zeta_a(k) + w_a{}^T\phi_a(k)\right)^T R\left(\zeta_a(k) + w_a{}^T\phi_a(k)\right) + \gamma_3\varepsilon_{cm}{}^2 \\
&\le -\gamma_3(1 - \alpha_c\gamma^2\left\|\phi_c(k)\right\|^2)e_c^2(k) - \gamma_3\gamma^2\zeta_c^2(k) + \frac{\gamma_3}{4}\zeta_c^2(k-1) + \frac{\gamma_3}{4}(\gamma J^*(k) - J^*(k-1))^2 \\
&\quad + \frac{\gamma_3}{4}Q_{\max}\left\|e(k)\right\|^2 + \frac{\gamma_3}{8}R_{\max}\left\|\zeta_a(k)\right\|^2 + \frac{\gamma_3}{8}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2 + \gamma_3\varepsilon_{cm}{}^2
\end{aligned} \tag{A.5}
$$

where $Q_{\max}$ and $R_{\max}$ are the maximum eigenvalue of matrix Q and R, respectively.

$$\Delta L_{s4} = \gamma_4\left(\left\|\zeta_c(k)\right\|^2 - \left\|\zeta_c(k-1)\right\|^2\right) \tag{A.6}$$

Combining (A.1) - (A.6) yields

$$\begin{aligned}
\Delta L_s(k) \le & -\frac{\gamma_1}{3}\left(1-3l_{\max}^2\right)\left\|e_n(k)\right\|^2 + \gamma_1 g_{\max}^2\left\|\zeta_a(k)\right\|^2 + \gamma_1\left\|d_a(k)\right\|^2 - \gamma_2 g_{\min}\left\|\zeta_a(k)\right\|^2 \\
& -\gamma_2\left(g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right)\times\left\|\zeta_a(k) + \frac{\left(I-\alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2 \\
& +\gamma_2' n\left\|\zeta_c(k)\right\|^2 + \gamma_2' n\left\|J^*(k)+d_a(k)\right\|^2 - \gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k) - \gamma_3\gamma^2\zeta_c^2(k) \\
& +\frac{\gamma_3}{4}\zeta_c^2(k-1) + \frac{\gamma_3}{4}\left(\gamma J^*(k) - J^*(k-1)\right)^2 + \frac{\gamma_3}{4}Q_{\max}\left\|e_n(k)\right\|^2 + \frac{\gamma_3}{8}R_{\max}\left\|\zeta_a(k)\right\|^2 \\
& +\frac{\gamma_3}{8}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2 + \gamma_4\left(\left\|\zeta_c(k)\right\|^2 - \left\|\zeta_c(k-1)\right\|^2\right) + \gamma_3\varepsilon_{cm}{}^2 \\
= & -\left(\frac{\gamma_1}{3}\left(1-3l_{\max}^2\right) - \frac{\gamma_3}{4}Q_{\max}\right)\left\|e_n(k)\right\|^2 - \left(\gamma_2 g_{\min} - \gamma_1 g_{\max}^2 - \frac{\gamma_3}{8}R_{\max}\right)\left\|\zeta_a(k)\right\|^2 \\
& -\left(\gamma_3\gamma^2 - \gamma_2' n - \gamma_4\right)\left\|\zeta_c(k)\right\|^2 - (\gamma_4 - \frac{\gamma_3}{4})\left\|\zeta_c(k-1)\right\|^2 - \gamma_3\left(1-\alpha_c\gamma^2\left\|\phi_c(k)\right\|^2\right)e_c^2(k) \\
& -\gamma_2\left(g_{\min} - \alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2\right)\times\left\|\zeta_a(k) + \frac{\left(I-\alpha_a\left\|\phi_a(s(k))\right\|^2 g(x(k))\right)}{g_{\min}-\alpha_a\left\|\phi_a(s(k))\right\|^2 g_{\max}^2}\right\|^2 + D_M^2
\end{aligned} \tag{A.7}$$

where

$$D_M^2 = \frac{\gamma_2' n}{2}d_a^2(k) + \left(\frac{\gamma_3}{4}+\frac{\gamma_2' n}{2}\right)J_m^2 + \frac{\gamma_3}{6}R_{\max}\left\|w_a{}^T\phi_a(k)\right\|^2 + \gamma_1\left\|d_a(k)\right\|^2 + \gamma_3\varepsilon_{cm}{}^2 \tag{A.8}$$

For the standard Lyapunov analysis, equation (A.7) and (A.8) implies that $\Delta L_s \le 0$ as long as the conditions (32) – (35) are satisfied and following holds

$$\left\|e_n(k)\right\| \ge \frac{2\sqrt{3}D_M}{\sqrt{4\gamma_1\left(1-3l_{\max}^2\right) - 3\gamma_3 Q_{\max}}} \tag{A.9}$$

or

$$\left\|\zeta_a(k)\right\| \le \frac{2\sqrt{2}D_M}{\sqrt{8\gamma_2 g_{\min} - 8\gamma_1 g_{\max}^2 - \gamma_3 R_{\max}}} \tag{A.10}$$

or

$$\left\|\zeta_c(k)\right\| \le \frac{D_M}{\sqrt{\gamma_3\gamma^2 - \gamma_2' n - \gamma_4}} \tag{A.11}$$

According to the standard Lyapunov extension theorem [5], [26], the analysis above demonstrates that the tracking error $\|e_n(k)\|$ and the weights of the estimation errors are UUB. Considering $e_i(k) = x_i(k) - x_{id}(k) = e_n(k-n+i),\ i = 1,...,n$, one can readily conclude that $e_i(k),\ i = 1,...,n-1$ is also UUB. Further, the boundedness of $\|\zeta_a(k)\|$ and $\|\zeta_c(k)\|$ implies that the weight estimations $\|\hat{w}_a(k)\|$ and $\|\hat{w}_c(k)\|$ are also bounded.

Appendix B: Proof of Theorem 2

Proof: The proof of Theorem is similar to that of theorem 1. Since an additional observer NN is introduced to estimate the immeasurable states, we consider following Lyapunov function

$$\begin{aligned} L_o(k) &= \sum_{i=1}^{11} L_{oi} \\ &= \frac{\eta_1}{2}\sum_{i=1}^{n}\|\tilde{x}_i(k-1)\|^2 + \frac{\eta_2}{2}\sum_{i=1}^{n}\|\tilde{x}_i(k)\|^2 + \frac{\eta_3}{3}\sum_{i=1}^{n}\|e_i(k-1)\|^2 + \frac{\eta_4}{3}\sum_{i=1}^{n}\|e_i(k)\|^2 \\ &\quad + \frac{\eta_5}{\alpha_o} tr\left(\tilde{W}_o^T(k-1)\tilde{W}_o(k-1)\right) + \frac{\eta_6}{\alpha_o} tr\left(\tilde{W}_o^T(k)\tilde{W}_o(k)\right) + \frac{\eta_7}{\alpha_a} tr\left(\tilde{W}_a^T(k)\tilde{W}_a(k)\right) \\ &\quad + \frac{\eta_8}{\alpha_c} tr\left(\tilde{W}_c^T(k)\tilde{W}_c(k)\right) + \eta_9\|\zeta_c(k-1)\|^2 + \eta_{10}\|\zeta_c(k-1)\|^2 + \eta_{11}\|e_1(k-1)\|^2 \end{aligned} \tag{A.12}$$

where $\eta_i \in R^+$, $i = 1,...,11$ are design parameters. Hence, the first difference of the Lyapunov function is the summation of the difference for each term.

From (39), we have

$$\begin{aligned} \Delta L_{o1} &= \frac{\eta_1}{2}\sum_{i=1}^{n}\|\tilde{x}_i(k)\|^2 - \frac{\eta_1}{2}\sum_{i=1}^{n}\|\tilde{x}_i(k-1)\|^2 \\ &= \eta_1\|\zeta_o(k-1)\|^2 + \eta_1\|d_o(k-1)\|^2 - \frac{\eta_1}{2}\|\tilde{x}_1(k-1)\|^2 \end{aligned} \tag{A.13}$$

$$\begin{aligned}\Delta L_{o2} &= \frac{\eta_2}{2}\sum_{i=1}^{n}\left\|\tilde{x}_i(k+1)\right\|^2 - \frac{\eta_2}{2}\sum_{i=1}^{n}\left\|\tilde{x}_i(k)\right\|^2 \\ &= \eta_2\left\|\zeta_o(k)\right\|^2 + \eta_2\left\|d_o(k)\right\|^2 - \frac{\eta_2}{2}\left\|\tilde{x}_1(k)\right\|^2\end{aligned} \tag{A.14}$$

(49) yields

$$\Delta L_{o3} \le \eta_3 l_{1\max}^2\left\|e_n(k)\right\|^2 + \eta_3 g_{\max}^2\left\|\hat{\zeta}_a(k)\right\|^2 + \eta_3\left\|\hat{d}_a(k)\right\|^2 - \frac{\eta_3}{3}\left\|e_1(k)\right\|^2 \tag{A.15}$$

$$\Delta L_{o4} \le \eta_4 l_{1\max}^2\left\|e_n(k)\right\|^2 + \eta_4 g_{\max}^2\left\|\hat{\zeta}_a(k)\right\|^2 + \eta_4\left\|\hat{d}_a(k)\right\|^2 - \frac{\eta_4}{3}\left\|e_n(k)\right\|^2 \tag{A.16}$$

Considering (52), we have

$$\begin{aligned}\Delta L_{o5} &= \frac{\eta_5}{\alpha_o} tr\left(\tilde{W}_o^T(k)\tilde{W}_o(k) - \tilde{W}_o^T(k-1)\tilde{W}_o(k-1)\right) \\ &\le -\eta_5\left(1-\alpha_o\left\|\phi_o(k-1)\right\|^2\right)\left\|\zeta_o(k-1) + l_2\tilde{x}_1(k-1) + W_o^T\phi_o(k-1)\right\|^2 \\ &\quad -\eta_5\left\|\zeta_o(k-1)\right\|^2 + 2\eta_5 l_{2\max}^2\left\|\tilde{x}_1(k-1)\right\|^2 + 2\eta_5 w_{om}^2\phi_{om}^2\end{aligned} \tag{A.17}$$

$$\begin{aligned}\Delta L_{o6} &= \frac{\eta_6}{\alpha_o} tr\left(\tilde{W}_o^T(k+1)\tilde{W}_o(k+1) - \tilde{W}_o^T(k)\tilde{W}_o(k)\right) \\ &\le -\eta_6\left(1-\alpha_o\left\|\phi_o(k)\right\|^2\right)\left\|\zeta_o(k) + l_2\tilde{x}_1(k) + W_o^T\phi_o(k)\right\|^2 \\ &\quad -\eta_6\left\|\zeta_o(k)\right\|^2 + 2\eta_6 l_{2\max}^2\left\|\tilde{x}_1(k)\right\|^2 + 2\eta_6 w_{om}^2\phi_{om}^2\end{aligned} \tag{A.18}$$

where $l_{2\max} \in R$ is the maximum eigenvalue of matrix l_2. Similar to (A.4), we obtain

$$\Delta L_{o7} \le -\eta_7'\left(g_{\min} - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g_{\max}^2\right)\left\|\hat{\zeta}_a\left(k\right) + \frac{\left(I - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g(\hat{x}(k))\right)\beta(k)}{g_{\min} - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g_{\max}^2}\right\|^2$$
$$-\eta_7' g_{\min}\left\|\hat{\zeta}_a\left(k\right)\right\|^2 + \frac{\eta_7'}{4}\left\|\beta(k)\right\|^2$$
$$\le -\eta_7'\left(g_{\min} - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g_{\max}^2\right)\left\|\hat{\zeta}_a\left(k\right) + \frac{\left(I - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g(\hat{x}(k))\right)\beta(k)}{g_{\min} - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g_{\max}^2}\right\|^2 \quad (A.19)$$
$$-\eta_7' g_{\min}\left\|\hat{\zeta}_a\left(k\right)\right\|^2 + 2\eta_7'\left\|\zeta_o\left(k\right)\right\|^2 + 2\eta_7'\left\|d_o\left(k\right)\right\|^2 + 2\eta_7' l_{1\max}^2\left\|\zeta_o\left(k-1\right)\right\|^2$$
$$+2\eta_7' l_{1\max}^2\left\|d_o\left(k-1\right)\right\|^2 + \eta_7'\left\|\zeta_c\left(k\right)\right\|^2 + \eta_7'\left\|W_c^T\phi_c\left(\hat{x}\left(k\right)\right) + \hat{d}_a(k)\right\|^2$$

where $\beta(k) = \tilde{x}_n(k+1) - l_1\tilde{x}_n(k) + \hat{Q}\left(k\right) + \hat{d}_a\left(k\right)$ and $\eta_7 = \eta_7'\eta_7''$ such that

$$\eta_7'' \frac{1 - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g_{\min}}{g_{\min} - \alpha_a \left\|\phi_a\left(\hat{s}\left(k\right)\right)\right\|^2 g_{\max}^2} \le \frac{1}{4}.$$

Similar to (A.5)

$$\Delta L_{o8} \le -\eta_8\left(1 - \alpha_c\gamma^2\left\|\phi_c(\hat{x}(k))\right\|^2\right)e_c^2(k) - \eta_8\gamma^2\zeta_c^2(k) + \frac{\eta_8}{3}\zeta_c^2(k-1) + \frac{\eta_8}{3}\left(\gamma J(k) - J(k-1)\right)^2$$
$$+\frac{\eta_8}{3}Q_{\max}\left\|e_1(k-1)\right\|^2 + \frac{2\eta_8}{3}R_{\max}\left\|\hat{\zeta}_a(k-1)\right\|^2 + \frac{2\eta_8}{3}R_{\max}\left\|w_a{}^T\phi_a(\hat{s}(k-1))\right\|^2 \quad (A.20)$$

Furthermore,

$$\Delta L_{o9} = \eta_9\left\|\hat{\zeta}_a(k)\right\|^2 - \eta_9\left\|\hat{\zeta}_a(k-1)\right\|^2 \quad (A.21)$$

$$\Delta L_{o10} = \eta_{10}\left\|\zeta_c(k)\right\|^2 - \eta_{10}\left\|\zeta_c(k-1)\right\|^2 \quad (A.22)$$

$$\Delta L_{o11} = \eta_{11}\left\|e_1(k)\right\|^2 - \eta_{11}\left\|e_1(k-1)\right\|^2 \quad (A.23)$$

Combining (A.13) – (A.23) yields,

$$
\begin{aligned}
\Delta L_o \le & -\left(\eta_6 - \eta_2 - 2\eta_7'\right)\left\|\zeta_o(k)\right\|^2 - \left(\eta_5 - \eta_1 - 2\eta_7' l_{1\max}^2\right)\left\|\zeta_o(k-1)\right\|^2 \\
& -\left(\eta_7' g_{\min} - \eta_4 g_{\max}^2 - \eta_3 g_{\max}^2 - \eta_9\right)\left\|\hat{\zeta}_a(k)\right\|^2 - \left(\eta_9 - \frac{2\eta_8}{3}R_{\max}\right)\left\|\hat{\zeta}_a(k-1)\right\|^2 \\
& -\left(\eta_8\eta^2 - \eta_7' - \eta_{10}\right)\zeta_c^2(k) - \left(\eta_{10} - \frac{\eta_8}{3}\right)\zeta_c^2\left(k-1\right) - \left(\frac{\eta_2}{2} - 2\eta_6 l_{2\max}^2\right)\left\|\tilde{x}_1(k)\right\|^2 \\
& -\left(\frac{\eta_1}{2} - 2\eta_5 l_{2\max}^2\right)\left\|\tilde{x}_1(k-1)\right\|^2 - \left(\frac{\eta_3}{3} - \eta_{11}\right)\left\|e_1(k)\right\|^2 - \left(\eta_{11} - \frac{\eta_8}{3}Q_{\max}\right)\left\|e_1(k-1)\right\|^2 \\
& -\left(\frac{\eta_4}{3} - \eta_4 l_{1\max}^2 - \eta_3 l_{1\max}^2\right)\left\|e_n(k)\right\|^2 - \eta_8\left(1 - \alpha_c\gamma^2\left\|\phi_c\left(\hat{x}(k)\right)\right\|^2\right)e_c^2\left(k\right) \\
& -\eta_7'\left(g_{\min} - \alpha_a\left\|\phi_a(\hat{s}(k))\right\|^2 g_{\max}^2\right)\times\left\|\hat{\zeta}_a\left(k\right) + \frac{\left(I - \alpha_a\left\|\phi_a(\hat{s}(k))\right\|^2 g(\hat{x}(k))\right)\beta(k)}{g_{\min} - \alpha_a\left\|\phi_a(\hat{s}(k))\right\|^2 g_{\max}^2}\right\|^2 \\
& -\eta_6\left(1 - \alpha_o\left\|\phi_o(k)\right\|^2\right)\left\|\zeta_o(k) + l_2\tilde{x}_1(k) + W_o^T\phi_o(k)\right\|^2 \\
& -\eta_5\left(1 - \alpha_o\left\|\phi_o(k-1)\right\|^2\right)\left\|\zeta_o(k-1) + l_2\tilde{x}_1(k-1) + W_o^T\phi_o(k-1)\right\|^2 + D_M^2
\end{aligned}
\tag{A.24}
$$

where

$$
\begin{aligned}
D_M^2 = & (2\eta_5 + 2\eta_6)w_{om}^2\phi_{om}^2 + \frac{2\eta_8}{3}R_{\max}w_{am}^2\phi_{am}^2 + 2\eta_7' w_{cm}^2\phi_{cm}^2 + \frac{\eta_8(2\gamma^2+1)}{3}J_m^2 \\
& +(\eta_1 + \eta_2 + 2\eta_7' + 2\eta_7' l_{1\max}^2)d_{om}^2 + (\eta_3 + \eta_4 2\eta_7')d_{am}^2
\end{aligned}
\tag{A.25}
$$

Referring to the standard Lyapunov analysis [5], [26], equation (A.24) and (A.25) implies that $\Delta L_o \le 0$ as long as the conditions (32) – (34) and (53) – (54) are satisfied and following holds

$$
\left\|\tilde{x}_1(k)\right\| \ge \frac{\sqrt{2}D_M}{\sqrt{\eta_2 - 4\eta_6 l_{2\max}^2}}
\tag{A.26}
$$

or

$$
\left\|e_n(k)\right\| \ge \frac{\sqrt{3}D_M}{\sqrt{\eta_4 - 3\left(\eta_3 + \eta_4\right)l_{1\max}^2}}
\tag{A.27}
$$

or

$$\|\zeta_o(k)\| \geq \frac{D_M}{\sqrt{\eta_6 - \eta_2 - 2\eta_7'}} \tag{A.28}$$

or

$$\left\|\hat{\zeta}_a(k)\right\| \geq \frac{D_M}{\sqrt{\eta_7' g_{\min} - (\eta_3 + \eta_4) g_{\max}^2 - \eta_9}} \tag{A.29}$$

or

$$\|\zeta_c(k)\| \leq \frac{D_M}{\sqrt{\eta_8 \gamma^2 - \eta_7' - \eta_{10}}} \tag{A.30}$$

According to the standard Lyapunov extension theorem [5], [26], the analysis above demonstrates that the tracking error $\|e(k)\|$ and the weights of the estimation errors are UUB. Further, the boundedness of $\|\zeta_o(k)\|$, $\left\|\hat{\zeta}_a(k)\right\|$ and $\|\zeta_c(k)\|$ implies that the weight estimations $\|\hat{w}_o(k)\|$, $\|\hat{w}_a(k)\|$ and $\|\hat{w}_c(k)\|$ are also bounded.

VITA

Qinmin Yang was born in Hengyang, China, on August 1, 1979. He received his Bachlor's degree in Electrical Engineering from Civil Aviation University of China in 1997, and his Master of Science Degree in Electrical Engineering from Institute of Automation, Chinese Academy of Sciences in 2004. At the age of 25, he came to the United States to pursue his dreams. He spent a happy 3 years in Rolla, Missouri and earned his Ph.D. Degree in Elctrical Engineering at the University of Missouri – Rolla in December 2007.