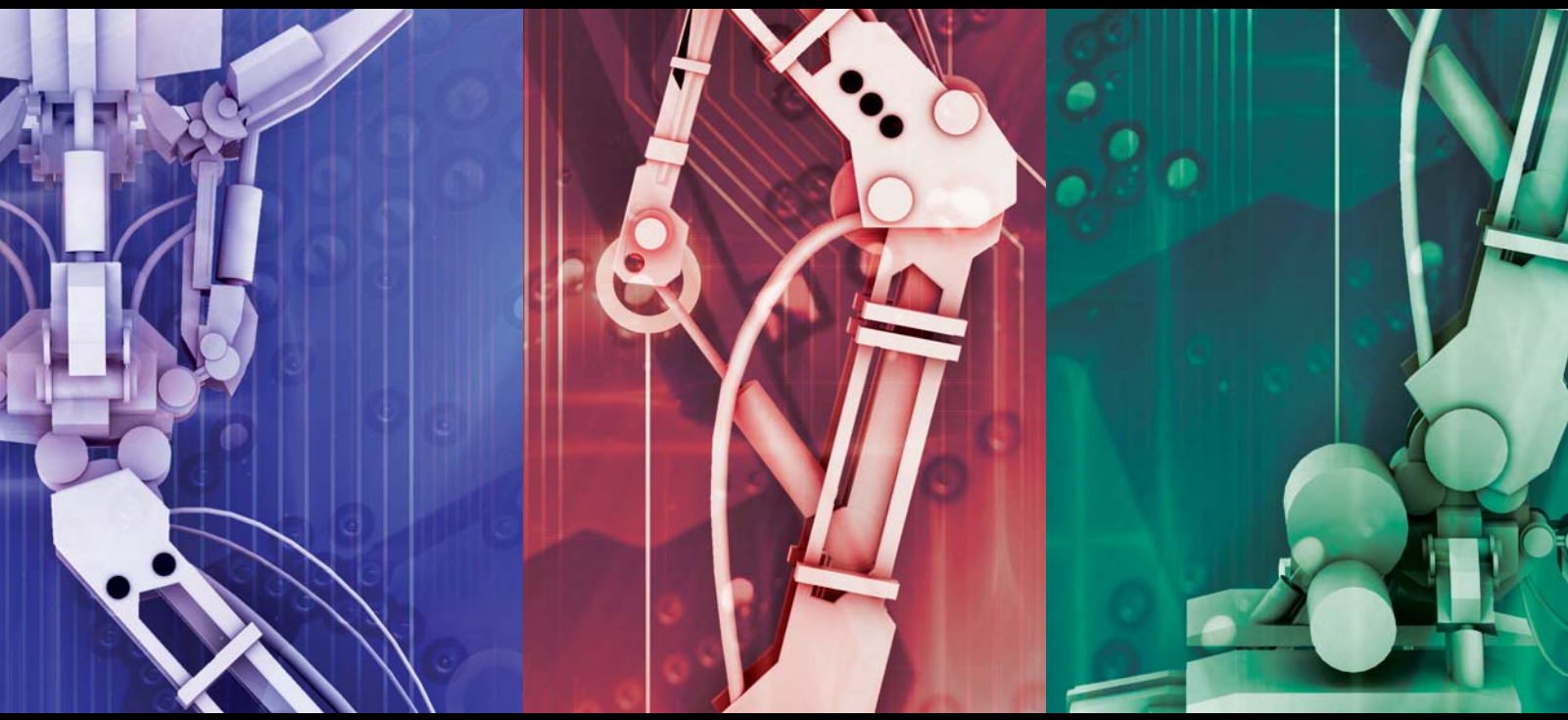
Journal of Robotics

Cognitive and Neural Aspects in Robotics with Applications 2011

Guest Editors: Madan M. Gupta, Ivo Bukovsky, Noriyasu Homma, Zeng-Guang Hou, and Ashu M. G. Solo



Cognitive and Neural Aspects in Robotics with Applications 2011

Journal of Robotics

Cognitive and Neural Aspects in Robotics with Applications 2011

Guest Editors: Madan M. Gupta, Ivo Bukovsky, Noriyasu Homma, Zeng-Guang Hou, and Ashu M. G. Solo

Editorial Board

Contents

Hindawi Publishing Corporation
Journal of Robotics
Volume 2012, Article ID 132360, 2 pages
doi:10.1155/2012/132360

Editorial

Cognitive and Neural Aspects in Robotics with Applications 2011

Madan M. Gupta,[1] Ivo Bukovsky,[2] Noriyasu Homma,[3] Zeng-Guang Hou,[4] and Ashu M. G. Solo[5]

[1] *Intelligent Systems Research Laboratory, College of Engineering, University of Saskatchewan, Saskatoon, SK, Canada S7N 5A9*
[2] *Division of Automatic Control and Engineering Informatics, Department of Instrumentation and Control Engineering, Faculty of Mechanical Engineering, Czech Technical University in Prague, Technicka 4, Prague, 16607, Czech Republic*
[3] *Cyberscience Center, Tohoku University, 6-3 Aoba, Aramaki, Aoba-Ku, Sendai 980-8578, Japan*
[4] *Institute of Automation, The Chinese Academy of Sciences, P.O. Box 2728, Beijing 100190, China*
[5] *Maverick Technologies America Inc., Suite 808, 1220 North Market Street, Wilmington, DE 19801, USA*

Correspondence should be addressed to Madan M. Gupta, madan.gupta@usask.ca

Received 19 January 2012; Accepted 19 January 2012

With the evolution of our complex technological society and the introduction of new notions and innovative theoretical tools such as cognitive and neural aspects of robotics, there have been some new evolutions of theoretical methodology. These evolving and innovative theoretical tools are providing some intelligence and robustness in robotic systems similar to what is found in natural biological species.

Cognition and intelligence—the ability to learn, understand, and adapt—are the creation of nature, and they play a key role in human actions as well as in many other biological species. Humans possess some robust attributes of learning and adaptation and that is what makes them so intelligent.

Humans react through the process of learning and adaptation on information received through a widely distributed network of sensors and control mechanisms in our body. The faculty of cognition which is contained in our carbon-based computer—the brain—acquires information from the environment through various sensory mechanisms such as vision, hearing, touch, taste, and smell. Then the process of cognition—cognitive computing—integrates this information through its intricate neural networks and provides appropriate actions. The cognitive process then advances further toward some attributes such as learning, recollection, reasoning, and control.

We are learning from the carbon-based cognitive computer—the brain—and are now in the process of inducing perception, cognition, and intelligence in robotics machines. One of our aims is to construct a robotic vehicle that can think and operate in uncertain and unstructured driving conditions. Robots in manufacturing, mining, agriculture, space and ocean exploration, and health sciences are just a few examples of challenging applications where human attributes such as cognition and intelligence can play an important role.

The proposal for this second special issue on cognitive and neural aspects of robotics with applications was conceived in late 2010, and we are now pleased to present 8 research papers that cover a wide aspect of cognition and intelligence. Initially, we received 15 research papers, but after going through a thorough review process, we have accepted only 8 research papers. These 8 accepted research papers cover some wider aspects of cognition and intelligence in the field of robotics, and for this special issue, we have divided these 8 research papers into three groups.

Four research papers cover the fields of cognition, perception, and neural learning in robotics. In the research paper entitled "Control loop sensor calibration using neural networks for robotic control," Kathleen A. Kramer and Stephen C. Stubberud present a technique referred to as a neural extended Kalman filter (NEKF) to provide both state estimation in a control loop and learn the difference between true sensor dynamics and the sensor model. The resulting sensor model provides better estimation capability and redundancy. In the research paper entitled "3D Assembly

group analysis for cognitive automation," Christian Brecher, Thomas Breitbach, Simon Müller, Marcel Ph. Mayer, Barbara Odenthal, Christopher Schlick, and Werner Herfs introduce a novel concept that allows for the cognitive automation of robotic assembly processes and use an assembly cell of two robots to verify the concept. In the research paper entitled "Optimal search strategy of robotic assembly based on neural vibration learning," Lejla Banjanovic-Mehmedovic, Senad Karic, and Fahrudin Mehmedovic present the implementation of an optimal search strategy in verification of a robotic assembly process based on neural vibration learning. In the research paper entitled "A cognitive model for generalization during sequential learning," Ashish Gupta, Lovekesh Vig, and David C. Noelle present a detailed analysis of the Leabra cognitive modeling framework for sequential learning of multiple tasks because traditional neural network models are trained for one task, and when they are trained to do a new task, they forget how to do the original task. Then they demonstrate the applicability of sequential learning to a pair of movement tasks using a simulated robotic arm.

Two research papers deal with neurocontrol and neural architecture in robotics. In the research paper entitled "Discovering and characterizing hidden variables using a novel neural network architecture (LO-net)," Soumi Ray and Tim Oates present a novel neural network architecture for solving problems related to theoretical entities, which are aspects of the world that cannot be sensed but are causally relevant, such as black holes and dark matter. Their novel neural network architecture discovers that theoretical entities exist, computes their number, and computes their values. In the research paper entitled "Design of an error-based adaptive controller for a flexible robot arm using dynamic pole motion approach," Ki-Yong Song, Madan M. Gupta, and Noriyasu Homma introduce a novel concept of dynamic pole motion (DPM) for the design of an error-based adaptive controller (E-BAC). The purpose of this novel design approach is to make the system response reasonably fast with no overshoot where the system may be time varying and nonlinear with only partially known dynamics. For illustrating the strength of this novel design approach, they give an example of a flexible robot with nonlinear dynamics.

Finally, two research papers focus on miscellaneous robotics applications including underwater robotics and the development of bio-machine. In the research paper entitled "Application of on-board evolutionary algorithms to underwater robots to optimally replan missions with energy constraints," Mae L. Seto presents an on-board knowledge-based agent based on a genetic algorithm to replan a near optimal survey mission for an autonomous underwater vehicle (AUV) given the AUV energy budget, mission duration, and survey area dimensions. Finally, in the research paper entitled "Development of bio-machine based on the plant response to external stimuli," Aditya. K., Ganesha Udupa, and Yongkwun Lee present research on plant intelligence. Specifically, they present research on plant action potential signals and their response to various light modes for control of a biomachine.

Thus, as can be seen, these 8 research papers represent a broad cross-section of cognitive and neural aspects of the field of robotics. Also, the authors of these research papers have used some cognitive and neural aspects of robotics in the design of learning and control algorithms. These research papers have been authored or coauthored by 24 researchers in the robotics field from 14 different research institutions or universities located in 7 different countries: Bosnia and Herzegovina, Canada, China, India, Japan, Korea, and U.S.A.

This second special issue including 8 contributed research papers, which are authored by international researchers and devoted to various aspects of cognition, perception, neural learning, and neurocontrol in robotics with various industrial applications, is an informative and useful addition to the field of robotics.

Acknowledgments

We, the guest editors, would like to express our sincere appreciation to the editorial board of the *Journal of Robotics* and Hindawi for their confidence and unwavering support for this second special issue. We acknowledge the efforts of the authors for their valuable research contributions to this special issue of the journal and that of the reviewers who adhered to the strict timeline for making this special issue a success.

Madan M. Gupta
Ivo Bukovsky
Noriyasu Homma
Zeng-Guang Hou
Ashu M. G. Solo

Hindawi Publishing Corporation
Journal of Robotics
Volume 2012, Article ID 542124, 10 pages
doi:10.1155/2012/542124

Research Article

Application of On-Board Evolutionary Algorithms to Underwater Robots to Optimally Replan Missions with Energy Constraints

M. L. Seto

Defence R&D Canada, Dartmouth, Nova Scotia, Canada B2Y 3Z7

Correspondence should be addressed to M. L. Seto, mae.seto@dal.ca

Received 16 July 2011; Revised 2 December 2011; Accepted 16 December 2011

Academic Editor: Ivo Bukovsky

The objective is to show that on-board mission replanning for an AUV sensor coverage mission, based on available energy, enhances mission success. Autonomous underwater vehicles (AUVs) are tasked to increasingly long deployments, consequently energy management issues are timely and relevant. Energy shortages can occur if the AUV unexpectedly travels against stronger currents, is not trimmed for the local water salinity has to get back on course, and so forth. An on-board knowledge-based agent, based on a genetic algorithm, was designed and validated to replan a near-optimal AUV survey mission. It considers the measured AUV energy consumption, attitudes, speed over ground, and known response to proposed missions through on-line dynamics and control predictions. For the case studied, the replanned mission improves the survey area coverage by a factor of 2 for an energy budget, that is, a factor of 2 less than planned. The contribution is a novel on-board cognitive capability in the form of an agent that monitors the energy and intelligently replans missions based on energy considerations with evolutionary methods.

1. Introduction

Autonomous underwater vehicles (AUVs) are robots used for underwater tasks that range from surveys, inspection of submerged structures (e.g., pipelines), searching for downed aircraft, tracking oceanographic features, laying undersea cable, undersea mapping, and finding mines, to name a few. Such robots work in an unstructured dynamic environment with unique *perception*, *communication*, and *decision* issues compared to land, air, or space robots. Means for *perception* and detection of underwater targets include magnetic, optical, electric field, thermal (infrared), hydrodynamic changes (pressure), and sound (acoustic). Sound is unsurpassed, compared to other means, for detection underwater. As an example, the sonar (sound navigation and ranging) is a popular underwater perception sensor that uses sound for detection, classification, and location of underwater targets. Having said that, there are acoustic propagation difficulties in the highly variable, noisy, and reverberant water medium. The ocean is a nonstationary and dynamic environment where the conductivity, temperature, and density of its water varies temporally and spatially and thus affects the propagation of acoustic signals within it. Add to this multi-path, absorptive losses (high attenuation) [1], and low bandwidth for acoustic signal propagation that also occur in nonpredictable ways. Consequently, underwater *communication* issues stem from the variability and poorness of acoustic propagation in water.

These propagation limitations impact how AUVs are employed since reliable acoustic communications with their operators (or other AUVs) is not easily possible. Consequently, out of the land, ocean, space, and air robot environments, the ocean one is difficult for tasks that require persistent and reliable communications. Robotic autonomy, where the robot makes *decisions* and autonomously alters its mission plans *in situ* without operator intervention, in fulfillment of its mission, is necessary to exploit the potential of underwater robots. Autonomy is one way of coping with the poor underwater communications issue.

Autonomy or *decision* issues are addressed to make AUVs truly autonomous—able to operate long periods without operator intervention. It is desirable that the AUV have the autonomy for decision making or problem solving to deal with unexpected robot or mission events in a timely fashion. Mission autonomy is the ability to adapt a mission to unanticipated conditions in the environment or *in situ*

Table 1: Common energy sources for AUVs.

Battery type	Cost ($/Wh)	Energy density (Wh/liter)	Specific energy (Wh/kg)	Typical cell dimensions (mm)
lithium ion	4.27	300	130	$225 \times 212 \times 9.5$
silver zinc	1.75	240	130	$178 \times 161 \times 8$
lead acid	0.17	65	30	$330 \times 228 \times 152$

intelligence that can be exploited to better perform the mission. For example, in-situ environmental measurements can be applied to collect more optimal sonar images. Robot autonomy addresses issues that increase the robot's fault tolerance so it can adapt to unexpected robot events (e.g., more energy consumed earlier in the mission than planned precipitating a shortage for the rest of the mission). Wherever possible, if on-board autonomy can replan, or adapt, in light of unexpected events, the mission can be completed or better performed. Otherwise, the mission could be scrubbed (or in a drastic case, the robot is lost) due to an unexpected event. At the root of the required autonomy or decision-making are cognitive abilities for the robot. A timely scenario that would benefit from such autonomy or decision making ability is discussed next.

Large AUVs, of which the Explorer class [2] is an example, can have ranges on the order of hundreds of kilometers. Such missions can occur over a week or more which is a long time for an ocean environment to remain stationary. Consequently, energy shortages over long deployments are a real possibility. Energy shortages on missions can occur due to unplanned events, as the AUV travels through stronger currents, goes off-course and requires more energy to finish the survey *and* be at the recovery point on time, on-board hotel load increased because instrumentation use was higher (e.g., extra sensors employed), or the AUV was not trimmed for the local salinity conditions (final AUV trim only confirmed when AUV is underway) to name a few. The impact of such unplanned events cumulate over the course of long duration missions to the point where the planned mission may not be achievable anymore. Once the AUV is launched and underway, it is not (easily) possible to monitor these conditions in order to be forewarned of critical issues in time to have an operator recover the AUV to replan the survey. As well, the operator is not necessarily nearby on a support ship. With AUVs now used on longer missions, the issue of unexpected energy shortages is timely and relevant. The different types of AUV on-board energy is briefly discussed.

Like most mobile robots, AUVs carry all their energy on-board for a mission. This energy powers on-board equipment, sensors, computers, propulsion/locomotion, and so forth. On long-duration surveys, the bulk of the energy is for propulsion/locomotion. The on-board volume allocated to carry energy, in the form of batteries, is fixed, so the objective is to carry batteries of the highest specific energy and density possible. Table 1 (not an exhaustive list) shows properties of a few common AUV energy sources. Operating cost and endurance are correlated with the energy source selected. Consequently, endurance is also correlated with AUV size. A fair amount of effort has gone into investigating AUV energy sources [3–5]. They can range from the familiar lead acid battery to modern fuel cells [4, 5]. Fuel cells are an emerging technology that will impact how AUVs are tasked for long survey missions and for allowing on-board computationally intensive calculations. Presently, fuel cells are not commonly used in AUVs. The impact of energy on mission-planning is described next.

AUV path planning (or mission planning) is an active area of research [6–14]. As in air and land robots, underwater path planning objectives include obstacle (moving or stationary) avoidance [6, 7], optimal area coverage [8], and transiting accurately and safely between points with high variability in the water conditions [9–11]. Most related work considers the energy budget and the desire to minimize energy use [9, 10]. However, there is little work that studies replanning specifically due to an unexpected energy shortage on long survey missions. There is even less work on missions that employ side scan sonars.

A mission is planned with the AUV carrying 10% (or more) surplus energy. However, if unexpected events occur, and that surplus is insufficient, there is currently little contingent mission replanning. The usual response is to recover the AUV, replan the mission with an operator, download the mission to the AUV, and redeploy the AUV. Many such survey interruptions are undesirable on long missions as they require additional time, expense, and a nearby support ship. It also detracts from the value of using an AUV to begin with.

This paper's objective is to show that on-board autonomy to replan an AUV side scan sonar mission due to an unexpected AUV energy shortage enhances the mission's success.

This on-board autonomy is in the form of a novel knowledge-based autonomous agent developed to monitor and replan missions due to unexpected energy shortages. The agent replans the mission with a genetic algorithm (GA) that evaluates the fitness of proposed replanned missions to achieve the mission objectives (survey remaining area with less energy than planned and make the recovery schedule). It uses knowledge of the AUV's nonlinear hydrodynamics, dynamics, and control response to proposed replanned missions. Additionally, it takes into account the AUV's real-time attitudes, speed through water, hotel load, and so forth, to evaluate a replanned mission's fitness. The agent continually compares actual energy expended against anticipated energy to expend for the mission. This is achieved through on-line calculations of the remaining energy/endurance to predict and evaluate the AUV likelihood of completing the mission given the measured energy consumption rate. If it appears, at any point in the mission, that the AUV will have insufficient energy to complete its mission as planned, the agent initiates a behavior to replan its mission given the survey area left, the

remaining power, and the original mission completion and AUV recovery times.

The rest of the paper is organized as follows. Section 2 defines the mission configuration followed by an overview of the autonomous agent. Then, details of the AUV model (Sections 2.1–2.3) used in the optimization to evaluate the AUV hydrodynamics, dynamics, control response, and subsequent energy consumption to missions proposed by the genetic algorithm are described. Section 2.4 describes the genetic algorithm implementation and how the aforementioned AUV models are used as the evaluation function for the GA. Section 3 describes the initial validation of the AUV energy model component in the agent. Section 4 analyzes an illustrative case by imposing a range of available energies, time constraints, and survey areas to evaluate optimal solutions proposed by the agent. It then compares the optimal solutions against what would have been achieved without autonomous replanning of the mission to highlight the agent's effectiveness. Section 5 concludes with a few remarks.

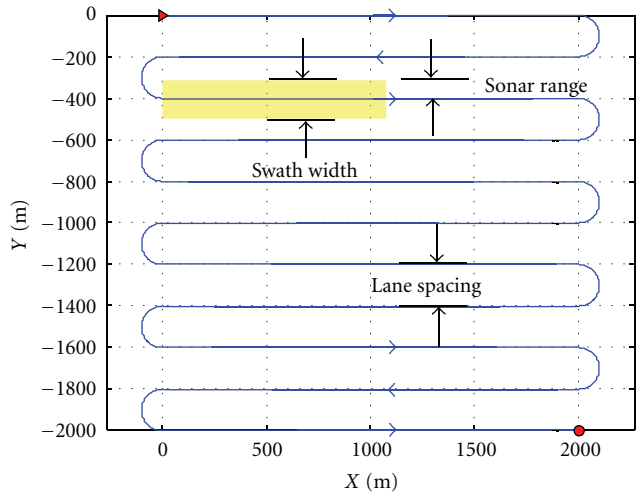


FIGURE 1: AUV side scan sonar survey mission (—) with 200 m lane spacing and 2 km survey leg lengths. Arrow heads indicate AUV travel direction (start point ▶; recovery point •).

2. Autonomous Agent Description

The objective of this section is to define the mission configuration, provide an overview of the knowledge-based agent, and then detail the AUV models (dynamics, hydrodynamics, controls, energy) to be used in the GA. This Section concludes with an overview description of the GA implementation.

The AUV mission is to survey an area using long straight transits capped by 180-degree turns which put the AUV on a reciprocal heading—the "lawn mower tracks" whose lane spacing defines the mission geometry or configuration (Figure 1). Highlighted in yellow is the swath scanned by the AUV's side-scan sonar on a constant heading. Side-scan sonars scan from both sides to a specific sonar range (shown as 100 m). The objective is to survey/scan an area through side-by-side swaths (strips) with overlap between the swaths (similar to lawn mowing). The overlap is required to co-register features in the images, perform feature-based navigation, and so forth. These constraints govern operation with side scan sonars. For the case study of Figure 1, a projected AUV energy shortage is detected at $(X, Y) = (0, 0)$ m with the AUV recovery point at $(X, Y) = (2000, -2000)$ m to occur no later than t_R of 30,000 seconds later. The arrows show the AUV travel direction.

The swath width relative to the lane spacing determines the amount of area overlap across neighboring swaths, A_{overlap}. The area surveyed is the total area scanned by the sonar as it passes through. If all the area is scanned from the start to recovery point, then a total of 4 km^2 of area has been surveyed (area in turns do not contribute to area surveyed).

When the agent replans a mission, it calculates the energy to transit directly to the recovery point from every point along the mission. The optimal replanned mission surveys until there is just enough energy to branch off and transit to the recovery point. If the mission is optimally planned, the branch off to the recovery point occurs near the end of the survey area.

The components of the agent are shown in Figure 2. Upon detection of a projected energy shortage, the agent replans a new mission by first determining dimensions of the remaining area to survey (agent knows survey area corners and AUV location within it) and checks the actual power on-board, and the time left to do the rest of the survey. These parameters are inputs to the genetic algorithm to generate the nominal optimal mission, $[\mathbf{MIS}]_o$. $[\mathbf{MIS}]_o$ consists of the desired AUV speed v_o, turn diameter D_o, and the total time t_o, to complete this mission. However, this is only a nominal optimal mission as it may or may not be realizable by the AUV. Function, N, takes v_o and D_o and uses them as set point values for the robot to achieve. Speed achieved, v_A, is determined based on the AUV propulsion, dynamics, hydrodynamics, and control as well as the water currents. v_A is then used to determine area surveyed in time t_A with the available energy. Note that there are three different time values, t_R, t_o, and t_A. The way the agent is implemented the recovery time, t_R is the largest of the three.

Finally, objective function, F, is applied to AUV response, $\mathbf{R}$, to determine the fitness value, f, for proposed mission $[\mathbf{MIS}]_o$. The details of the dynamic and control models used in the on-line evaluation of the AUV's response to a mission are briefly discussed next. This is followed by a description of the genetic algorithm implementation used.

2.1. Hydrodynamic and Dynamic Model. The on-board AUV hydrodynamics, dynamics and control models were implemented, and validated using the DRDC *Theseus* underwater vehicle [15] used in Arctic missions for laying cable under ice. The AUV equations of motion (1) for three rotational (yaw, pitch, roll) and three translational degrees-of-freedom are integrated. The notation used is shown in Figure 3.

X, Y, and Z, are the external forces due to the added masses, hydrodynamics, statics, and control fins. The control

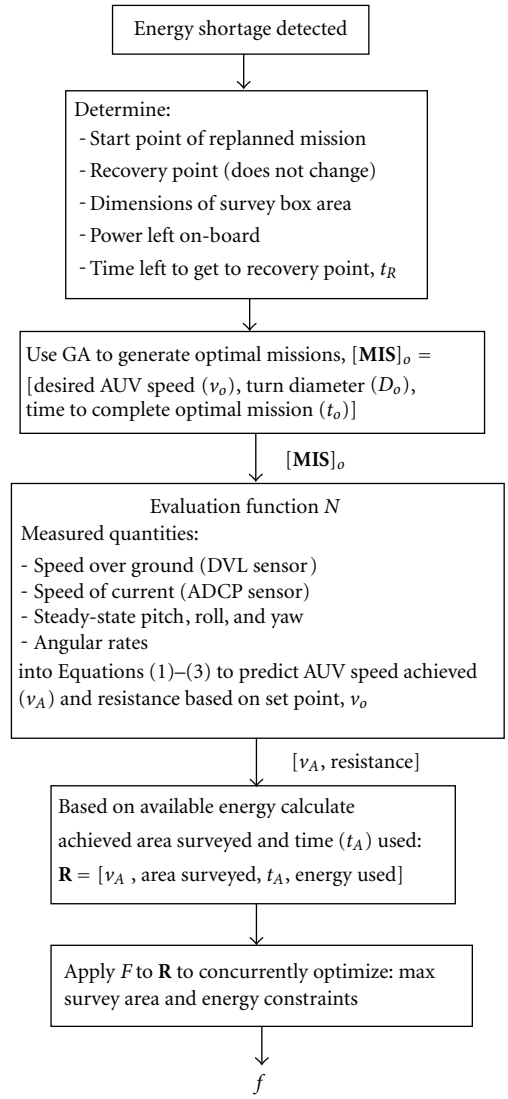


Figure 2: Overview of autonomous agent to replan AUV mission upon detection of an energy shortage.

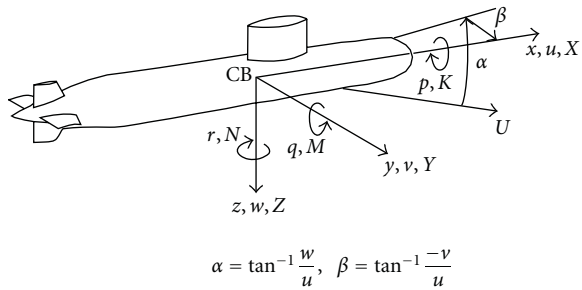


Figure 3: Nomenclature for AUV motion in body frame axes.

fin damping, natural frequency, and maximum rates are described with a second order model as part of the fin response to commanded deflections (where δ_{bp} = bow port fin, δ_{ss} = stern starboard fin, and δ_r = rudder). K, M, and N are the moments of the roll, pitch, and yaw external forces. For brevity, only the X external forces are detailed in (2) as it is most relevant to propulsive power (4). The other 5 external forces [16] are omitted here for brevity. Note ρ = water density, and the following pertain to the AUV: l = length, m = mass, W = vehicle weight, B = buoyancy, and X_{prop} = propulsive thrust. A more detailed description of the hydrodynamic terms is available [16]. The propulsion model, captured through X_{prop}, was validated to adequately capture the AUV behavior in level flight, diving, rising, and so forth [15].

$$X = m[\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})],$$

$$Y = m[\dot{v} - wp + ur - y_G(r^2 + p^2) + z_G(qr - \dot{p}) + x_G(qp + \dot{r})],$$

$$Z = m[\dot{w} - uq + vp - z_G(p^2 + q^2) + x_G(rp - \dot{q}) + y_G(rq + \dot{p})],$$

$$
\begin{aligned}
K &= I_x\dot{p} + \left(I_z - I_y\right)qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} \\
&\quad + (pr - \dot{q})I_{xy} \\
&\quad + m[y_G(\dot{w} - uq + vp) - z_G(\dot{v} - wp = ur)], \\
M &= I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{zx} \\
&\quad + (qp - \dot{r})I_{yz} \\
&\quad + m[z_G(\dot{u} - vr + wq) - x_G(\dot{w} + uq + vp)], \\
N &= I_z\dot{r} + \left(I_y - I_x\right)pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} \\
&\quad + (rq - \dot{p})I_{zx} \\
&\quad + m[x_G(\dot{v} - wp + ur) - y_G(\dot{u} - vr + wq)];
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
X &= -\frac{1}{2}\rho l^4\left[X_{pp}p^2 + X_{rr}r^2 + X_{qq}q^2 + X_{q|q|}q\,|q|\right] \\
&\quad + \frac{1}{2}\rho l^3[X_{\dot{u}}\dot{u} + X_{\dot{v}}\dot{v} + X_{\dot{w}}\dot{w}] \\
&\quad + \frac{1}{2}\rho l^4\left[X_{\dot{p}}\dot{p} + X_{\dot{q}}\dot{q} + X_{\dot{r}}\dot{r}\right] \\
&\quad + \frac{1}{2}\rho l^3\left[X_{vr}vr + X_{wq}wq\right] \\
&\quad + \frac{1}{2}\rho l^2\Big[X_{uu}u^2 + X_{vv}v^2 + X_{ww}w^2 + X_{\delta r\delta r}u^2(\delta r)^2 \\
&\qquad + \frac{1}{2}X_{\delta b\delta b}u^2\left(\delta_{bp}^2 + \delta_{bs}^2\right) \\
&\qquad + \frac{1}{2}X_{\delta s\delta s}u^2\left(\delta_{sp}^2 + \delta_{ss}^2\right) + X_{\text{prop}}\Big] \\
&\quad + (W - B)\sin\theta.
\end{aligned} \tag{2}
$$

2.2. AUV Control Model. The AUV in this study has two bow fins and a cruciform (+) stern fin configuration. The horizontal fins are used for pitch, roll, and depth control. Yaw (heading) control is achieved through the stern fins in the vertical plane. Both vertical stern fins are deflected together and constitute δ_r. This horizontal control is directly relevant to the mission/path-planning work in this study.

The fins are under closed-loop PID (proportional-integral-differential) control. Corrective fin deflections to minimize the attitude, heading, and depth errors are determined by applying the PID gains matrix to the AUV state vector: $[\psi, \theta, \varphi, \dot{\psi}, \dot{\theta}, \dot{\varphi}, z, \dot{z}]$ (yaw, pitch, roll, yaw rate, pitch rate, roll rate, depth, and depth rate) as shown in (3). $[\psi, \theta, \varphi]$ are the Euler angles that relate inertial frame axes to the body frame axes shown in Figure 3. The PID gains are arrived at through a combination of vehicle dynamic simulations and tuning during sea trials. In both cases, the AUV is commanded to do a variety of maneuvers that include holding and changing course, diving and rising, and turning at different speeds and rudder deflections, and so forth. In the case of the dynamic simulations, the AUV's hydrodynamic derivatives were both predicted and measured [17]. This controller has been studied in some detail and is near optimal in its control authority distribution to maintain the attitude and heading set points [18]:

$$
\delta = \begin{bmatrix} \delta_{bs} \\ \delta_{bp} \\ \delta_{ss} \\ \delta_{sp} \\ \delta_r \end{bmatrix} = \begin{bmatrix} \text{PID} \\ \text{GAINS} \\ \text{MATRIX} \end{bmatrix} \times \begin{bmatrix} \psi \\ \theta \\ \phi \\ \cdot \\ \psi \\ \cdot \\ \theta \\ \cdot \\ \phi \\ Z \\ \cdot \\ Z \end{bmatrix}. \tag{3}
$$

2.3. AUV Energy Consumption Model. The energy consumption of the AUV is modeled as [19]:

$$
e_t = \frac{\left(p_s + p_p + p_v\right) \times r}{3600 \times V}, \tag{4}
$$

such that: e_t = total energy on-board (kWh) [fixed for mission], r = survey range (km) [changes with mission geometry, e.g., in Figure 1], V = AUV speed through water (m/s), p_v = on-board vehicle equipment power (W) [hotel load #1, changes with control plane usage, sensors on different power settings, use of ballast system, intensity of computations, etc. over mission], p_s = sonar power, (W) [hotel load #2, changes with sonar, sonar range, and mission geometry in step function manner], p_p = propulsion power (W) [~AUV resistance (2) × AUV water speed, predict to change as per (1)–(3)]. AUV propulsion power, p_p, is captured through time-varying models based on (1)–(3) which calculate the hydrodynamic resistance in order to predict energy requirements for the remaining mission. As shown in Figure 2, the inputs to these equations are the measured on-board AUV speeds (both speed through water and speed over ground), steady-state yaw, pitch, and roll angles, angular rates, and so forth. The on-board equipment and sensor energy usage is described in terms p_v and p_s, respectively. A running average of the hotel loads are used as measures of p_v and p_s for input into the autonomous agent.

As the AUV speed increases so does the AUV's hydrodynamic resistance ($\sim v^2$) and, consequently, the energy is consumed at a higher rate. This could result in a shorter mission time if it does not consume all the energy before the mission is complete. On long deployments, the propulsion power is often the largest term of the three. It varies as ~ velocity^3× time. If the AUV speed is fast, it consumes energy quickly and may not survey much area before having to transit to the recovery point—thus mission success is

Table 2: Bounds on optimizing parmeters. (sonar range = 50 m on one side → survey swath width = 100 m).

Optimizing parameter	Bounds imposed	
Speed over ground	[0.5–2.0] meters/second	AUV performance
Turn diameter	[50–80] m ~ [50–20] % overlap of neighbour swaths	sensor performance
Time	[80–100] % of max time	mission requirement

limited by energy. If the AUV speed is slow, it may also not cover much area before it has to transit to the recovery point (mission success limited by time). The optimal solution is somewhere in between.

Thus, the agent uses knowledge of the robot performance through the dynamic, hydrodynamic, and controller models with input from measured on-board AUV states, to monitor and predict energy usage to detect an energy shortfall. The energy consumption prediction is performed every τ seconds. Typically, τ is the time to complete 90% of a survey leg which is just prior to the AUV changing heading. It is a good point in a mission to replan and change a mission if required. These energy projections are also compared against a time-dependent battery energy-consumption curve since the consumption rate is nonlinear and specific to a battery.

2.4. Genetic Algorithm Implementation. The application to path-planning of a search procedure based on Darwin's theories of natural selection and survival has been recognized [20]. In these methods, referred to as genetic algorithms, a population of possible solutions is maintained and the paths are iteratively transformed by genetic operations like crossover and mutation [20]. GAs are applied to a variety of path-planning problems [6, 10, 13]. They are especially adept at solving problems with objective functions that are not continuous, differentiable, or possessing a closed tractable form.

This is exactly the case for the AUV energy problem here. The objective function acts on quantities that are obtained through integration of differential equations. As well, the propulsion energy varies as $\sim v^3 \times$ time so the three quantities are inextricably linked, yet they are optimized for a calculated quantity, the area surveyed. The application of genetic algorithms for the posed AUV energy problem, over other methodologies, is quite appropriate.

The optimization objective is to concurrently

(1) maximize area surveyed as shown in Figure 1 (or maximize range, r, in (4)) and

(2) stay within energy budget.

The optimizing parameters are

(i) AUV speed (AUV performance),

(ii) swath width overlap (or turn diameter for a given sonar range) (AUV sensor performance), and

(iii) replanned mission time (mission requirement).

Bounds over which the optimizing parameters can vary are imposed to confine the GA to search space regions with feasible solutions. This is so the agent does not waste time looking in regions that are known to *not* yield physically achievable solutions. The bounds on the optimizing parameters are shown in Table 2. They are based on at-sea best practices. The speed bounds 0.5 m/s < AUV speed <2 m/s are achievable for the AUV used.

The lane spacing is a function of the sonar swath width and the area overlap between neighboring swaths. While high overlap is conducive to good target detection with side scan sonars, it does not make for an efficient survey. From just geometrical considerations (Figure 1) between overlapping rectangles, the lane spacing = sonar range $\times$ 2 $\times$ $(1 - A_{\text{overlap}})$.

The time to complete the survey and be at the recovery point is a mission requirement. It is desirable that the AUV be at the recovery point more-or-less on time where possible to meet the ship to ensure successful recovery of the AUV and data.

Formally, the multiple objectives are to optimize maximum coverage and to stay within the energy budget and time constraints. The optimization problem is thus posed as

given: objective function F: $A \rightarrow \Re$ from some set A of real numbers $\Re$

find: $[\mathbf{MIS}]_{\text{o}} \in A : F([\mathbf{MIS}]_{\text{o}}) \geq F([\mathbf{MIS}]) \forall [\mathbf{MIS}]$ in A,

where: A is the solution space spanned by solutions, $[\mathbf{MIS}]$ to F. $[\mathbf{MIS}]_{\text{o}}$ is the optimal solution and $\Re$ is the set of real numbers.

Applying function N takes mission parameter vector $[\mathbf{MIS}]_{\text{o}}$ as input to the AUV nonlinear equations of motion (1)–(3) to determine AUV response $\mathbf{R}$, that is,

$$\mathbf{R} = N([\mathbf{MIS}]_{\text{o}}) \longrightarrow \begin{bmatrix} \text{speed} & \text{area} & \text{time} & \text{energy} \\ \text{achieved} & \text{surveyed} & \text{needed} & \text{used} \end{bmatrix}. \tag{5}$$

Then, objective function F (6) assigns value, f, to response vector, $\mathbf{R}$, as a measure of a proposed $[\mathbf{MIS}]_{\text{o}}$ fitness to achieve the mission objectives. Weights w_i reflect the relative priorities of the different objectives and vary with mission, sensors, AUV, environmental conditions, and so forth:

$$F(\mathbf{R}) = f = w_1 \begin{pmatrix} \text{area} \\ \text{surveyed} \end{pmatrix} + w_2 \begin{pmatrix} \text{time} \\ \text{required} \end{pmatrix} + w_3 \begin{pmatrix} \text{energy left-} \\ \text{energy used} \end{pmatrix} + w_4 \begin{pmatrix} \text{hotel} \\ \text{load} \end{pmatrix} + w_5 \begin{pmatrix} \text{speed achieved-} \\ \text{speed desired} \end{pmatrix}. \tag{6}$$

Search space A is pruned to include solutions [**MIS**] whose:

(i) time for a complete area survey (based on proposed speed, lane spacing, etc.) is less than t_R, and

(ii) whose energy for a complete area survey (based on proposed speed, lane spacing, etc.) is less than e_t (4).

To account for operational requirements, the following heuristic constraints are imposed:

(i) energy consumed for a proposed solution (given proposed speed, lane spacing, and time) is less than $0.85 \times$ remaining power (though weight w_3)—as in practice, this saves energy for the AUV to get to the recovery point, and

(ii) the proposed mission time is less than $0.95 \times$ total time allocated (t_R), for the mission (through weight w_2), which ensures the AUV does not arrive at the recovery point too early.

With this scheme, the agent does not produce solutions exceeding energy budget e_t or maximum time t_R.

The genetic algorithm creates a population of solutions, [**MIS**] and uses cross-over and mutation [20] to generate better solutions. Then, it propagates the evolution of the better solutions by choosing the best solutions as parent solutions. Poor solutions, as in nature, can evolve and propagate. However, if sufficient generations are calculated, poor solutions do not survive. A reasonable solution that minimizes (or maximizes) the objective function is an optimal solution. Often there are several objectives to optimize, as in this case.

The genetic algorithm simulates the evolution of an [**MIS**] solution towards the optimal one where "survival of the fittest" is applied to a population of solutions. The steps in the genetic algorithm implementation are as follows:

(1) initialize space A spanned by a population of acceptable solutions, [**MIS**]—(5);

(2) evaluate each solution, perform $F(N[\mathbf{MIS}])$—(6);

(3) select a new population from the old population based on the fitness of the solutions;

(4) apply genetic operators cross-over and mutations to the new population to create new solutions;

(5) evaluate the newly-created solutions by applying $F(N[\mathbf{MIS}])$, and

(6) repeat steps 3–6 until termination criteria which is convergence of the fitness value f.

The initial solution space is seeded with similar historical in-water missions (i.e., good combinations of AUV velocity and swath spacing/turn diameter that surveyed a rectangular area within a prescribed time limit) that worked in the past. The GA achieves solutions with an initial population size of 40 solutions and a maximum 25 generations of evolution.

The chromosome representation to describe members of the solution population was real-valued over binary due to the greater (order of magnitude) computational efficiency of real valued representations [21]. This impacts the implementation of the GA and especially the way the nonuniform mutation and arithmetic cross-over operators are applied. Define parent solutions as vectors $\overline{X} = \{x_1, \ldots, x_i, \ldots, x_n\}$ and $\overline{Y} = \{y_1, \ldots, y_i \ldots, y_n\}$. The parameters of these vectors are AUV speed, swath spacing/turn diameter, and mission completion time. Nonuniform mutation changes one of the parameters of the parent solution based on a non-uniform probability distribution. This Gaussian distribution starts wide and narrows to a point distribution as the current generation approaches the maximum number of generations. Nonuniform mutation randomly selects one of the three (in this case) parameters, and sets it equal to a nonuniform random number:

$$x_i' = \begin{cases} x_i + (b_i - x_i) f(G) & \text{if } r_1 < 0.5 \\ x_i - (x_i + a_i) f(G) & \text{if } r_1 \geq 0.5 \\ x_i, & \text{otherwise,} \end{cases} \tag{7}$$

$$f(G) = \left(r_2 \left(1 - \frac{G}{G_{\max}} \right) \right)^b,$$

r_1, r_2 = uniform random numbers in range $[0, 1]$

G = the current generation number

$G_{\max}$ = maximum number of generations (set by user)

b = shape parameter (set by user)

a_i = minimum possible value for parameter x_i

b_i = maximum possible value for parameter x_i.

Note, there are two possibilities for the nonuniform random number depending on the value of random number r_1.

Arithmetic cross-over produces two complimentary linear combinations of the parent solutions $(\overline{X}, \overline{Y})$ where $r = a$ uniform random number between 0 and 1. The children of the cross-over operation, $(\overline{X}', \overline{Y}')$, are:

$$\begin{aligned} \overline{X}' &= r\overline{X} + (1 - r)\overline{Y}, \\ \overline{Y}' &= (1 - r)\overline{X} + r\overline{Y}. \end{aligned} \tag{8}$$

Normalizations on the new (children) solutions are not required—the results are immediately usable.

The posed optimization problem sets up the possibility of local extrema in solution space A. Given the objective is to survey as much area as possible with a fixed energy budget and to arrive at the recovery point by a given time, the optimization can be driven by energy, time, or both. The next section briefly discusses the validation of the AUV energy model.

3. Validation

A basic validation of the energy model in the agent was performed against sea trials data collected with DRDC's *Theseus* AUV [17]. The runs from that trial are maneuvers

Table 3: Optimized missions to survey 2 km × 2 km area with variable energy and time = 30,000 seconds until recovery.

Area surveyed (%)	Lane spacing (m)	AUV speed (m/s)	Energy used (kWh)	Energy avail (kWh)	Time-max 30 k sec (10 k sec)
76.34	78.29	1.46	133.0	150	29.837
83.58	79.83	1.59	179.3	200	28.112
91.61	80.00	1.72	231.0	250	28.600
94.98	**79.22**	**1.86**	**290.9**	**300**	**29.402**
98.59	75.91	1.92	335.9	350	29.741
98.76	76.05	1.95	270.3	400	29.652

Table 4: Optimized sonar lane spacing and resulting overlap with neighboring lanes (50 m Sonar Range).

Area surveyed (%)	Lane spacing (m)	Survey area overlap (%)
76.34	78.29	21.71
83.58	79.83	20.17
91.61	80.00	20.00
94.98	**79.22**	**20.78**
98.59	75.91	24.09
98.76	76.05	23.95

that are geometrically similar to those in Figure 1. While the resistance of the AUV in straight and level flight is understood and previously validated [17], the energy consumption from propulsion, in turn was not validated until this work. Energy consumption depends on whether the AUV is commanded to maintain closed-loop control over speed or power. In the validation runs, it maintained constant power which means the AUV decreased speed in a turn. The energy consumed was monitored and logged. With all the main components of the autonomous agent described the next section analyzes an illustrative example that highlights the agent's effectiveness.

4. AUV Path Planning Driven by Energy

To assess the agent behaviour, the dimensions of the survey area (X km $\times Y$ km) and the mission time, t_R, until recovery were fixed and the energy budget varied, from insufficient, to a surplus at the time the agent projects an energy shortage. The suspected appearance of local extrema in the solution space did materialize. Its manifestation depended on the energy budget. For the illustrative case, 300 kWh is known to be roughly enough to survey the area based on knowledge of the AUV from sea trials.

4.1. Agent Results and Discussions. The analysis on a representative case is used to highlight the agent's capabilities. The optimal mission parameters for a 2 km × 2 km survey area (sum of total distance travelled = r from (4)) is shown in Table 3. Around the AUV's known, energy budget of 300 kWh (e_t of (4)) is a solution, not necessarily optimal, that can survey the whole area which is both power and time limited. As the energy budget increases, the optimal solution becomes time-limited, as desired. As the energy budget decreases, it can be time limited without all the energy consumed (150 kWh case) or it can be neither energy or time limited (200 kWh case).

The lane spacing is bound to have between a 20%–50% area overlap with neighboring swaths given the range of the sonar. Obviously, minimal overlap area is desired for a survey since it takes less time. However, a lane spacing that achieves that overlap may or may not give an optimal survey energy-wise given the AUV speed and time constraints. The agent does produce lane spacing that creates overlaps close to the minimum 20% as shown in Table 4.

As shown in Table 3, even with generous energy budgets, the agent surveys no more than 98% of the area. This is due to the way the optimizer works. 98% of the area surveyed is a completed survey of the area. From the results shown in Table 3, if an energy shortage is declared by the agent with less than 300 kWh of energy and only 30,000 seconds left for the remaining mission, the area will not be surveyed completely. However, the AUV will be at the recovery point having achieved the optimal maximal coverage possible shown in column 1 of Table 3.

The fitness values from a proposed [**MIS**] can congregate around two main values depending on whether the GA pursues a time, or energy-limited solution. The fitness f values from these two "sets" of solutions can be different by an order of magnitude (not shown). In that case, the agent is designed to choose the energetically favorable solution.

The trends observed in this case do not change with proportionately different sized survey areas or mission times (not shown). As a benchmark for the agent's performance, the AUV dynamic response component can be computed eight times faster than real time. When the genetic algorithm and energy model components are accounted for, the time to compute a solution is about three times faster than real time. Convergence to a solution typically takes less than 25 generations. Time wise, this is only a little (15%) slower than real-time. These are reasonable responses from an on-board-the-vehicle agent.

The effectiveness of these new cognitive capabilities was evaluated by comparing results of replanned missions against that achieved with a mission that did not adapt to energy shortfalls. The agent effectiveness was measured against the case of the optimal solution at 300 kWh in Table 3. Specifically, a mission speed of 1.86 m/s and a turn diameter of 79.22 m. Normally, the operator will use this mission for the speed, energy, and time constraints.

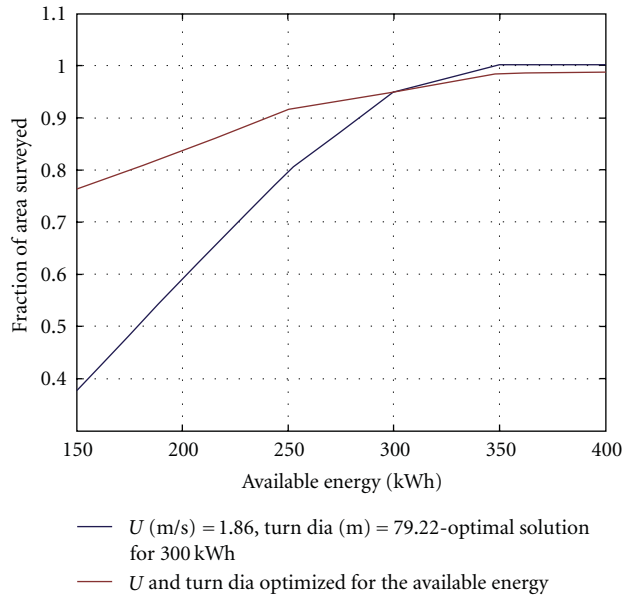


Figure 4: Effectiveness of the autonomous agent to replan AUV missions to adapt to an energy shortage.

As shown in Figure 4, the autonomous agent effectiveness is apparent as the available energy budget drops. The area surveyed with the fixed inflexible mission, near optimal for 300 kWh, and decreases rapidly with decreased available energy (blue plot). The replanned missions (red plot) which optimizes the speed and turn diameter for the available energy perform better. At the lowest energy considered (factor of 2 below sufficient), the surveyed area *improves* over the near optimal 300 kWh mission, by almost a factor of 2.

4.2. Current Work. Encouraged by the results, the current work is implementing the autonomous agent within the DRDC Multi-Agent System framework designed for collaborative underwater vehicles [22]. New developments include collaborative vehicles replanning a mission given a vehicle(s) has an energy shortage. Parallel work uses the agent to replan a mission in the presence of currents measured with an on-board current profiler. At-sea trials are on-going for all developments.

5. Conclusions

With AUVs used on increasingly long deployments, the issue of unexpected energy shortages is timely and relevant. Scripted *a priori* missions based on subsumption architecture [23] cannot adapt to the unstructured dynamic ocean environment and evolving changes within the AUV. This is especially true for AUVs on long deployments. The paper objective was to show that on-board mission replanning for an AUV sensor coverage mission, based on available energy, increases mission success in the event of an energy shortage.

This objective was successfully achieved through on-board cognitive abilities in the form of a novel knowledge-based autonomous agent that replans its missions underway using on-line evolutionary methods to optimize the replanned mission. The replanned missions take into account the AUV dynamic, hydrodynamic, and control performance, AUV-projected energy consumption, amount of survey area left, amount of available energy, time left for mission completion and AUV recovery, and amount of overlap desired in the side scan sonar images. The agent also makes use of on-line measurements of the AUV attitudes, speed through water, hotel load, and so forth and performs an on-going assessment of the AUV's ability to fulfil the mission energy wise.

This agent was tested on scenarios that varied the energy budget from below sufficient to well-above sufficient. With energy budgets that are insufficient to perform the survey mission, the agent can be either time- or energy-limited. With sufficient energy, the agent uses most of the energy and the time. With surplus energy, the agent is time limited. These are acceptable solutions for the replanned mission. For the illustrative case studied, the replanned mission can improve the survey area coverage by a factor of 2 for an energy budget that is a factor of 2 less than planned.

An effective on-board knowledge-based agent that can autonomously replan an optimal mission to intelligently adapt to an unexpected energy shortage has not been previously reported—especially one with considerations for side scan sonar requirements.

Acknowledgment

This work was supported in part by Defence R&D Canada.

References

[1] A. D. Waite, *SONAR for Practising Engineers*, John Wiley & Sons, New York, NY, USA, 3rd edition, 2002.

[2] T. Crees, C. Kaminski, J. Ferguson et al., "Preparing for UNCLOS—an historic AUV deployment in the Canadian high arctic," in *Proceedings of the Oceans / MTS Conference*, p. 8, 2010.

[3] J. G. Hawley and G. T. Reader, "A knowledge-based aid for the selection of autonomous underwater Vehicle Energy Systems," in *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology (AUV '92)*, pp. 177–180, 1992.

[4] O. Hasvold, K. H. Johansen, and K. Vestgaard, "The alkaline aluminium hydrogen peroxide semi-fuel cell for the hugin 3000 autonomous underwater vehicle," in *Proceedings of the Workshop on Autonomous Underwater Vehicles (AUV '02)*, pp. 89–94, June 2002.

[5] I. Yamamoto, T. Aoki, S. Tsukioka et al., "Fuel cell system of AUV Urashima," in *Proceedings of the MTS/IEEE Oceans Conference*, pp. 1732–1737, November 2004.

[6] Z. Chang, Z. Tang, H. Cai, X. Shi, and X. Bian, "GA path planning for AUV to avoid moving obstalces based on forward looking sonar," in *Proceedings of the 4th International Conference on Machine Learning and Cybernetics (ICMLC '05)*, pp. 1498–1502, August 2005.

[7] H. Kawano and T. Ura, "Navigation algorithm for autonomous underwater vehicle considering cruising mission using a side scanning SONAR in disturbance," in *Proceedings of the MTS/IEEE Oceans*, vol. 1, pp. 403–440, November 2001.

[8] A. Kim and R. M. Eustice, "Toward AUV survey design for optimal coverage and localization using the Cramer Rao lower

bound," in *Proceedings of the MTS/IEEE Oceans Conference*, pp. 1–7, Biloxi, Miss, USA, October 2009.

[9] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004.

[10] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, pp. 4265–4270, 2007.

[11] G. Yang and R. Zhang, "Path planning of AUV in turbulent ocean environments used adapted inertiaweight PSO," in *Proceedings of the 5th International Conference on Natural Computation (ICNC '09)*, pp. 299–302, August 2009.

[12] K. Carroll, S. McClaran, E. L. Nelson et al., "AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple overlapping, time-dependent exclusion zones," in *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology (AUV '92)*, pp. 79–84, Washington, DC, USA, 1992.

[13] W. Hong-jian, A. Jie, B. Xin-qian, and S. Xiao-cheng, "An improved path planner based on adaptive genetic algorithm for autonomous underwater vehicle," in *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA '05)*, pp. 857–861, Niagara Falls, Canada, 2005.

[14] Z. Chang, M. Fu, Z. Tang, and H. Cai, "Autonomous mission management for an unmanned underwater vehicle," in *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA '05)*, pp. 1456–1459, August 2005.

[15] J. Thorliefson, T. Davies, M. Black et al., "The theseus autonomous underwater vehicle: a Canadian success story," in *Proceedings of the IEEE/MTS Oceans Conference and Exhibition*, pp. 1001–1008, 1997.

[16] J. Feldman, "DTNSRDC Revised standard submarine equations of motion," Tech. Rep. DTNSRDC/SPD-0393-09, p. 31, 1979.

[17] M. L. Seto and G. D. Watt, "Dynamics and control simulator for the THESEUS AUV," in *Proceedings of the 10th International Conference of Society of Offshore and Polar Engineers*, p. 6, Montreal, Canada, 2000.

[18] M. L. Seto, "An agent to optimally re-distribute control in an underactuated AUV," *International Journal of Intelligent Defence Support Systems*, vol. 4, no. 1, pp. 3–19, 2010.

[19] ISE Ltd., Design an Explorer, 2000.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.

[21] A. Michaelewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, AI, Springer, New York, NY, USA, 1994.

[22] H. Li, A. Popa, C. Thibault, and M. Seto, "A software framework for multi-agent control of multiple autonomous underwater vehicles for underwater mine counter-measures," in *Proceedings of the IEEE International Conference on Autonomous and Intelligent Systems (AIS '10)*, pp. 1–6, Povoa de Varzim, Portugal, June 2010.

[23] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Transactions on Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, 1986.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2012, Article ID 375642, 18 pages
doi:10.1155/2012/375642

Research Article

3D Assembly Group Analysis for Cognitive Automation

Christian Brecher,[1] Thomas Breitbach,[1] Simon Müller,[1] Marcel Ph. Mayer,[2] Barbara Odenthal,[2] Christopher M. Schlick,[2] and Werner Herfs[1]

[1] *Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, 52074 Aachen, Germany*
[2] *Institute for Industrial Engineering and Ergonomics, RWTH Aachen University, 52062 Aachen, Germany*

Correspondence should be addressed to Thomas Breitbach, th.breitbach@wzl.rwth-aachen.de

Received 15 July 2011; Accepted 8 November 2011

Academic Editor: Ivo Bukovsky

A concept that allows the cognitive automation of robotic assembly processes is introduced. An assembly cell comprised of two robots was designed to verify the concept. For the purpose of validation a customer-defined part group consisting of Hubelino bricks is assembled. One of the key aspects for this process is the verification of the assembly group. Hence a software component was designed that utilizes the Microsoft Kinect to perceive both depth and color data in the assembly area. This information is used to determine the current state of the assembly group and is compared to a CAD model for validation purposes. In order to efficiently resolve erroneous situations, the results are interactively accessible to a human expert. The implications for an industrial application are demonstrated by transferring the developed concepts to an assembly scenario for switch-cabinet systems.

1. Introduction

One of the effects of globalization in public view is the reduction of production in high-wage countries especially due to job relocation abroad to low-wage countries, for example, towards Eastern Europe or Asia [1–3]. Based on this, a competition between manufacturing companies in high-wage and low-wage countries typically occurs within two dimensions: value-orientation and planning-orientation. Possible disadvantages of production in low-wage countries concerning process times, factor consumption and process mastering are compensated by low productive factor costs.

In contrast, companies in high-wage countries try to utilize the relatively expensive productivity factors by maximizing the output (economies of scale). Another way to compensate the arising unit cost disadvantages is customization or fast adaptation to market needs (economies of scope), even though the escape into sophisticated niche markets does not seem to be a promising way for the future anymore.

Within the dimension planning-orientation companies in high-wage countries try to optimize processes with sophisticated, investment-intensive planning approaches, and production systems while value-orientation offers the benefit of shop floor-oriented production with little planning effort. Since processes and production systems do not exceed the limits of an optimal operating range, additional competitive disadvantages for high-wage countries emerge.

In order to achieve a sustainable competitive advantage for manufacturing companies in high-wage countries with their highly skilled workers, it is therefore not promising to further increase the planning orientation of the manufacturing systems and simultaneously improve the economies of scale. The primary goal should be to wholly resolve the so-called polylemma of production, which is analyzed in detail by Klocke [4]. Economies of scale and economies of scope must be maximized at the same time, while additionally the share of added-value activities must be further maximized without neglecting the planning quality. Therefore, according to the "law of diminishing returns" a naive increase in automation will likely not lead to a significant increase in productivity but can also have adverse effects. According to Kinkel et al. [5], the amount of process errors is in average significantly reduced by automation, but the severity of potential consequences of a single error increases disproportionately. These "Ironies of Automation" [6] which were identified by Lisanne Bainbridge as early

as 1987 can be considered as a vicious circle [7], where a function that was allocated to a human operator due to poor human reliability is automated. This automation results in higher function complexity, finally increasing the demands on the human operator for planning, teaching, and monitoring, and hence leading to a more error-prone system.

In order to break the cited vicious circle, one essential step is the application of cognitive control mechanisms by means of simulation of human cognition within the technical system. Such cognitive production cells can generally be understood as a further development of autonomous production cells. Admittedly, autonomous production cells only possess limited abilities in self-optimization and self-adaption to changing production tasks. These abilities are the fundamental approach of cognitive production cells and are currently one challenge in research and development [8]. Based on these functions, the concept of cognitive automation was introduced by Onken and Schulte in 2010 [7]. However, their original concept was strongly influenced by the research field of unmanned vehicles. The corresponding concept of the "cognitive plant" by Zaeh et al. [9] transfers the cognitive approach onto production systems. This concept successfully integrates cognitive mechanism into manufacturing systems, but the superior subject of using cognitive modules including the "human factor" as an operator and surveillant, however, still remains unexplored. Based on artificial cognition, technical systems shall not only be able to (semi)autonomously perform process planning, adapt to changing manufacturing environments or objectives, and be able to learn from experience, but also to simulate goal-directed human behavior and therefore significantly increase the conformity with operator expectations. Within this focus, a highly debated issue is the software architecture of a cognitive system. For this purpose, various architectures were proposed as a basic framework for the simulation of cognitive functions [10, 11]. Herein, a popular approach is the three-layer model with a cognitive, an associative, and a reactive layer of regulation [12, 13]. Comparative structures can be found within the Collaborative Research Centre 614 "Self-optimizing concepts and structures in mechanical engineering" (CRC 614) [14] as well as within the "cognitive controller" at the Technical University of Munich [15, 16]. Further broad researches within the field of cognitive technical systems can be found in Onken and Schulte [7] and, with special focus on the production environment, in Ding et al. [17]. Herein, the implementation of cognitive abilities within security systems for plant control focused. In this context, especially the safety of human-machine interaction and safety at work are taken into account. Additional concepts and methods can be found within the automotive sector as well as within space and aeronautic research [18, 19].

Within the Cluster of Excellence "Integrative Production Technology for High-Wage Countries" at RWTH Aachen University, a Cognitive Control Unit (CCU) and its ergonomic user-centered human machine interface are developed for a robotized production unit [20–23] which partially transfers non-value-adding planning and implementation tasks of the skilled worker to the technical cognitive system. The validation and interaction with the human expert requires a technical recognition system which is able to measure the current state of the environment and provide feedback upon that information. This system needs to be integrated into the software architecture of the CCU.

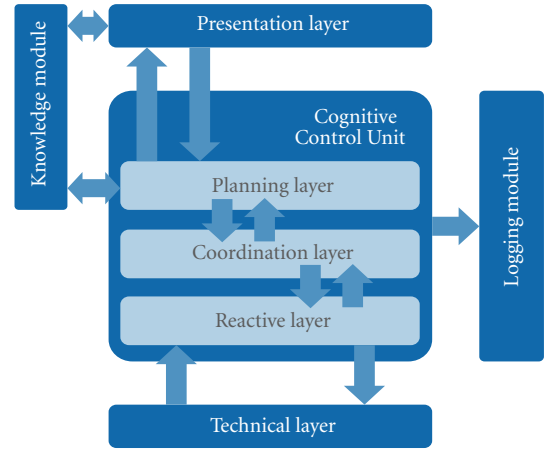


Figure 1: Concept of the cognitive architecture.

2. Cognitive Control Unit and Evaluation Scenario

Due to the fact that the CCU is a system, which needs to be comprised of several different software components, the design required a modular structure. It allows for an integration and enhancement through distributed software modules.

2.1. Cognitive Architecture. The software architecture of the CCU is separated into five layers and shown in Figure 1. The presentation layer incorporates the human machine interface and an interface for editing the knowledge base. The planning layer is the deliberative layer in which the actual decision for the next action in the assembly process is made. The services that the coordination layer provides can be invoked by the planning layer to start action execution. The reactive layer is responsible for a low response time reaction of the whole system, for example, in order to efficiently respond to emergency situations. The knowledge module contains the necessary domain knowledge of the system in terms of production rules.

2.2. Human Operator. In regard to the role that humans play in standard automated production, the main task involves managing and monitoring the manufacturing system. In the advent of malfunction, they must be able to take over manual control and return the system to a safe, productive state. This concept, termed "supervisory control" by Sheridan [24], involves five typical, separate subtasks that exist in a cascading relationship to one another: plan, teach, monitor, intervene, and learn.

After receiving an (assembly) order, the human operator's first task usually involves planning the assembly process.

To do so, he or she must first understand the functions in the relevant machine and the physical actions involved to be able to construct a mental model of the process. Using this basic understanding, the operator then develops a concrete plan that contains all specific subtargets and tasks necessary. "Teaching" involves translating these targets and tasks into machine-readable format—for example, NC or RC programs—which allow for a (partially) automated process. The resulting automation must be monitored to ensure that it runs properly and generates products of the desired quality. The expectations for the process are drawn from the mental model the operator created at the start. In cases where reality significantly deviates from this model or where there are anomalies, the human operator can intervene for example, by modifying the NC or RC program or by manually optimizing the process parameters. Ultimately, every intervention involves the human operator continually adapting his/her mental model, while existing process information, characteristic values, and trend analyses help the operator to better understand the process and develop a more detailed mental model.

With a cognitively automated system, the tasks change gradually, but in a conceptually relevant way. In this system, the human operator defines the assembly tasks based on the status of the subproduct or end product, carries out adaptations or sets priorities as needed, compiles rough process plans, and sets initial and boundary conditions. The information-related stress on the human operator is considerably reduced in the areas of detailed planning and teaching, since they are handled by the cognitive system. But shifting this load from the human to the machine can result in the human operator forming an insufficient mental model of the state variables and state transition functions in the assembly process. In order to ensure the conformity with the operator's expectations during the supervision of the assembly process [25], the first step is the use of motion descriptors to plan and execute the assembly process, since motions are familiar to the human operator from manually performed assembly tasks [26]. Therefore, the Methods-Time Measurement (MTM) system as a library of fundamental movements was chosen [25, 27]. Even though the sequence of fundamental movements (e.g., reach, grasp, move, position, release) is explainable a posteriori, the sequence of parts positioned after another is not predictable a priori due to a lack of elaboration knowledge [22]. Odenthal et al. [28] and Mayer et al. [29] identified human assembly strategies that were formulated as production rules. When the reasoning component is enriched with these human heuristics, a significant increase of the robot's predictability when assembling the products can be achieved.

Further, if an error occurs which the system cannot identify or solve, the human operator must receive all information relevant to the situation in an easily understandable form so that he/she can intervene correctly and enable system recovery.

2.3. Assembly Scenario. To test and develop a CCU in a near-reality production environment in a variety of different assembly operations, a robotic assembly cell was set up [21].

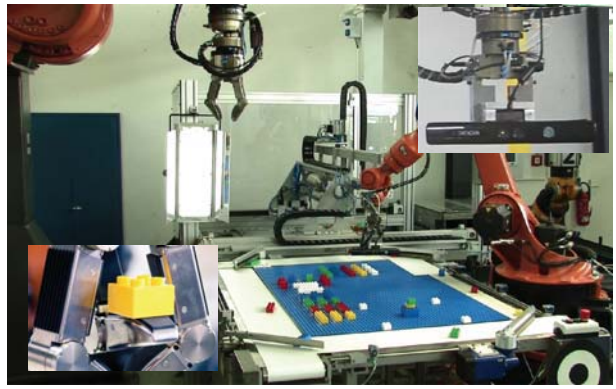


Figure 2: Assembly cell.

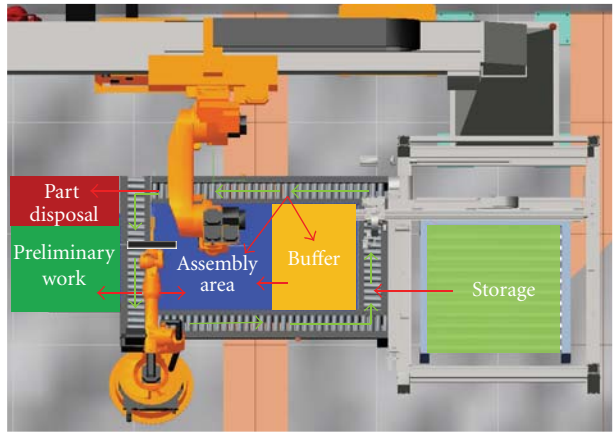


Figure 3: Assembly and storage areas in the assembly cell.

The layout of this cell is shown in Figure 2. The scenario was selected to address major aspects of an industrial application ("relevance") and at the same time to easily illustrate the potential of a cognitive control system ("transparency") [30].

2.3.1. General Setup of the Assembly Cell. The main function of the demonstrator cell is the assembly of predefined objects. Part of the cell is made up of a circulating conveyor system comprising six individually controllable linear belt sections. Several photoelectric sensors are arranged along the conveyor route for detection of components. Furthermore, two switches allow components to be diverted onto and from the conveyor route. Two robots are provided, with one robot travelling on a linear axis and carrying a tool (a flexible multifinger gripper) and a color camera. Several areas were provided alongside the conveyor for demand-driven storage of components and as a defined location for the assembly (see Figure 3). One area is provided for possible preliminary work by a human operator. This is currently separated from the working area by an optical safety barrier. The workstation has a multimodal human-machine interface that displays process information ergonomically, allowing it to provide information on the system state as well as help for solving problems, if necessary. To simultaneously achieve

a high level of transparency, variability, and scalability in an (approximate) abstraction of an industrial assembly process, building an assembly of Hubelino bricks was selected as the assembly task. These are in size and shape very similar to LEGO Duplo bricks. To take into account the criterion of flexibility for changing boundary conditions, the bricks are delivered at random. In terms of automation components, the system consists of two robot controllers, a motion controller, and a higher-ranking sequencer.

The initial state provides for a random delivery of required and nonrequired components on a pallet. A FESTO handling system successively places the components onto the conveyor. The automatic-control task now consists of coordinating and executing the material flow, using all the technical components in a way such that only the assembled product is in the assembly area at the end.

2.3.2. Actions and Sequences. The assembly scenario is as follows. An engineer has designed a mechanical assembly of medium complexity by composing it, for example, with a CAD system containing any number of subcomponents. The human operator assigns the desired assembly goal to the cognitive system via the presentation layer (see Figure 1). The desired goal is transferred to the planning layer where the SOAR-based reasoning component derives the next action based on the actual environmental state (current state on the conveyor, the assembly area, and the buffer) and the desired goal. The environmental state is based on the measured vector from the sensors in the technical application system (TAS). In the coordination layer the raw sensor data is aggregated to an environmental state. The next best action derived in the planning layer is sent back to the coordination layer, where the abstract description of that action is translated into a sequence of actor commands which are sent to the TAS. There, the sequence of commands is executed and the changed environmental state is measured again by the sensors.

2.3.3. Motivation for Assembly Group Analysis. The last step, an image-based recognition process of the assembly object's state in the assembly area, is focused on this contribution and is described in detail later on. If the current state differs from the target state, the human operator is informed so that he/she can detect and correct occurred errors.

Generally four types of errors are possible, when positioning a brick in the assembly group:

(1) It might occur during assembly that a brick is not placed in the assembly group at all. This is the case, for example, when the gripper loses the brick during the transportation from conveyor to assembly area.

(2) A generated assembly sequence might not be correct and a brick is placed at a false location. In practice, this error has never occurred, but its existence needs to be considered.

(3) The brick is placed at the correct position, but not fitted properly. This error case refers to possible tolerances for both the brick and the position of the robot. For example, in situations where a brick has to be positioned between two other bricks, the accuracy of the robot's position might not be sufficient. For the most part this leads to a part lying on top of the assembly board rather than being assembled onto that board.

(4) At last it is also possible that a brick has correctly been placed and fitted at the right location—but it was the wrong brick, for example, a blue brick instead of a green one. These errors are mostly related to the image processing system, which detects the moving part on the conveyor, failing at high conveyor speeds.

It can be stated that in practice most of the errors that occur during the course of the assembly are related to the failure of some sort of component. In order to efficiently interact with a human expert, the type of error needs to be identified. Since error type 1 should be reflected in the data as a whole set of missing data points and for error type 3 only as a small shift at a specific location, it becomes apparent that the reliability of this classification varies for the different error types.

The human operator must be supported with more information in case of an assembly error during the operation, for example, if the image-based control of the assembly step leads to a deviation between the current and the target state. Under this assumption, a first prototype of a supporting system was developed dealing with the task of error identification in an assembly object (incorrect construction of the assembly object). More precisely in this case, a prototype of an augmented vision system (AVS) was developed and implemented with the focus on the presentation of the assembly information. The aim of using this system was to place the human operator or skilled worker, respectively, in a position to detect the construction errors in a fast and adequate way. Therefore, a laboratory test was carried out in order to investigate different display types and different modes of visualizable assembly information from an ergonomic point of view [23, 31, 32]. Within a second step, the AVS was extended to assist the human operator in the disassembly of the erroneous object in order to correct the detected error in cooperation with the robot [33].

While a detailed overview of the cognitive control can be found in Kempf [30], there was still the need for an automated verification of the assembly group. In the following, the technical recognition of an assembly group within the assembly area of the cognitively automated assembly cell is described in detail.

3. Recognition Process

In this scenario, individual assembly groups are established which consist of an arbitrary combination of different basic elements. For demonstration, the actual assembly groups consist of Hubelino parts which differ in color (yellow, orange, red, light green, dark green, and blue) and length. The length of each part varies from 32 mm to 192 mm in steps of 32 mm. Each element has a width of 32 mm and a height of 16 mm.

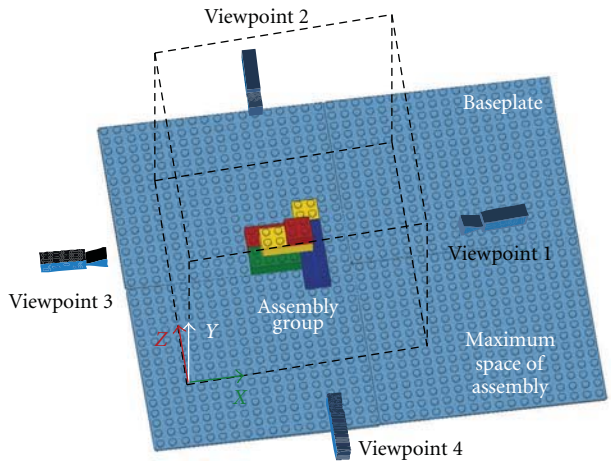


FIGURE 4: General design of the recognition process.

Since the Hubelino parts are plugged onto the baseplate, respectively, onto each other, their positions are defined by the round shapes on top of each Hubelino part (see Figure 4). Each Hubelino part is laminated with a glossy surface that has a high light-reflecting coefficient.

3.1. Technical Recognition Systems. Within the recognition process, the assembly has to be checked if it is constructed correctly. Therefore, the assembly group is analyzed from four different viewpoints using a contactless 3D measurement system. Figure 4 shows the general design of the scenario. The dashed cuboid defines the maximum space of assembly.

At each position the Hubelino brick, that is present in the real assembly group, potentially differs in color as well as in size from the corresponding part defined in the virtual model. Thus information about these two properties needs to be measured for all of the possible assembly positions.

3.1.1. Contactless Measurement Methods. In the past few years, contactless 3D measurement systems have become an important tool for quality control. The two most used contactless measurement methods [34] are (i) triangulating the image data with several color images, (ii) time-of-flight method.

Within the first method, a scene is inspected from several viewpoints and identical "landmarks" inside each color images are identified (e.g., edges or corners). The pixel position of those landmarks combined with the camera's intrinsic calibration (relation between a pixel position and the directions relative to the camera center) yields in the vectors pointing from the camera center to the direction of each landmark. Determining this vector in at least two camera images and considering the relative position between these cameras, the special position of this landmark can be triangulated (compare Figure 5(a)). For this measurement method, at least two RGB cameras are necessary as well as an accurate spatial transformation T_{ij} between the cameras. The disadvantages of this technology are the huge computing

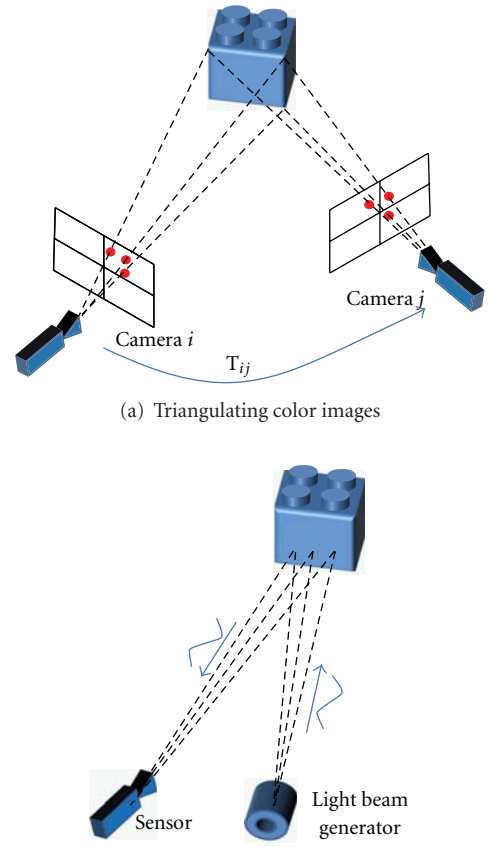


FIGURE 5: Principle of both contactless 3D measurement techniques.

power necessary to calculate several high-resolution color images as well as finding concurrent and unique landmarks in each image. Consequently, this measurement method fails in image regions with a structured surface. Details concerning this technology are described in [34].

The second widespread contactless measurement method is the time-of-flight technique (see Figure 5(b)). Here, the time that a frequency-modulated beam of light needs to reach an object and to get back to the sensor is measured indirectly by comparing the phase of the emitted light with the phase of the received signal. For that reason, this method requires a device emitting the light beam as well as a light sensitive sensor detecting the reflected light beam. Accordingly, this method depends on the light-reflecting coefficient of the measured surface. Hence, objects with a low reflectivity coefficient (i.e., windows) are not detected as well as objects with a very high reflectivity coefficient (i.e., bright surfaces).

Widely used types of application for this technology are 2D laserscanners. Within 2D-laserscanners, a single light beam is diverted by a mirror. Concerning the angular position of the mirror and the time of flight of the single light beam, the depth values of a line can be measured. By panning the 2D-laserscanner, a complete 3D depth image of the environment can be computed. Recently, time of flight sensors are available measuring the depth values not only of

Table 1: Sensor requirements.

Requested attribute for assembly group analysis	Available for the 3D time-of-flight method	Available for triangulating color images
Depth image and color image available	Only depth information available	Both supported
High accuracy of the depth image	About 1 cm	Depends on resolution and calibration accuracy; recently ca. 1 mm
Depth image independent of the surface	Depends on light-reflecting coefficient	Fails in region with structured surfaces
Mapping between color and depth image	Not available	Available for each camera
High resolution of the depth image	Up to 320 × 240 Pixels	Equals the number of identified landmarks
Dependency on ambient light	Independent	Fails if scene is under/overexposed

a single point or line, but also of a complete matrix. Hereby, a 3D depth image is measured directly without requiring mechanical parts for panning the sensor [35, 36].

Concerning the recognition process, both described measurement methods have disadvantages. For the recognition both the color and the exact spatial dimensions of the assembly are considered. Therefore a color image as well as an accurate 3D depth image and the mapping between the two are required. In Table 1, the requirements for analyzing an assembly are faced with the characteristics of both contactless measurement methods.

According to Table 1 the main disadvantages of the 3D time-of-flight method are that it does not meet the requirements in terms of accuracy for depth data and that it does not provide a color image. However, compared to the triangulation method, it provides a higher measurement frequency and is independent of ambient light.

Still, both measurement methods are not completely independent of the surface of the measured objects. On the one hand, the triangulation method cannot detect the depth values of structured surfaces, while on the other hand the time-of-flight method relies on the reflectivity of the surface.

However, the actual measurement task requires the detection of the color and the exact spatial dimensions of small devices with a completely structured and glossy surface. Hence a combination of both measurement methods is required.

3.1.2. The Kinect as a Recognition Device. Within the field of 3D measuring, the Kinect sensor, developed by Microsoft and PrimeSense, was presented to the public. Initially developed for game consoles, this device combines a 3D depth sensor with a resolution of 640×480 pixels and a standard color camera with a resolution up to 1280×1024 pixels. Additionally, a microphone array, a position sensor which measures the vector of gravitation, and an electrical motor for tilting the unit are integrated. Table 2 gives a short overview of the relevant specifications [37]. Combining the measured data of those devices, the Kinect sensor provides multiple innovative opportunities for research and development within the field of environment recognition as well as in the field of man-machine interaction [38].

The Kinect's 3D depth sensor combines aspects of both contactless 3D measurement methods described above, whereby the scene is continuously illuminated with infrared structured light [39]. Structured light measures a 3D scene

Table 2: Specifications of the Kinect.

Device	Specifications
Color sensor	32 bit RGB 1280 × 1024 at 15 fps
Depth sensor	16 bit Depth 640 × 480 at 30 fps
Depth sensor range	650 mm–3500 mm
Spatial Resolution (at 2 m distance)	*x/y* resolution: 2 mm, *z* resolution: 1 cm
Field of view	Horizontal 57 degree, vertical 43 degree
Tilt range	27 degree
Microphones	2 microphones, each 16 bit audio at 16 kHz

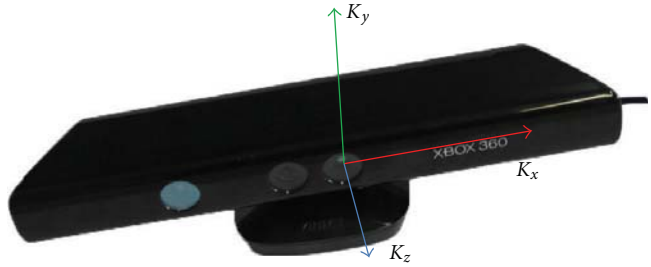


Figure 6: Coordinate system defined by OpenNI.

by projecting a known pattern of light onto the environment and recording it with a standard camera. The way this pattern deforms when striking surfaces allows calculating the depth information of the objects in the environment. Song [40] provides a detailed description of the computations. The Kinect projects a matrix of single IR dots onto the scene and provides a depth image with an accuracy of 1 cm at 2 m distance.

In order to accomplish the assembly group analysis, only depth and color information of the Kinect are merged. Communication and data exchange with the Kinect sensor are realized with the official driver from PrimeSense and its modifications for OpenNI. OpenNI uses a right-hand coordinate system for processing the depth data (see Figure 6).

As the Kinect depth sensor illuminates the room with an "IR Light Coding" and evaluates the reflected image,

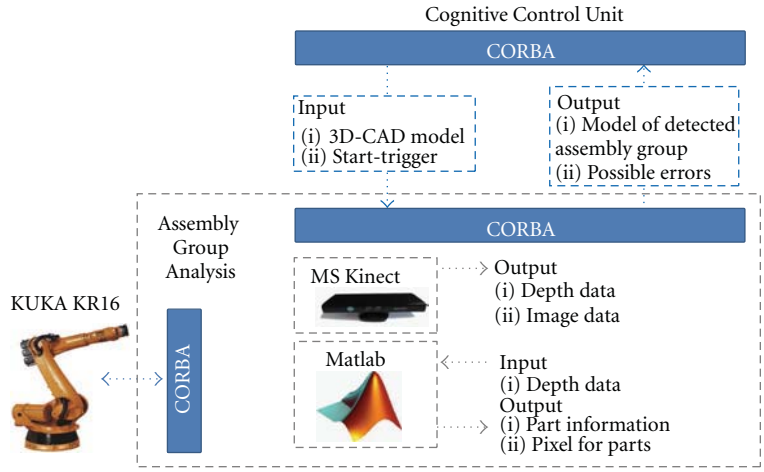


Figure 7: Software structure of the AGA.

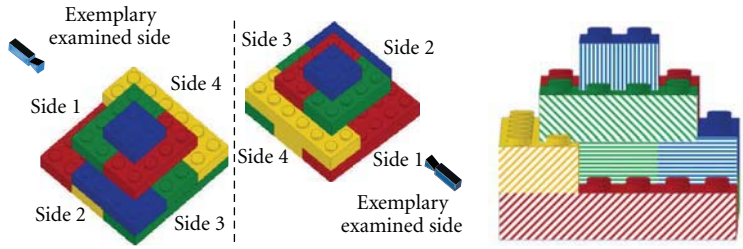
the efficiency depends on the reflectivity coefficient of the measured surface. Additionally, the correct operation of the Kinect depth sensor relies on the incidence angle between the emitted light beam and the measured object. If the incidence angle is undersized, the light beam cannot be reflected back to the sensor with a sufficient intensity to be detected. The other way round, an incidence angle, which is nearly 90 degrees on an object with a high reflectivity coefficient, reflects the light beam with a very high intensity. This combination results in a cross-talk between adjacent pixels, whereby a blind spot within the Kinect's depth data occurs. As the blind spot occurs only under the described conditions, these need to be taken into consideration when the Kinect is used for analyzing an object. However, since the Kinect allows for a seamless integration of both depth and color data, it presents an appropriate device for the analysis of an assembly group.

3.2. Assembly Group Analysis (AGA). In order to create a perceived model, the assembly group is observed from multiple perspectives. For the generation of this model two approaches will be presented. In the first approach only general assumptions regarding the structure of a Hubelino model will be used. This approach is able to create a perceived model for any Hubelino model without previous knowledge about the bricks that should have been assembled, but is only valid in Hubelino scenarios where the general assumptions can be applied. A second approach takes advantage of the given customer-defined CAD data to derive recognizable patterns. On the one hand this approach is not able to construct a perceived model, if no CAD data is present. On the other hand it eliminates the general assumptions and thus is applicable not only in Hubelino scenarios, but also in any scenario, provided the condition that CAD data exists. Both approaches include the perception of both depth and color data and perform error detection. The superior goal of verifying a real assembly group of Hubelino bricks is achieved by comparing the perceived model with the virtual model from the customer-defined CAD file. The AGA is implemented as a separate software component within the framework of the Cognitive Control Unit.

3.2.1. Software Structure of the AGA. In order to communicate with multiple components without having to implement a new interface for every new component, an abstract interface was used, which is able to transparently connect multiple devices. This abstract interface uses the CORBA Middleware and thus provides a real-time connection between devices and the CCU. This leads to an architecture, where the commands that the CCU issues are translated into CORBA calls that are directed to a specific device, in this case the AGA component. Hence the AGA component needs to provide interfaces to the CCU, which allow for a comprehensive communication. Since Matlab is a common tool for data visualization and the computation of complex algorithms, it was used to perform the computationally intensive tasks. A complete overview of the resulting architecture and their connections is provided in Figure 7.

In order to be able to verify the correct composition of an assembly group, the AGA component needs to know what the correct model looks like. Thus the CCU needs to submit a CAD file of the assembly group's expected state to the analysis component. This communication takes place for every verification, since it is possible for a model to change over time. The CAD model is given as an LDRAW file, which is an appropriate file format for the assembly of Hubelino parts [41]. After the analysis is completed, the AGA provides feedback to the presentation layer of the cognitive control framework (see Figure 1). This feedback contains the detected parts within the assembly group as well as possible errors. This allows for the CCU to present the results in the presentation layer, hence enabling a human operator to take action based on the received results.

In order to calculate a 3D model with a single sensor, the sensor needs to change its position relative to the object to examine all sides. Consequently, the Kinect sensor is attached to the flange of the KUKA KR 16 which can explore the assembly from an arbitrary position. Once the CCU requests a new



(a) Sample assembly group

(b) Viewpoint of the Kinect sensor

Figure 8: Assembly group.

analysis, the AGA component creates a connection to the Kinect as well as to the KUKA KR16 via CORBA. Within the analysis process, the assembly is examined from four different positions (compare Figure 4). Each viewpoint is aligned perpendicular to the particular side of the assembly with the intention of avoiding perspective masking of single Hubelino parts. For each viewpoint, the assembly group is analyzed and a discrete model for each side is calculated. Afterwards, these four discrete models are combined into one complete model of the assembly group.

3.2.2. Data Acquisition. The following description of the recognition process is based on the assembly group as shown in Figure 8. For demonstration purposes, side 1 is examined.

Figure 8(a) displays the exemplarily used assembly from two different points of view. Figure 8(b) represents the same assembly but presented from the viewpoint of the Kinect sensor. Concerning the Kinect's viewpoint, the Hubelino parts constitute planes parallel to the Kinect's x-y-layer. Those planes are marked by different striped patterns.

For the data acquisition, the positioning of the viewpoints is essential, thus the sensor characteristic needs to be taken into account. As described above, the Kinect sensor fails if the emitted light beam is reflected perpendicularly by an object with a high reflectivity coefficient. Within the given task, all objects to be recognized have a high reflectivity coefficient. Therefore, each viewpoint needs to fulfill the following conditions:

(1) The distance to the measured object has to be greater than 650 mm.

(2) The light beams should not be reflected perpendicularly.

(3) The particular side of the assembly group needs to be examined perpendicularly to avoid perspective masking of single Hubelino bricks.

Consequently, each viewpoint is at least 700 mm away from the assembly group and the assembly group itself is about 200 mm beneath the center of the depth image. Figure 4 shows their position and chronology. This choice of viewpoints has another important effect: in those viewpoints, the baseplate does not reflect the emitted light beams, according to the undersized angle of incidence. Hence the baseplate will not be recognized by the Kinect, whereby the effort of interpreting the measured depth values is reduced.

After defining the viewpoints, the depth values need to be acquired. Thus, the maximum dimensions of the assembly group are limited by a virtual cube and only depth values within this cube are used for identifying the assembly (see Figure 4 dashed cuboid).

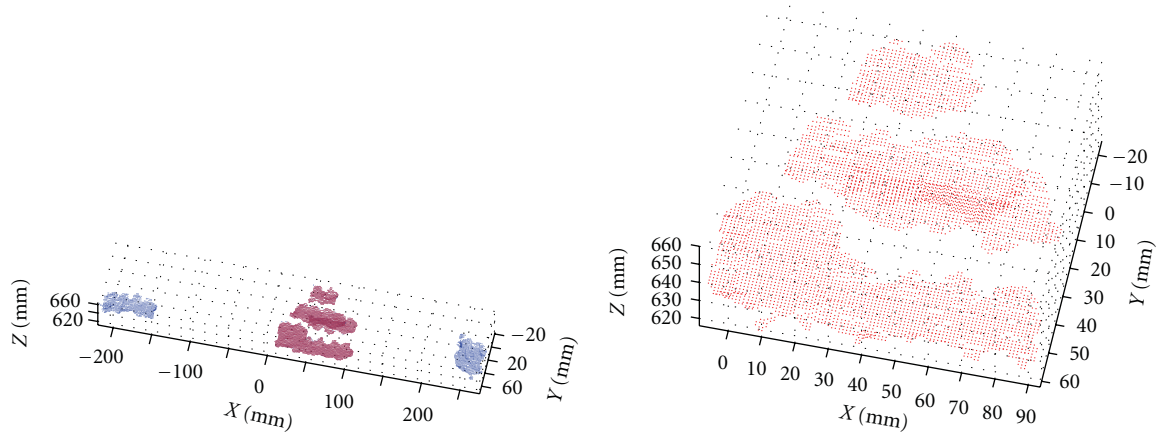
3.2.3. Prefiltering. Before starting to fit Hubelino parts, the depth data needs to be filtered. The first step is to identify the assembly group within the scatter plot. Within the defined cube, some parts of the baseplate or some object not belonging to the assembly, respectively, may be perceived by the Kinect sensor. Those objects disturb the recognition of the assembly group and need to be filtered.

Therefore, the data points are classified by their distance to each other. If the distance between two points is larger than a specified value, both points get a different classification. Otherwise they will be sorted into the same class. The distance for separating different scatter plots is 100 mm.

After passing this filter step, the data points are separated into their particular scatter plots. Based on the notion that the assembly group is the largest object within the defined cube, only the class with the most members is used for fitting the Hubelino parts. The result is shown in Figure 9. From the incoming measurement data (Figure 9(a)) only the red marked points are used for fitting the Hubelino parts (Figure 9(b)).

As described, the Hubelino parts can only be positioned in discrete places and the viewpoint of the sensor is perpendicular to each side of the assembly group. Hence the measured depth values are positioned on planes parallel to the sensors x-y-plane (compare Figures 8(b) and 9(b)). Since all sensors are disturbed by noise, the depth values are not measured exactly and the planes of the assembly group need to be reconstructed.

3.2.4. Identifying Planes. For this task the RANSAC algorithm is used and optimized for finding planes. The RANSAC algorithm is an iterative method to estimate parameters of a mathematical model from a set of sensor data [42, 43]. Within the scope of reconstructing planes out of disturbed depth data, the mathematical model of a plane is described by three 3D points.



(a) All depth data within the cuboid

(b) Filtered depth data

Figure 9: Prefiltering measured data.

The algorithm finds the plane as follows:

(1) Define a plane by randomly taking three points from the depth data.

(2) If this plane is parallel to the sensor's x-y-plane then continue, otherwise choose different points.

(3) Calculate the distance between each 3D point within the depth data and the randomly generated plane.

(4) Determine the score S of the plane as follows:

$$S = \sum_{i=0}^{i=n} p(\text{distance}) \quad \text{with } p = \begin{cases} 1 - \dfrac{\text{distance}}{\text{maximum distance}}, & \text{if distance} \leq \text{maximum distance}, \\ 0, & \text{if distance} > \text{maximum distance}. \end{cases} \tag{1}$$

(5) Repeat steps 1 to 4 for a predefined number of iterations.

(6) The plane with the highest score is the best estimation of a plane within the depth data. If the highest plane score is beneath 80% of the theoretical minimum plane score, no plane is found.

The value assigned to the maximum distance results from the discrete positions of the Hubelino parts. As those positions are defined by the round shapes on top of each part, it is obvious that the minimum distance between two planes within the depth image is 16 mm (compare Figure 8). In order to allocate each depth value to a plane, the maximum distance between a measured point and the corresponding plane is 8 mm.

The decision, if a found plane is valid, depends on the minimum theoretical plane score. This parameter is equal to the theoretical number of measurements on the surface of the smallest possible plane (see Figure 10).

The theoretical number of depth data within a certain area yields from the sensor characteristics. As the Kinect depth sensor has a resolution of 640×480 pixels and the lens has an opening angle of 57 degree horizontally, 43 degree vertically, respectively, the spatial resolution decreases with increasing distance. This interrelationship is given in Figure 11.

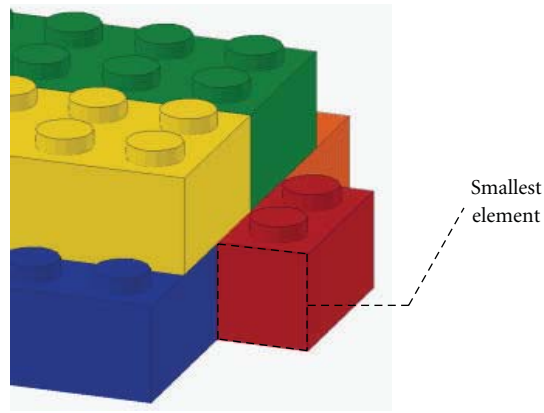


Figure 10: Definition of the smallest element.

As Figure 11 shows, the theoretical minimum plane score depends on the distance of the plane and thus needs to be determined individually for each plane. Considering the fact

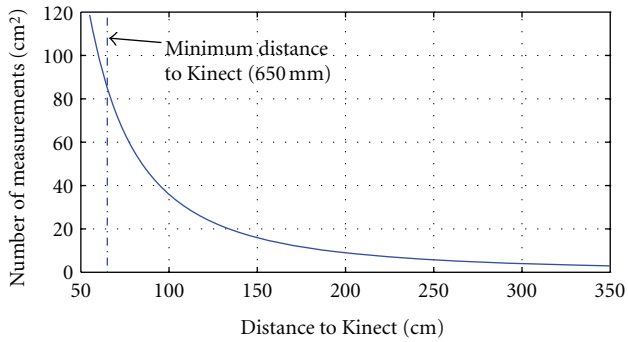


Figure 11: Interrelationship of spatial resolution and distance.

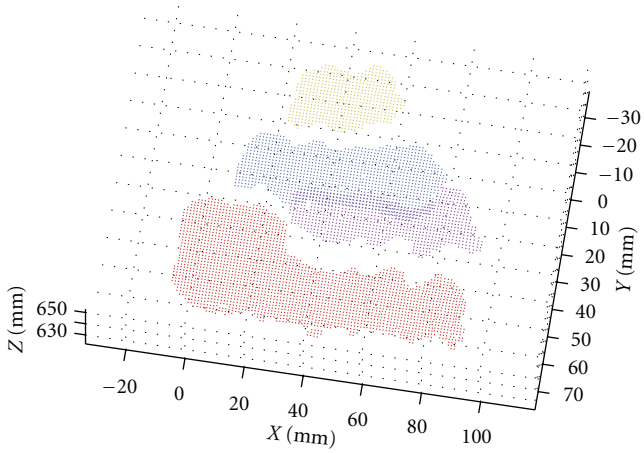


Figure 12: Found planes.

that the Kinect sensor is disturbed by noise, the found plane is valid, if its score is at least 80% of the theoretical score.

As the assembly group generally consists of more than one plane, the RANSAC algorithm has to be executed several times. Therefore the depth values belonging to the already found planes are deleted, and within the remaining depth data another plane is searched. This procedure is repeated until no further plane can be found. The result is shown in Figure 12.

3.2.5. Fitting of Virtual Hubelino Bricks. After finding the planes within the depth data, the single Hubelino parts can be fitted. Since the Hubelino parts can only be placed at discrete positions they can occlude each other. Therefore, the smallest visible element is half of one side of a 32 mm × 32 mm Hubelino brick (see Figure 10). Hence, only rectangles with a height of 32 mm and a width of 16 mm are positioned.

The fitting process itself is based on the assumption that size and shape of the found planes can only consist of multiples of the smallest elements (compare Figure 13). Hence for each plane the surrounding rectangle is calculated (light blue line). Regarding the sensor disturbance, the dimensions of this rectangle are not necessarily a multiple of the smallest element. Consequently, the size of the rectangle

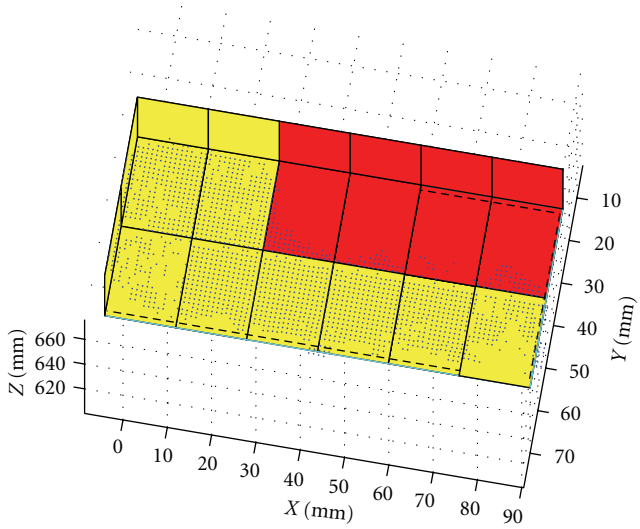


Figure 13: Fitting elements into plane number 4.

is approximated towards the nearest multiple of the smallest element's size (black dashed line). Afterwards, the resulting rectangle is filled with smaller rectangles that have the shape of the smallest elements (red and yellow cuboids). This procedure is repeated for each found plane.

Since not all of those fitted elements really exist, the next step is to decide whether an element belongs to the assembly group or not. This decision is based on the quotient of the actual number of measured 3D depth points on the element and the theoretical maximum number of data points on the element. This maximum is given by Figure 11. Since the edge line of the found planes is irregular, the quotient is generally not equal to one. Thus a threshold of 90% is defined. Concerning this threshold, the decision whether a fitted element belongs to the assembly group can be evaluated by the following steps:

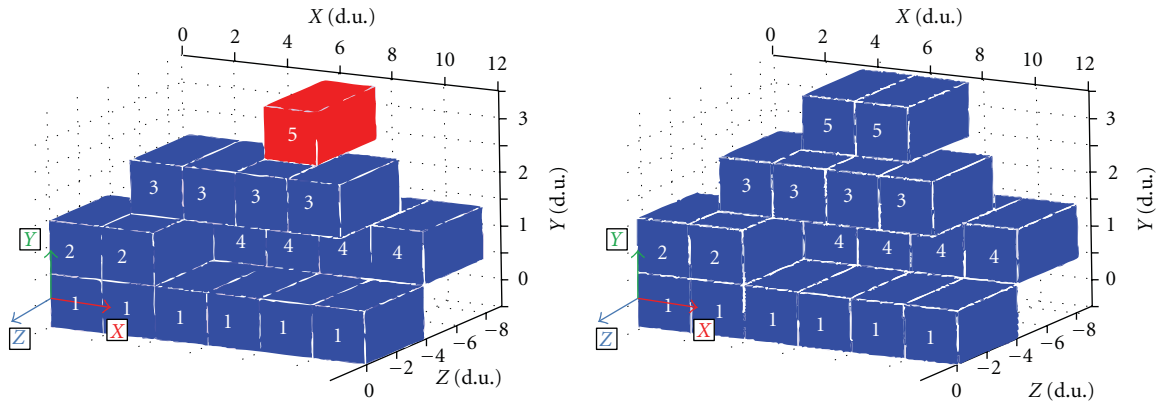
(1) Determine the number of measured depth data within the element.

(2) Based on the distance of the element calculate how many data points should be on the element.

(3) If the result of step 1 is at least 90% of the result of step 2, then the element belongs to the assembly group.

This procedure is repeated for each element on each found plane and results in a model for the examined side. Within Figure 13, the red marked cuboids do not belong to the assembly group.

3.2.6. Error Correction for a Single Side. After the Hubelino parts are positioned according to the established planes, the resulting model is checked for errors. This step proved to be necessary, after several initial evaluation tests failed due to frayed shapes of the identified planes (compare Figure 13).



(a) Defective Side 1 (b) Corrected Side 1

Figure 14: Correction of side 1.

Within the correction process, the found model has to meet the following requirements:

(1) A single smallest element can only appear if the rest of the Hubelino part is masked (compare Figure 10).

(2) Each line of the model must consist of an even number of smallest elements.

(3) All found elements have to be placed at a discrete position.

In order to check these conditions, first the coordinate system is changed. Since the presented calculations contain the measured data, they are based on the coordinate system defined by OpenNI. This measured data is represented by the fitted elements which are placed in discrete positions. Hence, the unit of the new coordinate system is given in dimension units (d.u.). Thereby, the size of two dimension units equals the width of the smallest element. The new coordinate system is placed on the lower left element of the scanned side (see Figure 14). Within this coordinate system the values of the x- and y-axis are positive and those for the z-axis are negative. In Matlab the Hubelino model is contained in an array according to discrete positions of smallest element's multiples (see Figure 10).

The error correction process is illustrated in Figure 14. First, the found element are grouped (numbers on the elements in Figure 14) based on their position within the model. Therefore, all elements within the same coordinates in y and z are grouped together. For each group with a noneven number of members (red marked element) the fitting of the element inside the plane is rechecked. Within this check, the unused elements of a plane are analyzed. Of special interest are the elements at the left and right of the defective group, respectively. Those elements meet only one conditions listed in Table 3 and result in the corresponding error correction.

Each increase as well as decrease refers to the width of an element. For each correction process only the dimension of the surrounding rectangle changes. The position of its center remains the same. The advantage of this error correction process is that it can be calculated straightforward without iteratively searching the best solution.

Table 3: Possible errors and corresponding corrections.

Error	Correction
Both elements were not chosen	Increase the width of the surrounding rectangle
Both elements were chosen	Decrease the width of the surrounding rectangle
One element was chosen	Decrease the width of the surrounding rectangle

Figure 15 demonstrates this error correction process for the defectively fitted group presented in Figure 14(a). For this figure it is obvious that the error correction finds a better mapping between the surrounding rectangle (light blue line) and the measured data.

Since the calculated model is based on disturbed sensor data, it is generally not the exact model of the actual assembly group. Hence, the last step is to reconstruct the complete 3D model of the assembly group and to check if all four sides match.

3.2.7. Reconstruction of the Complete Model. In order to reconstruct the complete model of the assembly group out of the models for each side, first the transformations between adjacent sides have to be calculated. Considering the anti-clockwise chronology of the scan positions (see Figure 4), the values of the right-hand rotation matrix to rotate side B into the coordinate system of side A are given as follows:

(i) Rotation around the x-axis: 0°.

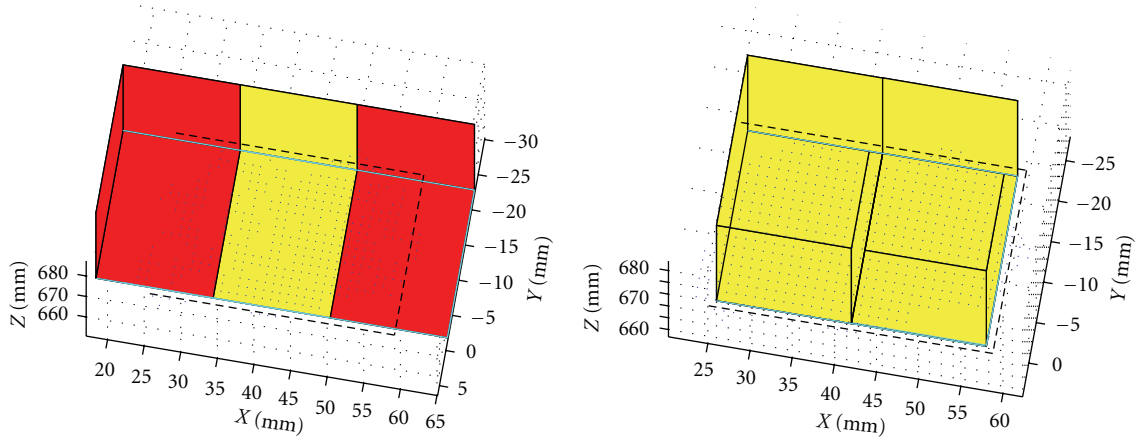
(ii) Rotation around the y-axis: 90°.

(iii) Rotation around the z-axis: 0°.

The translation vector between both sides is calculated as follows:

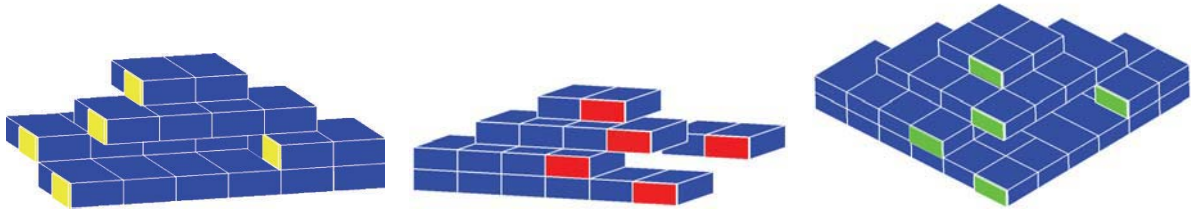
(a) Starting from the assumption that model A is correct, calculate those parts within model A that are expected to be part of side B.

(b) Within side B, calculate those elements, which side A would expect to be part of side B.



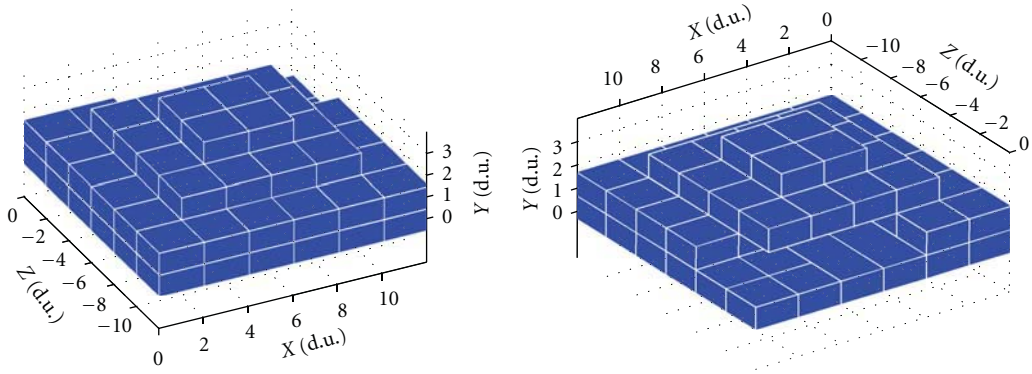
(a) Defective fitting of elements (b) Corrected fitting of elements

Figure 15: Failure correction for plane 1 within side 1.



(a) Scan of side 1 (b) Scan of side 2 (c) Fitted sides

Figure 16: Fitting of side 1 and 2.



(a) (b)

Figure 17: Reconstructed assembly group.

(c) Rotate side B into the coordinate system of side A.

(d) Find the translation vector mapping most elements of step 1 to those of step 2.

If not all elements of step (b) are mapped to those of step (a), at least one of the involved sides contains errors. Figure 16 shows this procedure for side 1 and 2 (compare Figure 8). The calculated elements of steps (a) and (b) are marked in yellow and red, respectively. Afterwards those elements are mapped onto each other by the found transformation (green elements).

Those transformations are calculated between all adjacent sides. If none of the sides contains errors, the 3D model of the complete assembly group can be calculated. The 3D model of the analyzed assembly group is shown in Figure 17.

3.2.8. Color Detection. In order to recreate an exact virtual model of the assembly group, information about the color of the detected parts needs to be acquired. In case of a scenario, that involves a color camera as the sole source of information, usually very complex algorithms have to be applied in order to efficiently detect regions of certain colors. In the given

Table 4: RGB values for the main colors used in the scenario.

	R	G	B
Red	196	40	27
Green	40	127	70
Blue	13	105	171
Yellow	245	205	47

case, however, this task is extremely simplified, since it is already known which pixels belong to a certain part. This information can be used to identify the color for a detected part. As an example, it can be assumed that an identified pixel region $x_m \cdots x_n$ contains j pixels for one side of a detected part. For the assumption that all of those pixels belong to the same part and should thus roughly contain the same color information, the basic approach of calculating the mean RGB value should suffice. Thus the resulting RGB value for the detected part can easily be calculated as:

$$cd.R = \frac{\sum_{k=m}^{n} x_k.R}{j} \qquad cd.G = \frac{\sum_{k=m}^{n} x_k.G}{j}$$
$$cd.B = \frac{\sum_{k=m}^{n} x_k.B}{j}. \tag{2}$$

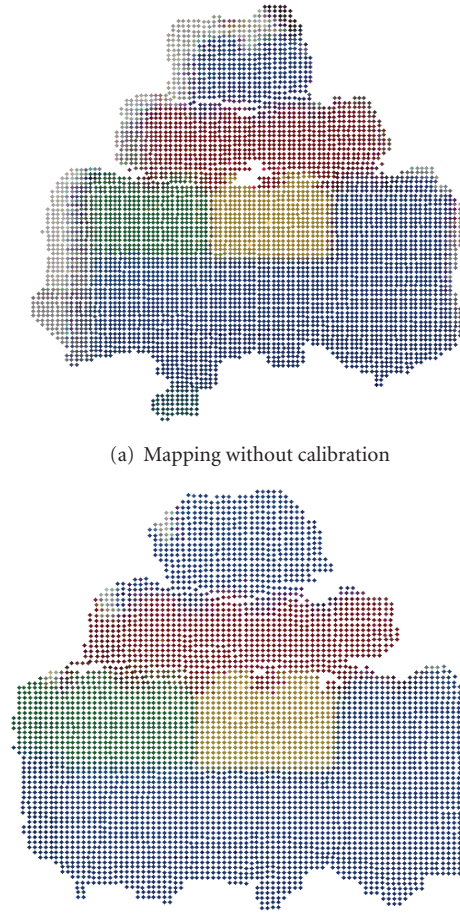
Since the RGB values for most Hubelino parts are publicly available, see for example, [44], these can be used to identify the corresponding part color. The RGB values for the parts used in this example are presented in Table 4.

Given the fact that the different colors are saved in an array z, the resulting part color can be calculated:

$$\|(z_i.R - cd.R)\| + \|(z_i.G - cd.G)\| + \|(z_i.B - cd.B)\| = \min. \tag{3}$$

However, since only four different part colors are possible, the significant distance between the different RGB values partly contributes to the efficiency of this approach. Since the RGB values are determined by the manufactured parts and thus do not relate to the specific conditions (i.e., lighting, reflection), even better results can be achieved if these factors are taken into consideration. Hence, sample values for the different colors have to be recorded manually. Since lighting is a significant factor, the best results can be achieved by recording different color values for each of the four viewpoints.

In order to achieve even better results, the calculation of the RGB values for a detected part can be improved. In the previous approach, all of the points that were considered as belonging to a specific part were used for the determination of that part's color. Inasmuch as all of those color values are roughly the same, this approach is sufficient. However, the approach can be improved if the mean RGB value for a part is calculated using only the ten most central points for a specific brick. On the one hand this approach eliminates the problem that the part's color tends to change at the part's border. On the other hand this might lead to false detections in case that local reflection takes place near a part's center.



(a) Mapping without calibration

(b) Final mapping

Figure 18: Mapped sensor data.

Given the situation that a limited solution space of colors is present, the problem of reflection can be reduced by comparing each of a part's pixels with the possible colors. A color is assigned to a pixel if

$$\|(z_i.R - cd.R)\| + \|(z_i.G - cd.G)\| + \|(z_i.B - cd.B)\| = \min \cap$$
$$\|(z_i.R - cd.R)\| + \|(z_i.G - cd.G)\| + \|(z_i.B - cd.B)\| < \delta. \tag{4}$$

Thus it is possible to dynamically apply different δ for different scenarios and lighting conditions. Subsequently, the color for a part is computed by averaging over the pixels that were assigned a specific color. This computation reduces the problem of reflection, since colors that are out of range do not contribute to the resulting part color.

As shown in Figure 18(a) another problem, that needed to be resolved, is based on the fact that there is a partial displacement between the color information and the depth information. Since the color detection algorithms rely on the assumption that for each pixel in the depth information the image data corresponds to that exact same spot in the real world, this displacement leads to miscalculations of a part's color. As the Kinect is attached to the flange of a robot,

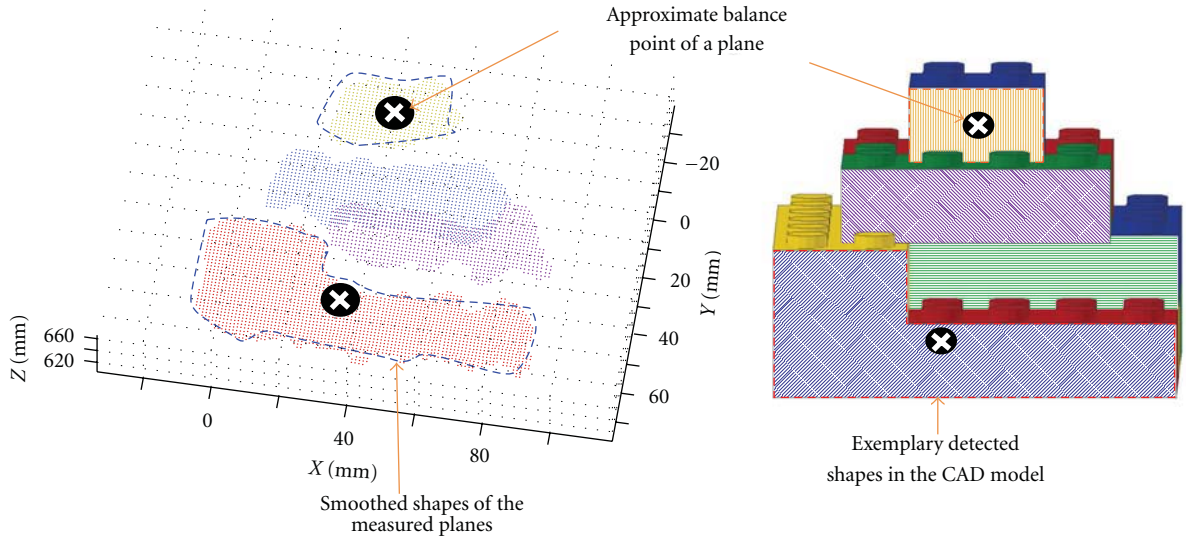


Figure 19: Model-based approach.

however, the information about the current position of the robot's tool center point (TCP) can be utilized for the exact determination of a viewpoint's displacement within the assembly area. The resulting image containing the mapping of both depth and color data is shown in Figure 18(b).

3.3. Generalized, Model-Based Approach. As a modification to the presented approach, the elimination of assumptions for the matching process offers to broaden the range of detected parts without additional engineering efforts. Due to the fact that the model needs to be verified, it is apparent that there is already a CAD model present for the desired assembly group. The superior goal is to use this model in order to match CAD and Kinect data without the need for additional assumptions regarding the presence of an object. This task is often achieved by photogrammetry methods [45, 46], but normally relies on different conditions regarding time and accuracy [47–49].

As a first step to realize this approach, the CAD model has to be analyzed in regard to different layers that can be perceived from different viewpoints. The result of this step is a "virtual model" of the assembly group that should be built. The model that is recorded with the Kinect forms the "perceived model". For this generalized approach it can be stated, that the point of view from which an assembly group is looked at in the virtual and the perceived perspective have to match to achieve the best results. But since it is always possible to correct the perspective for both the virtual and perceived model, there is no need for an exact calibration between these two perspectives. The only requirement is that the basic "sides," from which an assembly group is looked at, match. As previously stated, the four viewpoints correspond to the four sides of the rectangle that defines the maximum assembly space (see Figure 4). Thus for each of those sides the visible layers need to be extracted from the virtual model. The fact that the sides are either parallel to the x-y-layer or the y-z-layer of the modeling environment simplifies this task, inasmuch as that for the different viewpoints the polygons with the minimum or maximum x or z value, respectively, form the foremost layer for each height, which is given in the y-coordinate. As an example for viewpoint 1 the maximum x values for a given height y define the visible planes. The polygons themselves are extracted from the underlying CAD model. All of the different polygons that are adjacent to each other comprise a region. In order to utilize these regions for further processing, their shape and balance point are determined.

Once the different layers have been identified in the virtual model, the same steps as in the approach mentioned above—from recording the sensor data to the RANSAC algorithm—are performed. Hence the results of these steps are the identified regions for the perceived data. For these regions the different shapes and balance points are extracted from those results. The shape herein is defined by the outmost data points for a specific region. Figure 19 shows this process. The process is illustrated exemplarily for two out of four planes within the assembly. In the left of Figure 19 the planes, found by the RANSAC algorithm within the measured data, are presented. For both planes their smoothed shapes and balance points are shown. The right side illustrates both planes within the underlying virtual model as well as their ideal shapes and ideal balance points.

As a last step, this information about shapes and balance points is used to match the regions of the virtual and the perceived model. This is realized by first comparing the number of balance points for the virtual and perceived model. If there is a difference between those numbers then it becomes apparent that there is an error. After that the Gauss algorithm (cp. [50]) is applied to match the balance points of the different models. The algorithm completes when either the distance between the balance points is small enough or the Gauss algorithm exceeds a certain number of iterations. Next the balance points of the two models can be mapped by taking the smallest difference between two points as the main

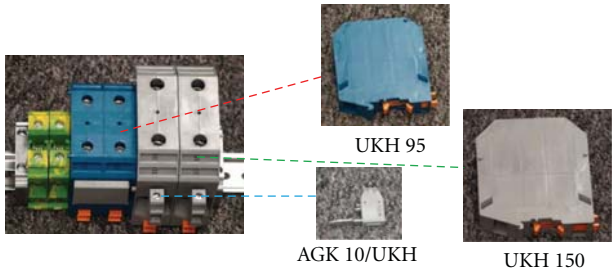


Figure 20: Final assembly.

criteria. If the numbers of balance points were different, only points that have a minimum distance are considered. In the following the regions that belong to a balance point in the perceived model are projected onto the region of the corresponding balance point in the virtual model. By calculating the overlapping areas it is possible to determine if the two models match or if it is more likely that there is a difference between them.

4. Industrial Application

In order to apply the achieved results in an industrial application, the concepts already tested with the demonstrator cell were simulated, in collaboration with automation technology manufacturer Phoenix Contact, on a scenario for the assembly of switch cabinets. This approach addresses the customer-driven switch cabinet assembly system, which Phoenix Contact operates in addition to the final assembly line in its mass-production-style manufacturing system. The assembly of switch cabinets consists of components (control units, terminals, etc.) in configurations predefined by the customer that are mounted onto top-hat sections and passed on as completed modules to switch-cabinet production (module assembly). Additionally some of these components need to be equipped with specific clip combinations. Figure 20 shows an example of this system.

A model of each customer-specific and mass-production-style switch cabinet is available in a CAD format. Currently this data is used to derive assembly instructions which are printed out and handed to a human worker. Finally the manually assembled switch cabinet is analyzed by an image recognition system and the results are compared to the mounting requirements.

Since the target assembly is available electronically, it is feasible to cognitively automate the process, hence achieving economic advantages. The main challenges involved are as follows:

(i) Construction of a continuous flow of information from the CAD system to the manufacturing system.

(ii) Robust system components and joining processes.

(iii) Logistical concepts and components for efficient stock placement during the assembly process.

(iv) Automated verification of the final assembly and man-machine interaction in case of errors.

Figure 21: Mapped depth and color data for the switch-cabinet scenario (one viewpoint).

The switch-cabinet scenario bears strong similarities to the demonstrator-cell scenario, that is, in regard to the customer-defined CAD models and requirements regarding an automated verification of the final assembly.

Hence, in order to verify the capabilities of the Assembly Group Analysis, the perception and verification process using the Kinect was simulated for a given scenario. Mapped depth and color data for the final assembly indicating the different parts that it consists of is given in Figure 21.

For the verification process, three different cases are taken into consideration. In the first case, the assembly was built correctly. In the second case, one of the larger feed-through modular terminal blocks UKH 95 has not been placed onto the mounting rail. In the third case only one of the smaller pick-off terminal blocks AGK 10/UKH has not been connected to the UKH 150. Figure 22 presents the resulting Kinect data for the different cases. For the first erroneous case the missing UKH 95 results in a smaller region of the matching depth data. In comparison with the Hubelino scenario, this might be the case, if two adjacent bricks that have the same color are present in the desired CAD model, but only one of them has been placed in the real scenario. For the second error case the AGK 10/UKH defines its own region, since it is not directly connected to any other component. A comparable Hubelino scenario would be that a single brick, that is, one that does not comprise a layer with any other brick, has not been placed into the assembly group.

Through transferring the described approaches in cognition as well as in 3D Assembly Group Analysis onto this scenario, an autonomous assembly process can be established. Particularly, within the focus of customer-defined switch-cabinet assemblies, a cognitive production cell is useful as it easily adapts the production process on changing assemblies. Additionally, an automated analysis process relieves the human worker and is less error-prone.

5. Conclusion

In order to resolve the problem that the commissioning and programming of complex robotized assembly takes considerable planning efforts which do not directly contribute to

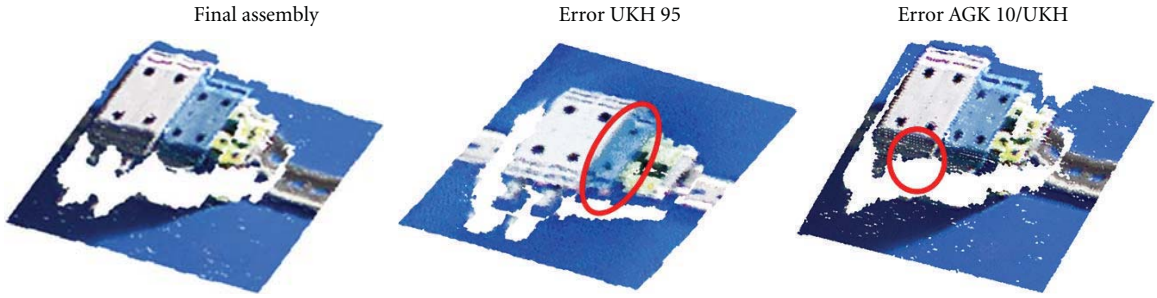


Figure 22: Resulting depth data for switch-cabinet scenario.

the added value, the overall approach was to shift planning tasks to the level of execution. Hence, within the Cluster of Excellence "Integrative Production Technology for High-Wage Countries" at RWTH Aachen University, a Cognitive Control Unit had been created, which is able to plan and execute action flows autonomously for a given goal state. The combination with an ergonomic user-centered human machine interface allows for an interaction with a human expert in order to partially transfer planning and implementation tasks to the technical cognitive system as well as to present him/her the opportunity to intervene in case of assembly errors. To test and further enhance the CCU in a near-reality production environment in a variety of different assembly operations, a robotic assembly cell had been set up. The scenario comprises major aspects of an industrial application, while at the same time easily illustrating the potential of a cognitive control system. As a demonstration scenario, the building of an assembly group comprised of Hubelino bricks, which is given to the CCU in the form of a CAD file, was chosen.

In order to verify an assembled group, the CCU requires a recognition process. Additionally, the human operator must be supported in case of assembly errors. This verification process including the feedback to a human expert was realized using the Microsoft Kinect as a perception device. The Kinect was chosen since it merges the capability of a depth sensor and a color camera in a single device, hence providing an innovative approach for research and development for the field of environment recognition.

In the scenario, the Kinect was attached to the flange of an industrial robot, allowing for a positioning at multiple viewpoints in order to create a virtual image of the current assembly group. For each viewpoint the depth data as well as color information is recorded and analyzed to create a discrete model for each side. Afterwards, the discrete models are combined into a complete model of the assembly group. Finally, the superior goal of verifying a real assembly group of Hubelino bricks was achieved by comparing the resulting perceived model with the virtual model from the customer-defined CAD file.

In order to demonstrate the implications in an industrial application, the concepts that were tested with the demonstrator cell were transferred to a scenario for the assembly of switch-cabinets. This scenario bears strong similarities to the demonstrator cell, that is, in regard to the customer-defined CAD models and requirements regarding an automated verification of the final assembly.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for its kind support of the research on cognitive automation within the Cluster of Excellence Integrative Production Technology for High-Wage-Countries. No other sponsorship or economic interests were involved which ensured free evaluation of the experiments.

References

[1] E. von Weizsäcker, *Globalisierung der Weltwirtschaft - Herausforderungen und Antworten*, vol. 5 of *Politik und Zeitgeschichte*, 2003, http://www.bpb.de/publikationen/U27MV5,0,Globalisierung_der_Weltwirtschaft_Herausforderungen_und_Antworten.html.

[2] DIHK, "Produktionsverlagerung als Element der Globalisierungsstrategie von Unternehmen: Ergebnisse einer Unternehmensbefragung," 2011, http://www.dihk.de/ressourcen/downloads/produktionsverlagerung.pdf.

[3] DIHK, "Auslandsinvestitionen in der Industrie," 2011, http://www.muenchen.ihk.de/mike/ihk_geschaeftsfelder/starthilfe/Anhaenge/DIHK-Auslandsinvestitionen-2010.pdf.

[4] F. Klocke, "Production technology in high-wage countries: from ideas of today to products of tomorrow," in *Industrial Engineering and Ergonomics: Visions, Concepts, Methods and Tools*, C. Schlick, Ed., Festschrift in Honor of Professor Holger Luczak, Springer, New York, NY, USA, 2009.

[5] S. Kinkel, *Arbeiten in der Zukunft: Strukturen und Trends der Industriearbeit*, Ed. Sigma, Berlin, Germany, 2008.

[6] L. Bainbridge, "Ironies of automation," in *New Technology and Human Error*, J. Rasmussen, K. Duncan, and J. Leplat, Eds., pp. 775–779, John Wiley & Sons, Chichester, UK, 1987.

[7] R. Onken and A. Schulte, *System-Ergonomic Design of Cognitive Automation: Dual-Mode Cognitive Design of Vehicle Guidance and Control Work Systems*, Springer, Berlin, Germany, 2010.

[8] C. Brecher, Ed., *Integrative Produktionstechnik für Hochlohnländer*, Springer, Berlin, Germany, 2011.

[9] M. F. Zäh, M. Beetz, K. Shea et al., "The cognitive factory," in *Changeable and Reconfigurable Manufacturing Systems*, H. A. El-Maraghy, Ed., Springer, Berlin, Germany, 2009.

[10] S. Karim, L. Sonenberg, and A. H. Tan, "A hybrid architecture combining reactive plan execution and reactive learning," in

Proceedings of the 9th Biennial Pacific Rim International Conference on Artificial Intelligence (PRICAI '06), China, 2006.

[11] E. Gat, "On three-layer architectures," in *Artificial Intelligence and Mobile Robots*, D. Kortenkamp, R. Bonnasso, and R. Murphy, Eds., pp. 195–211, AAAI Press, Menlo Park, Calif, USA, 1998.

[12] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson Education, Upper Saddle River, NJ, USA, 2003.

[13] K. Paetzold, "On the importance of a functional description for the development of cognitive technical systems," in *International Design Conference*, Dubrovnik, Croatia, 2006.

[14] P. Adelt, J. Donath, J. Gausemeier et al., "Selbstoptimierende Systeme des Maschinenbaus," in *HNI-Verlagsschriftenreihe*, J. Gausemeier, F. Rammig, and W. Schäfer, Eds., Westfalia Druck GmbH, Paderborn, Germany, 2009.

[15] M. Zäh and M. Wiesbeck, "A model for adaptively generating assembly instructions using state-based graphs," in *Manufacturing Systems and Technologies for the New Frontier*, The 41st CIRP Conference on Manufacturing Systems, Tokyo, Japan, 2008.

[16] M. Zäh, M. Wiesbeck, F. Engstler et al., "Kognitive Assistenzsysteme in der manuellen Montage," *in wt Werkstatttechnik*, vol. 97, no. 9, pp. 644–650, 2007.

[17] H. Ding, S. Kain, F. Schiller, and O. A. Stursberg, "Control architecture for safe cognitive systems," in *10. Fachtagung Entwurf komplexer Automatisierungssysteme*, Magdeburg, Germany, 2008.

[18] S. Kammel, J. Ziegler, B. Pitzer et al., "Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.

[19] H. J. Putzer, *Ein uniformer Architekturansatz für Kognitive Systeme und seine Umsetzung in ein operatives Framework*, Dr. Köster, Berlin, Germany, 2004.

[20] D. Ewert, E. Hauck, A. Gramatke, and S. Jeschke, "Cognitive assembly planning using state graphs," in *Proceedings of the 3rd International Conference on Applied Human Factors and Ergonomics*, Miami, Fla, USA, 2010.

[21] T. Kempf, W. Herfs, and C. Brecher, "Cognitive Control Technology for a Self-optimizing Robot Based Assembly Cell," in *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conferences (DETC '08)*, New York, NY, USA, August 2008.

[22] M. Mayer, B. Odenthal, M. Faber et al., "Cognitive engineering for direct human-robot cooperation in self-optimizing assembly cells," in *First International Conference on Human Centered Design (HCD '09)*, M. Kurosu, Ed., San Diego, Calif, USA, July 2009.

[23] C. Schlick, B. Odenthal, M. Mayer et al., "Design and evaluation of an augmented vision system for self-optimizing assembly cells," in *Industrial Engineering and Ergonomics: Visions, Concepts, Methods and Tools*, C. Schlick, Ed., Festschrift in Honor of Professor Holger Luczak, Springer, New York, NY, USA, 2009.

[24] T. B. Sheridan, *Humans and Automation: System Design and Research Issues*, Human Factors and Ergonomics Soc, Santa Monica, Calif, USA, 2001.

[25] M. Mayer, B. Odenthal, and M. Grandt, "Task-oriented process planning for cognitive production systems using MTM," in *Proceedings of the 2nd International Conference on Applied Human Factors and Ergonomic*, Louisville, Ky, USA, 2008.

[26] V. Gazzola, G. Rizzolatti, B. Wicker, and C. Keysers, "The anthropomorphic brain: the mirror neuron system responds to human and robotic actions," *NeuroImage*, vol. 35, no. 4, pp. 1674–1684, 2007.

[27] E. Drumwright, "Toward a vocabulary of primitive task programs for humanoid robots," in *Proceedings of the International Conference on Development and Learning (ICDL '06)*, Bloomington, Ind, USA, 2006.

[28] B. Odenthal, M. Mayer, N. Jochems, and C. Schlick, "Cognitive engineering for human-robot interaction—the effect of subassemblies on assembly strategies," in *Advances in Human Factors, Ergonomics, and Safety in Manufacturing and Service Industries*, pp. 1–10, CRC Press, Boca Raton, Fla, USA, 2011.

[29] M. Mayer, C. Schlick, D. Ewert et al., "Automation of robotic assembly processes on the basis of an architecture of human cognition," *Production Engineering Research and Development*, vol. 5, no. 4, pp. 423–431, 2011.

[30] T. Kempf, *Ein kognitives Steuerungsframework für robotergestützte Handhabungsaufgaben*, Apprimus, Aachen, Germany, 1st edition, 2010.

[31] B. Odenthal, M. Mayer, W. Kabuß, B. Kausch, and C. Schlick, "Investigation of error detection in assembled workpieces using an augmented vision system," in *Proceedings of the 17th World Congress on Ergonomics (IEA '09)*, Beijing, China, August 2009.

[32] B. Odenthal, M. Mayer, W. Kabuß, and C. Schlick, "Einfluss der Bildschirmposition auf die Fehlererkennung in einem Montagebautei," in *Neue Arbeits- und Lebenswelten gestalten: Vom 24. - 26. März*, M. Schütte, Ed., pp. 203–206, GfA-Press, Dortmund, Germany, 2010.

[33] B. Odenthal, M. Mayer, W. Kabuß, B. Kausch, and C. Schlick, "An empirical study of disassembling using an augmented vision system," in *3rd International Conference on Digital Human Modeling (ICDHM '11)*, Orlando, Fla, USA, 2011.

[34] C. Beder, B. Bartczak, and R. Koch, "A comparison of PMD-cameras and stereo-vision for the task of surface reconstruction using patchlets," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '07)*, June 2007.

[35] C. Sa, "Time of Flight Camera Technology," *Technology*, 2009.

[36] D. Schauer, *Integration of a 3D time of flight camera system into a robot system: integration, validation and comparison*, VDM Verlag Dr. Müller, Saarbrücken, Germany, 2011.

[37] Microsoft, Ed, "Programming Guide: Getting Started with the Kinect for Windows SDK Beta from Microsoft Research," 2011.

[38] D. Laing, "Kinect - Could this technology have a future in industrial automation?" http://blog.vdcresearch.com/industrial_automation/2010/11/kinect-could-this-technology-have-a-future-in-industrial-automation.html.

[39] D. L. Andrews, *Structured light and its applications: an introduction to phase-structured beams and nanoscale optical forces*, Academic Press, Amsterdam, The Netherlands, 2008.

[40] Z. Song, *Use of structured light for 3D reconstruction*, Hong Kong, 2008.

[41] K. Clague and M. Agullo, *LEGO software power tools*, Syngress, Rockland, Mass, USA, 2002.

[42] Z. Yaniv, "Random Sample Consensus (RANSAC) Algorithm, A Generic Implementation," Information Systems Journal, 2010, http://isiswiki.georgetown.edu/zivy/writtenMaterial/RANSAC.pdf.

[43] R. Szeliski, "Computer vision: Algorithms and applications," Computer Vision, 2011, http://www.worldcat.org/oclc/700473658.

[44] C. Stephens, "Lego Color List - official names, numbers, CYMK, RGB, Pantone values," 2011, http://isodomos.com/Color-Tree/Lego-List.html.

[45] C. Zhang, N. Xi, and Q. Shi, "Object-orientated registration method for surface inspection of automotive windshields," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*, pp. 3553–3558, September 2008.

[46] X. Liang, J. Liang, J. Liu, and C. Guo, "A rapid inspection method for large water turbine blade," in *IEEE International Conference on Mechatronics and Automation (ICMA '10)*, pp. 712–716, August 2010.

[47] F. Bosché, "Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction," *Advanced Engineering Informatics*, vol. 24, no. 1, pp. 107–118, 2010.

[48] L. Yue and X. Liu, "Application of 3D optical measurement system on quality inspection of turbine blade," in *16th IEEE International Conference on Industrial Engineering and Engineering Management (IE and EM '09)*, pp. 1089–1092, October 2009.

[49] A. Tellaeche, R. Arana, A. Ibarguren, and J. Martínez-Otzeta, "Automatic quality inspection of percussion cap mass production by means of 3D machine vision and machine learning techniques," in *Lecture Notes in Computer Science, Hybrid Artificial Intelligence Systems*, M. Graña Romay, E. Corchado, and M. Garcia Sebastian, Eds., pp. 270–277, Springer, Berlin, Germany, 2010.

[50] B. Muralikrishnan and J. Raja, *Computational Surface and Roundness Metrology*, Springer, 2009.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 726807, 9 pages
doi:10.1155/2011/726807

Research Article

Design of an Error-Based Adaptive Controller for a Flexible Robot Arm Using Dynamic Pole Motion Approach

Ki-Young Song,[1] Madan M. Gupta,[1] and Noriyasu Homma[2]

[1] *Intelligent Systems Research Laboratory, College of Engineering, University of Saskatchewan, Saskatoon, Sk, Canada S7N5A9*
[2] *Research Division on Advanced Information Technology, Cyberscience Center, Tohoku University, Sendai 980-8579, Japan*

Correspondence should be addressed to Madan M. Gupta, madan.gupta@usask.ca

Received 16 July 2011; Accepted 12 October 2011

Academic Editor: Ivo Bukovsky

Design of an adaptive controller for complex dynamic systems is a big challenge faced by the researchers. In this paper, we introduce a novel concept of *dynamic pole motion* (DPM) for the design of an error-based adaptive controller (E-BAC). The purpose of this novel design approach is to make the system response reasonably fast with no overshoot, where the system may be time varying and nonlinear with only partially known dynamics. The E-BAC is implanted in a system as a nonlinear controller with two dominant dynamic parameters: the dynamic position feedback and the dynamic velocity feedback. For illustrating the strength of this new approach, in this paper we give an example of a flexible robot with nonlinear dynamics. In the design of this feedback adaptive controller, parameters of the controller are designed as a function of the system error. The position feedback $K_p(e,t)$ and the velocity feedback $K_v(e,t)$ are continuously varying and formulated as a function of the system error $e(t)$. This approach for formulating the adaptive controller yields a very fast response with no overshoot.

1. Introduction

Recently, there has been an increasing interest in the design of feedback controllers: from the design of conventional approaches to the design of intelligent-based approaches. One such approach is on the design of adaptive controller for controlling a complex dynamic system containing nonlinearity like flexible joints. During the past, there has been a common practice to approximate a nonlinear system by a linear system in limited operating ranges and then make use of the conventional controller design approaches. However, the nonlinearity of a system is inevitable since many systems in practice involve nonlinear relationships among the variables such as electromechanical systems, hydraulic systems, and pneumatic systems [1]. For decades, various schemes of adaptive control have been proposed, and adaptive control for nonlinear systems with complex dynamics has received great attention. However, not many of these approaches are suitable for complex nonlinear systems [2–4]. Up to the present, inverse optimal controller [5–7] using the Lyapunov function has been considered as one of the most effective way for designing controllers for nonlinear systems.

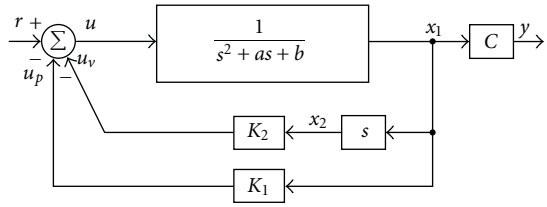
In this paper, we introduce a new notion of controller called error-based adaptive controller (E-BAC) with a novel conception based upon dynamic pole motion (DPM) approach. In general, for the design of E-BAC, we consider two dominant parameters, the position feedback $K_p(e,t)$ and velocity feedback $K_v(e,t)$, and a proper design of these two feedback parameters will yield a faster and stable response of the system with no overshoot. The feedback parameters are adapted by the system error $e(t)$ and its states $\mathbf{x}(t)$.

The rest of the paper is organized as follows. In Section 2, we introduce some important observations of a step response for a typical linear second-order system. In Section 3, we describe the notion of dynamic pole motion (DPM) and the design of error-based adaptive controller (E-BAC) in detail. A flexible robotic joint control is presented in Section 4 with E-BAC and DPM as a case study. Section 5 concludes this paper with a discussion and future works.

2. Some Important Observations in the Step Response for a Second-Order Linear System

In our study, we consider a typical open-loop second-order plant $G_p(s)$ defined as

$$G_p(s) = \frac{1}{s^2 + as + b}, \tag{1a}$$



(1b)

$$\text{state vector: } \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{C} y \\ \dot{x}_1 \end{bmatrix}, \tag{1c}$$

$$\text{output: } y = Cx_1 = (b + K_1)x_1.$$

Equations (1a), (1b), and (1c) represent a typical second-order system with position (K_1) and velocity (K_2) feedbacks.

As shown in (1b), with position and velocity feedback controller, the transfer function of the closed-loop system is given by

$$\frac{Y(s)}{R(s)} = \frac{C}{s^2 + (a + K_2)s + (b + K_1)}, \tag{2}$$

where $C = b + K_1$. This transfer function can be compared with a general linear second-order system model as

$$\frac{(b + K_1)}{s^2 + (a + K_2)s + (b + K_1)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}. \tag{3}$$

Thus, we see that

$$\omega_n^2 = (b + K_1) \triangleq K_p, \quad \omega_n\text{: system natural frequency,}$$
$$2\zeta\omega_n = (a + K_2) \triangleq K_v, \quad \zeta\text{: system damping ratio,} \tag{4}$$

where the parameters K_p and K_v are defined as position feedback and velocity feedback, respectively.

Generally the dynamic behavior of a second-order system can be described in terms of two dominant parameters, the natural frequency (ω_n) and the damping ratio (ζ). The transient response of a typical control system often exhibits damped oscillations before reaching the steady state. In specifying the transient response characteristics of a second-order control system to a unit-step input, the following

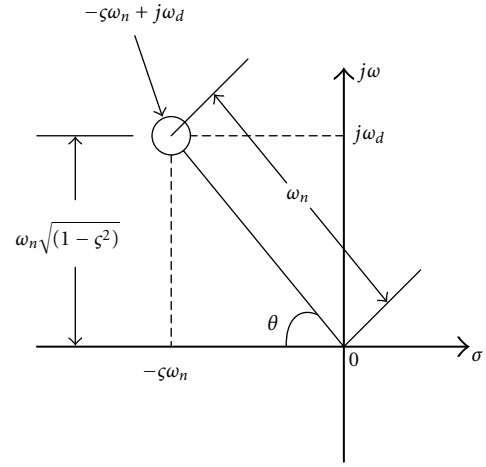


Figure 1: Definition of the various parameters in the complex σ-$j\omega$ plane: natural frequency: ω_n, damping ratio: ζ, damped natural frequency: $\omega_d = \omega_n\sqrt{(1-\zeta^2)}$, and $\theta = \cos^{-1}\zeta$.

transient parameters in the design of a controller are usually considered [1, 8, 9]:

$$\text{rise time: } T_r = \frac{\pi - \theta}{\omega_d} = \frac{\pi - \theta}{\omega_n\sqrt{1 - \zeta^2}}, \quad \theta = \cos^{-1}(\zeta),$$

$$\text{settling time: } T_s = \frac{4}{\zeta\omega_n} \quad (2\%\text{ criterion}),$$

$$\text{maximum overshoot: } M_p = e^{-\zeta\pi/\sqrt{1-\zeta^2}} \times 100 \quad (\%),$$

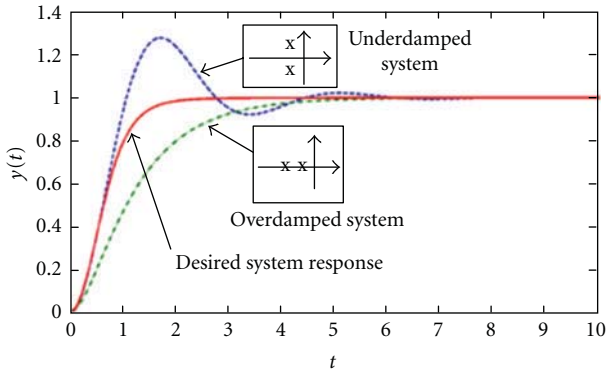
$$\text{bandwidth: } \omega_{\text{BW}} = \omega_n\sqrt{(1 - 2\zeta^2) + \sqrt{4\zeta^4 - 4\zeta^2 + 2}}. \tag{5}$$

It is important to note that in the step response of the second-order system, the dominant transient parameters T_r, T_s, and M_p are dependent upon the natural frequency (ω_n) and the damping ratio (ζ) of the system. Thus, the positions of the poles of the system are determined by the values of ω_n and ζ as shown in Figure 1.

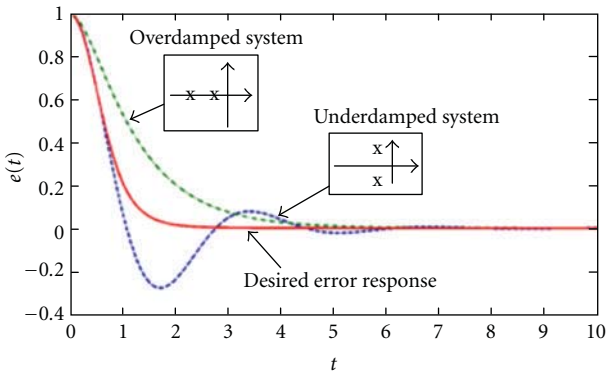
As shown in Figure 2, it is also to be noted that, in typical transient responses an underdamped system ($\zeta < 1$) yields a faster rise time (T_r) at the expense of a large overshoot (M_p) and a large settling time (T_s), whereas an overdamped system ($\zeta > 1$) yields no overshoot, that is, $M_p = 0$, but it yields large T_r and T_s.

3. Development of an Error-Based Adaptive Controller (E-BAC): Some Design Criteria

For the design of an appropriate feedback controller, let us consider the system error $e(t)$ as an important signal in our feedback design. In our design methodology developed in this paper, we will make the parameters of the feedback controller as functions of the error. From the transient responses shown in Figure 2(b), we can emphasize that for large errors a small ζ and a large ω_n, (i.e., an underdamped dynamics with large bandwidth) will yield a very fast



(a) The system response curves



(b) The error response curves of the systems

FIGURE 2: System responses to a unit-step input with two different locations of poles (i) underdamped case ($\zeta < 1$) and (ii) overdamped case ($\zeta > 1$). The desired system response curve initially follows the underdamped curve for large errors and then settles down to a steady-state value (following the overdamped curve) for decreasing errors.

response with a very small rise time T_r. On the other hand, for small errors a large ζ and a small ω_n (i.e., an overdamped system with a small bandwidth) will inhibit any overshoot. Since ζ and ω_n are dependent upon the parameters of position feedback (K_p) and velocity feedback (K_v), if we define $K_p(e,t)$ and $K_v(e,t)$ as functions of the system error, $e(t) = r(t) - y(t)$, then we can achieve a very fast dynamic response with no overshoot.

From these qualitative observations on the transient response of the step response, we derive the following design criteria for the E-BAC [10].

Design Criteria for the Error-Based Adaptive Controller (E-BAC)

(i) If the system *error is large,* then keep the *damping ratio* ζ *very small* and *natural frequency* ω_n *very large.* A *large* ω_n *and small* ζ will result into a *large bandwidth* of the system, thereby a shorter rise time and fast response.

(ii) If the system *error is small,* then keep the *damping ratio* ζ *large* and *natural frequency* ω_n *small.* This will result into a *small bandwidth* of the system. For small errors, a large damping ratio in the system will avoid any overshoot in the system response.

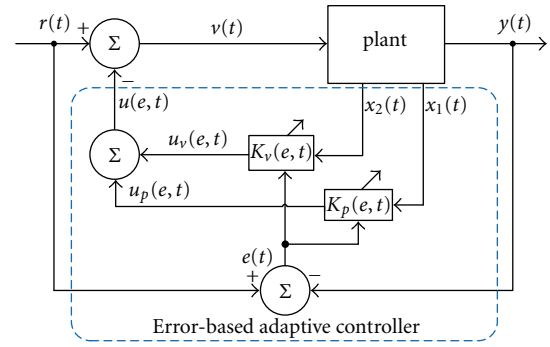


FIGURE 3: The proposed error-based adaptive controller (E-BAC): $x_2(t) = \dot{x}_1(t)$. $K_p(e,t)$ and $K_v(e,t)$ are defined in (8) and (12).

Design of Parameters for the E-BAC

(i) Position feedback K_p controls the natural frequency ω_n, ($K_p = \omega_n^2$), and, therefore, the bandwidth of the system.

(ii) Velocity feedback K_v controls the damping ratio ζ, ($K_v = 2\zeta\omega_n$).

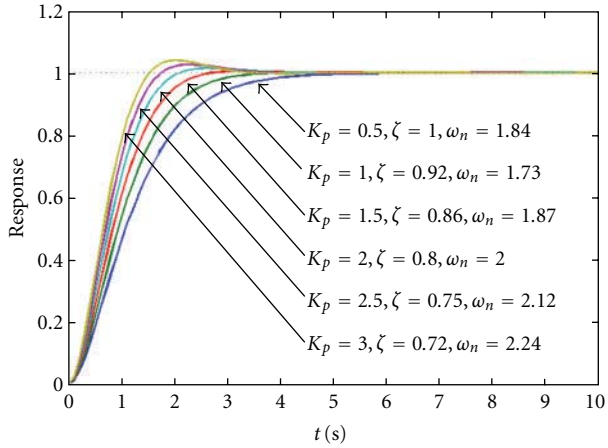
Thus, we design the adaptive controller parameters which, in this case, are the position feedback $K_p(e,t)$ and velocity feedback $K_v(e,t)$ as functions of the error, $e(t)$. This procedure for designing the adaptive controller will introduce a dynamic motion in the poles of the system keeping the system response at an acceptable level. Here thus, we introduce a new notion of the movable poles and give it the name *Dynamic Pole Motion* (DPM). The proposed novel E-BAC is illustrated in Figure 3.

This novel design philosophy for adaptive controller is translated into the following linguistic algorithm:

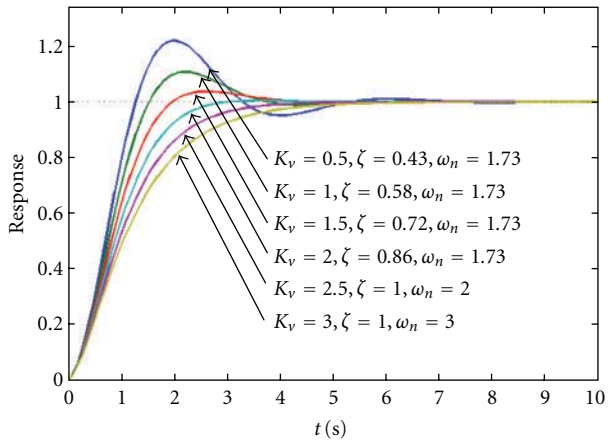
> *As error decreases from a large value to a small value,* $K_p(e,t)$ $(=\omega_n^2(t))$ *is continuously decreased from a very large value to a small value, and simultaneously,* $K_v(e,t)$ $(=2\zeta(t)\omega_n(t))$ *is increased from a small value to a large value.*

This linguistic control algorithm causes a larger bandwidth with a smaller damping ratio for large errors and smaller bandwidth with larger damping ratio for small errors. Hence, as discussed above and shown in Figures 2 and 3, during the operation of the system a desired transient response from the systems can be achieved by varying ω_n and ζ as functions of error. As given in (4), ω_n and ζ are dependent upon the position feedback K_p and velocity feedback K_v, respectively. Some typical response curves for a second-order closed-loop system with varying K_p and K_v are shown in Figure 4.

3.1. Design of E-BAC Parameters $K_p(e,t)$ *and* $K_v(e,t)$. Using the design criteria for the adaptive controller stated above,



(a) A family of system response curves with various values of K_p and a constant $K_v = 3$



(b) A family of system response curves with various values of K_v and a constant $K_p = 1$

FIGURE 4: System response curves of a second-order system varying K_p (position feedback) and K_v (velocity feedback).

one can develop many types of functions for $K_p(e,t)$ and $K_v(e,t)$, which satisfy the design criteria with respect to the system error and time. Here, we give one such function for $K_p(e,t)$ and $K_v(e,t)$ by defining the system error as

$$e(t) = r(t) - y(t), \tag{6}$$

where the system output $y(t)$ is given by

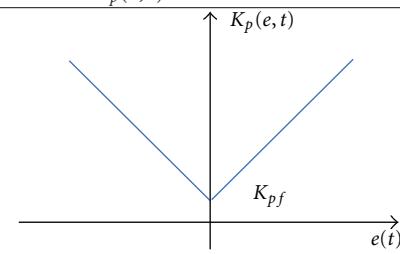
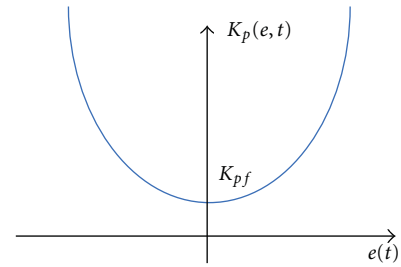
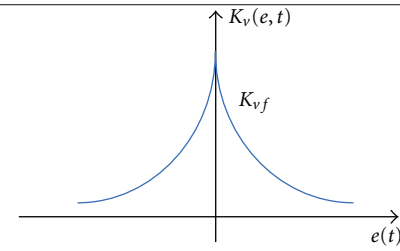
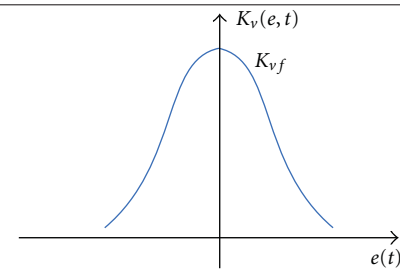
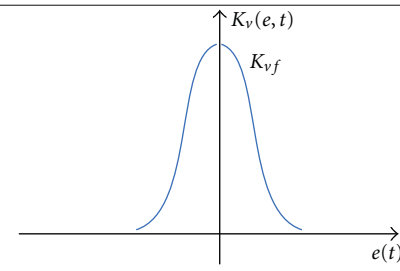
$$y(t) = K_p(e,t)x_1(t). \tag{7}$$

Thus, we define the position feedback $K_p(e,t)$ and the velocity feedback $K_v(e,t)$ gains as functions of $e(t)$ as

$$\begin{aligned} K_p(e,t) &= K_{pf}\left(1 + \alpha e^2(t)\right), \\ K_v(e,t) &= K_{vf}\exp\left[-\beta e^2(t)\right], \end{aligned} \tag{8}$$

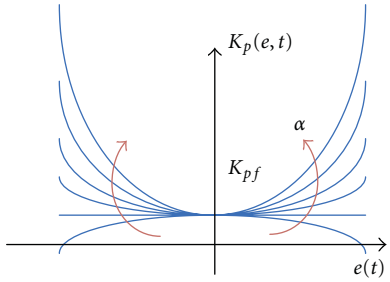
where α and β are some gain constants which decide the slope of the functions and affect the system response (see Figure 5), K_{pf} and K_{vf} are the final steady-state values of $K_p(e,t)$ and $K_v(e,t)$, and $\exp(\cdot)$ is the exponential function. The other possible functions for $K_p(e,t)$ and $K_v(e,t)$ are given in Table 1.

TABLE 1: Various possible functions and their graphic experessions for feedback gains $K_p(e,t)$ and $K_v(e,t)$.

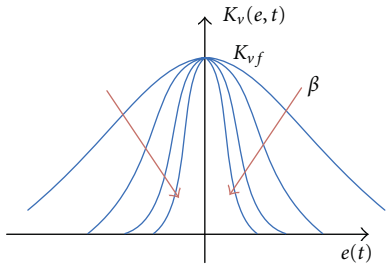
$K_p(e,t)$	
$K_{pf}(1 + \alpha\lvert e\rvert)$	
$K_{pf}(1 + \alpha e^2)$	
$K_v(e,t)$	
$K_{vf}\dfrac{1}{1+\beta\lvert e\rvert}$	
$K_{vf}\dfrac{1}{1+\beta e^2}$	
$K_{vf}\exp(-\beta e^2)$	

Many other functions can be derived for $K_p(e,t)$ and $K_v(e,t)$, for example, using the hyperbolic tangent and cosine functions.

3.2. Design of the Error-Based Adaptive Controller (E-BAC). The error-based adaptive control signal $u(e,t)$ is derived as a



(a) Changes in slope for $K_p(e,t) = K_{pf}(1 + \alpha e^2)$ for various values of α: the direction of arrows indicates the increasing value of α from negative to positive values



(b) Changes in slope for $K_v(e,t) = K_{vf}\exp[-\beta e^2(t)]$ for various values of β: the direction of arrows indicates the increasing value of β

FIGURE 5: The change of the slopes of $K_p(e,t)$ and $K_v(e,t)$ curves for various values of α and β.

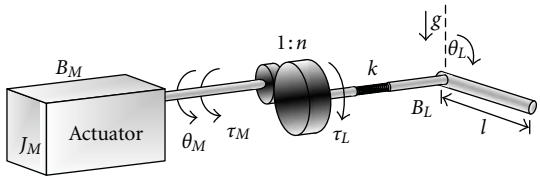


FIGURE 6: A schematic diagram of a single link manipulator with a flexible joint.

function of the error $e(t)$ and time t using the following two steps:

$$\begin{aligned} &\text{position feedback control: } u_p(e,t) = K_p(e,t)x_1(t),\\ &\text{velocity feedback control: } u_v(e,t) = K_v(e,t)x_2(t), \end{aligned} \tag{9}$$

where $K_p(e,t)$ and $K_v(e,t)$ are defined in (8) and (12), respectively. Thus, the total feedback signal $u(e,t)$ is given by

$$u(e,t) = u_p(e,t) + u_v(e,t), \tag{10}$$

and the control signal $v(t)$ (see Figure 3) is defined as

$$v(t) = r(t) - u(e,t). \tag{11}$$

4. A Case Study: Control of a Flexible Robot Arm Using E-BAC

In this section, we present the design and simulation studies of the proposed error-based adaptive controller (E-BAC) for a flexible joint of a robot arm.

4.1. Modeling of a Single Link Flexible Robot. As shown in Figure 6, a single link manipulator with flexible joint consists of an actuator connected through a gear train (harmonic drive) with the ratio n to a rigid link with length l, mass m, and moment of inertia $ml^2/3$.

Let us symbolize the rotor inertia of the actuator J_M, the viscous damping of the actuator B_M, the relative angular displacement of the joint actuator θ_M, a torque to the motor shaft τ_M, and the relative displacement of the end effector (load) θ_L. The joint flexibility is modeled by a linear torsional spring with stiffness k. The dynamics of the manipulator with a flexible joint can be represented by Euler-Lagrange equation defining $\tau_M = r$ as [11, 12]

$$\begin{aligned} \frac{ml^2}{3}\ddot{\theta}_L + B_L\dot{\theta}_L + \frac{mgl}{2}\sin\theta_L + k\left(\theta_L + \frac{\theta_M}{n}\right) = 0,\\ J_M\ddot{\theta}_M + B_M\dot{\theta}_M + \frac{k}{n}\left(\theta_L + \frac{\theta_M}{n}\right) = r. \end{aligned} \tag{12}$$

Equation (12) can be rewritten using the state variables x_i $(i = 1, 2, 3, 4)$ defining as

$$x_1(t) = \theta_M, \quad x_2(t) = \dot{\theta}_M, \quad x_3(t) = \theta_L, \quad x_4(t) = \dot{\theta}_L. \tag{13}$$

Thus, we have

$$\begin{aligned} \dot{x}_1(t) &= x_2(t),\\ \dot{x}_2(t) &= -a_1x_1(t) - a_2x_2(t) - a_3x_3(t) + br(t),\\ \dot{x}_3(t) &= x_4,\\ \dot{x}_4(t) &= -a_4x_1(t) - a_5x_3(t) - a_6\sin(x_3(t)) - a_7x_4(t), \end{aligned} \tag{14}$$

where

$$\begin{aligned} b = \frac{1}{J_M}, \quad a_1 = \frac{k}{J_Mn^2}, \quad a_2 = \frac{B_M}{J_M}, \quad a_3 = \frac{k}{J_Mn},\\ a_4 = \frac{3k}{mnl^2}, \quad a_5 = \frac{3k}{ml^2}, \quad a_6 = \frac{3g}{2l}, \quad a_7 = \frac{3B_L}{ml^2}. \end{aligned} \tag{15}$$

The block diagram of the system is shown in Figure 7. This system is a nonlinear and time varying system since the sine function in the feedback loop of the system causes nonlinearity in the system. The output of the system is dependent on the amplitude of the control signal, and if we use the conventional design tools, this nonlinearity causes some problems in designing an effective controller. In this paper, we present a novel approach to the design of a controller for this nonlinear timevarying system by using the error-based adaptive controller (E-BAC) and dynamic pole motion (DPM).

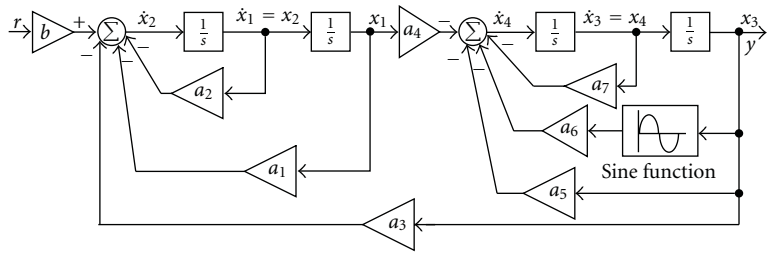


Figure 7: Block diagram of the single link manipulator with a flexible joint. The system has both linear and nonlinear feedbacks.

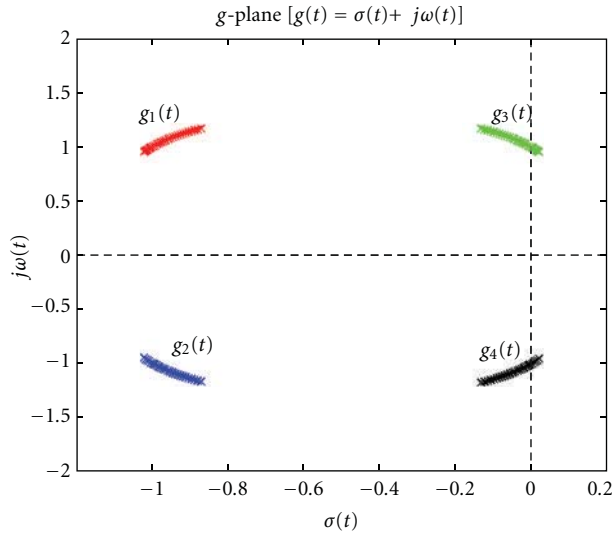


Figure 8: The sketch of dynamic pole motion (DPM) of the single link manipulator with a flexible joint system, (18), without a controller. Four poles are moving in the $g(t) = \sigma(t) + j\omega(t)$ plane with varying $x_3(t)$. Note that for certain values of $x_3(t)$ two dynamic poles move towards the right-hand side of the g-plane causing instability in the system.

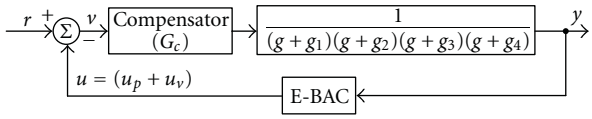


Figure 9: Schematic diagram of the flexible joint of a robot arm with a compensator (G_c) and E-BAC.

4.2. Design of Adaptive Controller for the Systems. In this case study, for simplicity we set the value of the parameters a_i ($i \in [1, 7]$) and b equal to 1. Thus, (14) can be rewritten as

$$\begin{aligned}
\dot{x}_1(t) &= x_2(t),\\
\dot{x}_2(t) &= -x_1(t) - x_2(t) - x_3(t) + r(t),\\
\dot{x}_3(t) &= x_4(t),\\
\dot{x}_4(t) &= -x_1(t) - \left\{1 - \frac{\sin(x_3(t))}{x_3(t)}\right\}x_3(t) - x_4(t).
\end{aligned} \tag{16}$$

Now we design an error-based adaptive controller for the single link robotic manipulator with a flexible joint. We

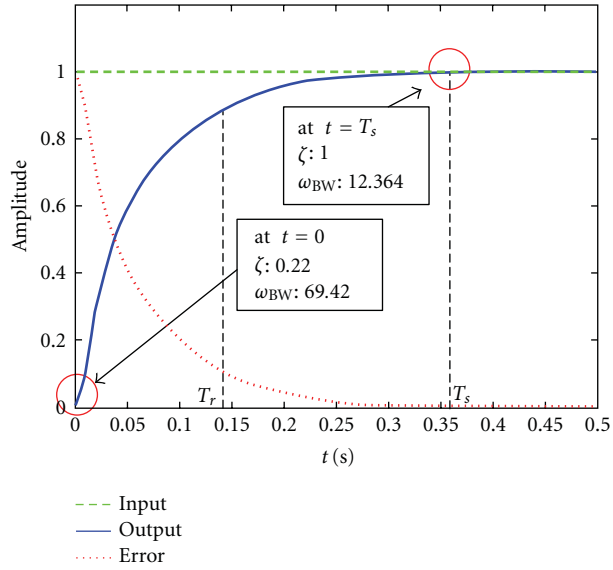


Figure 10: The system response with the E-BAC to a step reference input. Rise time $T_r = 0.14$ seconds; settling time $T_s = 0.36$ seconds.

first formulate the dynamic characteristic equation of the system. In this study for the nonlinear and timevarying robot arm, a new notion of timevarying complex variable $g(t) = \sigma(t) + j\omega(t)$ (g-plane) is applied instead of the time invariant complex variable $s = \sigma + j\omega$ (s-plane). The g-plane has the same properties of s-plane with an additional property of timevarying. The dynamic characteristic equation of the single-link manipulator with a flexible joint shown in Figure 7 and described in (16) is given by

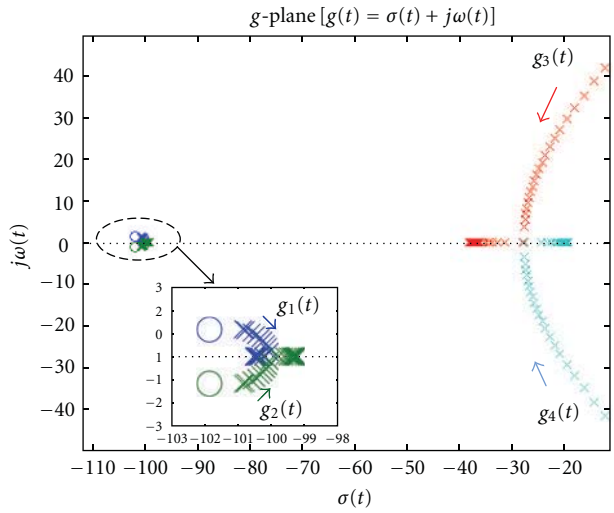
$$g^4(t) + 2g^3(t) + (3 + \psi(t))g^2(t) + (2 + \psi(t))g(t) + 2 + \psi(t) = 0, \tag{17}$$

where $\psi(t) = \sin(x_3(t))/x_3(t)$.

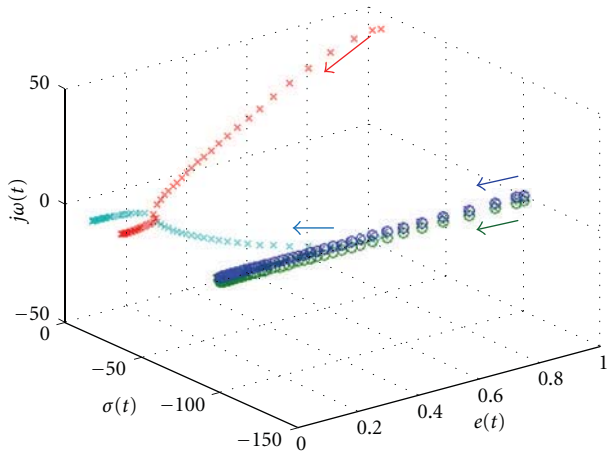
The dynamic roots of this characteristic equation of the transfer function can be calculated as

$$\begin{aligned}
g_{1,2}(t) &= \frac{1}{2}\left\{-1 - \sqrt{-2\psi(t) - 3 \pm 2\sqrt{\psi^2(t) - 4}}\right\},\\
g_{3,4}(t) &= \frac{1}{2}\left\{-1 + \sqrt{-2\psi(t) - 3 \pm 2\sqrt{\psi^2(t) - 4}}\right\}.
\end{aligned} \tag{18}$$

The nonlinear function, $\psi(t) = \sin(x_3(t))/x_3(t)$, covers the range $-0.22 < \psi(t) < 1$ for all values of $x_3(t)$ over $[-\infty, \infty]$.



(a) The dominant poles are moving from initial position $g_{3,4}(0) = -10.24 \pm j45.14$ to final position $g_3(0.5) = -37.9557$ and $g_4(0.5) = -19.2109$, respectively. This dynamic pole motion causes the system response to become from underdamped response to overdamped response



(b) 3D sketch of the dynamic poles and zeros motion of the controlled system. The location of the dynamic poles and zeros is a function of the system error

FIGURE 11: The sketch of motions of dynamic poles and zeros of the system with a compensator and an E-BAC (×: poles, ◯: zeros).

Thus, the roots of the dynamic characteristic equation are moving in the g-plane ($g(t) = \sigma(t) + j\omega(t)$).

The moving roots of the dynamic characteristic equation are named as *dynamic poles*. (Note that in linear time-invariant dynamic systems since the parameters of the system remain constant, the poles and zeros of the system are time invariant.) The plot of the four dynamic poles of this flexible robot arm without a controller is shown in Figure 8. From this figure, it is clear that for some values of x_3 two dynamic poles move towards the right-hand side (RHS) of the $g(t) = \sigma(t) + j\omega(t)$ plane causing instability in the system. The design criteria of our proposed error-based adaptive controller (E-BAC) for a system are as below.

Design Criteria of E-BAC for the Flexible Joint of a Robot Arm. For designing an E-BAC, we should consider the following important points.

(1) For introducing the stability in the robot arm system, we should move the dynamic poles on the left-hand side (LHS) on $g(t) = \sigma(t) + j\omega(t)$ plane for all values of $x_3(t)$.

(2) Realization of DPM using E-BAC.

   (a) For achieving the fast response time, the system must have a large bandwidth for large errors and small bandwidth for small errors. Thus, the position feedback K_p, the bandwidth parameter, must be a function of the system error $e(t)$.

   (b) For no overshoot in the system response, damping should be adjusted continuously as a function of the system error. The position feedback $K_p(e,t)$ and the velocity feedback $K_v(e,t)$ are designed such that they yield a small damping ratio with large bandwidth for large errors and a large damping ratio with small bandwidth for small errors.

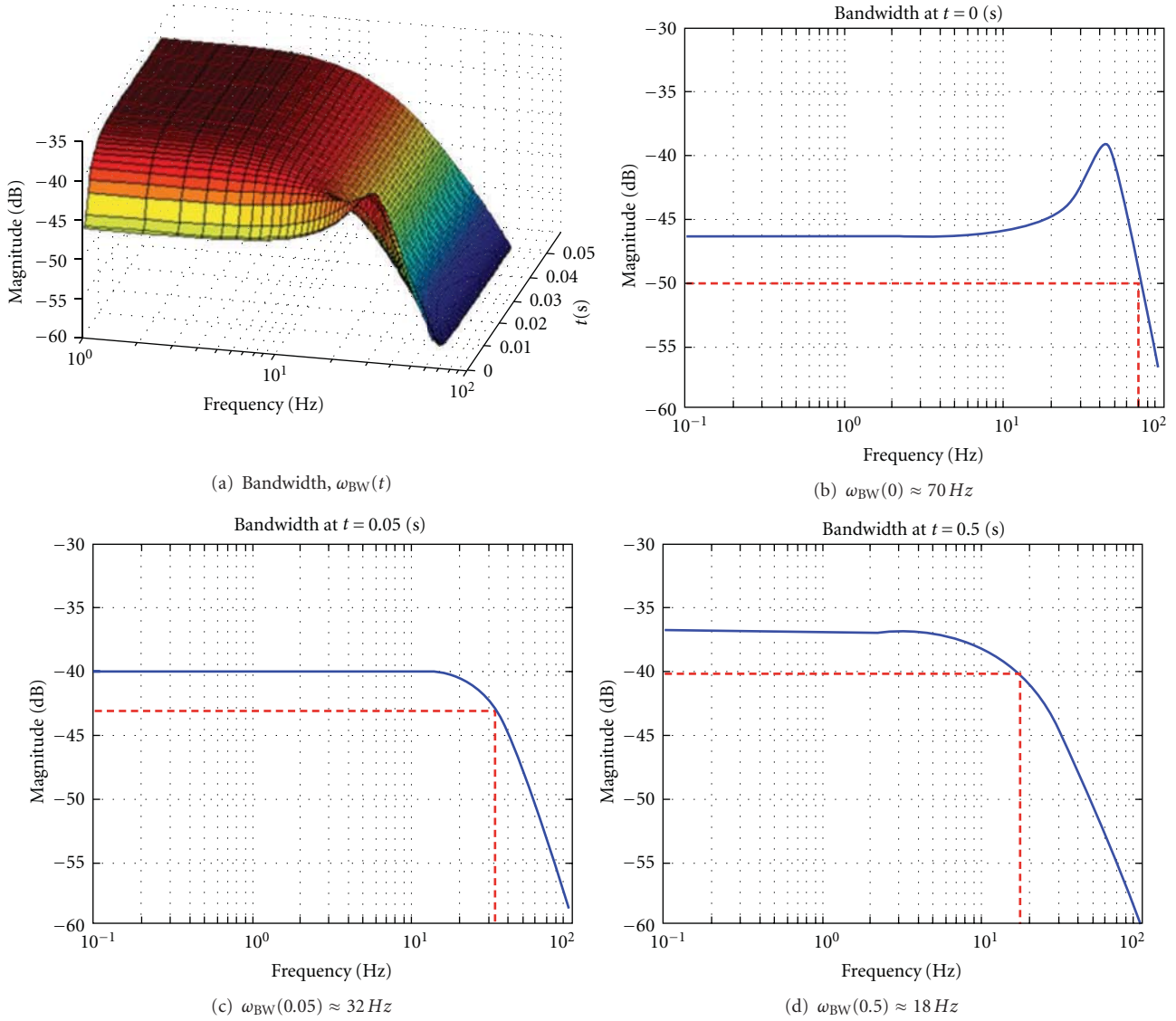
For achieving a good controller for this fourth-order flexible robot arm, we must first add a compensator to relocate two dynamic poles far away from the origin in the left-half of the g-plane. In this case study, we relocate $g_1(t)$ and $g_2(t)$ far away from the $j\omega(t)$-axis. These two relocated poles far away in the left-side of the g-plane will induce very small time constants, thereby, will have negligible effect in the system dynamic response. The other two poles that are closer to the imaginary axis are dominant poles and will cause an influence in the system dynamic response. Then, an E-BAC is added in the feedback loop with a position feedback $K_p(e,t)$ and a velocity feedback $K_v(e,t)$ defined in the previous sections. In this study, the compensator (G_c) used introduces two zeros in the forward loop with the position of zeros being $g(t) = -102 \pm j1.18$. This compensator provides a control over the plant poles [$g_1(t)$ and $g_2(t)$] keeping them far away from the $j\omega(t)$-axis. The feedback controller, E-BAC, provides a control over the two plant poles [$g_3(t)$ and $g_4(t)$]. The diagram of the system with a compensator and an E-BAC is illustrated in Figure 9.

The control input signal $v(t)$ is derived using (11) as

$$v(t) = r(t) - u(t), \tag{19}$$

where

$$\begin{aligned} u(t) &= \left\{u_p(e,t) + u_v(e,t)\right\}, \\ u_p(e,t) &= K_{pf}\left[1 + \alpha e^2(t)\right]x_3(t), \\ u_v(e,t) &= \exp\left[-\beta e^2(t)\right]x_4(t). \end{aligned} \tag{20}$$



(a) Bandwidth, $\omega_{BW}(t)$

(b) $\omega_{BW}(0) \approx 70\,Hz$

(c) $\omega_{BW}(0.05) \approx 32\,Hz$

(d) $\omega_{BW}(0.5) \approx 18\,Hz$

FIGURE 12: The variation of the bandwidth of the controlled system. (a) 3D sketch of the variation of the bandwidth with respect to the frequency at each time interval, (b) the bandwidth at $t = 0$ (sec), (c) the bandwidth at $t = 0.05$ (sec), and (d) the bandwidth at $t = 0.5$ (sec).

$x_3(t) = x$ and $x_4(t) = \dot{x}$ are the states of the system, K_{pf} and K_{vf} are the steady-state values of feedbacks $K_p(e,t)$ and $K_v(e,t)$, respectively, α and β are some gain constants for $K_p(e,t)$ and $K_v(e,t)$, respectively, $r(t)$ is the reference input of the system, and $e(t) = [y(t) - r(t)] = (r(t) - K_p(e,t)x_3(t))$ is the system error.

As described in the design criteria, the objective of the embedded E-BAC is to design the control $u(t)$ to make the system output $y(t)$ follow the reference input signal $r(t)$ as closely as possible with fast rise time$T_r$and small settling time$T_s$with no overshootM_p. Thus, we continuously change the dynamics of the close-loop system: initially for large errors, we make large bandwidth and very small damping ratio $\zeta(t)$, and as error decreases, the damping ratio $\zeta(t)$ is continuously increased and the system bandwidth is decreased.

In the design of the E-BAC, we have arbitrarily chosen the gains $\alpha = 2$, $\beta = 1$, $K_{pf} = 70$, and $K_{vf} = 5.5$. With these values, the controlled system responded as an underdamped system for large error at $t = 0$, which continuously moved towards an overdamped system with decreasing error.

4.3. Simulation Results. Using the gains $\alpha = 2$, $\beta = 1$, $K_{pf} = 70$, and $K_{vf} = 5.5$, the initial positions of the dynamic poles of the system are placed at $g_{1,2}(0) = -100.87 \pm j1.18$ (relocated to far from $j\omega(t)$-axis by the compensator), and $g_{3,4}(0) = -10.24 \pm j45.1$. During the operation of the system, as error is decreased to zero, the final positions of the dynamic poles are moved to -100.5639, -99.2695, -37.9557, and -19.2109 on the g-plane. The zeros are located at around -100 near the relocated poles by the compensator, and the zeros attract the relocated poles not

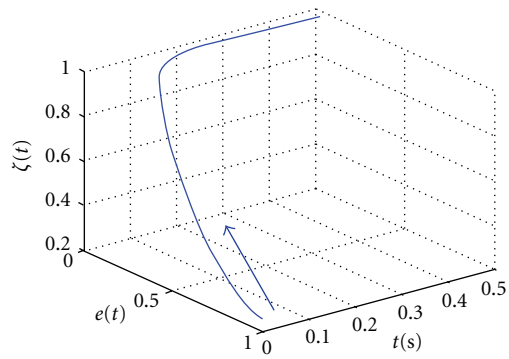


FIGURE 13: The variation of the damping ratio $\zeta(t)$ with respect to the system error as a function of time. The value of $\zeta(t)$ changes from a low value, $\zeta(t) = 0.221$, to a high value, $\zeta(t) = 1$.

to affect the dominant poles. The results of the simulation study of this case study are shown in Figure 10. Further, the maps of the dynamic poles motion (DPM) are illustrated in Figure 11. The output response initially follows the trajectory with a large bandwidth and a small damping ratio, which settles down with a large damping and a smaller bandwidth.

It is clear from the figure that the dynamic motion of poles of the system is decided by the value of the system error. The initial positions of the dominant dynamic poles are placed to generate a low damping ratio $\zeta(t)$ and large bandwidth of the system. Thus, initially the system is underdamped. Thereafter, the dynamic poles are optimized and shifted as the system error decreases reducing the bandwidth and increasing $\zeta(t)$. The final positions of the dominant dynamic poles make the system an overdamped system. Thus, the bandwidth becomes small, but $\zeta(t)$ becomes high. The variation of the bandwidth at each time interval is shown in Figure 12, and the variation of the damping ratio $\zeta(t)$ is shown with respect to the system error at each time interval in Figure 13.

5. Discussion and Conclusions

In this paper, we have proposed the design of an error-based adaptive controller (E-BAC) for controlling the dynamic response of a nonlinear system. The proposed E-BAC is the controller with continuously changing feedback parameters as functions of the system error: initially for large errors an underdamped system with large bandwidth which is forced to approach to become an overdamped system with small bandwidth for small errors. In the beginning, for large errors the system is underdamped, thus, it makes the system faster with a wider bandwidth. As the error decreases, the value of the feedback gains K_p decreases and K_v increases. The design of this adaptive controller is conceptually error-based and can be used to handle the complexity of systems. In order to support the novel controller, we introduce the notion of dynamic pole motion (DPM).

As a case study, we present a flexible joint of robot arm, which is a nonlinear dynamic system, and this system is controlled by the proposed E-BAC. Without a proper controller, the system is unstable due to the nonlinearity in the feedback loop of the system. However, as shown in Figure 10, with E-BAC the trajectory response of the system is very fast, $T_r = 0.36$ seconds, without any overshoot ($M_p = 0\%$). Also, as shown in Figures 12 and 13, in this step response, the initial bandwidth of the system is very high ($\approx$70 Hz) which settles down to about 18 Hz in the steady-state situation. The bandwidth of the system changes from a large value to a small value. Similarly but contrarily, the damping ratio $\zeta(t)$ varies from 0.221 ($t = 0$) to 1 ($t = 0.5$). From the simulation studies, it is shown that the proposed E-BAC is able to control nonlinear time varying systems. Conventionally, a proper design of the controller guarantees that the changing pole position is always positioned in the left-hand side (LHS) on g-plane. In this novel design approach, the dynamic poles are always located in LHS on g-plane, thus the stability of the controlled system is assured. Further work is under way to extend this E-BAC design philosophy for higher-order partially known and unknown complex dynamic systems.

References

[1] K. Ogata, *Modern Control Engineering*, Prentice Hall, Upper Saddle River, NJ, USA, 4th edition, 2002.

[2] Y. C. Chang, "An adaptive H∞ tracking control for a class of nonlinear multiple-input-multiple-output (MIMO) systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp. 1432–1437, 2001.

[3] D. G. Taylor, P. V. Kokotovic, R. Marino, and I. Kanellakopoulos, "Adaptive regulation of nonlinear systems with unmodeled dynamics," *IEEE Transactions on Automatic Control*, vol. 34, no. 4, pp. 405–412, 1989.

[4] Y. Liu and X. Y. Li, "Robust adaptive control of nonlinear systems represented by input-output models," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1041–1045, 2003.

[5] M. Jankovic, R. Sepulchre, and P. V. Kokotovic, "Global adaptive stabilization of cascade nonlinear systems," *Automatica*, vol. 33, no. 2, pp. 263–268, 1997.

[6] R. Sepulchre, M. Jankovic, and P. V. Kokotovic, "Integrator forwarding: a new recursive nonlinear robust design," *Automatica*, vol. 33, no. 5, pp. 979–984, 1997.

[7] E. D. Sontag, "A 'universal' construction of Artstein's theorem on nonlinear stabilization," *Systems and Control Letters*, vol. 13, no. 2, pp. 117–123, 1989.

[8] N. S. Nise, *Control Systems Engineering*, John Wiley & Sons, New York, NY, USA, 3rd edition, 2000.

[9] W. J. Palm III, *System Dynamics*, McGraw-Hill, New York, NY, USA, 2005.

[10] K. Y. Song, M. M. Gupta, D. Jena, and B. Subudhi, "Design of a robust neuro-controller for complex dynamic systems," in *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '09)*, Cincinnati, Ohio, USA, June 2009.

[11] M. W. Spong, K. Khorasani, and P. V. Kokotovic, "An integral manifold approach to the feedback-control of flexible joint robots," *IEEE Journal of Robotics and Automation*, vol. 3, no. 4, pp. 291–300, 1987.

[12] L. Jin, M. M. Gupta, and P. N. Nikiforuk, "Dynamic recurrent neural networks for modeling flexible robot dynamics," in *Proceedings of the 10th IEEE International Symposium on Intelligent Control*, pp. 105–110, August 1995.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 193146, 16 pages
doi:10.1155/2011/193146

Research Article

Discovering and Characterizing Hidden Variables Using a Novel Neural Network Architecture: LO-Net

Soumi Ray and Tim Oates

Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA

Correspondence should be addressed to Soumi Ray, soumi.ray@gmail.com

Received 1 June 2011; Revised 27 September 2011; Accepted 28 September 2011

Academic Editor: Ivo Bukovsky

Theoretical entities are aspects of the world that cannot be sensed directly but that, nevertheless, are causally relevant. Scientific inquiry has uncovered many such entities, such as black holes and dark matter. We claim that theoretical entities, or hidden variables, are important for the development of concepts within the lifetime of an individual and present a novel neural network architecture that solves three problems related to theoretical entities: (1) discovering that they exist, (2) determining their number, and (3) computing their values. Experiments show the utility of the proposed approach using discrete time dynamical systems, in which some of the state variables are hidden, and sensor data obtained from the camera of a mobile robot, in which the sizes and locations of objects in the visual field are observed but their sizes and locations (distances) in the three-dimensional world are not. Two different regularization terms are explored that improve the network's ability to approximate the values of hidden variables, and the performance and capabilities of the network are compared to that of Hidden Markov Models.

1. Introduction

Humans, like robots, have limited sensory access to the physical world. Despite this fact, thousands of years of scientific inquiry have uncovered much hidden structure governing the behavior and appearance of the world, along the way proposing a vast array of entities that we cannot see, hear, taste, smell, or touch. Just like Gregor Mendel who discovered genes in the middle of the 19th century, you and I cannot see, hear, taste, smell, or touch genes, but we can observe their effects on the world in the color of our friends' eyes or the scent of a rose. That is, genes may be unobservable, but they have causal power that manifests itself in ways that can be sensed directly. For Mendel, one such manifestation was the color of the peas of the pea plants that he bred with one another. Whether a plant would have yellow or green peas could not be predicted accurately based solely on observable properties of the parent plants. The desire to explain this apparent nondeterminism led Mendel to posit the existence of *a causally efficacious entity of the world that could not be sensed directly*, that is, genes. Such entities are called *theoretical entities*.

No one has ever seen a black hole, yet most physicists believe that they exist because black holes accurately explain a wide array of observational data. We propose to develop algorithms that will allow robots to discover the existence of theoretical entities in much the same way and for many of the same reasons that human scientists do. First, theoretical entities will be posited to explain nondeterminism in action outcomes. For example, a mobile robot in a home environment might discover that sometimes when it is turned on the power required to move forward is 2.5 watts and at other times only 1.5 watts are required, but there is no way to anticipate this prior to actually moving. The robot will posit the existence of a theoretical entity—some aspect of the world that causes this difference but cannot be directly sensed. In this case, we know that the posited entity corresponds to whether the robot starts out on carpet or on a hardwood floor. Second, only those entities that account for multiple nondeterministic observations will be retained. Does knowledge of the fact that the power required to move is 2.5 watts (i.e., the robot is on carpet) make it possible to predict anything else more accurately, such as whether the cleaning function selected by the robot's owner

will be sweep or vacuum? If so, the robot can decide for itself which cleaning function is appropriate simply by moving and measuring power consumption. The goal for the robot, as for the human scientist, is to acquire knowledge that makes it possible to better understand, predict, and control the world around it.

Another example of a domain where finding hidden variables is of great importance is the control of aluminum smelting pots. The United States produces more than 22 billion pounds of aluminum each year. Optimal control of the smelting process is important as suboptimal control not only reduces the yield but also increases energy consumption. But optimal control is very difficult because the smelting process occurs at extremely high temperatures and involves such caustic chemicals that very few sensors are able to survive. This is a case where performance is affected by insufficient sensing. Discovering variables that affect performance, yet are unobservable, can enable better understanding, prediction, and control of the smelting process.

A domain that has motivated much of this work is the discovery of hidden variables where the environment is partially observable with respect to robots. A robot can build a model of the environment to predict future values of state variables based on current and past observations. If hidden variables are present in the environment, then this predictive model will be inaccurate. When the sensor data are insufficient to predict outcomes due to theoretical entities, hidden variables need to be introduced in the model. The main goal of this modeling process is to acquire knowledge that makes it possible to better understand and predict the environment. There are three important tasks to be addressed, each of which is central to this work. The first is to discover the existence of hidden variables; the second is to find the number of hidden variables; and the third is to determine their values.

Predictive models can be developed based on current and past observations in the presence of hidden variables. Hidden variables arise in a wide variety of contexts as an important consideration for explaining observed data. However, most of the previous work in hidden variable (or latent variable or theoretical entity) models have imposed either a simple model or a strong structure on the observed dependencies, be it intervariable dependencies or temporal dependencies. We developed new methodologies that allow for a more data-driven approach to the discovery of hidden variables. This work describes a novel method that has been developed for the discovery of hidden variables using an augmented neural network called the LO-net (Latent and Original network architecture). Almost all of the current hidden variable literature aims to build models that can efficiently learn even when hidden variables are present. Their aim is *not* to discover the hidden variables and quantify them but to maintain or improve predictive performance even when hidden variables might be present.

We compare the LO-net to regular neural networks and also Hidden Markov Models (HMMs). The LO-net always does better than the regular neural network model. The prediction performance using HMMs is comparable to that of the LO-net, but the LO-net provides information about the hidden variables. In HMMs, the hidden states are not informative, hence even if the performance of prediction is good, they do not provide any information about the hidden variables. This work provides a new perspective in finding hidden variables. If hidden variables can be discovered and quantified, then we can have a much better understanding of the overall structure of the system.This can then enable us to find answers to many questions that otherwise could not be accounted for.

Section 2 describes some of the work done on finding hidden variables in different contexts. Section 3 gives a description of the method that we have introduced to find hidden variables, the LO-net. Section 4 lists the different experimental domains that have been used and provides a brief explanation for each of the domains. It describes data obtained from a physical robot data, which is one of our main data sets, and many different dynamical systems. Section 5 presents experimental results using the LO-net on the different experimental domains. Section 6 introduces an extension of the LO-net by adding regularization terms in the network. Section 7 presents experimental results using the regularized LO-net. Section 8 shows the results of comparison of the LO-net architecture with Hidden Markov Models (HMMs). Finally, Section 9 describes the contribution of this work and suggests future directions toward improving these methods.

2. Background

A robot may not be able to observe all of the causally relevant variables in its environment. In the real world, a robot is expected to learn something useful given whatever sensors and effectors are available. Even if the robot's task and the environment change, the hardware typically stays the same. Most commonly, inadequate foresight or sensor technology leads to a gap between what the robot can sense and what it would sense in an ideal world. That is, a robot's environment is almost always *partially observable*. This is particularly problematic in the context of learning because most machine learning algorithms never explicitly consider that there is more to the world than what is provided in their input. For example, decision tree induction algorithms split the data with tests on observed features [1]. Constructive induction algorithms go one step further and build new features out of observed features by combining them in various ways [2]. Kernel methods implicitly project the input data into high-dimensional spaces, some with infinitely many dimensions, but each dimension in these spaces corresponds to a composite feature computed from the inputs [3]. Fahlmans and Lebiere cascade correlation architecture dynamically adds hidden nodes to a neural network, but these nodes do not correspond to hidden features of the environment, they provide a more expressive internal representation based on the inputs [4].

There are many algorithms for learning and reasoning in the face of hidden data, and almost all of them are based on the Expectation Maximization (EM) algorithm [5]. EM is used for parameter estimation when the data required to solve the estimation problem are of the form $X = [Y, Z]$ and Y is observable but Z is not. EM uses a parameterized model

to compute the expected value of the hidden data and then uses the expected value to reestimate model parameters. Therefore, EM requires a priori knowledge about the existence, quantity, and role (in the estimation problem) of hidden data. For example, [6] builds a Bayesian network in which hidden variables are manually added to the network as parents of all observable variables.

Similarly, work on planning under uncertainty using, for example, the Partially Observable Markov Decision Process (POMDP) framework assumes knowledge of the number of underlying hidden states [7]. The agent whose world is characterized by the POMDP does not have access to the state that it actually occupies. Rather, the agent maintains a belief state, or probability distribution over states that it might be occupying. This belief state is Markovian, meaning that no additional information from the past would help increase the expected reward of the agent. Again, the goal of this work is not to discover the existence of a hidden state, but to behave optimally given knowledge of the existence of hidden state. More recently, [8] showed that dynamical systems can be represented using Predictive State Representations (PSRs), or multistep, action-conditional predictions of future observations, and that every POMDP has an equivalent PSR. PSRs can look at the past and summarize what happened, and they can also look to the future and predict what will happen. A PSR is a vector of tests [9] which stores the predictions for a selected set of action-observation sequences. Unobserved or hidden states can be fully captured by a finite history-based representation called a looping prediction suffix tree (PST) [10]. They focus on cases of POMDPs where the underlying transition and observation functions are deterministic. Again, the goal of this work is not to discover the existence of hidden state, but to behave optimally given knowledge of the existence of hidden state.

Work on hidden states in the context of reinforcement learning has been going on for some time [11]. McCallum et al. proposed a method called *instance-based state identification*, where raw data from previous experiences is stored directly. The simplest such instance-based technique is the *nearest sequence memory*, which is based on k-nearest neighbors. This technique, though simple, improves learning performance. The main disadvantage of this technique is that, though it learns good policies quickly, it does not always learn the optimal policy. There is a relatively new method known as the deep belief nets (DBNs) which are probabilistic generative models that are composed of multiple layers of stochastic, latent variables with no intralayer connections [12]. Gaussian Process Dynamical Models (GPDMs) have also been built to generalize well from small datasets [13].

Latent variables are important in the psychology and social science research. [14] described three definitions of latent variables: *local independence*, *expected value true score*, and *nondeterministic functions of observed variables* and introduced a new notion of latent variables called "*sample generalizations*." Latent variables can be defined *nonformally* or *formally*. Nonformal latent variables can be considered as *hypothetical variables* or unobserved variables as a data reduction device. Hypothetical variables are variables considered imaginary, that is, not existing in the real world. The third nonformal definition of latent variables defines them as a data reduction device that can be used to describe a number of variables by a small number of factors.

There is a tremendous body of work on time series analysis, much of it in the neural networks literature [15], with a variety of models aimed at predictive tasks and pattern discovery. That body of literature is too vast to review in any meaningful way here. Suffice it to say that very little of it is devoted to the explicit representation of the number and values of hidden variables in multivariate streaming time series.

In most of the previous work, the aim was to design better models that work with hidden variables. Our aim is to actually discover the hidden variables and quantify them. We describe a method that is developed for the discovery of hidden variables using augmented neural networks.

3. Method

Our neural network architecture is called the LO-net which is, at its heart, a multilayer feed-forward network. Conceptually, it consists of a single O-net (original net) and zero or more L-nets (latent nets). The network and its training are structured in such a way that the single output of each L-net approximates the value of one hidden (latent) variable. The outputs of the L-nets then serve as inputs to the O-net along with the values of the observable variables. The O-net is trained to produce as output the values of the observable variables on the next time step.

Somewhat more formally, let the state of some system at time t be the pair (X_t, Z_t), where $X_t \in \Re^n$ and $Z_t \in \Re^m$. X_t is a vector of n observable values and Z_t is a vector of m unobservable values. Crucially, the state of the system at time $t+1$ depends on both X_t and Z_t so that predictions made using solely X_t, the observable values, will be less accurate than those made using Z_t as well. Standard practice would dictate building a model, f, to predict X_{t+1} based on X_t, that is, $\hat{X}_{t+1} = f(X_t)$, where $\hat{X}_{t+1}$ is a prediction that will be compared against the actual values observed at time $t+1$, that is, X_{t+1}. The model f might be a neural network with n inputs and n outputs and a sufficiently large number of hidden units. Note that the network's hidden layer representation is merely a projection of the observable values into a different (usually higher dimensional) space. Clearly this is key aspect of the performance of neural networks, but the hidden layer representation is not meant to, and typically does not, convey information about unobservable variables, that is, elements Z_t.

One standard way of dealing with hidden state in situations like this is to rely on a history of observable values. A theorem by Takens [16] says that for discrete-time deterministic dynamical systems of n variables, it is possible to exactly recover the topology of the system by treating each window of $2n$ consecutive values of just one variable as a state. Practically, this means that predictions can be improved when the input to the model consists of X_t and some number of additional observation vectors from the recent past. Let $\Phi(X_t, \ldots, X_{t-k+1})$ be a vector of length kn obtained by concatenating X_t, $X_{t-1}, \ldots, X_{t-k+2}, X_{t-k+1}$. In general, a model

predicting X_{t+1} will be more accurate given $\Phi(X_t, \ldots, X_{t-k+1})$ as input for an appropriate value of k than one given simply X_t as input when there is hidden state. While in this case it is reasonable to assume that the hidden layer representation in a neural network trained with $\Phi(X_t, \ldots, X_{t-k+1})$ as inputs will extract information about the hidden variables Z_t, it is impossible to tell from the hidden layer representation how many hidden variables there are (i.e., m) or what their values might be at any given time. The LO-net is explicitly designed to produce exactly that information.

Figure 1 shows the structure of an LO-net with one latent network for which $k = 3$. Note that the latent network is a standard 3-layer feed-forward network that takes as input $\Phi(X_t, X_{t-1}, X_{t-2})$ and produces a single output. The O-net is also a standard 3-layer feed-forward network that takes as input $\Phi(X_t, X_{t-1}, X_{t-2})$ and the output of the L-net. The O-net is trained using backpropagation to predict the value of X_{t+1}. Assuming that $k = 3$ is sufficient, the L-net is initially superfluous. That is, the O-net can ignore the output of the L-net and use the history of values to accurately predict X_{t+1}. Nevertheless, we train the network for some period of time, backpropagating the errors in the usual way through both the O-net and the L-net.

Now that the LO-net as a whole is in a good part of the weight space and can predict X_{t+1} with reasonable accuracy, we change the inputs to the O-net. Let $\mathbf{0}^n$ be a vector of n zeros. We initiate training of the LO-net again, starting with its current weight vector but using $\Phi(X_t, X_{t-1}, \mathbf{0}^n)$ instead of $\Phi(X_t, X_{t-1}, X_{t-2})$ as its input. The L-net is unchanged, and its output is still an input to the O-net. This gives the O-net less information about the hidden variables and causes it to rely more on the output of the L-net. Because the L-net has full access to the history of observables, it is forced to provide information about that history to the O-net through its one output. The most informative and parsimonious way to do that is for the L-net to compute the current value of the hidden variable (assuming $m = 1$) and produce that as output. Note that there is an initial drop in accuracy, but it quickly rebounds as the L-net learns to provide a more informative output. Training continues until the error of the O-net stabilizes, the inputs to the O-net are changed one more time so that they consist of $\Phi(X_t, \mathbf{0}^n, \mathbf{0}^n)$, placing even more pressure on the L-net to provide information about the value of the hidden variable, and training continues until the error of the O-net stabilizes.

To estimate the number of hidden variables, we run this procedure with a sequence of LO-nets starting with no latent networks. If adding an additional latent network significantly improves prediction accuracy, we have evidence for a hidden variable. We keep adding latent networks until doing so has no significant impact on the accuracy of the O-net at predicting X_{t+1}. Empirically, we have strong evidence that the number of latent networks in the previous LO-net corresponds to the number of hidden variables and their outputs are approximations (up to a scaling factor) of the values of the hidden variables. Note that in contrast to a standard neural network (and most other kinds of models one might use in this case), the LO-net makes explicit information about hidden variables.

Note that this method makes no assumptions about the numbers of observable or unobservable variables and will work with histories of arbitrary size. Figures 2(a) and 2(b) show the one- and two-LO-net architectures, respectively. Note that in the latter the outputs of both L-nets act as inputs to the O-net and their values are presented in parallel.

The L-net and the O-net have the same network structure. The neural network architecture has been implemented in Matlab, using Matlab's Neural Network Toolbox. The Neural Network Toolbox provides a flexible network object type that allows many kinds of networks to be created and then used with other functions in the toolbox. This flexibility is possible because the networks have an object-oriented representation, allowing a user to define various architectures and assign different algorithms to those architectures. To create a custom network, we start with an empty network (obtained with the network function) and set its properties as desired. The network object consists of many properties that can be set to specify the structure and behavior of the network.

In our implementation, the network has each of its layers' weights and biases initialized with the Nguyen-Widrow layer initialization method [17]. The Nguyen-Widrow method generates initial weight and bias values for each layer so that the active regions of the layer's neurons are distributed approximately evenly over the input space. The values contain a degree of randomness, so they are not the same each time this function is called. The training function used to update the weight and bias values in the network is gradient descent with momentum and adaptive learning rate backpropagation. The parameter lr indicates the learning rate, similar to simple gradient descent. The parameter mc is the momentum constant. mc is set between 0 (no momentum) and values close to 1 (high momentum). A momentum constant of 1 results in a network that is completely insensitive to the local gradient and therefore does not learn properly. The learning rate (lr) chosen is 0.01. The momentum constant used was 0.9. For each epoch, if performance decreases toward the goal, then the learning rate is increased by the factor lr-inc (1.05). If performance increases by more than the factor max-perf-inc (1.04), the learning rate is adjusted by the factor lr-dec (0.7) and the change that increased the performance is not made. A transfer function is used to calculate the ith layer's output given the layer's net input during simulation and training. Backpropagation is used to calculate the derivatives of performance (perf) with respect to the weight and bias variables X, where performance is measured according to the mean squared error. Each variable is adjusted according to gradient descent with momentum given by

$$dX = \text{mc} * dX\text{prev} + \text{lr} * (1 - \text{mc}) * \frac{d\text{perf}}{dX}, \tag{1}$$

where dXprev is the previous change to the weight or bias. Gradient descent with momentum depends on two training parameters. The transfer function used to calculate the hidden layer's output is the *tan-sigmoid* transfer function and the output layers use a *linear* transfer function. The hyperbolic tangent sigmoid transfer function takes the net inputs and returns a value squashed into $[-1, 1]$.

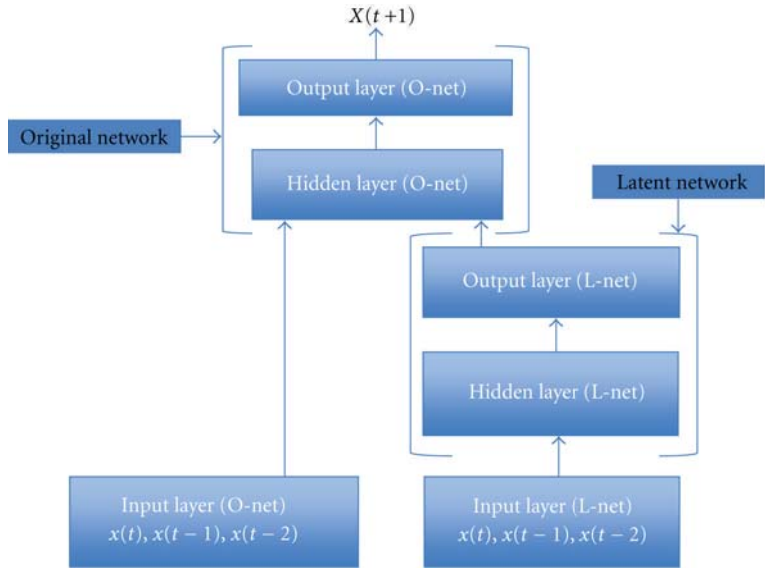
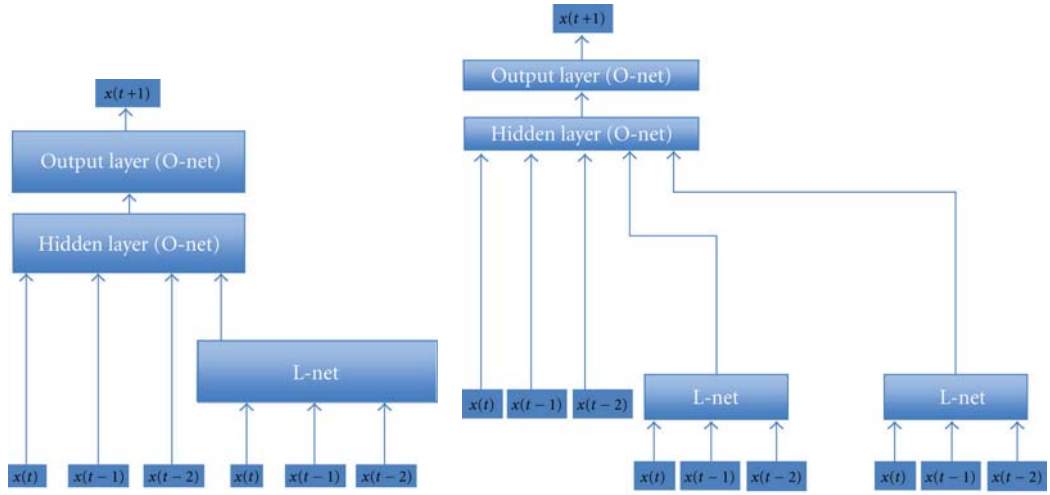


Figure 1: A generic LO-net with one latent network.



(a) An LO-net with one L-net

(b) An LO-net with two L-nets

Figure 2: Example LO-net architectures.

4. Test Domains

This section describes the various domains from which data were drawn to test the LO-net approach. In all cases, the goal is to determine the number of hidden variables and to approximate, as accurately as possible, their values. In practice, the trained LO-net would be used in two qualitatively different ways. The first is by humans to better understand the domain. Knowing, for example, that there are three hidden variables that affect the price of a financial security or the blood pressure of an intensive care unit patient, and being able to see how they change over time would be extremely valuable for domain experts. The second way in which the results would be used is by machines for making more accurate predictions. The output of the O-net is more accurate by virtue of having the L-nets, and the values produced by the L-nets can be used as inputs to other models constructed for other purposes in the underlying domain.

4.1. Robot Data. We take a robot mounted camera taking pictures of 2D objects. Consider the coordinate frame shown in Figure 3, center of projection at the origin and an image plane orthogonal to the z axis. The robot's vision system extracts the following information for a given box:

s_t: the size of the object in the image plane,

x_y: the x coordinate of the centroid of the object in the image plane,

y_t: the y coordinate of the centroid of the object in the image plane.

Each object has an objective size S_t and objective location (X_t, Y_t, Z_t), relative to the origin of the coordinate frame.

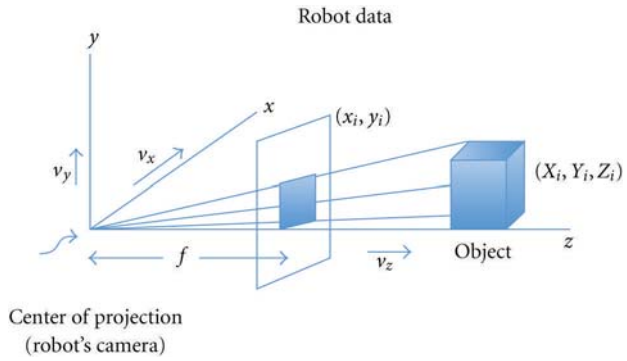


Figure 3: Perspective projection of the centroid of an object in 3D space onto the robot's image plane.

Assume a simple case where the robot is moving along the z-axis with a constant velocity $v_z = 1$. The quantities required to predict the next value of x_{t+1} and y_{t+1} are observable except Z, the distance of the robot from the box.

$$x_{t+1} = x_t + \frac{v_z x_t}{z_t}, \qquad y_{t+1} = y_t + \frac{v_z y_t}{z_t}. \tag{2}$$

The perceived size of an object s_t depends on the objective size S and the distance Z of the object as follows:

$$s_t = \frac{S_t}{Z_t^2}. \tag{3}$$

The robot's perception of the size of the target thus changes with the distance from the target, though the target itself is of constant size. The quantities S and Z are not observable, so s cannot be directly estimated.

Real-world data was provided for this project by a surveyor SRV-1 Blackfin robot. The robot consists of a camera mounted on a pair of tank style treads that can be controlled remotely through a user interface on a laptop. The robot was placed in a fairly uniform environment (in this case, the UMBC Department of Computer Science lobby) and driven by a human around several targets. The targets are brightly colored boxes, easily distinguishable from the surrounding environment by standard image processing software. The surveyor would approach a target from different angles, keeping it in view the entire time for some trials, and occasionally facing in different directions for others. Each frame transmitted from the surveyor's camera was recorded for later processing. The computation done on these frames consisted of counting the number of pixels that were present in a certain color range (giving us the surveyor's perception of the size of the box), and the centroid of the pixels of that color. Before each experiment, the color range was calibrated to avoid the surveyor mistaking other things for its target.

4.2. Lorenz Attractor. The Lorenz attractor [18] is a dynamical system which has three variables x, y, and z. It is defined by a system of three equations that give the first derivative of each of the three variables

$$\frac{dx}{dt} = \sigma(y - x), \qquad \frac{dy}{dt} = x(\rho - z) - y, \qquad \frac{dz}{dt} = xy - \beta z. \tag{4}$$

We can test our methods using the Lorenz attractor, or any other dynamical system, by generating data for x, y, and z over time and then hiding one or two of the variables.

4.3. Character Trajectories Dataset. We also used a practical data set from the UCI machine learning repository [19], the character trajectories data set. The data consists of 2858 character samples. We randomly picked one character sample from the data set to run the experiments. The data stored for each character was the x and y coordinates and the pen tip force during writing a character and were captured at 200 Hz. The pen tip force will depend on both the x and y coordinates.

4.4. A Dynamical System with Four Variables. Next we generated a dynamical system with four variables, x, y, z, and h. This dynamical system is defined by the following set of equations:

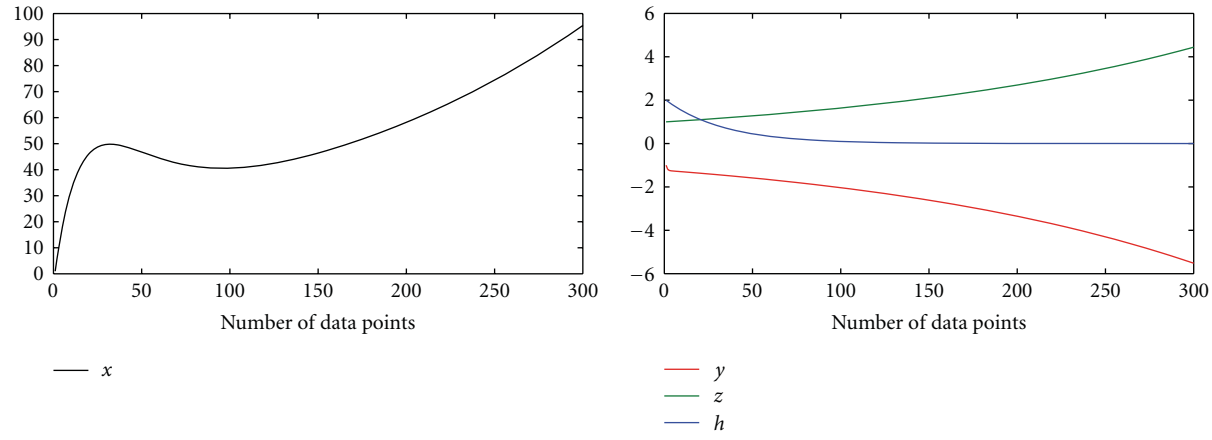
$$x_{t+1} = ay_t + cx_t + z_t + h_t^2, \qquad y_{t+1} = ay_t - z_t, \qquad z_{t+1} = bz_t, \qquad h_{t+1} = ch_t, \tag{5}$$

where $a = 0.2$, $b = 1.005$, and $c = 0.97$. The initial values of the variables are $x_0 = 1$, $y_0 = -1$, $z_0 = 1$, and $h_0 = 2$. Figures 4(a) and 4(b) plot the values of (x_t, y_t, z_t, h_t) over time.

Note that the variable x depends on all four variables x, y, z, and h. This dynamical system will be referred henceforth as the *four-variable dynamical system* (FVDS).

5. Experimental Results Using the Unregularized LO-Net

5.1. Simulated Robot Data. The first set of experiments are performed with simulated robot data. A robot collects data by moving towards a box with a constant velocity. It records the x and y coordinates of the centroid of the box and the size of the box in its vision. The network is trained for 400 epochs since around that time the mean squared error (MSE) converges to a very low value. The dataset is divided into training and test datasets. The MSE is calculated over the test dataset. Figure 8(a) plots the MSE for predicting the next value of x. The solid line shows the performance when the current value of x (x_t) is input and the next value of x (x_{t+1}) is predicted, using only the original network. The dashed line shows the performance when the current and the previous



(a) Variable x

(b) Variables y, z, and h

FIGURE 4: Sample values from the four-variable dynamical system.

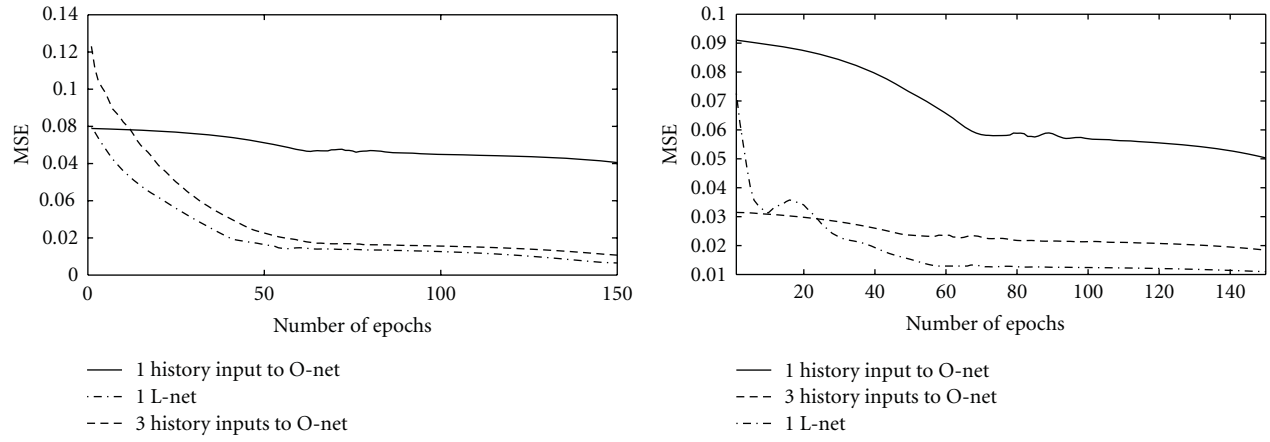
two x values (x_t, x_{t-1}, x_{t-2}) are input to the original network and the next value of x x_{t+1} is predicted. The dash-dot line and the dotted lines show the performance with one and two latent networks, respectively. Initially for the first 100 epochs, the current and the previous two x values (x_t, x_{t-1}, x_{t-2}) are input to the original and latent networks. The output of the latent networks are also given as an input to the original network as shown in Figure 1. In the next 100 epochs, one history value x_{t-2} is dropped from the original network and training is continued. In the last 200 epochs, the original network is fed with only the current value of x (x_t) and the output from the latent network. All the figures plot the MSE versus the number of epochs. The plots show the MSE of the last 150 epochs where the input to the original net is the current value of the variable and the output from the latent net. The x-axis plots the number of iterations and the y-axis plots the mean square error (MSE) in the following figures.

Figures 5(a) and 5(b) plot the performance curve for predicting x_{t+1} and y_{t+1}, respectively. In both cases, only z is hidden. It is observed that the maximum performance improvement is achieved using one-LO-net architecture in both cases. Figure 6 plots the performance for predicting the size s of the box when there is more than one box in the robot's field of vision. There are two hidden variables: z, the distance of the robot from the box, and S, the actual size of the box. The performance is best in this case with the two-LO-net architecture. The performance decreases when a third latent network is added. Figure 7 shows the outputs from the L-nets in the two-LO-net architecture for predicting the size s of the box.

5.2. Real Robot Data. The next set of experiments are performed with real robot data. The experimental setup is described above. It is the exact same scenario as for the simulated robot data. From Figure 8(a), the performance of the network with three history values for predicting x_{t+1} is better than the performance with just the current value. The one-LO-net architecture performs better than the two-LO-net architecture. There is one value which is unobservable for the prediction of x_{t+1}, which is the distance of the robot from the box. While trying to predict the next value of x with just the previous value of x, one variable is hidden to x, on which it is dependent. The output from the latent network in the LO-net architecture provides input to the original net that improves its performance. The latent network posits the presence of a hidden variable. It approximately learns the value of the hidden variable. Initially three history values are provided as input to the original network but when more learning history values are dropped, the input from the latent network becomes more important. We propose that the backpropagation algorithm updates the weights of the latent network so as to approximate the hidden variable. Similar results can be seen in the case of predicting y_{t+1}. The two-LO-net architecture performs best when predicting the size of the box as seen in Figure 8(b). The size of a box depends on the actual size of the box and the distance of the box from the robot. Hence for predicting the next value of the size perceived by the robot, the two hidden variables are the actual size of the box and the distance of the robot from the box. Adding a third latent network again reduces the performance of learning.

From these results, we conclude that the performance of prediction of the future values can be improved by using the LO-net architecture. Not only does it suggest the existence of hidden variables, but it also gives an estimate of the number of hidden variables. For example, in this case where x depends only on x and Z where Z was hidden, one-LO-net architecture gave the maximum performance improvement. While predicting future values of s which depends on two hidden variables S and Z, the two-LO-net architecture performs the best.

5.3. Lorenz Attractor. For the Lorenz attractor, the performance of the network with three history values for predicting x_{t+1} is better than the performance with just the current value. The performance of the one-LO-net architecture is the best in this case as seen in Figure 9(a). It performs better than the two-LO-net architecture. The variable x in the Lorenz



(a) MSE when predicting x_{t+1}

(b) MSE when predicting y_{t+1}

Figure 5: Simulated robot data.

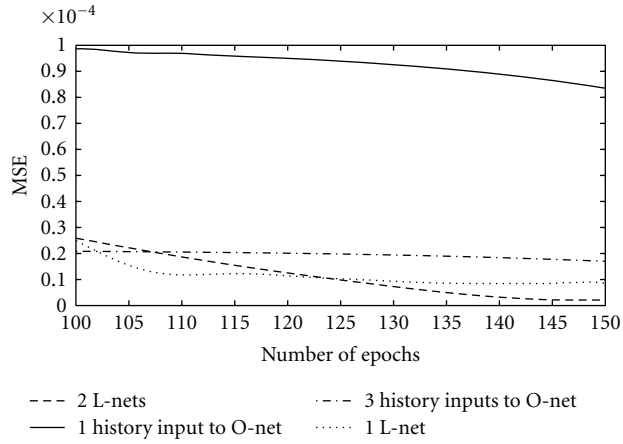
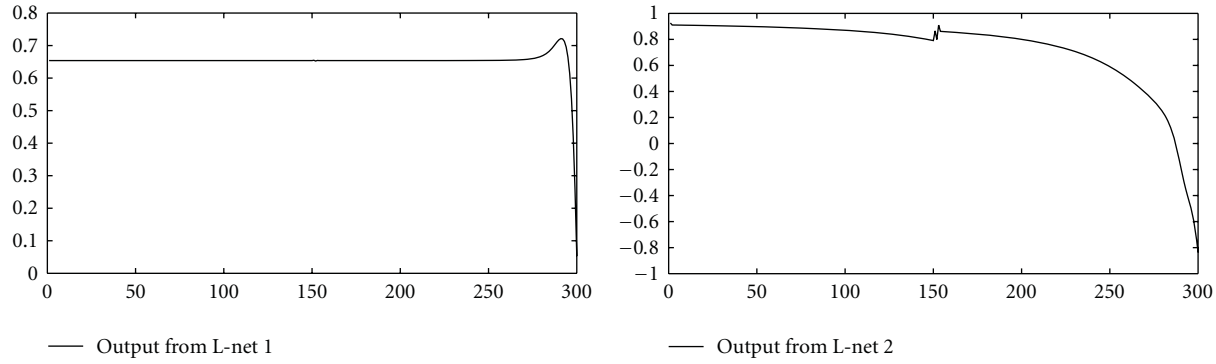


Figure 6: Simulated robot data: MSE when predicting s_{t+1}.

attractor depends only on variables x and y. While trying to predict the next value of x with just the previous value of x, variable y is missing on which x is dependent. The output from the latent network in the LO-net architecture provides input to the original network that improves its performance. Figure 9(b) shows the plot of the MSE for predicting variable y. Variable y depends on all the three variables x, y, and z. Here the performance is best in the case of the two-LO-net architecture. One latent network does improve the performance from that of using just the original network with history inputs. The two latent networks in this case approximate the two hidden variables x and z.

5.4. Character Trajectory Dataset. The LO-net was also tested on a practical data set from the UCI machine learning repository [19], the character trajectories data set. The network predicts the next value of the pen tip force using the current and the history values of the pen tip force. Since the pen tip force depends on both the x and y coordinates these are the two hidden variables. Here the network is trained for 300 epochs and the plots show the MSE of the last 100 epochs. Figure 10 shows that the two-LO-net architecture gives the maximum performance improvement for predicting the next value of the pen tip force.

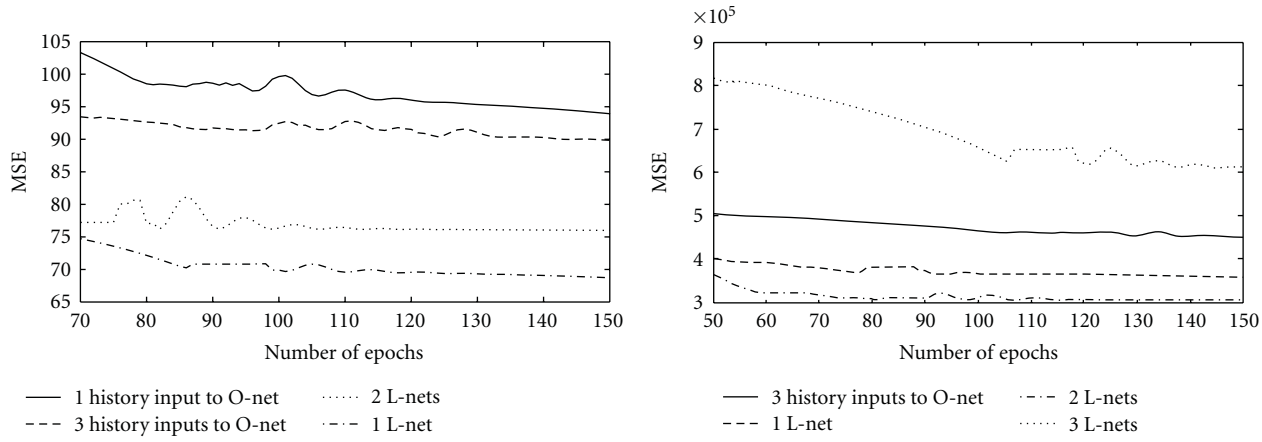
5.5. Four-Variable Dynamical System (FVDS). Figure 11 shows the performance curve for predicting x_{t+1}. The value of x_{t+1} depends on the variables x_t, y_t, z_t, and h_t. The O-net and the LO-net are provided with only the current and history values of x. Hence, for predicting x_{t+1}, there are three hidden variables present in the system. The maximum performance improvement for predicting x_{t+1} is achieved by the LO-net architecture with three L-nets as seen in Figure 11. When a fourth latent network is added, the performance decreases. These results show that the LO-net architecture works well even in the presence of larger numbers of hidden variables. It can detect the true number of hidden variables, and scaling this architecture up as the number of hidden variables increases works well. The correlations between the values of the hidden variables and the outputs from the L-nets are computed. In the one-LO-net architecture, the output from L-net approximates the value of the output from the O-net. As a result, the performance using the one LO-net architecture improves over the performance with three history values as input to the O-net, as seen in Figure 12. Figures 13, 14, and 15 compare the nature of the outputs from the L-nets and the hidden variables in the three-LO-net architecture. The correlation between the output from L-net 2 and the hidden variable y is 0.8964. Correlation between output form L-net 1 and hidden variable z is 0.8499. The correlation between the output from L-net 3 and the hidden variable h is 0.6808. We have developed a neural network architecture that can find the existence of hidden variables. The number of hidden variables can be found by iteratively adding latent networks to the original network until adding a new latent network does not significantly help.



(a) Output from L-net 1 for predicting s_{t+1} in 2-LO-net architecture (z and S are the hidden variables here)

(b) Output from L-net 2 for predicting z_{t+1} in 2-Lo-net architecture (z and S are hidden variables here)

Figure 7: Simulated robot data: outputs from the L-nets.



(a) MSE when predicting x_{t+1}

(b) MSE when predicting the size s_{t+1} of the box with two boxes in the robot's field of vision

Figure 8: Real robot data.

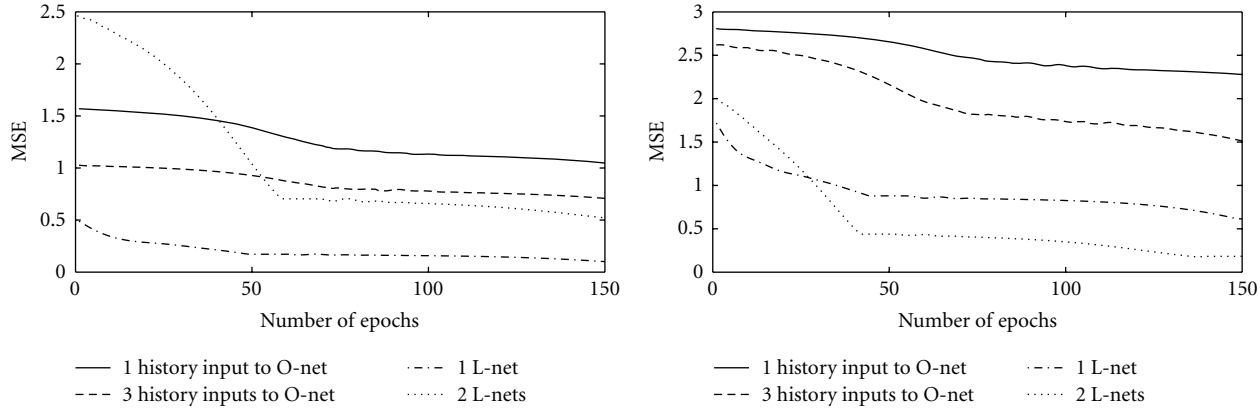
6. Regularization of the LO-Net Using Two Different Penalty Terms

The latent network sometimes tries to approximate the predicted output from the original network in the LO-net architecture. When we are trying to predict x_{t+1} with history values of x as input to the LO-net, the L-net can sometimes try to approximate the predicted value x_{t+1} instead of the hidden variable, that is y in this case. Since the latent network is not provided with any example outputs, the only way it learns is from the errors back-propagated to it. In many cases, the predicted variable and the hidden variable are highly correlated; then the latent variable might just approximate the predicted variable. Even when the predicted variable and the latent variable are not correlated, the overall network may "cheat" by forcing the latent network to do all of the work. We introduce a penalty term which is added to the error for the latent network to prevent this situation. The penalty term is formulated so that the latent network does not learn the output of the original network. The penalty term is applied to the output of the L-nets, and the gradient descent calculation is modified to include both the weight changes back-propagated from the output of the O-net and the weight changes suggested by the penalty term. That is, there is now pressure inside the L-nets to change the weights so as to minimize the penalty term in addition to the pressure to change the weights to improve the accuracy of the LO-net as a whole. These pressures are balance by selecting the value for a single parameter.

6.1. Distance Penalty P1. The first penalty term is given by

$$P = w * e^{-(O-L)^2}, \tag{6}$$

where O is the output from the original network, that is, the output of the whole network and L is the output of the latent network, and w is the weight of the penalty term which lies between zero and one [20]. Whenever the output from the original network is close to the output from the latent network, the penalty term is high, otherwise it is low. If the value of w is high, then the penalty term plays an important role in the learning of the latent network. The output from the latent network in this case will be



(a) MSE when predicting variable x_{t+1}

(b) MSE when predicting variable y_{t+1}

Figure 9: Lorenz attractor.

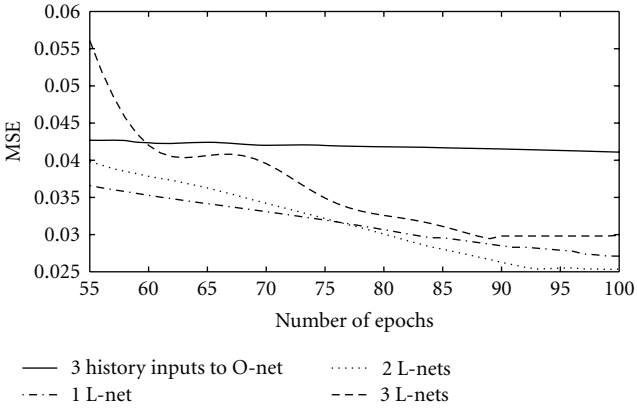


Figure 10: Performance curve for the pen tip force (character trajectories data set from the UCI machine learning repository).

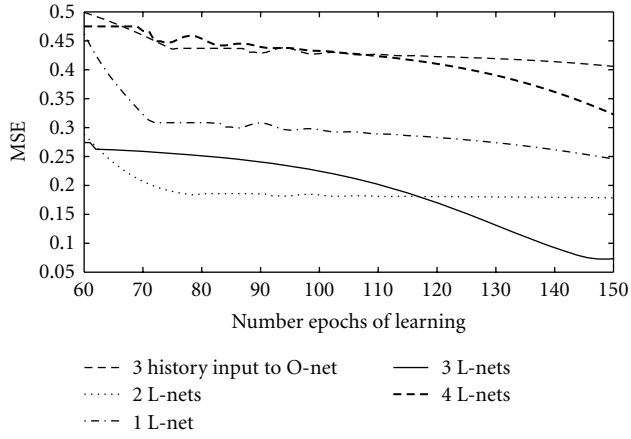


Figure 11: FVDS: Performance curve for predicting x_{t+1} with O-net (inputs are three history values x_t, x_{t-1}, and x_{t-2}) and LO-net architecture with one, two, three, and four L-nets.

significantly less correlated to the output from the original network. If the predicted variable and the hidden variable are highly correlated, then a high value of w will result in the latent net not learning the hidden variable because it will get a high penalty on being close to the predicted output and will deviate from that. Deviating from learning the predicted output will also move the network from learning the hidden variable since the hidden variable might be very close to the predicted output. Hence choosing an appropriate value of w for maximum performance improvement is crucial.

In a case where the two-LO-net architecture gives the maximum performance improvement, there are two different penalty terms for the two latent networks. For the first latent network, the penalty is same as before that is,

$$P = w * e^{-(O-L1)^2}, \tag{7}$$

where again O is the output from the original network and $L1$ is the output from the first latent network. The penalty term for the second latent network is

$$P = w * e^{-(L1-L2)^2}, \tag{8}$$

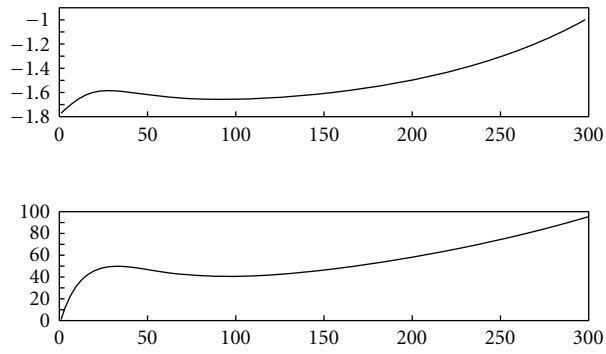


Figure 12: FVDS (one-LO-net architecture): comparing output from L-net 1 and variable x.

where again $L1$ is the output from the first latent network and $L2$ is the output from the second latent network. We do not want the second latent network to learn the output from the first latent.

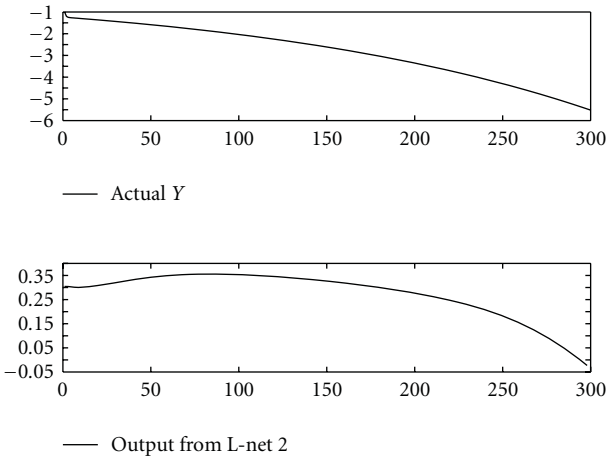


Figure 13: FVDS (three-LO-net architecture): comparing output from L-net two and hidden variable y, correlation is 0.8964.

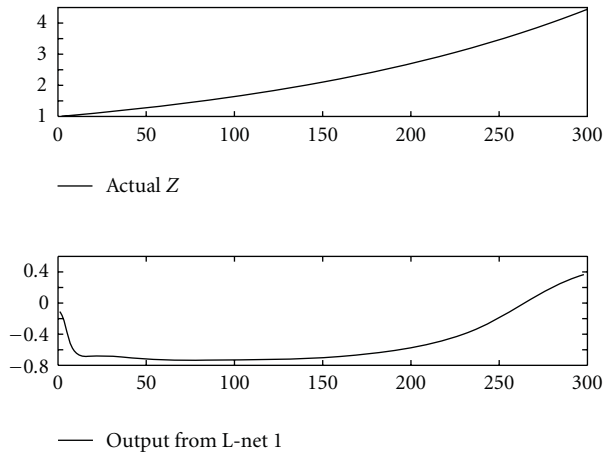


Figure 14: FVDS (three-LO-net architecture): comparing output from L-net one and hidden variable z, correlation is 0.8964.

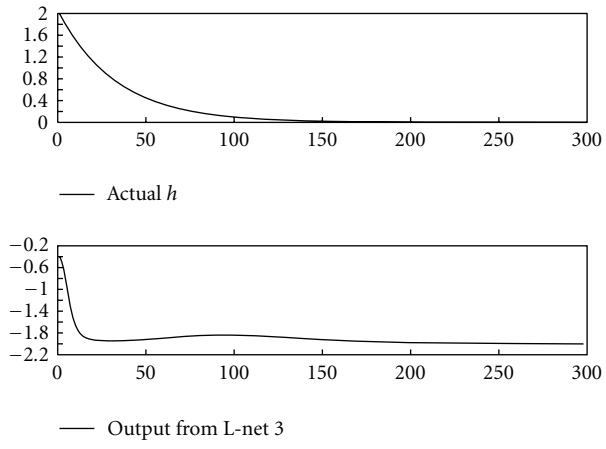


Figure 15: FVDS (three-LO-net architecture): comparing output from L-net 3 and hidden variable h, correlation is 0.6808.

6.2. Decorrelation Penalty P2. In some cases, the output from the latent network has to be decorrelated from the output of the original network. Decorrelation is a process that is used to reduce the autocorrelation within a dataset or the cross-correlation between two data sets while preserving the other aspects of the data. The extent of decorrelation depends on the weight of a decorrelation penalty. A new penalty term $P2$ is introduced which is the square of the covariance between the output of the O-net and L-net. The idea of this penalty term was adopted from [21]. The penalty term is

$$P2 = (\mathrm{Cov}_t(O, L))^2, \tag{9}$$

where O is the output from the O-net and L is the L-net. This can be written as

$$P2 = [\mathrm{mean}(O,\ L) - \mathrm{mean}(O)\ \times\ \mathrm{mean}(L)]^2. \tag{10}$$

The neural network toolbox in Matlab provides a method, calcgrad.m, which computes the gradient of the error with respect to the input weights, layer weights, and bias weights. We adapted the code in this file to incorporate the penalty term. The gradients of the penalty term with respect to the layer weights of the hidden layer in the latent network are computed and added it to the gradients of the error. The derivative of the penalty term $P2$ with respect to the weights of the hidden layer can be written as

$$\frac{dP2}{dw} = 2 \times P2 \times \frac{dP2}{dL} \times \frac{dL}{dw}, \tag{11}$$

where w is the weight vector of the hidden layer of the latent network. The derivative of L with respect to w is the output from the hidden nodes, $I_{h\times N}$, where h is the number of neurons in the hidden layer and N is the number of instances of the training set, that is

$$\frac{dL}{dw} = I. \tag{12}$$

Note that O and L are vectors of size N.

Next the derivative of $P2$ with respect to L (the output of the latent network) is computed. We know that

$$P2 = [\mathrm{mean}(O,\ L) - \mathrm{mean}(O)\ \times\ \mathrm{mean}(L)]^2. \tag{13}$$

This can be written as

$$P2 = \frac{O \cdot L}{N} - \left(\frac{O \cdot A}{N}\right) * \left(\frac{L \cdot A}{N}\right), \tag{14}$$

where A is a vector of ones of size N.

$$\begin{aligned}\frac{dP2}{dL} &= \frac{d(O \cdot L/N - (A \cdot O/N) \times (A \cdot L/N))}{dL} \\ &= \left(L \cdot \frac{dO}{dL} + O \cdot \frac{dL}{dL}\right)/N \\ &\quad - \left(\left(A \cdot \frac{dO}{dL}\right)/N \times \left(\frac{A \cdot L}{N}\right)\right) \\ &\quad + \left(\left(A \cdot \frac{dL}{dL}\right)/N \times \left(\frac{A \cdot O}{N}\right)\right),\end{aligned} \tag{15}$$

where dO/dL and dL/dL are $h \times N$ matrices computed numerically. Finally the gradient of the penalty with respect to the weights is

$$\frac{dP2}{dw} = 2 \times P2 \times \frac{dP2}{dL} \times L, \tag{16}$$

which is a vector of size h. These gradients are added to the gradient of the errors with respect to the weights of the latent network.

7. Experimental Results Using the Regularized LO-Net

7.1. Simulated Robot Data. The results of using the regularized LO-net on the simulated robot data are shown in this section. The performance of predicting x is best with the one LO-net architecture using the first penalty term $P1$ with weight $w = 0.0001$ (Figure 16(a)). The performance of predicting y is also best with the one LO-net architecture using the first penalty term $P1$ with weight $w = 0.0001$ (Figure 16(b)).

7.2. Lorenz Attractor. Regularization of the latent network improves the approximation of the hidden variables in the Lorenz Attractor domain. In Figure 17(a), the solid line shows the performance of predicting x with three history values of x. The dash-dot line shows the performance with the one LO-net architecture. The dash line shows the performance with the one LO-net architecture using the first penalty term (squared error $P1$), and the dotted line shows the performance with the second penalty term (covariance $P2$). It is observed from Figure 17(a) that the regularization with the second penalty terms performs the best in this case. The next value of y in the Lorenz attractor depends on both x and z, hence there are two hidden variables. As seen before, in this case, the two-LO-net architecture performs the best. The maximum performance improvement is achieved with the second penalty term $P2$, as seen in Figure 17(b). Similar results are observed when predicting z. The decorrelation penalty term performs better than the distance penalty term for cases where the observed variables are periodic.

7.3. Real Robot Data. Next the effect of adding a penalty term in the latent network to that of using no penalty is compared in the real robot data domain. All of the results shown are the best of ten iterations of the same experiment. Both the x and y coordinates of the box observed by the robot depend on the hidden variable Z, which is the distance of the robot from the box. Therefore, the maximum performance improvement is achieved using the one LO-net architecture, as explained before. Figures 18(a) and 18(b) show the performance of adding a penalty term to the error back-propagated to the latent network for predicting the x and the y coordinates. Similar results were seen in the domain when predicting the y coordinate with regularization. For predicting the next value of x, x_{t+1}, the maximum performance improvement is achieved for $w = 0.05$. For predicting y_{t+1}, the maximum improvement in performance is achieved for $w = 0.5$. The robot in the real world randomly starts from a point in the coordinate space and moves towards the object. In our set of experiments, the values of variable x are higher than the values of variable y. Therefore, the outputs from the O-net are larger for x than for y. It is observed that the outputs from the L-nets are small in both cases, which makes the difference $(O - L)$ larger for x compared to y. Hence the maximum improvement in performance is achieved with a smaller value of weight in the case of x than in the case of y. It is seen that the value of w plays an important role in the amount of performance improvement.

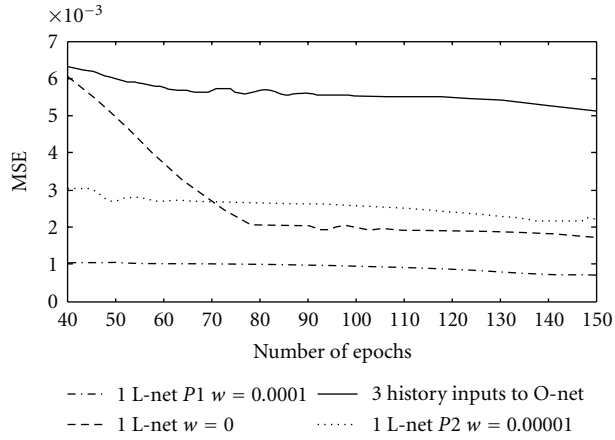
Figure 19 shows the effect of adding a penalty term for predicting the next value of size, s_{t+1}. The size of the box observed by the robot depends on two hidden variables: S, the actual size of the box and Z, the distance of the robot from the box. In this case, the two-LO-net architecture gives the maximum performance improvement. The maximum performance improvement is achieved for $w = 0.005$.

8. Comparison of the LO-Net Architecture with Hidden Markov Models (HMMs)

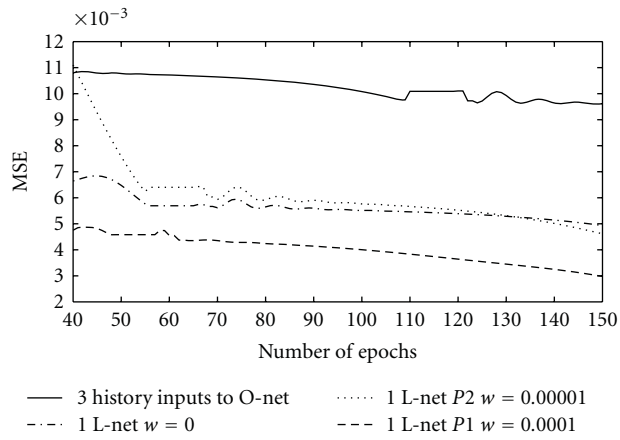
Hidden Markov Models (HMMs) are Markov processes with unobserved or hidden states [22]. In HMMs, the states are not directly observable, but the outputs, which are dependent on the states are observable. HMMs have a wide variety of applications in speech and handwriting recognition, music scores, and other time series domains. We have used ergodic continuous HMMs, in which the outputs given the hidden states are modeled using Gaussian distributions. Maximum likelihood parameter estimation is performed using the Baum-Welch algorithm. The Verterbi path (the most probable sequence of the hidden states) is computed, and the predicted output at each time step is computed as the estimated mean of the Gaussian distribution conditional on the most probable hidden state at that time step. Tables 1 and 2 show the MSEs for predicting x and z variables of the Lorenz attractor data using HMMs and LO-net. The next value of variable x depends on the current values of the variables x and y. There is one hidden variable for predicting x, namely, y. The next value of variable z depends on the current values of x, y, and z. Hence there are two hidden variables for predicting z, namely, x and y. The MSEs for predicting x using both the models are very similar again as seen in Table 1. The MSE for predicting z is lower using the LO-net architecture than the HMM model shown in Table 2.

Tables 3 and 4 show the MSEs of predicting the x- and y-coordinates in the simulated robot data using HMMs with the number of states varying from 100 to 250, as well as the LO-net. The input to the networks is a vector of 300 timesteps of the values of the observed variable. Recall that both x and y depend on one hidden variable, the distance of the robot from the object z. The MSEs for prediction are very similar using both the HMM model and the LO-net architecture.

Table 5 shows the MSEs for predicting the pen tip force from the Character Trajectory Dataset again using HMMs and LO-net. In this case, LO-net performs better than the

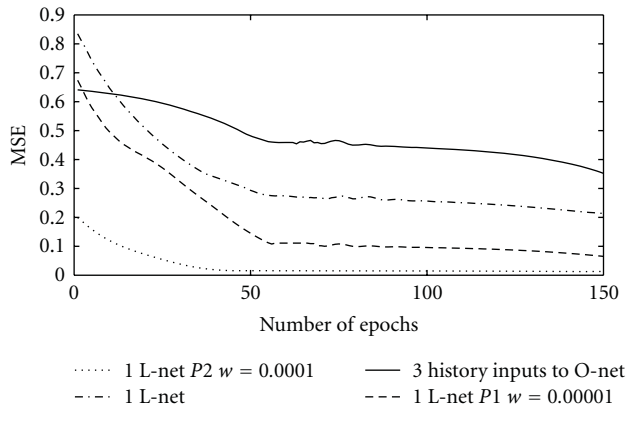


(a) Performance curve for predicting x_{t+1} (net with regularization)

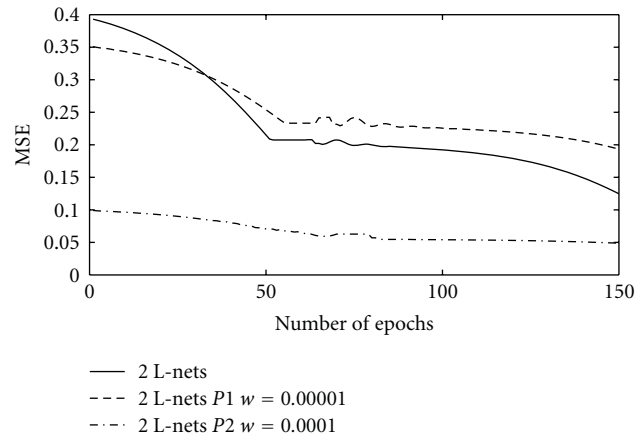


(b) Performance curve for predicting y_{t+1} (net with regularization)

Figure 16: Simulated robot data.

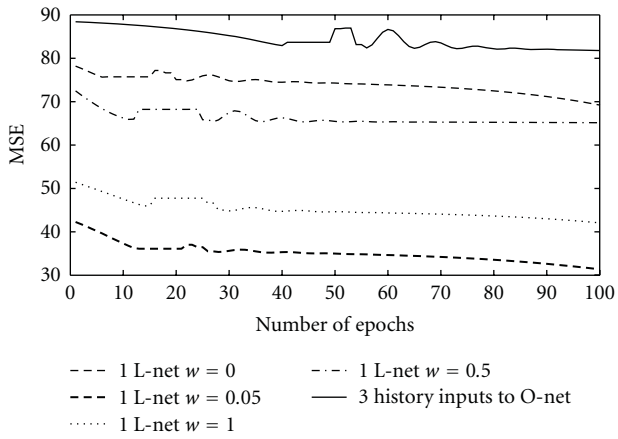


(a) Performance curve for predicting variable x_{t+1} (net with regularization)

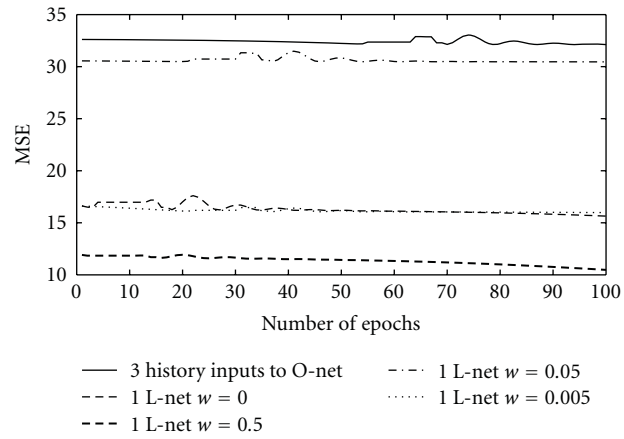


(b) Performance curve for predicting variable y_{t+1} (net with regularization)

Figure 17: Lorenz attractor.



(a) Performance curve for predicting x_{t+1} with different values of the weight (w) for the distance penalty term ($P1$)



(b) Performance curve for predicting y_{t+1} with different values of the weight (w) for the distance penalty term ($P1$)

Figure 18: Real robot data.

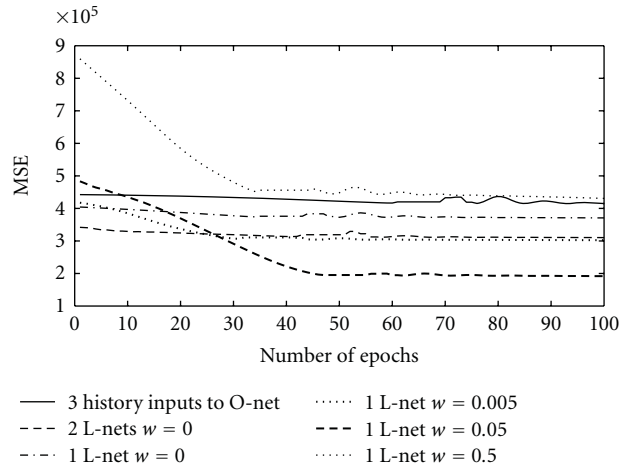


Figure 19: Real robot data: performance curve for predicting s_{t+1} with different values of the weight (w) for the penalty term (distance penalty $P1$).

Table 1: Lorenz attractor (predicting x_{t+1}): mean squared errors (MSEs).

HMM		LO-net
No. of states	MSEs	MSE
150	0.0383	0.0108
200	0.0275	
250	0.017	

Table 2: Lorenz attractor (predicting z_{t+1}): mean squared errors (MSEs).

HMM		LO-net
No. of states	MSEs	MSE
150	0.0739	0.058
200	0.071	
250	0.0701	

Table 3: Simulated robot data (predicting x_{t+1}): mean squared errors (MSEs).

HMM		LO-net
No. of states	MSEs	MSE
100	0.004	0.0008
150	0.0036	
200	0.0027	
250	0.00086	

HMMs. The pen tip force depends on two hidden variables the x and the y coordinates of the pen tip on the paper.

The performances using the HMM model and the LO-net architecture are comparable in the presence of one hidden variable. It is observed that the LO-net architecture performs better than the HMM model when two hidden variables are present. This might be because when there is one hidden variable the hidden states correspond to the one hidden variable. When there are two hidden variables, more hidden states are needed to capture the nature of the two hidden variables. The HMM model does not scale well as the number of hidden variables increases.

Table 4: Simulated robot data (predicting y_{t+1}): mean squared errors (MSEs).

HMM		LO-net
No. of states	MSEs	MSE
150	0.071	0.0045
200	0.0099	
250	0.0076	

Table 5: Character trajectories dataset (predicting the pen tip force): mean squared errors (MSEs).

HMM		LO-net
No. of states	MSEs	MSE
50	0.0157	0.0067
100	0.0131	

The HMM model does improve prediction in the presence of hidden variables, but it does not provide any information *about* the hidden variables. Adding hidden states to the HMM can improve its predictive performance, but the number of hidden states in the model is not a reflection of the number of hidden variables in the system. Our LO-net architecture, on the other hand, is successful not only at discovering the number of hidden variables but also in providing estimates of the values of the hidden variables.

9. Conclusion and Future Work

We presented a novel neural network architecture, the LO-net, that was able to address three basic tasks related to hidden variables: (i) discovery, (ii) cardinality, and (iii) estimation. The LO-net is a combination of two networks, the O-net and L-nets, where the outputs from the L-nets are inputs to the O-net. The network is trained using gradient descent backpropagation. History values are provided as inputs to both the networks in the LO-net. As learning progresses, history values are dropped from the O-net. At the end of learning, the L-nets approximate the values of the hidden variables. Experiments showed the utility of the proposed approach using sensor data obtained from the camera of a mobile robot in which the sizes and locations of objects in the visual field are observed but their sizes and locations (distances) in the three-dimensional world are not. The LO-net was also tested on a variety of nonlinear dynamical systems and simulated data.

In the real robot data, which is a domain that has motivated much of this work, we observed that the performance for predicting x_{t+1} and y_{t+1}, which are the coordinates of an object in the robot's visual field, improved by adding an L-net to the O-net. In the robot's field of vision, the distance of the robot from the object is hidden and the x and the y coordinates depend on this hidden variable. We also found that the output from the L-net approximated the value of the hidden variable. The output from L-net

does not provide the exact value of the hidden variable but its output approximates the effect of the hidden variable. Outputs from the L-nets were shown and compared to the values of the hidden variables in the presence of one or two hidden variables. The correlations between the outputs from the L-nets and the values of the hidden variables were computed; in most cases the correlations were high (in the range of 0.4 to 0.9). However, in some cases the outputs from the L-nets are more correlated with the output from the O-net than that with the hidden variables. That is, the L-nets were approximating the output from the O-net and not the values of the hidden variables. This led to regularize the L-nets to prevent the L-net from mimicking the output from the O-net.

A penalty term was added to the error back-propagated to the L-net. Experiments were performed using two different forms of the penalty term. The first penalty term, $P1$, is the square of the difference between the outputs from the O- and L-nets, which provides a high penalty if the O- and L-net outputs become similar. This penalty term is successful in pushing the outputs from the L-nets away from the O-net in some domains. However, $P1$ cannot always reduce the correlation between the outputs from the L-nets and O-net. A second penalty term, the decorrelation penalty, $P2$, is introduced to try to decorrelate the two outputs. This additional penalty is able to improve performance of the LO-net in some domains. The performance of the regularized LO-net is compared to that of the vanilla (unregularized) LO-net across many different dynamical systems as well as our robot data.

We also compared the performance of the LO-net with HMMs and found them to be comparable when only one hidden variable is present. The LO-net performs better than HMMs when two hidden variables are present. A disadvantage of HMMs compared to the LO-net is that HMMs do not scale well with the number of observations or the number of hidden variables and do not provide estimation and cardinality detection of the hidden variables unlike the LO-net. The LO-net has the advantage of providing insights into the nature of the system being investigated through its ability to detect and estimate hidden variables.

While our results are extremely encouraging, there are clearly limitations of this work, some of which we are currently exploring. Note that L-nets are added greedily. That is, if adding an L-net helps, then we keep it and try another. If the number of hidden variables is large and their interactions are complex and diffuse over the observable variables, it may not be possible for a single or even a small number of L-nets to extract any useful information. In cases like this, we would declare, wrongly, that there are simply no hidden variables. Further, we assume that k, the number of past observations provided to the network, is sufficient for recovering hidden state information. It may be that k is too small, or that the underlying system is just not amenable to this type of use of history. In both cases, we will have difficulty extracting useful information. Finally, it could be the case that an L-net learns a function of multiple hidden variables, leading us to underestimate their number. However, we would claim that in these cases the hidden variable we really care about for the purposes of predictive accuracy is the one that is a combination of more primitive hidden variables. Nonetheless, there are clearly a number of ways in which our approach can underestimate the true number of hidden variables in a system.

This work can be extended by finding other forms of the regularization penalties that will prevent the L-net from "cheating." Furthermore, it would be extremely useful to estimate the regularization weights from data rather than doing so manually. An obvious solution is to keep a portion of the data aside and algorithmically explore the space of weights (perhaps along a log scale) until weights are found for which prediction accuracy is optimized. The main goal of this regularization process is to better optimize our estimates of the hidden values. We will continue to investigate the scalability of the method to several hidden variables in different domains. Our current experiments have shown this method to be successful in domains with up to three hidden variables. There may be issues with computational speed as the number of hidden variables increases.

References

[1] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[2] J. Wnek and R. S. Michalski, "Hypothesis-driven constructive induction in aq17- hci: a method and experiments," *Machine Learning*, vol. 14, no. 2, pp. 139–168, 1994.

[3] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.

[4] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *Advances in Neural Information Processing Systems*, vol. 2, pp. 524–532, 1990.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incom- plete data via the em algorithm," *Journal of the Royal Statistical Society Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[6] N. Friedman, "Learning belief networks in the presence of missing values and hidden variables," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 125–133, Morgan Kaufmann, 1997.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[8] M. L. Littman, R. S. Sutton, and S. Singh, "Predictive representations of state," in *Advances in Neural Information Processing Systems*, vol. 14, pp. 1555–1561, MIT Press, 2002.

[9] R. L. Rivest and R. E. Schapire, "Diversity-based inference of finite automata," *Journal of the ACM*, vol. 41, no. 3, pp. 555–589, 1994.

[10] M. P. Holmes and C. L. Isbell, "Looping suffix tree-based inference of partially observable hidden state," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 409–416, ACM, New York, NY, USA, June 2006.

[11] A. Mccallum, "Instance-based state identification for reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 7, pp. 377–384, MIT Press, 1994.

[12] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models," in *NIPS*, pp. 1441–1448, MIT Press, 2006.
[14] K. A. Bollen, "Latent variables in psychology and the social sciences," *Annual Review of Psychology*, vol. 53, pp. 605–634, 2002.
[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1998.
[16] F. Takens, "Detecting strange attractors in turbulence," *Lecture Notes in Mathematics*, pp. 366–381, 1981.
[17] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '90)*, vol. 3, pp. 21–26, June 1990.
[18] N. Lorenz, "Deterministic non-periodic flows," *Journal of Atmospheric Science*, 1963.
[19] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.
[20] S. Ray and T. Oates, "Discovering and characterizing hidden variables in streaming multivariate time series," in *Proceedings of the 9th International Conference on Machine Learning and Applications (ICMLA '10)*, pp. 913–916, IEEE Computer Society, 2010.
[21] J. Bergstra and Y. Bengio, "Slow, decorrelated features for pretraining complex cell-like networks," in *NIPS*, 2009.
[22] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 617613, 12 pages
doi:10.1155/2011/617613

Research Article

A Cognitive Model for Generalization during Sequential Learning

Ashish Gupta,[1] Lovekesh Vig,[2] and David C. Noelle[3]

[1] *Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235-1826, USA*
[2] *School of Comptational and Integrative Sciences, Jawaharlal Nehru University, New Delhi 110067, India*
[3] *School of Engineering, University of California, Merced, Merced, CA 95343, USA*

Correspondence should be addressed to Lovekesh Vig, lovekeshvigin@gmail.com

Received 31 May 2011; Revised 2 September 2011; Accepted 20 September 2011

Academic Editor: Ivo Bukovsky

Traditional artificial neural network models of learning suffer from *catastrophic interference*. They are commonly trained to perform only one specific task, and, when trained on a new task, they forget the original task completely. It has been shown that the foundational neurocomputational principles embodied by the Leabra cognitive modeling framework, specifically fast lateral inhibition and a local synaptic plasticity model that incorporates both correlational and error-based components, are sufficient to largely overcome this limitation during the sequential learning of multiple motor skills. Evidence has also provided that Leabra is able to generalize the subsequences of motor skills, when doing so is appropriate. In this paper, we provide a detailed analysis of the extent of generalization possible with Leabra during sequential learning of multiple tasks. For comparison, we measure the generalization exhibited by the backpropagation of error learning algorithm. Furthermore, we demonstrate the applicability of sequential learning to a pair of movement tasks using a simulated robotic arm.

1. Introduction

Humans acquire many different skills, behaviors, and memories throughout their lifetime. Lack of use of a particular skill, behavior, or memory results in its slow degradation. It is a common observation that reacquisition of any knowledge is typically rapid as compared to its initial acquisition. This retention of knowledge, often in a latent form, is known as *savings*.

Why do we observe a degradation in performance when a particular piece of knowledge fails to be used? The initial acquisition of task knowledge is driven by synaptic plasticity, shaped by experience. This plasticity continues even when that task knowledge is not regularly used. Thus, experience with other activities could shape the neural circuits in a manner that interferes with the initial knowledge that was acquired.

What could be the neural basis of savings? Savings could emerge due to neural specialization. Some of the neurons employed by an initial task might not be reused when learning a subsequent task. To the degree that the sets of neurons associated with different tasks are disjoint, learning one task will not affect the synapses associated with another. Note, however, that when neurons are shared between tasks savings are still possible. Savings could arise from subthreshold residual synaptic weights associated with the initial task—weights that have been driven down by interfering experiences to below their threshold for neural firing, but not all the way down to their initial values. Finally, tasks may share components or "subtasks." To the degree that such components have isolated neural representations, learning a new task may actually reinforce portions of a previously learned task.

Traditional artificial neural network models of human learning, including those based on the powerful *backpropagation of error* learning algorithm, fail to display adequately robust savings when tasks are learned sequentially. A set of synaptic connection weights serve as the neural network's memory, and any task is learned by modifying these weights. This is the strength of neural networks since they can learn almost any possible input-output mapping. However, this is also the source of problems, as the networks have an inherent tendency to abruptly and completely forget previously learned knowledge when presented with new training inputs. This phenomenon is known as catastrophic interference [1]. This prevents artificial neural networks

from exhibiting savings and, therefore, leads to questions about their biological plausibility.

In our previous work, we proposed that the biological constraints imposed by the structure of cortical circuitry may embody the properties that are necessary to promote savings, as observed during human skill acquisition [2]. Specifically, we examined the neurocomputational principles forming the Leabra cognitive modeling framework [3], and we found that these biologically motivated principles give rise to savings without the need for any auxiliary mechanisms. Our findings suggest that the Leabra implementation of fast acting lateral inhibition acts in concert with its synaptic plasticity mechanism in order to produce adequately sparse representations to support skill savings.

Sparse representations involve patterns of neural firing in which only a small fraction of neurons in a "pool" or "layer" are strongly active at any one time. The use of sparse representations results in different sets of neurons being used for the different tasks that the network has been trained to perform. There is good reason to believe that, when learning multiple tasks, humans are able to generalize the common structure, if any, that exists between them. In our previous work, we provided preliminary evidence that Leabra networks, even with a sparse internal representation enforced by a biologically plausible lateral inhibition mechanism, are able to generalize the common subtasks shared by multiple tasks, when doing so leads to appropriate responses. In this paper, we provide a more detailed analysis of the generalization seen in Leabra networks. We show that the generalization seen in a Leabra network when learning two tasks sequentially is comparable to the generalization seen when the two tasks are learned in an interleaved manner. In comparison, a backpropagation-based artificial neural network not only does not show any generalization, but it also does not show any savings either.

Most neural-network-based controllers require substantial training on a particular task and need to be retrained if the network subsequently learns a different task [4, 5]. A network model that exhibits savings offers potential benefits for such applications. The retraining time required for a previously learnt task is considerably reduced if the network exhibits savings. Such a network would be capable of learning multiple tasks sequentially without the need to interleave these tasks.

A wide variety of sequential key pressing tasks have been used to investigate human motor skill learning [6–8], and a number of interesting findings have resulted. There is a period of rapid improvement in performance during the early stages of training. During this stage, learning is effector independent (e.g., switching hands does not substantially degrade performance). Further, interfering with the frontal systems involved in the controlled pathway during this period seriously disrupts performance. Interfering with the automatic pathway, however, does not affect performance during this early period. In this paper, we demonstrate the applicability of our Leabra network model to learning of movement tasks in a simulated robotic manipulator. The network is able to exhibit savings whilst learning arm movement tasks sequentially, thereby substantially reducing the retraining time required.

This paper is organized as follows. In the next section, we provide some background and an overview of related work. Section 3 provides a description of a Leabra-based task learning model and some simulations of the model. Section 4 describes the results of these simulation experiments. We end the paper with a discussion of the results, as well as some conclusions.

2. Background

2.1. Leabra. The Leabra framework offers a collection of integrated cognitive modeling formalisms that are grounded in known properties of cortical circuits while being sufficiently abstract to support the simulation of behaviors arising from large neural systems. It includes dendritic integration using a point-neuron approximation, a firing rate model of neural coding, bidirectional excitation between cortical regions, fast feedforward and feedback inhibition, and a mechanism for synaptic plasticity [3] (refer to the appendices for a more detailed description). Leabra models have successfully illuminated cognitive function in a wide variety of domains, including perception, object recognition, attention, semantic memory, episodic memory, working memory, skill learning, reinforcement learning, implicit learning, cognitive control, and various aspects of language learning and use. Of particular relevance to skill savings are Leabra lateral inhibition formalism and its synaptic learning rule.

In the neocortex, two general patterns of connectivity have been observed involving inhibitory neurons and their interactions with excitatory neurons, namely, *feedforward* and *feedback* inhibition [3]. Feedforward inhibition occurs when the inhibitory interneurons in a cortical region are driven directly by the inputs to that region, producing rapid inhibition of the excitatory neurons in that area. Feedback inhibition occurs when the same neurons that excite nearby inhibitory interneurons are, in turn, inhibited by the cells they excite, producing a kind of negative feedback loop.

The effects of inhibitory interneurons tend to be strong and fast in the cortex. This allows inhibition to act in a regulatory role, mediating the positive feedback of bidirectional excitatory connections between brain regions. Simulation studies have shown that a combination of fast feedforward and feedback inhibition can produce a kind of "set-point dynamics," where the mean firing rate of cells in a given region remains relatively constant in the face of moderate changes to the mean strength of inputs. As inputs become stronger, they drive inhibitory interneurons as well as excitatory pyramidal cells, producing a dynamic balance between excitation and inhibition. Leabra implements this dynamic using a *k-Winners-Take-All (kWTA)* inhibition function that quickly modulates the amount of pooled inhibition presented to a layer of simulated cortical neural units, based on the layer level of input activity. This results in a roughly constant number of units surpassing their firing threshold. The amount of lateral inhibition within a layer

can be parameterized in a number of ways, with the most common being the percentage of the units in the layer that are expected, on average, to surpass threshold. A layer of neural units with a small value of this k parameter (e.g., 10–25%) will produce sparse representations, with only a small fraction of the units being active at once.

With regard to learning, Leabra modifies the strength of synaptic connections in two primary ways. An error-correction learning algorithm changes synaptic weights so as to improve network task performance. Unlike the backpropagation of error algorithm, Leabra error-correction scheme does not require the biologically implausible communication of error information backward across synapses. In addition to this error-correction mechanism, Leabra also incorporates a Hebbian correlational learning rule. This means that synaptic weights will continue to change even when task performance is essentially perfect. This form of correlational learning allows Leabra to capture certain effects of overlearning.

2.2. Catastrophic Interference. Many past studies have shown that artificial neural networks suffer from a kind of catastrophic interference that is uncharacteristic of human performance. The seminal example of catastrophic interference is the experiment performed by McCloskey and Cohen [1], in which the authors tried to employ a standard backpropagation network to perform the AB-AC list-learning task. In this task, the network begins by learning a set of paired associates (A-B) consisting of a nonword and a real word (e.g., "pruth-heavy"). Once, this learning was completed, they trained the network to associate a new real word with each of the original nonwords (A-C). The authors found that as soon as the training on the AC list started, the network completely forgot the AB list.

Since the original observation of catastrophic interference in artificial neural networks, a number of computational mechanisms have been proposed to overcome it. Most of these involve segregating the neural units associated with different skills in order to avoid the damage caused by "reuse" of synaptic weights [9]. For example, forcing layers of neural units to form sparse representations reduces the probability that a given unit will be active while performing a given skill, thereby reducing the probability of interference when learning multiple skills in sequence. Leabra offers a biologically justified mechanism for producing sparse representations. With a low k parameter, Leabra kWTA lateral inhibition implementation limits the overlap between the neural representations used for different tasks. This has been shown to improve performance on the AB-AC list learning task [3].

One extreme form of segregation between neurons devoted to different tasks involves isolating them into discrete modules. Modular artificial neural network architectures have been proposed in which differences between tasks are explicitly detected during learning, and a "fresh" module of neural units is engaged to learn the task, protecting previously trained modules from interference [10, 11]. Importantly, overlearning of a task can strengthen its consolidation in a module, increasing resistance to interference, as is observed in humans [12, 13]. While such modular models can exhibit robust savings (and appropriately limited forms of interference), the biological plausibility of a reserve of untrained neural modules awaiting assignment when a new task is to be learned is questionable. Even if we assume the existence of unused modules, questions still remain about their granularity—do we need a different module even for different variants of the same task? It also poses a question concerning the total number of available modules. Some modular networks do show limited forms of generalization through combining the outputs from multiple modules [11], but they still need to use a fresh module for most cases [14].

Modular approaches of this kind should be distinguished from the hypothesis that the hippocampus and the neocortex form distinct learning systems [15]. This hypothesis asserts that catastrophic interference is alleviated through the use of a fast hippocampal learning system that uses sparse representations. While neocortical systems are assumed to use less sparse representations, making them more vulnerable to interference, problems are avoided through a hippocampally mediated process of consolidation, where neocortical networks receive interleaved "virtual" practice in multiple skills. Through the mediation of the hippocampus, multiple skills continue to be essentially learned "together," rather than sequentially, one after the other.

One successful computational learning strategy that is similar in nature to hippocampally mediated consolidation involves the use of "pseudopatterns" [16]. Trained artificial neural network models are used to generate virtual training experiences—pseudopatterns—which are similar to previously learned patterns. These pseudopatterns are mixed with the new patterns, corresponding to new learning experiences, and the network is given interleaved training on all of these patterns. It has been observed that the use of pseudopatterns alleviates the problem of catastrophic interference substantially [17, 18]. However, the biological mechanisms giving rise to the generation and use of these patterns have yet to be fully explicated. It is also difficult to imagine the generation of pseudopatterns for the maintenance of all kinds of knowledge. This is particularly true if hippocampus is believed to be the sole site for the maintenance of mixed training pattern sets. All kinds of knowledge and skill acquisition do not depend upon the hippocampus. For example, there is evidence that humans can continue to learn new motor skills even after complete removal of the hippocampus [19].

There are many other factors that contribute to savings [2, 20]. Contextual cues help in disambiguating between different tasks and could also lead to the use of different sets of neurons for their representation. Overlearning a particular task results in a sharpening of its representation, which is more resistant to perturbation. Finally, it is possible that synaptic changes during the learning of an interfering task may drive certain neurons associated with a previously learned task below their firing threshold—but just below that threshold, allowing them to recover quickly once practice of the previous task is resumed.

We have shown, in previous work, that the sparse representations enforced by Leabra lateral inhibition mechanism,

in conjunction with its synaptic learning rule, causes Leabra simulations of cortical circuits to escape the most dire pitfalls of catastrophic interference when those circuits are required to sequentially learn multiple temporally extended motor trajectories [2].

2.3. Generalization. The use of sparse representation results in the use of different sets of neurons for different tasks. This implies that the network learns the tasks in a highly specialized manner. Hence, the network may not be able to generalize the common substructure existing between different tasks. There is good reason to believe that humans are able to make such generalizations, even if the tasks are learned sequentially. Also, artificial neural network models of human learning show generalization when multiple tasks are learned in an interleaved manner [21]. Such generalization also emerges from networks that generate pseudopatterns of previously learned tasks [17, 18].

Is it possible for a network with a sparse representation scheme to still exhibit generalization when tasks are learned sequentially? We know that the use of a small value for the k parameter for internal hidden layers creates the possibility of using different units for different tasks. A contextual cue layer increases this probability by providing inputs to distinct hidden layer neurons for the different tasks. Contextual cues are biologically justified, and they improve savings significantly. However, if the cue signal is too strong, it enforces a separate representation even for the common subtask, thus hindering generalization. We found that the strength of the cues can be set to an optimum value, such that the network continues to show significant savings by enforcing orthogonal representation [22], while still allowing the use of the same neurons for the common subtask, thus enabling generalization as well.

2.4. Neural Network Controllers. The neural-network-based control technique has been widely used to solve problems commonly encountered in control engineering. The most useful property of neural networks in control is their ability to approximate arbitrary linear or nonlinear mappings through learning. It is because of the above property that many neural-network-based controllers have been developed for the compensation of the effects of nonlinearities and system uncertainties in control systems so that the system performance such as the stability and robustness can be improved.

There has been considerable research towards the development of intelligent or self-learning neural-based controller architectures in robotics.Riedmiller and Janusz [23] proposed a neural self-learning controller architecture based on the method of asynchronous dynamic programming [24]. The capabilities of the controller are shown on two typical sequential decision problems. Johnson et al. [25] were the first to utilize a backpropagation network to train a robotic arm. The backprop was able to develop the inverse kinematics relationship for the arm after being trained on the arm data. Hesselroth et al. [26] employ a neural map algorithm to control a five-joint pneumatic robot arm and gripper through feedback from two video cameras. The pneumatically driven robot arm (SoftArm) employed in this investigation shares essential mechanical characteristics with skeletal muscle systems. To control the position of the arm, 200 neurons formed a network representing the three-dimensional workspace embedded in a four-dimensional system of coordinates from the two cameras and learned a three-dimensional set of pressures corresponding to the end-effector positions.

Bouganis and Shanahan [27] present a spiking neural network architecture that autonomously learns to control a 4-degrees-of-freedom robotic arm after an initial period of motor babbling. Its aim is to provide the joint commands that will move the end-effector in a desired spatial direction, given the joint configuration of the arm. Vaezi and Nekouie [28] utilize a new neural networks and time series prediction-based method to control the complex nonlinear multivariable robotic arm motion system in 3D environment without engaging the complicated and voluminous dynamic equations of robotic arms in the controller design stage.

2.5. Sequential Learning. The implications and advantages of sequential training of partially unconnected tasks on a given network architecture remain mostly unexplored, unclear or have even been seen negatively in form, for example, catastrophic interference [20, 29]. In robotics and machine learning, though, a number of attempts have been made in analysing enhanced learning through a sequence of training environments [30–35]. Saal et al. [36] consider the problem of tactile discrimination, with the goal of estimating an underlying state parameter in a sequential setting. They present a framework that uses active learning to help with the sequential gathering of data samples, using information theoretic criteria to find optimal actions at each time step. Kruger explores the effects of sequential training in the form of shaping in the cognitive domain. He considers abstract, yet neurally inspired, learning models and proposes extensions and requirements to ensure that shaping is beneficial using a long term memory model. While this model can learn sequentially and mitigate catastrophic interference, it is reliant on an explicit memory model whereas our model avoids catastrophic interference by simulating the biological processes in real neurons such as lateral inhibition. Kruger [37] explores the effects of sequential training in the form of shaping in the cognitive domain. He considers abstract, yet neurally inspired, learning models and proposes extensions and requirements to confirm that shaping is beneficial for sequential learning. However, to the best of our knowledge no system has yet been designed to explicitly utilize the phenomenon of savings in sequential environments by utilizing generalization of neural models in a cognitive framework.

2.6. RoboSim Robotic Arm Simulator. The RoboSim simulated robotic manipulator [38] was used to demonstrate the applicability of our approach to learning of movement sequences. RoboSim is a simulation system for a 6-degrees-of-freedom robot manipulator. For our experiments we used only three joints and kept the other 3 fixed to result in

Table 1: Similarities between various tasks as compared to Base Task.

Task	Similarity
Nothing Same Task	Nothing in common with Base Task
5 Patterns Same Task	5 of the 18 patterns same as in Base Task
10 Patterns Same Task	10 of the 18 patterns same as in Base Task

a manipulator with 3 degrees of freedom. The manipulator may be moved either in joint coordinates or in cartesian coordinates. RoboSim allows joint weights to be altered, in order to favor movement of individual joints versus others. The simulation system includes both forward and inverse manipulator kinematics. The system also allows the user to teach the manipulator a sequence of consecutive movements.

3. Method

3.1. Simulated Tasks with the Leabra Network. Four different tasks were used in our simulations. Each task contained 18 different input-output pairs, with the input and output vectors generated randomly. While the output vectors for each task differed, the input vectors were kept identical for all tasks. This resulted in different input-output mappings being learnt by the network for different tasks.

After training on the Base Task, the network was trained on one of the interfering tasks. Table 1 briefly describes the similarity between the Base Task and the interfering tasks. After training on the interfering task, retained performance on the Base Task was assessed. Each of the input and output patterns was encoded in the Leabra network over a pool of 36 neural units.

3.2. The Network. Figure 1 shows the Leabra network used in our simulations. At each step, the network was provided with a 36-unit input vector, encoding one of the 18 patterns comprising a task. Complete interconnections from this input layer to a 100-unit hidden layer produced an internal representation for the current pattern, with the sparsity of this representation controlled by lateral inhibition within the hidden layer (i.e., by its k parameter). Complete bidirectional excitatory connections map this internal representation to an output layer that is intended to encode the output pattern. During training, the output layer was also provided with a target signal, indicating the correct output. The context layer contained two units, each corresponding to one of the two learned tasks, indicating which of the two tasks was to be produced by the network. This layer was connected to the hidden layer with an 80% chance of interconnection between a given context layer unit and any given hidden layer unit. This random interconnection structure was expected to increase the chances of orthogonal representations for different tasks.

Most of the network parameters used in the simulations were Leabra default values. Connection weights between units were randomly initialized with a mean of 0.5 and a variance of 0.25. The minimum weight value for any interconnection was 0 and the maximum was 1. The activation of any node could range between 0 and 1. Leabra uses a mixture of the GeneRec learning algorithm and a form of Hebbian learning. GeneRec is an error-driven learning algorithm [3]. Hebbian learning was strengthened in our simulations, as compared to the Leabra default, contributing to 1% of synaptic weight changes rather than the default 0.1%. The learning rate for training was set to 0.01. Error tolerance was set to 0.25. This means that any output unit could be within ± 0.25 of its desired activity without prompting error-correction learning. In order to facilitate a sparse internal representation, a value of $k = 10$ was used for the hidden layer. For comparison, a backpropagation of error network (BP), matching the Leabra network in structure and size, was also examined.

There are two common measures of savings: *exact recognition* and *relearning* [9]. The exact recognition measure assesses the percentage of the original task that the network performs correctly after it has learned a second task. The relearning measure examines how long it takes the network to relearn the original task. The two measures are usually correlated. We used an exact recognition measure to assess savings. In particular, we measured the sum-squared error (SSE) of the network output on the first task after training on the second task. In order to contrast this SSE value with "complete forgetting" of the first task, the SSE was also recorded prior to the first task training, and we report the ratio of SSE after interference training to SSE of the untrained network. A value of one or more for this ratio indicates complete forgetting of the initial task, while lower values indicate savings.

To measure the similarity of the hidden layer representation for the common subtask between the two tasks, two different measures were used. First, we computed the root-mean-square (RMS) value of the difference between the hidden layer activities for the common subtask using the following formula:

$$\mathrm{RMS} = \sqrt{\frac{\sum_{i=1}^{N} \left(A_i^1 - A_i^2\right)^2}{N}}. \tag{1}$$

Here, A_i^1 is the activity of the ith hidden layer unit for the common subtask after the network has been trained on the Base Task. Similarly, A_i^2 is the activity of the ith hidden layer unit for the common subtask after the network has been trained on the interfering task. N is the total number of units in the layer. The minimum RMS value is 0. This would occur if the hidden layer activity is exactly the same for the two tasks. Since the activity of the units is restricted to the range between 0 and 1, the maximum RMS value is 1. This would occur if the hidden layer activity is completely orthogonal and every unit is firing at its maximum possible activation value for one of the tasks and is not firing at all for the other task.

Second, we measured the percentage of total active units (at least 0.05 activation) that were common for the common subtask at the two relevant points in training. This percentage is a good measure of similarity in representation for the

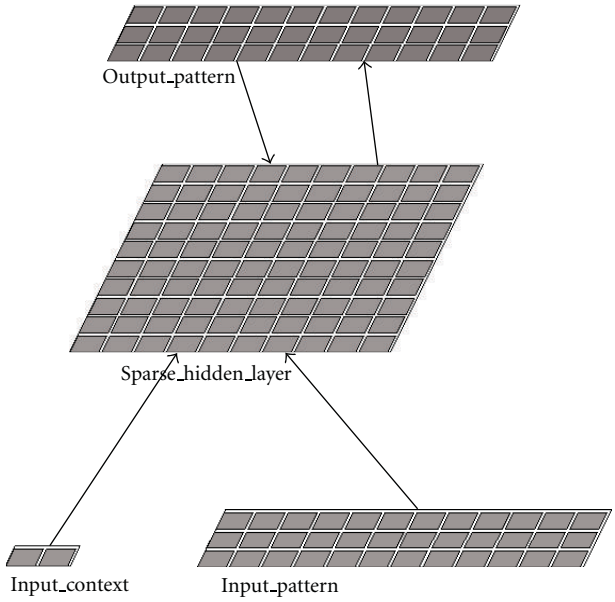


Figure 1: The Leabra network.

Leabra network, since roughly only 10 units are active at any time (since $k = 10$ for the hidden layer). However, this measure is not appropriate for the BP network, since BP uses almost all the hidden units to represent a task.

We repeated each experiment five times in order to deal with stochastic variations (arising from such factors as random initial weights and the random order of pattern presentation) in our simulations. We report the average of these repetitions.

4. Results

4.1. Generalization. A randomly initialized network was trained on the Base Task. Then, the network was trained on an interfering task. Using the "Nothing Same Task" as the second task; the variation of the SSE ratio as a function of the hidden layer kWTA parameter is shown in Figure 2.

We found a regular decrease in the error ratio with decrease in k. The reason for this decrease is the decrease in overlap between the Base Task and Nothing Same Task hidden layer activation patterns as representations became sparser. We measured the percentage of total active units common for the two tasks. This percentage was large for dense representation and decreased considerably for sparse representations. Thus, increasing lateral inhibition produced more distinct internal representations between the tasks and resulted in improved savings. The network showed complete forgetting (SSE ratio greater than 1) of the base task for k greater than 40. This suggests that the savings exhibited by the Leabra network are due to the kWTA mechanism and not due to its recurrent architecture. It was found that the BP network exhibited no savings at all. This was as expected since there is no explicit mechanism to facilitate nonoverlapping hidden layer representation in the BP network.

Table 2: Mean and standard error of SSE ratios for Base Task, when different interfering tasks were used. The first row gives the ratio for a BP network, the second row for a Leabra network with contextual cues of strength 1.0, the third row for a Leabra network with contextual cues of strength 0.35, and the fourth row for a Leabra network trained in an interleaved manner for the two tasks. A smaller ratio signifies more savings.

Network	5 Patterns Same	10 Patterns Same
BP	1.269 (± 0.0242)	0.841 (± 0.0425)
Context = 1.0	0.196 (± 0.0606)	0.144 (± 0.0457)
Context = 0.35	0.147 (± 0.0350)	0.108 (± 0.0168)
Interleaved	0.000 (± 0.0000)	0.000 (± 0.0000)

Next, we fixed $k = 10$ for the hidden layer and performed similar experiments with all of the other interfering tasks. Table 2 shows the SSE ratio for the various cases. It was found that the SSE ratio was close to 1 for the BP network. This suggests that the BP network consistently experienced catastrophic interference. On the other extreme was the Leabra network that was given interleaved training for the two tasks. In this case, the network learned both of the tasks completely. The Leabra network with contextual cues (of activation magnitude 1.0 and 0.35) that was given sequential training on the two tasks displayed significant savings, represented by a very small SSE ratio.

To test if the networks were able to come up with a generalized representation for the common subtask, we compared the hidden layer activity for the common subtask between the learning of the two tasks. Table 3 shows the percentage of active units that were common across the two tasks. As explained before, this percentage has been computed only for the Leabra networks.

We found that the Leabra network with contextual cues of strength 0.35 shows generalization comparable to the Leabra network that is trained on the two tasks in an interleaved manner. On the other hand, the Leabra network with contextual cues of strength 1.0 shows very little generalization. For comparison, we also measured the percentage of common active units when the "Nothing Same Task" was used as the interfering task and the strength of contextual cues was 1.0. We found that this percentage was zero, signifying completely orthogonal internal representations.

To compare the results of generalization seen in the Leabra networks with that seen in the BP network, we computed the root-mean-square (RMS) difference in the hidden layer activity for the common subtask between the two tasks. Table 4 shows the RMS values for the different cases. Once again, it was observed that the RMS difference is comparable for the Leabra network with contextual cues of strength 0.35 and the Leabra network that was given interleaved training. Also, the RMS value for the Leabra network with contextual cues of strength 1.0 is comparable to that of a BP network. For comparison, we also measured the RMS value when the "Nothing Same Task" was used as the interfering task. The RMS value for the BP network was 0.3279 ± 0.0098, and for the Leabra network it was 0.3517 ± 0.0027.

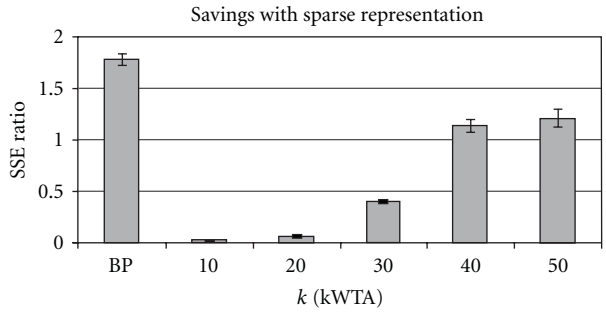


Figure 2: Savings as a function of sparsity. An SSE ratio of one or more indicates no savings, while lower values indicate retention of Base Task knowledge. Nothing Same Task was used as the interfering task. The k parameter roughly equals the percentage of active units in the hidden layer. Error bars display standard errors of the mean.

Table 3: Mean and standard error of percentage of common units for the common subtask between Base Task and the interfering task. The first row gives the ratio for a Leabra network with contextual cues of strength 1.0, the second row for a Leabra network with contextual cues of strength 0.35, and the third row for a Leabra network trained in an interleaved manner for the two tasks.

Network	5 Patterns Same	10 Patterns Same
Context = 1.0	12.748 (±1.695) %	12.724 (±1.975) %
Context = 0.35	71.246 (±2.036) %	66.702 (±2.750) %
Interleaved	68.882 (±1.785) %	67.340 (±2.300) %

Table 4: Mean and standard error of RMS difference in the hidden layer activation for the common subtask between Base Task and the interfering task. The first row gives the RMS value for a BP network, the second row for a Leabra network with contextual cues of strength 1.0, the third row for a Leabra network with contextual cues of strength 0.35, and the fourth row for a Leabra network trained in an interleaved manner for the two tasks.

Network	5 Patterns Same	10 Patterns Same
BP	0.324 (±0.0080)	0.381 (±0.0067)
Context = 1.0	0.299 (±0.0044)	0.329 (±0.0060)
Context = 0.35	0.159 (±0.0060)	0.147 (±0.0052)
Interleaved	0.156 (±0.0053)	0.156 (±0.0083)

4.2. Savings. To uncover the degree to which our model exhibits savings, we conducted simulations to record how the network changes during training and extinction. Animals are faster to reacquire an extinguished behavior, as compared to initial acquisition, and they are faster to extinguish a reacquired behavior, as compared to initial extinction [39–44]. A randomly initialized network was trained to respond upon the presentation of a pattern. Once this training reached criterion, the network was trained to not respond upon the presentation of the pattern. This process was repeated 5 times. Figure 3 shows the number of trials required for successive acquisition and extinction trainings. Note that the required time quickly decreases. The model predicts that the required number of trials will asymptote to a small value after just a few acquisition-extinction iterations.

The network starts with small initial synaptic weights. Hence, a large change in weights is required for success during the first acquisition training session. During the first extinction training session, the weights to the acquisition neurons start decreasing and the weights to the extinction neurons start increasing. As soon as the extinction neurons win the inhibitory competition, the acquisition neurons tend to fall below their firing threshold. At this stage, the weights to the acquisition neurons stop decreasing, as these neurons are no longer contributing to erroneous outputs. Hence, a significant amount of acquisition-related association strength is retained through the extinction process. During the reacquisition training, the weights to the acquisition neurons increase once again and the weights to the extinction neurons decrease. Once again, the weights stop changing as soon as the extinction neurons lose the inhibitory competition. Hence, most of extinction-related plasticity is retained through the acquisition process. In this manner, subsequent acquisition and extinction trainings require a very small change in weights (Figure 4). Effectively, acquisition and extinction associations are maintained side by side in the network, allowing for the rapid switching between them based on recent training feedback.

4.2.1. Savings in Robotic Arm Movements. We have used our model to simulate the learning of arm movement sequences. Our model controls a simulated 3-joint planar arm which moves over a 3-dimensional space, as shown in Figure 5. The state of the arm at any point in time is represented by the vector (q1, q2, q3), where q1, q2, and q3 are the three joint angles. The joint angles range between -180° and 180°. Two unrelated trajectories are taught to the manipulator with each trajectory represented as a sequence of arm states over the successive time steps. During training, the arm is essentially guided along the desired trajectory, with differences between the motor output of the arm controller and the configuration of the arm, as specified by the guide, acting as a measure of error to drive synaptic weight change.

Figure 6 shows the Leabra network used for our simulations. The Sensory_Input layer provides the current state of the arm as input to the network and the Motor_Output layer is to produce the desired arm state for the next time step. Each joint angle is encoded over a pool of 36 neural units. Each of the 36 units has a preferred angle, ranging from -180° to 180°, in 10° increments. To encode a given joint angle, the closest unit with regard to preference, as well as its two neighbors, is set to its maximal firing rates. Similarly, patterns of activity over each row of 36 units in the Motor_Output are decoded by identifying the preferred angle of the unit in the middle of the three adjacent units that are all active. Other patterns of activity in the Motor_Output layer are considered to be ill-formed. With each joint angle encoded over 36 units in this way, the complete arm configuration is encoded over 108 units. The context layer was used to encode contextual information related to the tasks.

The tasks consist of 20 consecutive arm positions that the manipulator was trained to achieve in order. After the manipulator had learnt the movements associated with

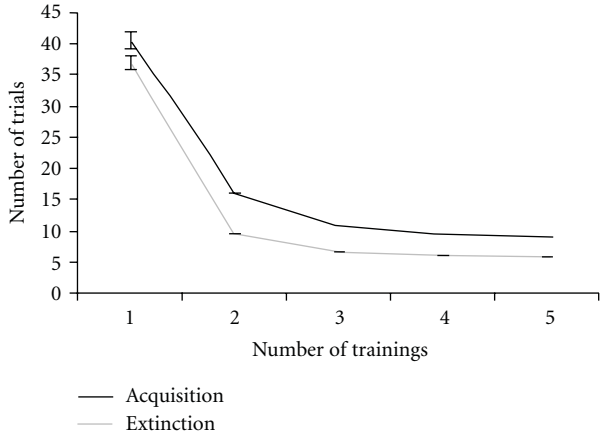


Figure 3: The number of training trials required to reach criterion (y-axis) decreases as the number of prior acquisition and extinction training sessions (x-axis) increases. Error bars report standard errors of the mean.

Table 5: SSE ratio and retraining epochs.

Task	SSE ratio	Training epochs	Retraining epochs
Task 1	0.522 (±0.0033)	10.8 (±0.24)	4.2 (±0.20)
Task 2	0.467 (±0.0023)	12.2 (±0.24)	4.2 (±0.20)
Task 3	0.413 (±0.0097)	15.4 (±0.24)	2.4 (±0.48)
Task 4	0.328 (±0.0049)	13.8 (±0.24)	2.4 (±0.48)
Task 5	—	13.6 (±0.24)	—

the first task, the network was subsequently trained on the second task. The tasks were generated to ensure that there were no common patterns between the tasks. The manipulator was then tested and retrained on the first task. The manipulator was able to accurately remember the training for 15 out of 20 movements for the first tasks with an SSE ratio of 0.234. More importantly, the training time for relearning the first time was just 2 epochs as opposed to 15 epochs for the untrained network.

To further investigate the applications of savings, the manipulator network was sequentially trained on five manipulator tasks (task 1 to task 5). The SSE ratio for all the tasks was shown in Table 5. As expected, the SSE ratio is the lowest for the most recently trained tasks, indicating greater savings for these tasks. More importantly, the retraining time for all tasks was significantly less than the original training time. Figure 7 shows the variation in savings for successively trained tasks with different values of hidden layer kWTA. As the kWTA parameter is increased, the network is able to retain less information about previously trained tasks.

5. Conclusion and Discussion

We have shown that the neurocomputational principles embodied by the Leabra modeling framework are sufficient to produce generalization, whilst exhibiting significant savings during the sequential learning of multiple tasks. No auxiliary computational mechanisms are needed. Generalization

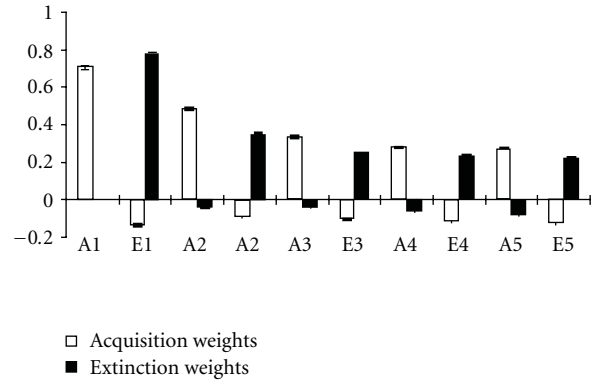


Figure 4: This graph plots the change in the summed connection weights in the acquisition pathway and in the extinction pathway (y-axis) during the successive acquisition and extinction trainings (x-axis). The change in weights decreases in both the pathways as the number of prior acquisitions and extinctions training sessions increases. There seems to be a slow upward going trend in the weights in both the pathways, which appears to be a quirk of the simulator.

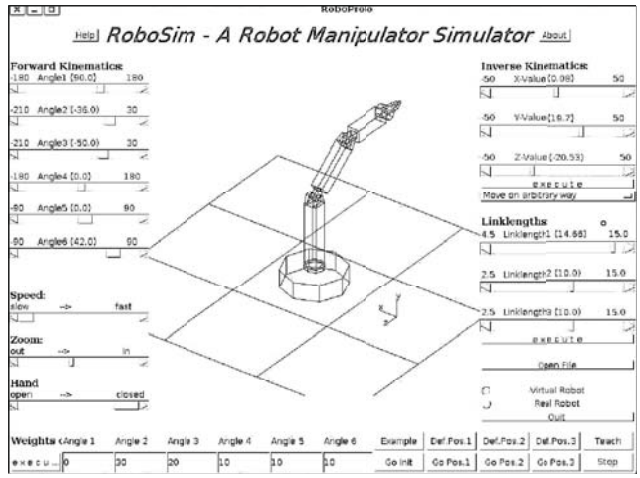


Figure 5: The RoboSim Robotic Arm Simulator. The simulator allows for both forward and inverse kinematics, learning of a sequence of movements, varying degrees of freedom and link lengths, and speed control.

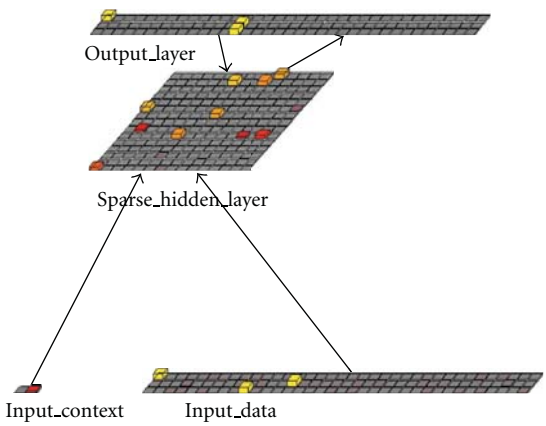


Figure 6: The Leabra network for robot manipulator simulations.

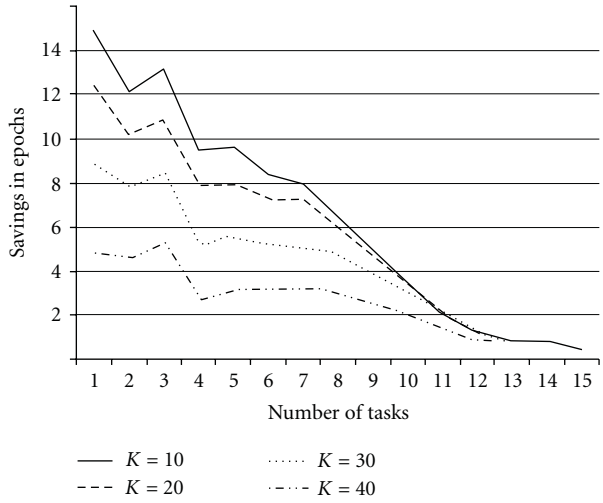


Figure 7: Savings in successive tasks.

is found to be sensitive to the strength of the contextual cues input. Certain neurons develop strong connection weights for the common subtask during the base task training. Due to strong weights, these neurons receive stronger input for the common subtask even during the interfering task training. Hence, they have a greater chance of firing. Contextual cues, however, provide strong inputs to certain other neurons in the hidden layer that compete with the shared neurons. If the cue signal is too strong, these other neurons win the competition, and the network fails to generalize the common subtask. We have shown that the cue strength can be set to an optimum value such that cues are strong enough to enforce savings through the generation of orthogonal representations, while still allowing the emergence of similar representation for the common subtasks between different tasks. The generalization observed during sequential learning of the tasks is comparable to the generalization observed when the two tasks are learned in an interleaved manner.

While our previous work [2] demonstrated how to overcome catastrophic interference via generalization on a Leabra Network, the current paper demonstrates the application of savings to a simulated robotic arm. We have shown that the application of biologically plausible savings during the sequential learning of robotic manipulator tasks has the additional benefit of retention of previously learnt tasks. We demonstrated that with a minimal amount of retraining the manipulator is able to perform the original task accurately. This highlights the advantages of designing a network which is capable of learning multiple tasks in sequence. It saves the user the trouble of having to interleave the tasks or generating artificial pseudopatterns. While human motor skill learning is considerably more nuanced, this framework offers insight into the rapid reacquisition after extinction exhibited by subjects during conditioning experiments. Many more applications of such networks may exist.

Even though biological plausibility has been the focus of our work, research in this direction is also significant for many engineering domains that use artificial neural networks. In many real-world scenarios, the full range of training data is not available up front. In such cases, traditional neural networks need to be updated as new information arrives. The techniques described in this paper can be used to improve the efficiency of such systems by enabling the networks to learn tasks in a sequential manner.

Appendices

A. Dendritic Integration

A fundamental function of neurons is the transformation of incoming synaptic information into specific patterns of action potential output. An important component of this transformation is synaptic integration, the combination of voltage deflections produced by a myriad of synaptic inputs into a singular change in membrane potential. Leabra simulates this integration at the dendrite of the neuron via a weighted summation of all the input activations followed by functional transformation (normally sigmoidal) of the sum.

B. Point Neuron Approximation

Leabra uses a point neuron activation function that models the electrophysiological properties of real neurons, while simplifying their geometry to a single point. This function is nearly as simple computationally as the standard sigmoidal activation function, but the more biologically based implementation makes it considerably easier to model inhibitory competition, as described below. Further, using this function enables cognitive models to be more easily related to more physiologically detailed simulations, thereby facilitating bridge-building between biology and cognition.

C. Lateral Inhibition

The processes involved in lateral inhibition are particularly relevant to the model presented in this paper. Lateral inhibition allows for competition between neurons involved in the encoding of stimuli. Along with the mechanisms of synaptic learning, this competition separates the neurons that associate the stimulus with responding, or acquisition neurons, from those which associate the stimulus with nonresponding, called extinction neurons. The class of in-hibitory functions that Leabra adopts are known as k-winners-take-all (kWTA) functions. A kWTA function ensures that no more than k units out of a total of n in a layer are active at any given point in time. This is attractive from a biological perspective because it captures the *setpoint* property of inhibitory interneurons, where the activity level is maintained through negative feedback at a roughly constant level (i.e., k).

D. kWTA Function Implementation

The k active units in a kWTA function are the ones receiving the most excitatory input (g_e). Each unit in the layer computes a layer-wide level of inhibitory conductance (g_i) while updating its membrane potential such that the top k units

will have above threshold equilibrium membrane potentials with that value of g_i, while the rest will remain below firing threshold. The function computes the amount of inhibitory current g_i^θ that would put a unit just at threshold given its present level of excitatory input, where θ is the threshold membrane potential value. Computing inhibitory conductance at the threshold (g_i^θ) yields

$$g_i^\theta = \frac{g_e^* \bar{g_e}(E_e - \theta) + g_l \bar{g_l}(E_l - \theta)}{\theta - E_i}, \tag{D.1}$$

where g_e^* represents the excitatory input minus the contribution from the bias weight and $g_l \bar{g_l}, g_e \bar{g_e}$ are the total conductances from the potassium and sodium channels, respectively. E_l and E_e are the equilibrium potentials for the the potassium and sodium channels, respectively [3]. g_i is computed as an intermediate value between the g_i^θ values for the kth and $k+1$th units as sorted by level of excitatory conductance (g_e). This ensures that the $k+1$th unit remains below threshold, while the kth unit is above it. Expressed as a formula this is given by

$$g_i = g_{k+1}^\theta + q\left(g_i^\theta(k) - g_i^\theta(k+1)\right), \tag{D.2}$$

where $0 < q < 1$ determines where the inhibition lies between the k and $k+1$th units.

E. Leabra Learning Algorithms

Leabra provides for a balance between Hebbian and error-driven learning. Hebbian learning is performed using a conditional principal component analysis (CPCA) algorithm. Error-driven learning is performed using GeneRec, which is a generalization of the recirculation algorithm and approximates Almeida-Pineda recurrent backpropagation.

F. Hebbian Learning

The objective of the CPCA learning rule is to modify the weights for a given input unit (x_i) to represent the conditional probability that the input unit (x_i) is active when the corresponding receiving unit (y_j) is also active. This is expressed as

$$w_{ij} = P(x_i = 1 \mid y_i = 1) = P\left(x_i \mid y_j\right). \tag{F.1}$$

In (F.1) the weights reflect the frequency with which a given input is active across the subset of input patterns, represented by the receiving unit. If an input pattern occurs frequently with such inputs, then the resulting weights from it will be relatively large. On the other hand if the input pattern occurs rarely across such input patterns, then the resulting weights will be small. The following weight update rule achieves the CPCA conditional probability objective represented by (F.1)

$$\Delta w_{ij} = \epsilon\left[y_j x_i - y_j w_{ij}\right] = \epsilon y_j\left(x_i - w_{ij}\right), \tag{F.2}$$

where ϵ is the learning rate parameter. The weights are adjusted to match the value of the sending unit activation x_i, weighted in proportion to the activation of the receiving unit (y_j). Thus inactivity of the receiving unit implies that no weight modification will occur. Conversely, if the receiving unit is very active (near 1), the update rule modifies the weight to match the input unit's activation. The weight will eventually come to approximate the expected value of the sending unit when the receiver is active (consistent with (F.1)).

G. Error-Driven Learning

GeneRec implements error backpropagation using locally available activation variables thereby making such a learning rule biologically plausible. The algorithm incorporates the notion of plus and minus activation phases. In the *minus phase*, the outputs of the network represent the expectation or response of the network, as a function of the standard activation settling process in response to a given input pattern. Then, in the *plus phase*, the environment is responsible for providing the outcome or target output activations.

The learning rule for all units in the network is given by

$$\Delta w_{ij} = \epsilon\left(y_j^+ - y_j^-\right)x_i^- \tag{G.1}$$

for a receiving unit with activation y_i and sending unit with activation x_i. The rule for adjusting the bias weights is just the same as for the regular weights, but with the sending unit activation set to 1:

$$\Delta \beta_{ij} = \epsilon\left(y_j^+ - y_j^-\right). \tag{G.2}$$

The difference between the two phases of activation is an indication of the unit contribution to the overall error signal. Bidirectional connectivity allows output error to be communicated to a hidden unit in terms of the difference in its activation states during the plus and minus states. $(y_j^+ - y_j^-)$.

Acknowledgments

The authors would like to thank three anonymous reviewers for their valuable feedback which helped to significantly improve the quality of this paper.

References

[1] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: the sequential learning problem," in *Psychology of Learning and Motivation*, G. H. Bower, Ed., vol. 24, pp. 109–164, Academic Press, New York, NY, USA, 1989.

[2] A. Gupta and D. C. Noelle, "The role of neurocomputational principles in skill savings," in *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pp. 863–868, 2005.

[3] R. C. O'Reilly and Y. Munakata, *Computational Explorations in Cognitive Neuroscience*, MIT Press, 2000.

[4] Y. Zhao and C. C. Cheah, "Position and force control of robot manipulators using neural networks," in *Prroceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, pp. 300–305, December 2004.

[5] M. J. Er and Y. Gao, "Robust adaptive control of robot manipulators using generalized fuzzy neural networks," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 3, pp. 620–628, 2003.

[6] R. S. Bapi, K. Doya, and A. M. Harner, "Evidence for effector independent and dependent representations and their differential time course of acquisition during motor sequence learning," *Experimental Brain Research*, vol. 132, no. 2, pp. 149–162, 2000.

[7] O. Hikosaka, K. Nakamura, K. Sakai, and H. Nakahara, "Central mechanisms of motor skill learning," *Current Opinion in Neurobiology*, vol. 12, no. 2, pp. 217–222, 2002.

[8] M. K. Rand, O. Hikosaka, S. Miyachi et al., "Characteristics of sequential movements during early learning period in monkeys," *Experimental Brain Research*, vol. 131, no. 3, pp. 293–304, 2000.

[9] R. M. French, Catastrophic interference in connectionist networks, Macmillan Encyclopedia of the Cognitive Sciences, 2011.

[10] T. Brashers-Krug, R. Shadmehr, and E. Todorov, "Catastrophic interference in human motor learning," *Advances in Neural Information Processing Systems*, vol. 7, pp. 19–26, 1995.

[11] M. Haruno, D. M. Wolpert, and M. Kawato, "MOSAIC model for sensorimotor learning and control," *Neural Computation*, vol. 13, no. 10, pp. 2201–2220, 2001.

[12] T. Brashers-Krug, R. Shadmehr, and E. Bizzi, "Consolidation in human motor memory," *Nature*, vol. 382, no. 6588, pp. 252–255, 1996.

[13] R. Shadmehr and H. H. Holcomb, "Neural correlates of motor memory consolidation," *Science*, vol. 277, no. 5327, pp. 821–825, 1997.

[14] C. Miall, "Modular motor learning," *Trends in Cognitive Sciences*, vol. 6, no. 1, pp. 1–3, 2002.

[15] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory," *Psychological Review*, vol. 102, no. 3, pp. 419–457, 1995.

[16] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, pp. 123–146, 1995.

[17] B. Ans, "Sequential learning in distributed neural networks without catastrophic forgetting: a single and realistic self-refreshing memory can do it," *Neural Information Processing*, vol. 4, no. 2, pp. 27–32, 2004.

[18] B. Ans, S. Rousset, R. M. French, and S. Musca, "Preventing catastrophic in- terference in multiple-sequence learning using coupled reverberating elman networks," in *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, 2002.

[19] I. H. Jenkins, D. J. Brooks, P. D. Nixon, R. S. J. Frackowiak, and R. E. Passingham, "Motor sequence learning: a study with positron emission tomography," *Journal of Neuroscience*, vol. 14, no. 6, pp. 3775–3790, 1994.

[20] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[21] M. Botvinick and D. C. Plaut, "Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action," *Psychological Review*, vol. 111, no. 2, pp. 395–429, 2004.

[22] R. M. French, "Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference," in *Proceedings of the 16th Annual Cognitive Society Conference*, 1994.

[23] M. Riedmiller and B. Janusz, "Using neural reinforcement controllers in robotics," in *Proceedings of the 8th Australian Conference on Artificial Intelligence*, 1995.

[24] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 81–138, 1995.

[25] J. Johnson, R. Challoo, R. A. McLauchlan, and S. I. Omar, "A multi-neural network intelligent path planner for a robot arm," in *Proceedings of the Artificial Neural Networks in Engineering (ANNIE '96)*, 1996.

[26] T. Hesselroth, K. Sarkar, P. P. Van der Smagt, and K. Schulten, "Neural network control of a pneumatic robot arm," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 1, pp. 28–38, 1994.

[27] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *Proceedings of the IEEE World Congress on Computational Intelligences (WCCI '10)*, 2010.

[28] M. Vaezi and M. A. Nekouie, "Adaptive control of a robotic arm using neural networks based approach," *International Journal of Robotics and Automation*, vol. 1, no. 5, pp. 87–99, 2011.

[29] M. McCloskey, "Catastrophic interference in connectionist networks: the sequential learning problem," *Psychology of Learning and Motivation*, vol. 24, pp. 164–169, 1989.

[30] S. P. Singh, "Transfer of learning by composing solutions of elemental sequential tasks," *Machine Learning*, vol. 8, no. 3-4, pp. 323–339, 1992.

[31] L. M. Saksida, S. M. Raymond, and D. S. Touretzky, "Shaping robot behavior using principles from instrumental conditioning," *Robotics and Autonomous Systems*, vol. 22, no. 3-4, pp. 231–249, 1997.

[32] M. Dorigo and M. Colombetti, "Robot shaping: developing situated agents through learning," Tech. Rep. TR-92-040, International Computer Science Institute, 1993.

[33] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the 21st International Conference on Machine Learning (ICML '04)*, pp. 1–8, July 2004.

[34] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference On Machine Learning (ICML '09)*, pp. 41–48, June 2009.

[35] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: a survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[36] H. Saal, J. Ting, and S. Vijayakumar, "Active sequential learning with tactile feedback," *Journal of Machine Learning Research*, vol. 9, pp. 677–684, 2010.

[37] K. A. Kruger, *Sequential learning in the form of shapingas a source of cognitive exibility*, Ph.D. thesis, Gatsby Computational Neuroscience Unit, University of London, 2011.

[38] R. Pollak, J. Schuetznerz, and T. Braunl, "Robot Manipulator Simulation," 1996, http://robotics.ee.uwa.edu.au/robosim/.

[39] C. Balkenius and J. Moren, "Computational models of classical conditioning: a comparitive study," Tech. Rep. LUCS 62, 1998.

[40] R. A. Rescorla, "Retraining of extinguished Pavlovian stimuli," *Journal of Experimental Psychology: Animal Behavior Processes*, vol. 27, no. 2, pp. 115–124, 2001.

[41] R. A. Rescorla, "Savings tests: separating differences in rate of learning from differences in initial levels," *Journal of Experimental Psychology: Animal Behavior Processes*, vol. 28, no. 4, pp. 369–377, 2002.

[42] R. A. Rescorla, "Comparison of the rates of associative change during acquisition and extinction," *Journal of Experimental Psychology: Animal Behavior Processes*, vol. 28, no. 4, pp. 406–415, 2002.

[43] R. A. Rescorla, "More rapid associative change with retraining than with initial training," *Journal of Experimental Psychology: Animal Behavior Processes*, vol. 29, no. 4, pp. 251–260, 2003.

[44] G. S. Reynolds, *A Primer of Operant Conditioning*, Scott, Foresman and Company, 1975.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 549489, 9 pages
doi:10.1155/2011/549489

Research Article

Optimal Search Strategy of Robotic Assembly Based on Neural Vibration Learning

Lejla Banjanovic-Mehmedovic,[1] Senad Karic,[2] and Fahrudin Mehmedovic[3]

[1] *Faculty of Electrical Engineering, University of Tuzla, 75000 Tuzla, Bosnia and Herzegovina*
[2] *H&H Inc., 75000 Tuzla, Bosnia and Herzegovina*
[3] *ABB, 71000 Sarajevo, Bosnia and Herzegovina*

Correspondence should be addressed to Lejla Banjanovic-Mehmedovic, lejla.mehmedovic@untz.ba

Received 17 July 2011; Accepted 8 November 2011

Academic Editor: Ivo Bukovsky

This paper presents implementation of optimal search strategy (OSS) in verification of assembly process based on neural vibration learning. The application problem is the complex robot assembly of miniature parts in the example of mating the gears of one multistage planetary speed reducer. Assembly of tube over the planetary gears was noticed as the most difficult problem of overall assembly. The favourable influence of vibration and rotation movement on compensation of tolerance was also observed. With the proposed neural-network-based learning algorithm, it is possible to find extended scope of vibration state parameter. Using optimal search strategy based on minimal distance path between vibration parameter stage sets (amplitude and frequencies of robots gripe vibration) and recovery parameter algorithm, we can improve the robot assembly behaviour, that is, allow the fastest possible way of mating. We have verified by using simulation programs that search strategy is suitable for the situation of unexpected events due to uncertainties.

1. Introduction

The planning is a key ability of intelligent systems, increasing their autonomy, reliabilities, efficiently and flexibility through the construction of sequences of actions to achieve their goals [1]. In artificial intelligence, planning originally meant a search for a sequence of logical operators or actions that transform an initial world state into a desired goal state. Robot motion planning usually ignores dynamics and considers other aspects, such as uncertainties, differential constraints, modeling uncertainties, and optimality. The robotic assembly, wheelchair navigation, sewer inspection robot, autonomous driving system in urban and off-road environments, and machine's task planning for the robotic system all are examples of autonomous systems, which solve path planning/replanning problems [2, 3].

Dynamic replanning is necessary because at any time during execution of its tasks the robot might unexpectedly run into problems [2]. The typical approach used for replanning is repair plans, which are prepared in advance and invoked to deal with specific exceptions during execution. This class of approaches may work well in relatively static and predictable environment. In more dynamic and uncertain environment where it is hard to anticipate possible exceptions, the replanning generates a (partially) new plan in case when one or more actions have problems during execution [4].

Very interesting area of research is using planning strategies in robot assembly. The example components can be assembled faster, gentle, and more reliably using the intelligent techniques. In order to create robot behaviours that are similarly intelligent, we seek inspiration from human strategies date [5]. The working theory is that the human accomplishes an assembly in phases, with a defined behaviour and a subgoal in each phase. The human changes behaviours according to events that occur during the assembly, and the behaviour is consistent between the events. The human's strategy is similar to a discrete event system in that the human progresses through a series of behavioural states separated by recognizable physical events.

The primary source of difficulty in automated assembly is the uncertainty in the relative position of the parts being assembled [6]. The crucial thing in robot assembly is how to enable a robot to accomplish a task successfully in spite of the inevitable uncertainties. Often a robot motion may fail and result in some unintended contact between the part held by the robot and the environment.

There are generally three types of approaches to tackle this problem. One is to model the effect of uncertainties in the off-line planning process, but computability is the crucial issue. A different approach is to rely on on-line sensing to identify errors caused by uncertainties in a motion process and to replan the motion in realtime based on sensed information. The third approach is to use task-dependent knowledge to obtain efficient strategies for specific tasks rather than focusing on generic strategies independent of tasks.

A systematic replanning approach which consisted of patch planning based on contact analyses and motion strategy planning based on constraints on nominal and uncertainty parameters of sensing and motion is introduced in [7]. In order to test the effectiveness of the replanning approach, they have developed a general geometric simulator SimRep which implements the replanning algorithms, allows flexible design of task environments and modeling of nominal and uncertainty parameters to run the algorithms, and simulates the kinematics robot motions guided by the replanning algorithms in the presence of uncertainties.

Another possibility in achieving acceptably fast robot behavior with assuring contact stability is learning unstructured uncertainties in robot manipulators date. The example components can be assembled faster, gentle, and more reliably using the intelligent techniques. Many promising intelligent control methods have been investigated [5, 8]. For example, work in [9] describes intelligent mechanical assembly system. Correct assembly path is chosen by using form of genetic algorithm search, so the new vectors are evolved from most successful "parents." Another possibility in achieving acceptably fast robot behavior with assuring contact stability is learning unstructured uncertainties in robot manipulators date. The paper [10] presents implementation of intelligent search strategy based on genetic algorithm in verification of assembly process in the presence of uncertainties.

The main contribution of our work is using optimal search strategy in combination with robot learning from experimental setup. The research platform is the complex robot assembly of miniature parts in the example of mating the gears of one multistage planetary speed reducer. Assembly of tube over the planetary gears was noticed as the most difficult problem of overall assembly, and favorable influence of vibration and rotation movement on compensation of tolerance was also observed. For robotic assembly, the tolerance is especially difficult problem because in process of mating it must be compensated, but it takes time and requires corresponding algorithms. In order to compensate tolerance during robot assembly, we plan motion, involving path alternatives to yield minimum distance. The neural-network-based learning gave us new successful vibration solutions for each stage of reducer [11]. In this paper, we introduce optimal search strategy based on vibration parameters state in order to overcome uncertainties during motion planning.

2. Robot Assembly

The first part of our research was the complex robot assembly of miniature parts in the example of mating the gears of one multistage planetary speed reducer. The main difficulty in assembly of planetary speed reducers is the installation of tube over planetary wheels. Namely, the teeth of all three planetary wheels must be mated with toothed tube. Figure 1 presents planetary speed reducer (cross-section 20 mm, height of five stages 36 mm), which has been used for experiments.

In this research, it has not been considered the complete assembly of each part of planetary reducer but only the process of connecting the toothed tube to five-stage planetary reducer. By solving the problem of assembly of the gears, there will be no problem to realise complete assembly of planetary speed reducer.

For the process of assembly, the vertical-articulated robot with six degrees of freedom, type S-420i of the firm FANUC has been used, completed by vibration module (Figure 2), developed at Fraunhofer-IPA in Stuttgart, Germany.

Namely, the analysis of assembly process showed that movement based on vibration and rotation acted positively on the course of process. Vibration module produced vibration in x- and y-direction and rotation around the z-axis.

By starting the robot work, vibration module vibrated with determined amplitude (± 2 mm) and frequency (to max. 10 Hz) for each stage of reducer. The ideal Lisague figures (double eight, circle and line) have been used as figures of vibration for extensive experiments. The vibration figure horizontal EIGHT (Figure 3) was selected for waiter experiments, because we achieved the best performance in assembly process. In that case, the frequency ratio between down and above plate was $f_D/f_U = 2$.

During the robot assembly of two or more parts, we encountered the problem of tolerance compensation.

According to the functioning, the individual systems of tolerance compensation can be divided into:

(i) controllable (active) system for tolerance compensation in which, on base of sensor information on tolerance, the correction of movement is made for the purpose of tolerance compensation;

(ii) uncontrollable (passive) system for tolerance compensation in which the orientation of external parts is achieved by the means of advanced determined strategy of searching or forced by connection forces;

(iii) combination of above two cases.

For this system of assembly, the passive mechanism of tolerance compensation has been used with specially adjusted vibration of installation tools [12]. The assembly process started with gripe positioning together with toothed tube exactly 5 mm above the base part of planetary reducer and continued in direction of negative z-axis (Figure 4).

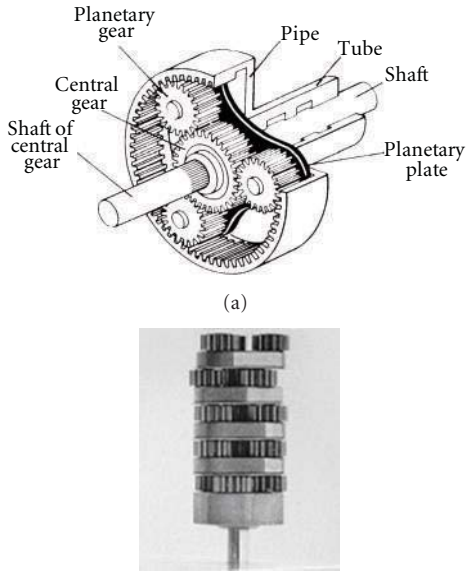


Figure 1: (a) One stage of planetary reducer, (b) planetary speed reducer with 5 stages.

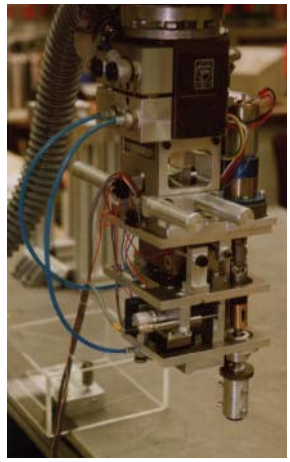


Figure 2: Vibration module.

In order to compensate tolerance during robot assembly, in experimental setup, we used the "search strategy", which adjusted amplitudes and frequencies gained from experimental experience (amplitudes of upper and down plate, frequencies of upper and down plate). As optimum values of amplitudes of down and above plate that were valid for all stages of reducer are $A_D = A_U = 0.8$ mm.

From experiments, we gained that smaller frequencies of vibration were better ($f_D/f_U = 4/2$ or $6/3$) for 1-2 stage (counting of stages starts from up to down), while for each next stage the assembly process was made better with higher frequencies ($f_D/f_U = 8/4$ or $10/5$).

In case of jamming from different physical reasons (position, friction, force, etc.), robot returned to beginning of current reducer stage, where the jamming was made. It

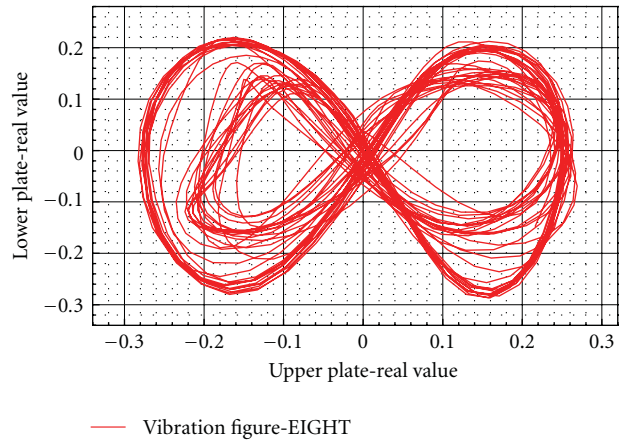


Figure 3: One example of vibration figure—EIGHT ($f_D/f_U = 8/4$, $A_D/A_U = 0.5/0.5$).

exploited the technique of blind search in optimal parameter space with repeated trials at manipulation tasks. When the jamming had been overcome, robot kept moving until it reached the final point in assembly.

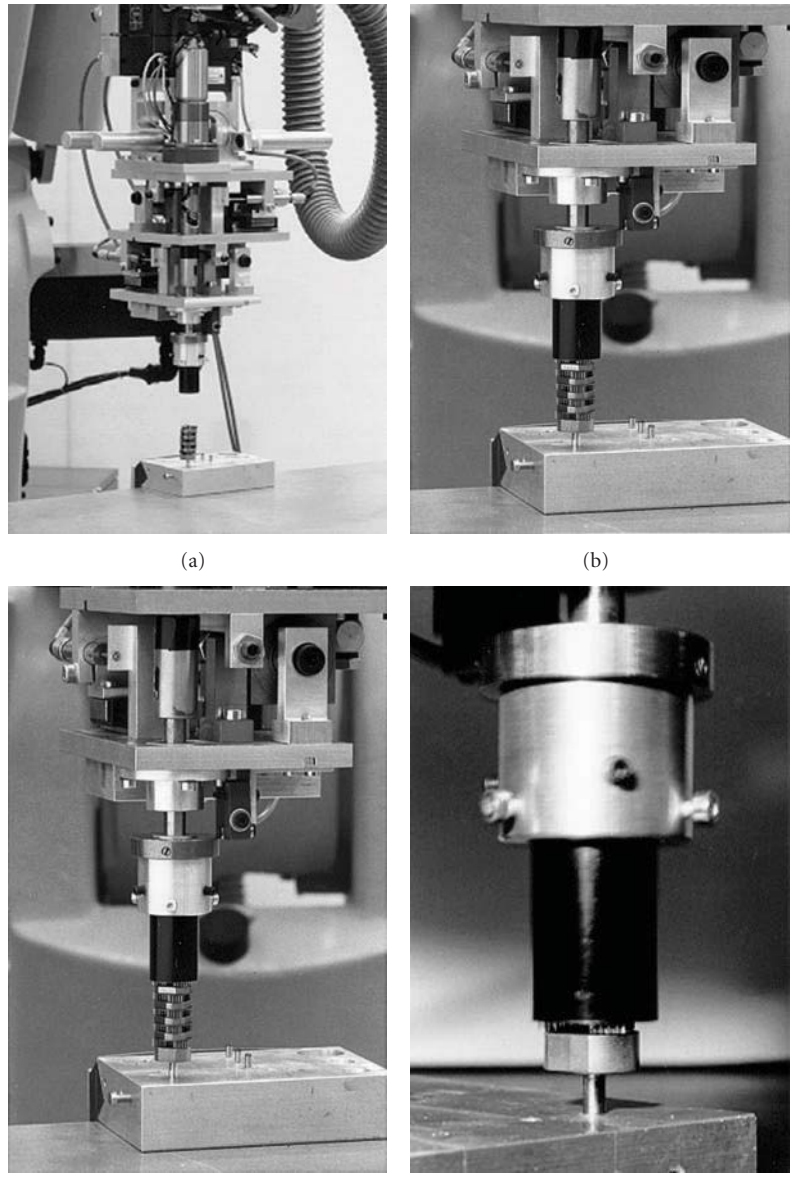
The used speeds of robot were from 4–20 mm/s. The time of complete assembly process for a given range of speeds was a function of frequency, amplitude of upper and lower plate of vibration module, amplitude and frequency of motor rotation, and the speed of motor movement in z-direction. The fastest process of assembly was for robot movement speed of 16 mm/s. Then, the complete process of assembly was only 4 s.

There were extensive experimental complex investigations made for the purpose of finding the optimum solution, because many parameters had to be specified in order to complete assembly process in defined realtime. But, tuning those parameters through experimental discovering for improved performance is a time-consuming process. To make this search strategy more intelligent, additional learning software was created to enable improvements of performance.

3. Advanced Replanning

The planning involves the representation of actions and world models, reasoning about the effects of actions, and techniques for efficiently searching the space of possible plans. Famous search algorithms tailored to planning problems are heuristic search algorithms (A^*, D^*, and Dijkstra algorithm) and their variants.

The planning under uncertainty is a hard job and requires replanning task structure. For example, the robot has to be able to plan the demonstrated task before executing it if the state of the environment has changed after the demonstration took place. The objects to be manipulated are not necessarily at the same positions as during the demonstration, and thus the robot may be facing a particular starting configuration it has never seen before.



(a) (b)

(c) (d)

Figure 4: Particular phases of assembly process.

The replanning is used as specific case of planning process (in case of jamming). Combining with the planning operation, we can describe the replanning strategy as follows.

(1) Given the initial planning problem $P_a = (S, G)$, where S is an initial state parameter, G is a goal state parameter, a plan P_a is a network of actions that lead from S to G (result from optimal search strategy is set of states parameters).

(2) If an action u in P_a fails, we define a replanning area RA $= \{u\}$.

(3) RA is treated as a partially/new plan, and construct a planning problem $P'_b = (S', G)$, where S' is a new start point used by RA. P'_b is a partially set of states parameters, produced by RA as effects of new optimal search strategy.

(4) We search for a plan for P'_b. P'_b replaces P_a in RA, and go to 5. If new action in P_a fails, we go to 2.

(5) Resume the execution of P'_b.

In order for the robots to react to stochastic and dynamic environments, they need to learn how to optimally adapt to uncertainty and unforeseen changes [13]. The robot learning covers a rather large field, from learning to perceive, to plan, to make decisions, and so forth. Learning control is concerned with learning control in simulated or actual physical robots. It refers to the process of acquiring a control strategy for a particular control system and particular task by trial and error.

In our research, we use this concept state and action order to describe the relationships between the parts being assembled. Namely, the states are assembly parameters—vibration amplitudes and frequencies for each planetary reducer stage and transition actions (minimal path) are used to move through assembly process from one stage to another of planetary reducer.

Our planning/replanning search strategy consists of three phase:

(i) learning phase (assembly through the states in X, try various actions, and data collecting),

(ii) planning phase,

(iii) replanning phase.

We use neural-network-based learning which gives us new successful vibration solutions for each stage of reducer. With this extended vibration parameters as source information for planning/replanning task, we introduce optimal search strategy for robot assembly (Figure 5).

The error model is used to model various dynamic effects of uncertainties and physical constraints by jamming. Combing the efforts of the planner and learned optimal values, the replanner is expected to guarantee that agent system enters the region of convergence of its final target location.

4. Neural Aspects of Robot Assembly Learning

Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence [14]. The changes might be either enhancement to already performing systems or synthesis of new system.

In order for the robots to react to stochastic and dynamic environments, they need to learn how to optimally adapt to uncertainty and unforeseen changes [13]. Artificial neural networks are capable of modeling complex mappings between the inputs and outputs of a system up to an arbitrary precision [15]. Process of "capturing" the unknown information is called "learning of neural network" or "training of neural network". In mathematical formalism to learn means to adjust the free parameters (synaptic weight coefficients and bias levels) in such a way that some conditions are fulfilled [16].

There exist many types of neural networks, but the basic principles are very similar. Neural-network-based learning is used in this research to generate wider scope of parameters in order to improve the robot behaviour. The amplitude and frequencies vibration data are collected during assembly experiments and are used as sources of information for the learning algorithm.

In our research, we used multilayer feed-forward neural networks (MLF) and Elman neural networks. MLFs, trained with a backpropagation learning algorithm, are the most popular neural networks. Elman neural network differs from conventional ones in that the input layer has a recurrent connection with the hidden one. Therefore, at each time step, the output values of the hidden units are copied to the input ones, which store them and use them for the next time step. This process allows the network to memorize some information from the past, in such a way to better detect periodicity of the patterns [17].

We expected that Elman neural network will be better than a standard MLF in our application, but we got the better results with MLF. Namely, MLF is better for learning in order to extend learning area parameters.

In our research, we used MLF neural network containing 10 tansig neurons in hidden layer and 1 purelin neuron in its output layer. The feed-forward neural networks were formed and tested for each stage of assembly process. Each one was initialized with random amplitudes $A_U = A_D = A_i$ between 0 and 2 and frequencies values f_i between 0 through 4. Namely, the range of the frequencies measurement is normalized by mapping from frequencies ratio $f_U/f_D = (4/2, 6/3, 8/4, 10/5)$ onto the range of the state frequencies values (0 through 4). To trains the MLF network, we used 35 vibrations sets for each 5 phases of assembly. The mean square errors (MSE) during the training of 5 MLF networks were achieved for 7–10 epochs. Two thousand data points were taken as a testing sample.

The feed-forward neural networks were formed and tested for each stage of assembly process. The following pictures (Figures 6, 7, and 8) present learning of new optimal stage vibration sets indicated by their respective picture.

The results show that the scope of adjusted vibration parameters obtained from autonomous learning is extended in respect to adjusted vibration sets from experimental robot assembly. We can see that critical moment in assembly process is second phase, which presents medium clutter position of optimal vibration parameter sets through stages. Second phase presents discontinuity between first and third phases in clutter space.

The search strategy involved in assembly experiments exploited the technique of blind search of optimal vibration values in repeated trials in each stage. If selected optimal value is in discontinuity area, then the path between one selected optimal stage parameter set and another will be outside of cone (Figure 9).

In this case, the tolerance compensation is not achieved, because position tolerance of some stage D is greater than admitted position tolerance $D0$. In order to solve this problem, we introduce optimal search strategy.

5. Optimal Search Strategy

Rather than being satisfied with any sequence of actions that leads to the goal set, we would like to propose a solution that would optimize some criterion, such as time, distance, or energy consumed, that is, we talk about optimal planning [1].

Consider representing the optimal planning problem in terms of states and state transitions. Let X be a nonempty, finite set of states, which is called the state space. Let $x \in X$ denote a specific state, x_I denote the initial state, and $x_G \in X$ denote a goal states.

For each $x \in X$, let $U(x)$ denote a nonempty, finite set of actions. Let L denote the function, which is applied to the sequence $u_1, \ldots, u_K$ of applied actions and achieved states

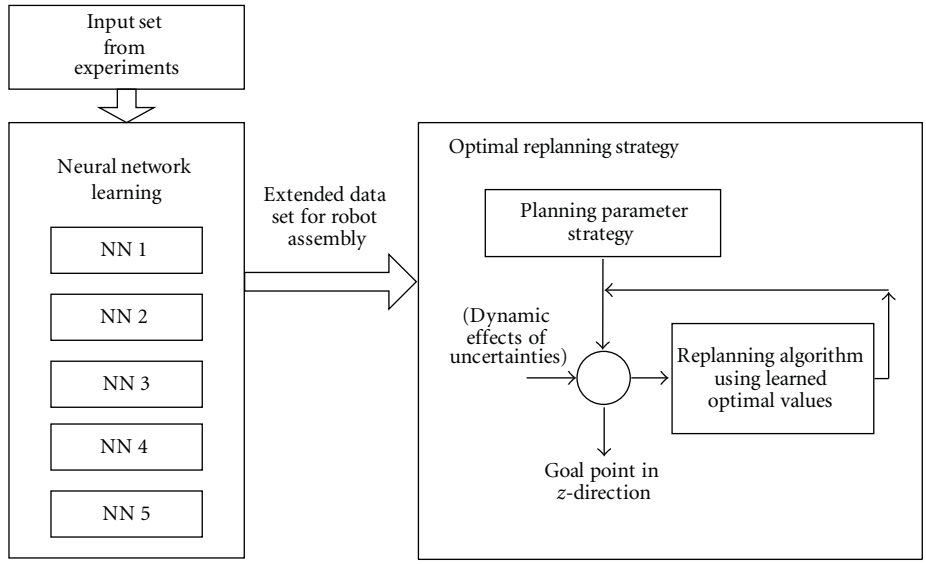


Figure 5: Optimal replanning strategy based on neural learning.

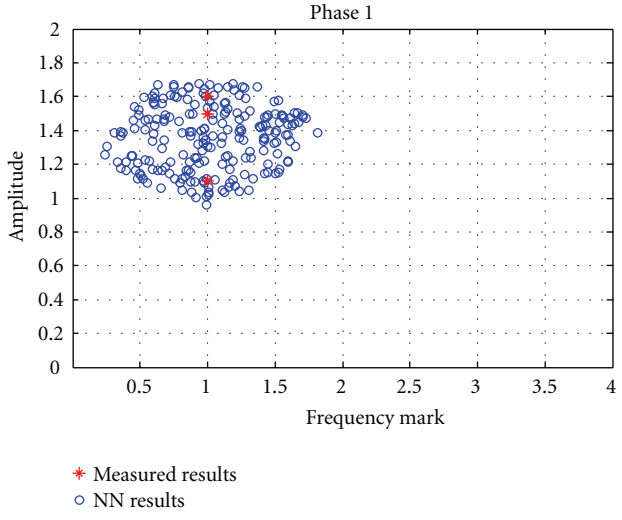


Figure 6: Results of neural network training for first stage.

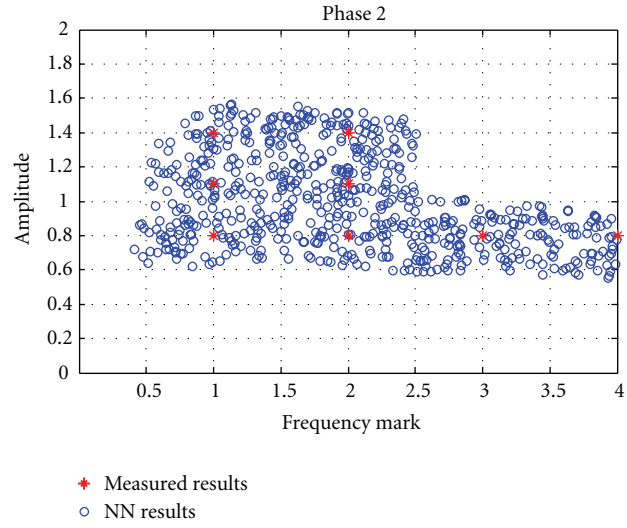


Figure 7: Results of neural network training for second stage.

$x_1, \ldots, x_G$. The task is to find a sequence of actions, $u_1, \ldots, u_K$ that minimizes cost for each segment L_k, that is,

$$L_k^* = \min_{u_k}(l_i(s_k, u_k)). \quad (1)$$

A path L is defined as a series of linear segments L_k connecting state points (P_k, P_{k+1}), $k = 1, \ldots, N$.

In our research, the problem with applied search strategy in experiments was in case of behaviour switching (case of assembly jamming). The search strategy tried to continue assembly process with another optimal, but blind chosen parameter state value. But, using *Optimal Search Strategy*, we use the transition action with minimal distance between vibration state sets:

$$L_k^* = \min_{d_i}(d_i((A_k, f_k), u_k)). \quad (2)$$

An algorithm finds *minimal distance vector* from selected optimal value (A_i, f_i), $i = 1, \ldots, N$ from current extended vibration state s_k gained from learning process towards next vibration state s_{k+1}. The minimal path between two phases is in cone, and we have compensated tolerance ($D < D0$), see Figure 10.

In case of jamming (in our simulator: *error event signal*), we propose *recovery parameter algorithm with learned optimal values*, which offers new plan for path tracking during simulation of robot assembly.

We can explain this with next example. Figure 10 presents next situation: system detects error event during second state of assembly and strategy try to continue assembly process with another optimal set value (A_2', f_2') from state s_2. This value is optimal parameter value, which distance is mean value of all distances from state s_1 to state s_2.

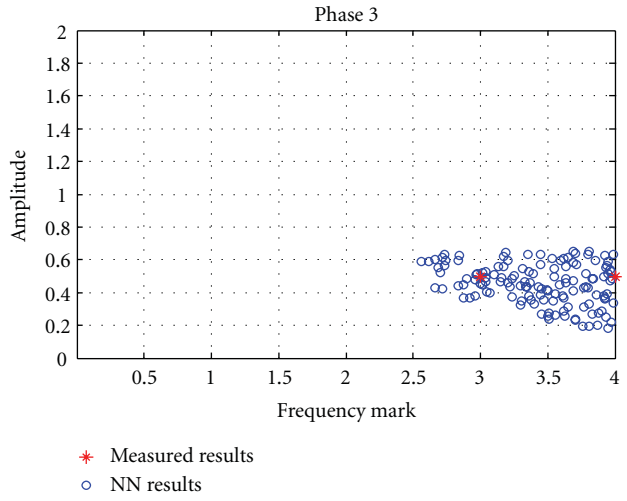


Figure 8: Results of neural network training for third stage.

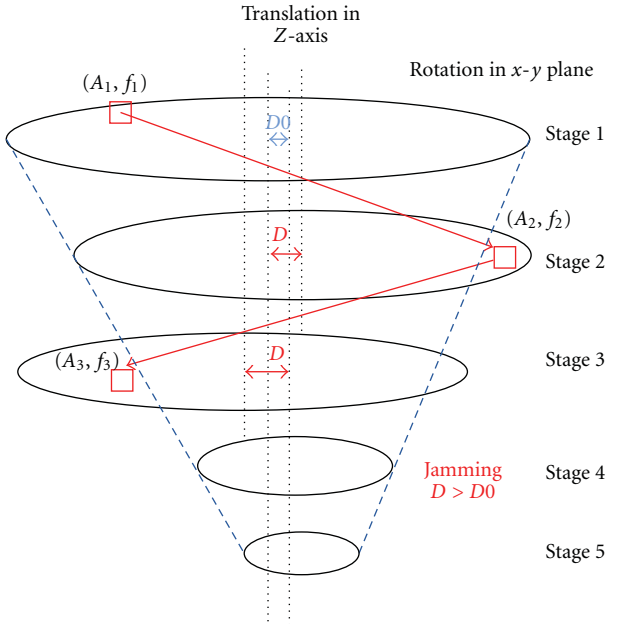


Figure 9: Transition problems between states inside Search Strategy.

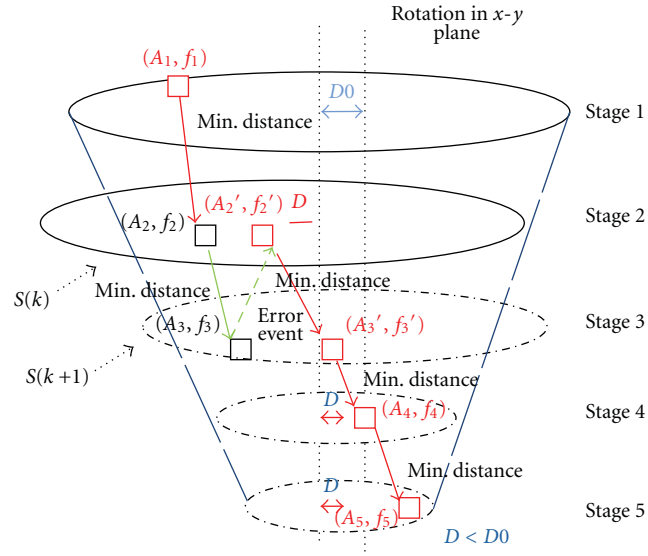


Figure 10: Optimal search strategy between states in jamming case.

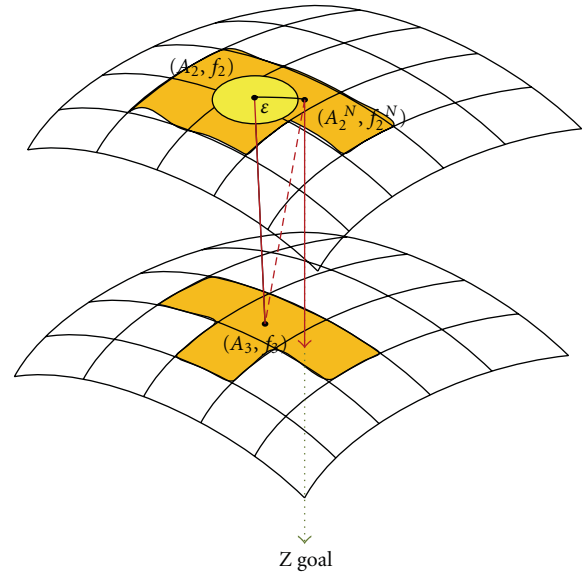


Figure 11: Choice of another optimal solution.

We make enough offset from this critical optimal point to another optimal solution (Figure 11). After that, strategy establishes action between values (A_2', f_2') and (A_3', f_3').

Backward formulation of the optimal cost of each segment to the goal:

$$L_k^* = \min_{d_i'} \left(d_i' \left(\left(A_k', f_k' \right), u_k' \right) \right). \quad (3)$$

To demonstrate the validity of this paradigm, we present test results obtained by implementation of robot assembly agent in Matlab. Some results of using optimal search strategy are demonstrated in Table 1.

Graphical demonstration of optimal search strategy without jamming (first case) is shown in Figure 12. For second and third cases, the replanning is necessary.

We consider the second case (Figure 13). In case of detecting of error event signal in first level, search strategy tries instead optimal value $(1.28, 1.68)$ to continue assembly process with another optimal assembly vibration parameter stage set value $(1.39, 0.85)^*$. New transition action is made from this new optimal value from current state with minimal path distance towards optimal vibration parameter stage set in next state $(1.42, 0.94)$. But, replanning is necessary with new optimal value for the second stage $(1.14, 2.25)^*$. Assembly process goes until it reaches the final point in assembly simulation process.

Table 1: Examples of using optimal search strategy.

N	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
(1)	(1.32, 1.66)	(1.31, 1.59)	(0.59, 2.72)	(0.59, 2.74)	(0.61, 2.67)
	—	—	—	—	—
(2)	(1.28, 1.68)	(1.42, 0.94)	(0.53, 2.60)	(0.56, 2.58)	(0.57, 2.59)
	$(1.39, 0.85)^*$	$(1.14, 2.25)^*$			
(3)	(1.36, 1.70)	(1.32, 1.75)	(0.47, 2.58)	(0.53, 2.74)	(0.39, 2.81)
		$(1.29, 0.72)^*$			

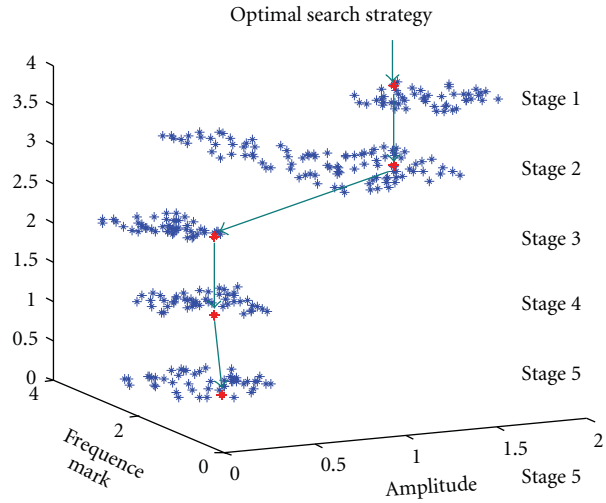


Figure 12: Presentation of Optimal search strategy without error event signals.

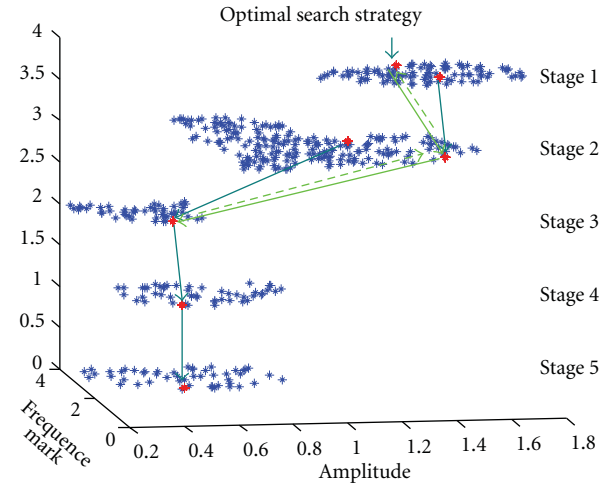


Figure 13: Presentation of Optimal search strategy with error event signals (backward case).

In third case, agent starts with vibration value (1.36, 1.70). In case of detecting of error event signal in second state, deterministic search strategy tries instead optimal value (1.32, 1.75) to continue assembly process with another optimal assembly vibration parameter stage set value $(1.29, 0.72)^*$. New transition action is made from this new optimal value from current state with minimal path distance towards optimal vibration parameter stage set in next state, until it reaches the final point in assembly simulation process.

6. Conclusion

In this paper, the problem of path planning/replanning due to unexpected events during robot assembly is presented. As an example of robot assembly, it was researched the complex assembly of toothed tube over planetary gears. Important contribution of paper is solving tolerance compensation's problem using combination search strategy and neural learning approach. Namely, we used an approach with task-dependent knowledge to obtain efficient strategy for specific task.

The supervised neural-network-based learning is used to generate wider scope of vibration state parameters in order to accommodate the uncertainty in complex assembly of tube over planetary gears in case of jamming. The optimal search strategy is used to reach a goal matting point with minimum segment cost, that is, with minimal time of robot assembly.

In order to verify this approach, we have tested the several model data by computer simulation. The results show this approach satisfactorily solves the complex problem of tolerance compensation under uncertainty regardless of their complexity. This intelligently based path replanner has evolved to be suitable for number forms of robot planning/replanning tasks.

Acknowledgment

Fahrudin Mehmedovic is ABB representative for Bosnia and Herzegovina.

References

[1] S. M. Lavalle, *Planning Algorithms*, Cambridge University Press, 2006.

[2] O. Adria, H. Streich, and J. Hertzberg, "Dynamic replanning in uncertain environments for a sewer inspection robot," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 33–38, 2004.

[3] R. Philippsen, S. Kolski, K. Macek, and R. Siegwar, "Path planning replanning and execution for autonomous driving in urban and offroad environments," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '97)*, 1997.

[4] J. F. Zhang, X. T. Nguyen, and R. Kowalczyk, "Graph-based multiagent replanning algorithm," in *Proceedings of the 6th*

International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '07), pp. 1–8, 2007.

[5] S. P. Chan, "A neural network compensator for uncertainties in robotic assembly," *Journal of Intelligent & Robotic Systems*, vol. 13, no. 2, pp. 127–141, 1995.

[6] E. G. Vaaler, *A machine learning based logic branching algorithm for automated assembly*, Ph.D. MIT, 1991.

[7] J. Xiao and L. A. Zhang, "Geometric simulator SimRep for testing the replanning approach toward assembly motions in the presence of uncertainties," in *Proceedings of the IEEE International Symposium on Assembly and Task Planning (ISATP '95)*, pp. 171–177, August 1995.

[8] L. Brignone, K. Sivayoganathan, V. Balendran, and M. Howarth, "Contact localisation: a novel approach to intelligent robotic assembly," in *International Joint Conference on Neural Networks*, pp. 2182–2187, July 2001.

[9] W. S. Newman, M. Branicky, and J.-H. Pao, "Intelligent strategies for compliant robotic assembly," in *Proceedings of the 11th Yale Workshop on Adaptive and Learning Systems*, pp. 139–146, 2001.

[10] L. Banjanovic-Mehmedovic, I. Bosankic, and F. Mehmedovic, "Robotic assembly planning motion based on genetic algorithm," in *Proceedings of the International Conference of Innovative Technologies (In-Tech '11)*, 2011.

[11] L. Banjanovic-Mehmedovic and S. Karic, "Robotic assembly re-planning agent based on specially adjustable vibrations," in *Advances in Reinforcement Learning*, B. A. Mellouk, Ed., INTECH, 2011.

[12] L. Banjanovic-Mehmedovic, "Robot assembly of planetary motor speed reducer," in *Proceedings of the VIII IEEE Electrotechnical and Computer Science Conference (ERK '99)*, pp. 267–270, Portoroz, Slowenia, 1999.

[13] J. Farrell and W. Baker, "Learning control systems," in *Introduction to Intelligent and Autonomous Control*, P. J. Antsaklis and K. M. Passino, Eds., Kluwer Academic Publishers, 1993.

[14] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[15] N. T. Siebel and Y. Kassahun, "Learning neural networks for visual servoing using evolutionary methods," in *Proceedings of the 6th International Conference on Hybrid Artificial Intelligence (IEEE HIS'06)*, 2006.

[16] D. Svozil, V. Kvasnička, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[17] S. Zappacosta, G. BaldassarreE, and S. Nolfi, "Elman neural networks and time integration for object recognition," in *Proceedings of the Atti del Terzo Workshop Italiano di Vita Artificiale (WIVA3 '06)*, A. Acerbi, S. Giansante, and D. Marocco, Eds., 2006.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 845685, 8 pages
doi:10.1155/2011/845685

Research Article

Control Loop Sensor Calibration Using Neural Networks for Robotic Control

Kathleen A. Kramer[1] and Stephen C. Stubberud[2]

[1] *Department of Engineering, University of San Diego, 5998 Alcalá Park, San Diego, CA 92110, USA*
[2] *Advanced Programs, Oakridge Technology, Del Mar, CA 92014, USA*

Correspondence should be addressed to Kathleen A. Kramer, kramer@sandiego.edu

Received 15 July 2011; Accepted 8 November 2011

Academic Editor: Ivo Bukovsky

Whether sensor model's inaccuracies are a result of poor initial modeling or from sensor damage or drift, the effects can be just as detrimental. Sensor modeling errors result in poor state estimation. This, in turn, can cause a control system relying upon the sensor's measurements to become unstable, such as in robotics where the control system is applied to allow autonomous navigation. A technique referred to as a neural extended Kalman filter (NEKF) is developed to provide both state estimation in a control loop and to learn the difference between the true sensor dynamics and the sensor model. The technique requires multiple sensors on the control system so that the properly operating and modeled sensors can be used as truth. The NEKF trains a neural network on-line using the same residuals as the state estimation. The resulting sensor model can then be reincorporated fully into the system to provide the added estimation capability and redundancy.

1. Introduction

In the target tracking applications, sensor systems are placed in various locations about the region of interest and provide measurements that are combined over time to provide an improved picture of the region. One of the significant problems with target tracking systems is that the sensors might not be properly calibrated, aligned, or modeled. This can wreak havoc with the performance of the involved tracking systems. When calibration and alignment are the main factors, the problem is referred to as sensor registration [1, 2]. Since the real-time activities of a tracking problem cannot be postponed, the correction of the registration or the mismodeling must be performed online. This problem can extend to other applications, including that of feedback-control systems.

While some control applications use a single sensor to provide measurements that are used in the feedback loop, many control systems use several sensors to provide the necessary measurements. Multiple sensors are used for a variety of reasons. Often times the sensors add redundancy, such as with navigation systems where multiple inertial navigation systems (INSs) are employed or for safety considerations to provide back-up in case of failure [3]. Sometimes multiple sensors are employed because more accurate sensors cannot provide measurements at the necessary update rates. In such cases, less accurate sensors might be used to provide reports at the sampling time between the measurement reports of the more accurate system. Finally, additional sensors are added in that they provide measurements that contain directly observable state information. In mobile robotic systems, for example, sensors include position sensors to determine absolute position, such as a global position system (GPS), or relative position such as radars, sonars, and imagery equipment, as well as speedometers and possible accelerometers that provide the velocity states more directly than the indirectly observed values from position.

While additional sensors are often used to provide improved overall accuracy, errors can arise in the accuracy of the measurements provided to the control law by individual sensors. This occurs with tracking sensors and is well known with gyroscopes in an INS. While a sensor that drifts from its

calibration point in a nonrandom and identifiable manner can be recalibrated, there is no remedy other than redundancy or replacing a sensor that fails or breaks completely since it is unable to provide a meaningful measurement. For a sensor that can be recalibrated, online calibration is desired because taking the inaccurate sensor offline may not be possible in the short term or could have a deleterious effect on the performance of the controller. When a closed-loop control system relies upon a sensor with an error, the result is that the control signal varies from the value needed to achieve the desired response. These effects are evident, even in a linear system [4]. Sensor calibration is thus an important issue for the control design [5]. For nonlinear systems, these issues become even more pronounced and can lead to a more significant effect on stability, as the feedback may not fall within the Lyapunov stability criteria [6].

In this work, an online calibration technique is proposed for the case where the control law utilizes a state estimator [7]. The approach uses a technique referred to as a neural extended Kalman filter (NEKF) [8–10]. The NEKF has the benefit of being able to train a function approximation online and provide the state estimation simultaneously. The NEKF provides a unified approach using the same residual for both components. Techniques such as in [11], in contrast, can provide the desired sensor correction for sensors in a control loop but are not integrated to the state estimator and corrections are performed outside of the control loop. The technique applied to this sensor correction problem is based upon an approach that was developed for a multisensor tracking system where the sensor model was in error or a sensor registration was detected [12]. In that development, one sensor was considered to be local while the other was considered to be off-board. All corrections were made to the off-board reports relative to the local reference frame. Unlike the tracking problem where the estimator is open loop, control applications require the consideration of closed-loop issues including stability. The NEKF is used to recalibrate the sensor model based on online operational measurements while in the control loop. It also provides the state estimation for the control law. Thus, the proposed technique estimates the states and provides the training paradigm in a single algorithmic implementation [13, 14], unlike other neural network sensor modeling techniques [15–17]. The software correction to the actual sensor measurements can be applied to the case where the sensor is still operational but poorly calibrated.

The NEKF algorithm requires a truth measurement from which to calibrate the sensor. Using additional sensors on the dynamic system, a poorly calibrated sensor can be modeled such that its reporting errors are removed prior to their incorporation to the feedback loop.

In Section 2, the NEKF algorithm and its implementation in the control loop is developed. Section 3 provides the first example system and the performance results of the online calibration using the NEKF for a two-sensor moving platform that has a range-bearing sensor and a miscalibrated velocity sensor. A navigation problem where an intermittent use of a position sensor such as GPS is available is presented in Section 4. These examples show the benefits of the NEKF calibration.

2. Recalibration Approach

In this effort, a recalibration technique that can be used online is developed. The approach relies upon other sensors in the multisensor system operating properly which are used to provide a level of truth from which to calibrate. The sensor to be recalibrated may have some inaccuracy but cannot be irreparably damaged.

The dynamics of a nonlinear system can be modeled as a set of recursive difference equations:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{r}_k) + \boldsymbol{\nu}_k, \\ \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\eta}_k, \end{aligned} \tag{1}$$

where $\mathbf{x}_k$ is the state vector representing the system dynamics, $\mathbf{f}(\cdot)$ is state-coupling model, $\mathbf{r}_k$ is the reference input vector, $\mathbf{z}_k$ is measurement vector, also considered the report from the sensor system, $\mathbf{u}_k$ is an external input to the sensor system often simply the reference signal, and $\mathbf{h}(\cdot)$ is the output-coupling function [18]. The vectors $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$ represent noise on the system states and the measurement states, respectively.

If the reference input $\mathbf{r}$ is assumed to be both the reference signal to the system and the external input to the sensor, then $\mathbf{r}$ is considered to be the same as $\mathbf{u}$. Then, the state estimation model used in the control law would be rewritten as

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= \hat{\mathbf{f}}(\mathbf{x}_k, \mathbf{u}_k), \\ \hat{\mathbf{z}}_k &= \hat{\mathbf{h}}(\mathbf{x}_k, \mathbf{u}_k). \end{aligned} \tag{2}$$

The accuracy of the models in the estimator determines the accuracies of state estimates. For this effort, it is assumed that the state-coupling function is very accurate while at least one of the sensor models has some inaccuracy. The error in this model is defined as

$$\boldsymbol{\varepsilon} = \mathbf{h}_{\text{true}}(\mathbf{x}, \mathbf{u}) - \mathbf{h}_{\text{model}}(\mathbf{x}, \mathbf{u}). \tag{3}$$

To reduce the measurement error, the function $\boldsymbol{\varepsilon}$ needs to be identified. A neural network that satisfies the function approximation requirements of the Stone-Weierstrauss Theorem [9] is proposed to approximate the error function $\boldsymbol{\varepsilon}$. Such a neural network can be defined as

$$NN(\mathbf{x}, \mathbf{w}) = NN(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\beta}) = \sum_j \beta_j g_j\left(\mathbf{x}, \boldsymbol{\omega}_j\right), \tag{4}$$

where $\mathbf{w}$ are the weights of the neural network, which are decomposed into the set of input weights $\boldsymbol{\omega}$ and the set of output weights $\boldsymbol{\beta}$. The hidden function $\mathbf{g}(\cdot)$ is a sigmoid function:

$$g(\mathbf{x}, \boldsymbol{\omega}) = \frac{1 - e^{-\omega^T \mathbf{x}}}{1 + e^{-\omega^T \mathbf{x}}} = \frac{1 - e^{-\sum_k \omega_{jk} x_k}}{1 + e^{-\sum_k \omega_{jk} x_k}}. \tag{5}$$

This creates a new measurement and more accurate measurement model:

$$e = \mathbf{h}_{\text{true}}(\mathbf{x}, \mathbf{u}) - \mathbf{h}_{\text{model}}(\mathbf{x}, \mathbf{u}) - NN(\mathbf{x}, \mathbf{u}, \mathbf{w}), \tag{6}$$

where $\|e\| \ll \|\varepsilon\|$.

The extended Kalman filter (EKF) is a standard for nonlinear estimation in a control loop [4, 19]. An approach developed in [7] referred to as a neural extended Kalman filter provides both the state estimation of the system and the training of the weights [20] of a neural network using the same measurement estimates.

The NEKF is a coupled state-estimation and neural-network training algorithm and has similarities to parameter estimation approaches [21]. In such implementations, the state vector of the EKF contains both the system states and the parameters of the model being estimated. Parameter estimation, though, is based on a known model structure. The neural network of (4), in contrast, is a general function with its weights as the parameters. Therefore, the augmented state of this NEKF is the state estimate and the input and output weights of the neural network:

$$\overline{\mathbf{x}}_k = \begin{bmatrix} \mathbf{x}_k \\ \hline \mathbf{w}_k \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k \\ \hline \boldsymbol{\omega}_k \\ \boldsymbol{\beta}_k \end{bmatrix}. \tag{7}$$

Incorporating this new state and the neural network affects all of the Kalman filter equations. In the Kalman gain equation and the state error covariance update equation, (8) and (10), the Jacobian of the output-coupling function is properly augmented. In (9), the neural network is incorporated into the state update. Finally, the prediction equations, (11) and (12), must be properly augmented for the augmented state. Thus, the NEKF equations for the sensor modeling become

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\overline{\mathbf{H}}^T\left(\overline{\mathbf{H}}\mathbf{P}\overline{\mathbf{H}}^T + \mathbf{R}\right)^{-1}, \tag{8}$$

$$\begin{aligned}\overline{\mathbf{x}}_{k|k} &= \begin{bmatrix} \mathbf{x}_{k|k} \\ \mathbf{w}_{k|k} \end{bmatrix} = \overline{\mathbf{x}}_{k|k-1} \\ &\quad + \mathbf{K}_k\left(\mathbf{z}_k - \left(\mathbf{h}\left(\mathbf{x}_{k|k-1}, \mathbf{u}_k\right) + NN\left(\mathbf{x}_{k|k-1}, \mathbf{u}_k, \mathbf{w}_{k|k-1}\right)\right)\right),\end{aligned} \tag{9}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \overline{\mathbf{H}}\mathbf{K}_k\right)\mathbf{P}_{k|k-1}, \tag{10}$$

$$\overline{\mathbf{x}}_{k+1|k} = \overline{\mathbf{f}}\left(\overline{\mathbf{x}}_{k|k}\right) = \begin{bmatrix} \mathbf{f}\left(\mathbf{x}_{k|k}\right) \\ \hline \mathbf{w}_{k|k} \end{bmatrix}, \tag{11}$$

$$\begin{aligned}\mathbf{P}_{k+1|k} &= \frac{\partial \overline{\mathbf{f}}(\overline{\mathbf{x}})}{\partial \overline{\mathbf{x}}}\mathbf{P}_{k|k}\frac{\partial \overline{\mathbf{f}}(\overline{\mathbf{x}})}{\partial \overline{\mathbf{x}}}^T + \mathbf{Q}_k \\ &= \begin{bmatrix} \mathbf{F} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\mathbf{P}_{k|k}\begin{bmatrix} \mathbf{F}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} + \mathbf{Q}_k.\end{aligned} \tag{12}$$

The augmented Jacobian of the sensor function model,

$$\overline{\mathbf{h}}(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \mathbf{h}(\mathbf{x}, \mathbf{u}) + NN(\mathbf{x}, \mathbf{u}, \mathbf{w}) \tag{13}$$

is therefore defined as

$$\begin{aligned}\overline{\mathbf{H}} &= \left[\frac{\partial \mathbf{h}(\mathbf{x}) + NN(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{x}} \frac{\partial NN(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}\right]_{\mathbf{x}=\mathbf{x}_{k|k-1}, \mathbf{w}_{k|k-1}} \\ &= \left[\mathbf{H} + \frac{\partial NN(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{x}} \frac{\partial NN(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{w}}\right]_{\mathbf{x}=\mathbf{x}_{k|k-1}, \mathbf{w}_{k|k-1}}.\end{aligned} \tag{14}$$

The necessary coupling between the weights and the dynamic states occurs as a result of the output-coupling Jacobian. This coupling permits the weights to be completely observable to the measurements and to train off of the same residual as the system states. This also implies that the NEKF trains the neural network online without the need for additional off-line training sets.

The NEKF defined in (8)–(12) requires a truth measurement for the neural network to learn. For a target tracking application where a surveyed target can be used to provide ground truth, this implementation is appropriate [9]. Control applications are different in that such "truth" is unavailable. Instead, the state of the system dynamics must be generated using sensors in the control system that are more accurately modeled. This requires a modification of the NEKF design to utilize the existing state estimates and the measurements from the other sensors to correct the output-coupling function of the mismodeled sensor using the neural network.

For this implementation, the detection of a sensor failure is assumed to have occurred. Such detection techniques abound in tracking theory and similar techniques exist in dynamic system implementation [22]. The detection of the faulty sensor initiates two changes in the NEKF. First, the process noise matrix is modified. The ratio of the process noise $\mathbf{Q}$ to the measurement noise $\mathbf{R}$ is reduced to significantly favor the dynamic state estimate over the residual. This reduces the effect of the measurements from the poor sensor on the system estimate. The process noise of the weights is increased to favor the residual over the weight estimates so that the weights will train to learn the error function. A similar approach of reducing the ratio of $\mathbf{Q}$ to $\mathbf{R}$ could be used for the reduction of state-coupling error [7, 8]. The second modification is that, for the first n steps of the correction of the NEKF, the dynamic state estimates are decoupled from the control loop. The resulting modified NEKF equations are given as

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\overline{\mathbf{H}}^T\left(\overline{\mathbf{H}}\mathbf{P}\overline{\mathbf{H}}^T + \mathbf{R}\right)^{-1}, \tag{15}$$

$$\begin{aligned}\overline{\mathbf{x}}_{k|k} &= \begin{bmatrix} \mathbf{x}_{k|k} \\ \hline \mathbf{w}_{k|k} \end{bmatrix} = \overline{\mathbf{x}}_{k|k-1} \\ &\quad + \mathbf{K}_k\left(\mathbf{z}_k - \left(NN\left(\mathbf{x}_{k|k-1}, \mathbf{u}_k, \mathbf{w}_{k|k-1}\right) + \mathbf{h}\left(\mathbf{x}_{k|k-1}, \mathbf{u}_k\right)\right)\right),\end{aligned} \tag{16}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \overline{\mathbf{H}}\mathbf{K}_k\right)\mathbf{P}_{k|k-1}, \tag{17}$$

$$\overline{\mathbf{x}}_{k+1|k} = \overline{\mathbf{f}}\left(\overline{\mathbf{x}}_{k|k}\right) = \begin{bmatrix} \mathbf{x}^{\text{acc}}_{k+1|k} \\ \hline \mathbf{w}_{k|k} \end{bmatrix}, \tag{18}$$

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} \mathbf{P}^{\text{acc}}_{k+1|k} & \mathbf{P}^{\text{Block1,2}}_{k|\text{k}} \\ \mathbf{P}^{\text{Block2,1}}_{k|k} & \mathbf{P}^{\text{Block2,2}}_{k|k} \end{bmatrix} + \mathbf{Q}_k, \tag{19}$$

where the superscript acc indicates the accurate covariance from the estimator using the other sensors. If more than one poor measurement is produced between accurate measurements, only the last measurement was used for the state estimator [7].

Two approaches for decoupling have been considered. In the first, the measurement noise for the faulty sensor is kept artificially high throughout the experiment. This reduces the effect of the poor sensor even as the neural network trains. In the second approach, as the weights of the neural network settle, the measurement noise is reduced from its artificially high values to values closer to the calibrated sensor noise covariance. This allows the improved model of the sensor to have greater effect on the control loop.

3. Control Example I

A simulated control example is used to demonstrate the capability of the NEKF technique to model a sensor change while the system is in operation. A small motorized vehicle is operating in a two dimensional space, as seen in Figure 1.

The goal is for the vehicle to move from one point in the operational grid to a second point in the operational grid. The vehicle maintains its position via a sensor system that provides a range-bearing measurement to a surveyed location in the operational grid. The platform also has a speedometer and compass system that is used to provide the vehicle's heading relative to a local magnetic pole. This is similar to INS systems that are GPS denied.

The state-space model of the vehicle-dynamics is defined as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} + \mathbf{g}(\mathbf{u}), \tag{20}$$

$$\begin{aligned} \mathbf{g}(\mathbf{u}) &= \begin{bmatrix} 0 & a_x & 0 & a_y \end{bmatrix}^T \\ &= \begin{bmatrix} 0 & a \cdot \cos(\theta + \psi) & 0 & a \cdot \sin(\theta + \psi) \end{bmatrix}^T, \end{aligned} \tag{21}$$

where a indicates acceleration, θ denotes the heading angle, and ψ denotes the change in heading angle. For a sample rate of dt, the discretized representation becomes

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k+1} \\ y_k \\ y_{k+1} \end{bmatrix} + \mathbf{g}_k(\mathbf{u}_k). \tag{22}$$

The speed sensor and heading sensor have combined into a single sensor package modeled in (23). The package has an update rate of 0.2 seconds. The accuracy of the compass heading is 2.0 degrees while speedometer accuracy is 0.001 m/s. The position sensor, providing a range (r), and bearing (α) relative to the surveyed location, is modeled in (24), also with an update rate of 0.2 seconds. The position sensor reports are interleaved with the velocity measurements, as shown in the time line of Figure 2. The

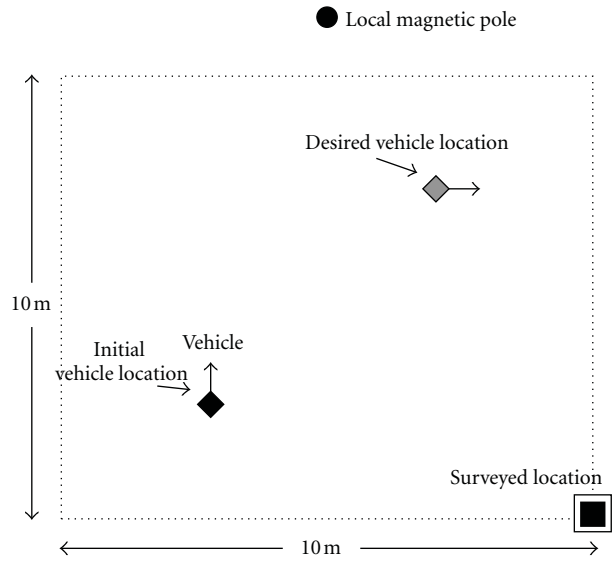


Figure 1: Example scenario of an autonomous vehicle using two sensors to achieve a specific end location.

accuracy of the position sensor is given as 1.0 degrees for bearing and 0.005 m for range:

$$\mathbf{z}_k^{\text{vel}} = \begin{bmatrix} s_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} \sqrt{\dot{x}_k^2 + \dot{y}_k^2} \\ \arctan\left(\dfrac{\dot{y}_k}{\dot{x}_k}\right) \end{bmatrix}, \tag{23}$$

$$\mathbf{z}_k^{\text{pos}} = \begin{bmatrix} r_k \\ \alpha_k \end{bmatrix} = \begin{bmatrix} \sqrt{\left(x_k - x_{\text{survey}}\right)^2 + \left(y_k - y_{\text{survey}}\right)^2} \\ \arctan\left(\dfrac{y_k - y_{\text{survey}}}{x_k - x_{\text{survey}}}\right) \end{bmatrix}. \tag{24}$$

For this effort, the control input has an input rate of 0.1 seconds to synchronize with the sensor reports. The initial location of the vehicle is defined at (−5 m, −6 m) with a heading of 0 degrees, while the desired final point is (0 m, 0 m) with a heading of 45.0 degrees. The range-bearing beacon is placed at (+3 m, −8 m), while the local north pole is defined at (−2 m, +3 m). The position sensor measurement uncertainty matrix had standard deviations of 0.005 m for the range and 1.0 radian for the bearing. For the velocity sensor, the measurement error covariance had standard deviations of 0.001 m/s for the speed and 2 radians for the heading. The broken velocity sensor reported speeds increased by a factor of 1.9 and added a 0.5-radian bias to the heading, in addition to the additive Gaussian noise.

For the NEKF, the process noise for the weights of the neural networks, $\mathbf{Q}_w$, and its initial error covariance, $\mathbf{P}_{wts}$, were increased to allow for changes based on the residuals using the broken sensor. The process noise for the input weights was set to 1.0 and for the output weights, it was set to 2.0. The initial state error covariance, $\mathbf{P}_{states}$, was set to 100.0.

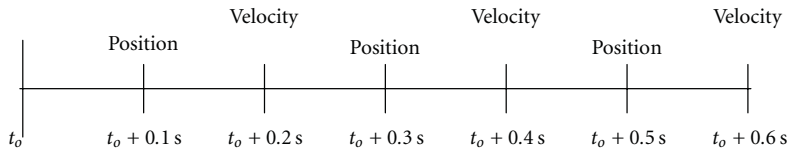


Figure 2: Sensor reports from different sensors are interleaved to provide uniform reporting.

The process noise, **Q**, was set to the integrated white noise model [15]:

$$\mathbf{Q} = q^2 \begin{bmatrix} \frac{dt^3}{3} & \frac{dt^2}{2} & 0 & 0 \\ \frac{dt^2}{2} & dt & 0 & 0 \\ 0 & 0 & \frac{dt^3}{3} & \frac{dt^2}{2} \\ 0 & 0 & \frac{dt^2}{2} & dt \end{bmatrix}, \tag{25}$$

with a factor q of 0.0017. For the NEKF, the initial value for $\mathbf{P}_{\text{wts}}$ was set to 1000. For comparison purposes, an EKF using the same values for the dynamic state components was generated as well.

Four separate cases, two with the EKF and two with the NEKF were implemented. The four cases are (1) an EKF with an inflation factor of 100 throughout the scenario, (2) an EKF with an inflation factor of 10,000,000 throughout the scenario, (3) an NEKF with an inflation factor of 100 throughout the scenario, and (4) an NEKF with an initial inflation factor of 1000 held for 80 iterations and then decayed by 15% for later iterations down to a minimum value of 1.

Inflating the uncertainty matrix, **R**, of the velocity sensor by a given inflation factor allows the state update component of EKF and of the NEKF to be less affected by these poor measurements. For the NEKF, a 4-node hidden layer was used. The results are shown in Figures 3(a)–3(d).

Figure 3(a) indicates that the EKF is unable to remain stable with the poor measurements having the reduced, but not completely insignificant, effect on the state estimate. By overcompensating with a significant increase in the measurement covariance, the EKF basically eliminates all of the poor sensor reports. This is seen in Figure 3(b). However, the vehicle never settles in at the location. The NEKF results are shown in Figures 3(c) and 3(d) where it learns sensor errors while on line to provide clearly improved control performance. Even with the same measurement covariance as the EKF in first case, as in Figure 3(c), the NEKF implementation remains stable. In both Figures 3(c) and 3(d), the results are quite similar with more changes after the measurement noise decays.

With multiple sensor systems, the NEKF is able to provide a fault correcting mechanism for sensors that are still providing information, although that information needs correction. The research of this effort has also shown that the slower the decay rate is on the inflation factor, the lower the initial inflation factor can be.

4. Control Example II

The small motorized vehicle operating in a two-dimensional space shown in Figure 1 is also the basis for the second control example. In this case, the vehicle maintains its position via a two-sensor system. One sensor provides a range-bearing measurement to a surveyed location in the operational grid and another, slower reporting, position sensor provides a linear position report. This would be similar to a GPS system updating an INS.

The state-space model of the vehicle-dynamics is defined as in (20)–(22). The range-bearing error is described as in (23). The position sensor has an update rate of 0.6 seconds and provides a linear report as seen in (26). The accuracy of the position sensor is assumed to be ± 0.01 m in both directions. The other position sensor, providing a range (r), and bearing (α) relative to the surveyed location, is modeled in (24) and has an update rate of 0.2 seconds. The accuracy of the faster, range-bearing position sensor is given as 1.0 degree of bearing accuracy and 0.005 m of range accuracy:

$$\mathbf{z}_k^{\text{pos}} = \begin{bmatrix} x_k \\ y_k \end{bmatrix}. \tag{26}$$

The sensor reports of positions are interleaved as shown in the time line of Figure 4. For this effort, the control input has an input rate of 0.2 seconds to synchronize with the range-bearing sensor report.

The initial location of the vehicle is defined at (-5 m, -6 m) with a heading of 0 degrees, while the desired final point is (0 m, 0 m) with a heading of 45.0 degrees. The range-bearing beacon is placed at ($+3$ m, -8 m). The position sensor providing linear reports provides measurements with a bias of (0.5 m, -0.25 m) added to the coordinates and is considered to be miscalibrated.

For the NEKF, the process noise for the weights of the neural networks, $\mathbf{Q}_{\text{w}}$, and its initial error covariance, $\mathbf{P}_{\text{wts}}$, were increased to allow for changes based on the residuals using the broken sensor. The process noise for the input weights was set to 1.0 and for the output weights, it was set to 2.0. The initial state error covariance, $\mathbf{P}_{\text{states}}$, was set to 10.0. The process noise, **Q**, was set to the integrated white noise model [14] of (25) with a factor q of 0.0017. For the NEKF, the initial value for $\mathbf{P}_{\text{wts}}$ was set to 0.1. The inflation factor for the broken sensor in both the NEKF and the EKF tests was set to 100. For comparison purposes, the EKF that was generated used the same values for the dynamic state components.

In this example case, when the system had two fully functional sensors, the vehicle took 3.6 seconds to reach its desired endpoint with a 0.001 m total distance error.

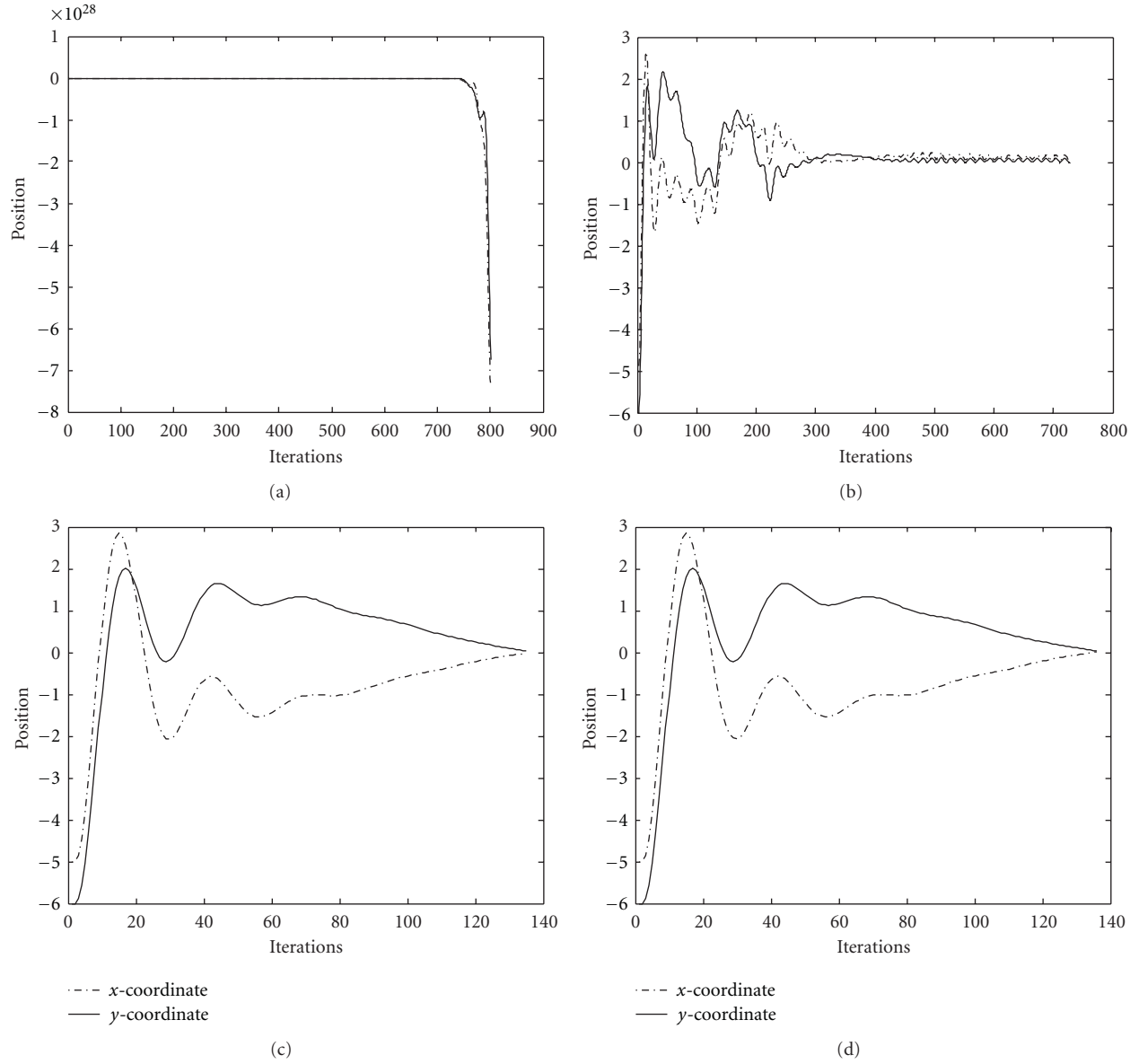


Figure 3: (a) Position coordinates using the EKF and a low inflation rate. (b) Position coordinates using the EKF and a high inflation rate. (c) Position coordinates using the NEKF and a low inflation rate. (d) Position coordinates using the NEKF and a higher inflation rate that is allowed to decay after 80 iterations.

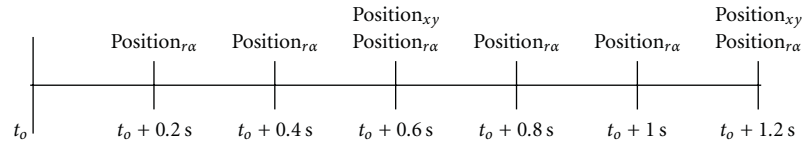


Figure 4: Sensor reports from different sensors are interleaved to provide uniform reporting.

A comparison between the EKF with the bad sensor and the NEKF with the bad sensor was generated. Two different comparisons were run. In the first comparison, an inflation factor of 10 for measurement error covariance matrix of the broken sensor was used. Inflating the uncertainty matrix, **R**, of the velocity sensor by a given inflation factor allows the state update component of EKF and of the NEKF to be less affected by these poor measurements. For the NEKF, a 4-node hidden layer was used. In this case, both the NEKF and the EKF went unstable.

In the second comparison, an inflation factor of 1000 was used. The results of the four states, x-position, x-velocity, y-position, and y-velocity are shown for the EKF in Figure 5. While the individual elements are hard to discern,

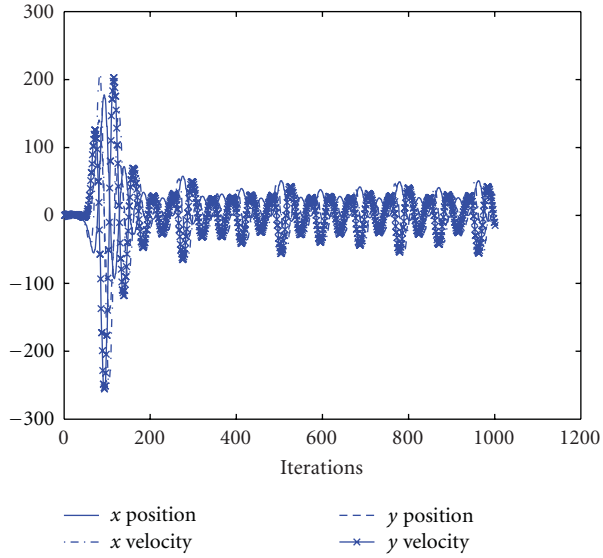


Figure 5: EKF state responses.

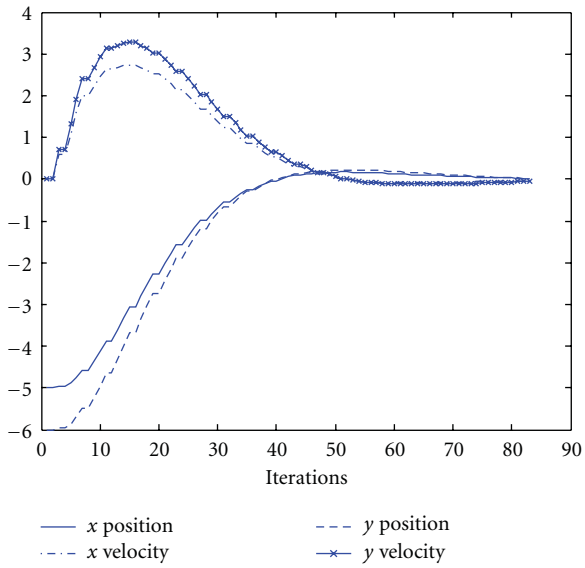


Figure 6: NEKF state responses.

the important fact is that none of the values converge. This causes the vehicle to leave the region of interest. Figure 6 depicts the NEKF results for the four states. As is clearly seen, both techniques achieve a result, but the NEKF trajectory is much smoother and the states are less erratic and require significantly fewer iterations to converge to the desired result (84 iterations versus 498 iterations).

If the position sensor was miscalibrated and not corrected but only had a reduced effect on the navigation estimator, as would be the case when using the EKF only and assumed it were inaccurate, the vehicle position becomes unstable causing it to leave the grid area within 10 seconds of beginning the test. When the NEKF was used, the vehicle was able to reach its desired endpoint in 38.4 seconds with 0.45 m total distance error and 0.3 m/s velocity error.

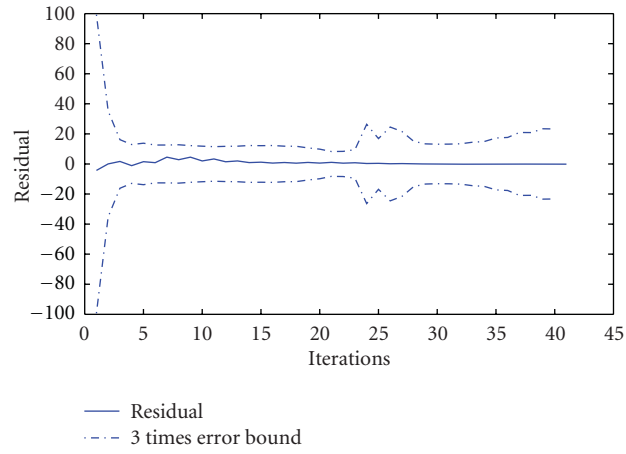


Figure 7: Comparison of EKF residual and the error covariance weighting.

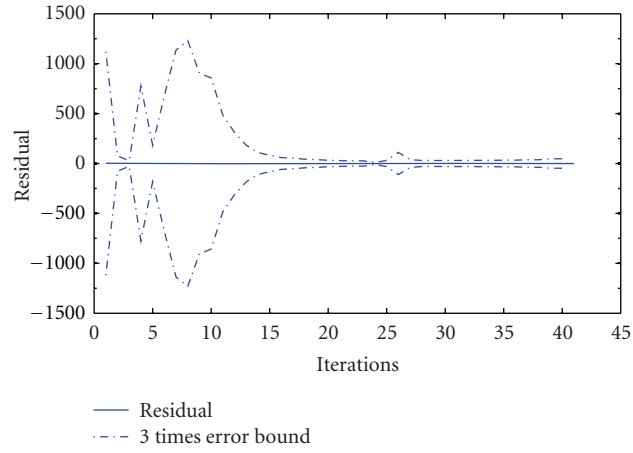


Figure 8: Comparison of NEKF residual and the error covariance weighting.

One of the interesting comparisons is that of the residual errors for heading in this example compared to a 3-sigma error based upon the system and sensor covariances. Figure 7 shows the results for the EKF, and Figure 8 shows the results for the NEKF. In both cases, the residuals are within the bounds, but the NEKF has a more accurate bound in the steady-state portion allowing for the Kalman gain to permit greater residual effect on the state update for the corrected sensor. Thus, without the NEKF, the system would go unstable with deleterious effects almost immediately. With the NEKF, the vehicle remains in the operational area and slowly converges to the desired point.

5. Conclusion

A new neural extended Kalman filter algorithm has been developed that can improve sensor modeling for a poorly modeled sensor in the control loop. The technique expanded on an approach originally developed for target tracking

that relied upon that availability of truth. In the algorithm, properly modeled, operating sensors were used to provide truth used to recalibrate the sensor that is poorly modeled. This NEKF approach decouples its state estimates from the accurate estimates using the measurement error covariance.

The NEKF was shown to have improved performance over that of the EKF in its application to estimate the states of the control loop of an autonomous vehicle for two example cases. Performance of the algorithm will continue to be evaluated in other applications and with different training parameters.

References

[1] R. Lobbia, E. Frangione, and M. Owen, "Noncooperative target sensor registration in a multiple-hypothesis tracking architecture," in *Signal and Data Processing of Small Targets*, vol. 3163 of *Proceedings of SPIE*, San Diego, Calif, USA, July 1997.

[2] S. Blackman, *Multiple-Target Tracking with Radar Applications*, Artech House, Norwood, Mass, USA, 1986.

[3] D. H. Titterton and J. L. Weston, *Strapdown Inertial Navigation Technology*, Institute of Electrical Engineers, London, UK, 1997.

[4] W. S. Levine, Ed., *The Control Handbook*, CRC Press, Boca Raton, Fla, USA, 1996.

[5] A. Martinelli, "State estimation on the concept of continuous symmetry and observability analysis: the case of calibration," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 239–255, 2011.

[6] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*, Prentice Hall, Englewood Cliffs, NJ, USA, 1989.

[7] S. C. Stubberud and K. A. Kramer, "Control loop sensor calibration using neural networks," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC '08)*, pp. 472–477, Victoria, Canada, May 2008.

[8] S. C. Stubberud, R. N. Lobbia, and M. Owen, "Adaptive extended Kalman filter using artificial neural networks," *International Journal of Smart Engineering System Design*, vol. 1, no. 3, pp. 207–221, 1998.

[9] A. R. Stubberud, "A validation of the neural extended kalman filter," in *Proceedings of the International Conference on Systems Engineering*, pp. 3–8, Coventry, UK, September 2006.

[10] S. C. Stubberud, R. N. Lobbia, and M. Owen, "Adaptive extended Kalman filter using artificial neural networks," in *Proceedings of the 34th IEEE Conference on Decision and Control*, pp. 1852–1856, New Orleans, La, USA, December 1995.

[11] D. Klimánek and B. Šulc, "Evolutionary detection of sensor discredibility in control loops," in *Proceedings of the 31st Annual Conference of the IEEE Industrial Electronics Society (IECON '05)*, vol. 2005, pp. 136–141, Raleigh, NC, USA, November 2005.

[12] S. C. Stubberud, K. A. Kramer, and J. A. Geremia, "On-line sensor modeling using a neural Kalman filter," in *Proceedings of the 23rd IEEE Instrumentation and Measurement Technology Conference (IMTC '06)*, pp. 969–974, Sorrento, Italy, April 2006.

[13] K. A. Kramer, S. C. Stubberud, and J. A. Geremia, "Target registration correction using the neural extended kalman filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 7, Article ID 5280378, pp. 1964–1971, 2010.

[14] S. C. Stubberud, K. Kramer, and A. R. Stubberud, "Parameter estimation using a novel nonlinear constrained sequential state estimator," in *Proceedings of the UKACC International Conference on Control*, pp. 1031–1036, Coventry, UK, September 2010.

[15] M. J. Gao, J. W. Tian, and K. Li, "The study of soft sensor modeling method based on wavelet neural network for sewage treatment," in *Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR '07)*, vol. 2, pp. 721–726, November 2007.

[16] Enderle, G. K. Kraetzschmar, Sablatnoeg, and Palm, "One sensor learning from another," in *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN '99)*, vol. 2, no. 470, pp. 755–760, September 1999.

[17] J. W. M. van Dam, B. J. A. Krose, and F. C. A. Groen, "Adaptive sensor models," in *Proceedings of the IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 705–712, December 1996.

[18] M. S. Santina, A. R. Stubberud, and G. H. Hostetter, *Digital Control System Design*, Saunders College Publishing, Fort Worth, Tex, USA, 2nd edition, 1994.

[19] A. K. Mahalanabis, "Introduction to random signal analysis and kalman filtering. Robert G. Brown," *Automatica*, vol. 22, no. 3, pp. 387–388, 1986.

[20] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended kalman algorithm," in *Advances in Neural Processing Systems I*, D. S. Touretsky, Ed., pp. 133–140, Morgan Kaufmann, 1989.

[21] S. C. Iglehart and C. T. Leondes, "Estimation of a dispersion parameter in discrete kalman filtering," *IEEE Transactions on Automatic Control*, vol. 19, no. 3, pp. 262–263, 1974.

[22] Brotherton, Johnson, and Chadderdon, "Classification and novelty detection using linear models and a class dependent—elliptical basis function neural network," in *Proceedings of the IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 2, pp. 876–879, Anchorage, Alaska, USA, 1998.

Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 124314, 7 pages
doi:10.1155/2011/124314

Research Article

Development of Bio-Machine Based on the Plant Response to External Stimuli

K. Aditya,[1,2] Ganesha Udupa,[1] and Yongkwun Lee[2]

[1] *Department of Mechanical Engineering, Amrita Vishwa Vidyapeetham, Amritapuri Campus, Kollam 690525, India*
[2] *Centre for Bionics, Korea Institute of Science and Technology, Seoul 136-791, Republic of Korea*

Correspondence should be addressed to Yongkwun Lee, yklee@kist.re.kr

Received 9 August 2011; Revised 31 October 2011; Accepted 11 November 2011

Academic Editor: Ivo Bukovsky

In the area of biorobotics, intense research work is being done based on plant intelligence. Any living cell continuously receives information from the environment. In this paper, research is conducted on the plant named *descoingsii* x *haworthioides* (Pepe) obtaining the action potential signals and its responses to stimulations of different light modes. The plant electrical signal is the reaction of plant's stimulation owing to various environmental conditions. Action potentials are responsible for signaling between plant cells and communication from the plants can be achieved through modulation of various parameters of the electrical signal in the plant tissue. The modulated signals are used for providing information to the microcontroller's algorithm for working of the bio-machine. The changes of frequency of action potentials in plant are studied. Electromyography (EMG) electrodes and needle-type conductive electrodes along with electronic modules are used to collect and transform the information from the plant. Inverse fast Fourier transform (IFFT) is used to convert signal in frequency domain into voltage signal for real-time analysis. The changes in frequency of the plant action potentials to different light modes are used for the control of the bio-machine. This work has paved the way for an extensive research towards plant intelligence.

1. Introduction

Much like humans, animals and plants have electrical signals which pass through them, but plants do not have nerves like humans and animals. Sanderson [1] was the first to discover the action potentials (APs) in the stimulation of a *Dionaea* leaf. Hence, electrical signals do not belong only to animal kingdom and humans [1]. Darwin [2] also found the response of some carnivorous venus fly trap plants [2]. Generally in humans and animals when the muscle is voluntarily contracted, action appears. In plants it is found that action potentials are the signals caused by the depolarization of plasma membrane [3, 4]. Green plants are able to show different electrical activity, which has been known long time ago [5]. Moreover, exhaustive studies in this field began only in the last decades of the former century along with the different contemporary experimental methods [6, 7]. Electrical phenomena in plants have complicated character. Plant tissues are very complicated, highly structured consisting of both conductive and insulative elements. Because of that the resistance of different plants is not only ohmic but also frequency dependent.

In 1926, Bose used the isolated vascular bundles of a fern (*Blechnum nudum*) to show that excitation was transmitted as an electrical disturbance that appeared to be controlled by similar physiological events as in animal nerves [8]. In 1968, Backster by using polygraph or "lie detector" can measure electrical resistance, and water would alter the resistance of the leaf [9, 10]. Various studies have been carried out on the plant signal and its intelligence. Recently researchers found that the transfer of volatile organic compounds (VOCs) signals among the plants. The signal is released by the emitter plant, and it is transported, absorbed, and perceived by the receiver plant [11]. The fastest methods of long-distance communication between the plant tissues and the organs are bio-electrochemical or electrophysiological signals. The effectiveness of such long-distance communication is clear, since plants can respond to external stimuli (e.g., changes in temperature or osmotic environment, illumination level,

wounding, cutting, mechanical stimulation, or water availability) and changes can be detected in the plant soon after the injury [7, 12]. The velocities of the propagation of electrical signals that have values from 0.5 mm to 4000 mm per second are sufficiently high to facilitate rapid long-distance communication, and these account for the rapid response phenomena observed in plants. The speed of propagation and the amplitude of action potential depend on the type of external stimulus [13]. It is also found that weak electrical signals of the chrysanthemum plant were tested by a touching test system of self-made double shields with platinum sensors [14]. Several studies have reported the effect of different stimuli that induce action potential gradients in plants, mainly light/dark [15, 16] temperature variations [17, 18], intense cold [19, 20], water availability [21, 22], mechanical wounding [23], and insects [22]. Also, it has been suggested that electrical signals could induce genetic programming [24, 25]. After the electrical signal is produced, it is transmitted through the plant to a specific organ or tissue, which generates immediate physiological actions in response to the stimulation [26]. Floranium lamp was developed to measure the voltage signal of the plant [27]. The Daisy team is working on the "The PLANTS" project that will enable the plant to control its own environment [28]. Ivanhenriques developed a Jurema action plant with electrodes clamped to the plant branches [29].

1.1. Types of Signals in the Plant. Two types of signals in the plant have been described: fast signals (action potentials, APs) and slow signals (variation potentials, VPs). A new type of electrical potential signals, called system potentials, has been postulated recently. The novel "system potentials" were detected in five different plant species, among them agricultural crops like tobacco (*Nicotian atabacum*), maize (*Zea mays*), Barley (*Hordeum vulgare*), and field bean (*Vicia faba*) [20]. Action potentials are an electrical waveform that is transmitted along the cell membrane [30], characterized by a response. Plants respond to the environment change according to its amplitude, frequency, and intensity [31, 32]. Thus, action potentials allow cells, tissues, and organs to transmit electrical signals over short and long distances in plants.

Variation potentials propagate in the plant, as temporal changes in the depolarization and repolarization of the cell membrane; this kind of signal varies with the intensity of stimulation and appears to be associated with changes in water tension or ion concentrations, creating a transient electrochemical unbalance in the xylem [8, 33].

1.2. Importance of Wounding Plant Leaf. If plant leaf is wounded, its action potential signal is stronger than that of unwounded leaf as shown in Figure 1 [20]. The strength of the inducing stimulus (wound signal) can influence the frequency when compared to that of systematic signal (unwounded signal).

In this research we are reporting a simple way for measuring the action potentials from the plant which is an approach different from those applied by earlier researchers.

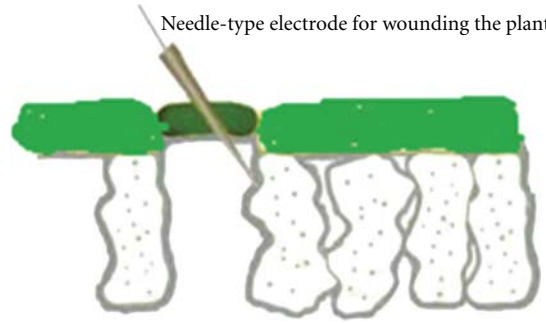


Figure 1: Insertion of electrodes through the stomata (small pores; the leaf is in dark green color) into the inner tissue. Plant electrical signal behavior can be changed by this process.

Figure 2: *descoingsii* x *haworthioides* plant for measuring the action potentials.

The speed of propagation duration and amplitude of action potentials depend on the location of the working electrodes and the reference electrode. The changes in frequency in the plant action potentials when it is exposed to different light modes are observed and used to control the bio-machine.

2. Materials and Methods

2.1. Plant and Laboratory Conditions. Different kinds of plants such as cactus (*Cactaceae*), soya beanplant (*Glycine max*), chrysanthemum (*Dendranthema* x *grandiflorum*), and Pepe (*descoingsii* x *haworthioides*) are purchased from the flower market and cultivated in the lab. It is observed that cactus is not responding and not showing any signal changes to the stimulus. The soybean and chrysanthemum plants are giving signals which are very weak to capture. The leaves of these plants are very thin, and it is difficult to injure them using the needle electrodes. The leaves of the *descoingsii* x *haworthioides* plant are very thick compared to the other plant leaves and sensitive to external stimulation. Hence *descoingsii* x *haworthioides* is selected for further research.

It is well known that plants follow diurnal cycle [15]. We assumed that plants are inactive during night time and rest

(a) (b)

Figure 3: (a) Uni-Patch Tyco EMG electrodes (b) needle-type conductive electrodes.

like human beings, and hence diurnal cycle is not considered during the experiment. Experiments are conducted during morning, afternoon, and evening, and it was found that there is no much variation in the results since the experiments are conducted in an air-conditioned room with almost similar environment conditions for all the light modes.

In Figure 2 *descoingsii* x *haworthioides* plant is grown in a flower pot. The laboratory in which the experiments are carried out is a room with windows, and the temperature is kept at 26°C and the humidity at 75%. The light in the room is turned off at 23:00 hrs, creating total darkness. In the morning once again the plant is exposed to normal room conditions. The plant is watered every alternate days.

2.2. Selection of Electrodes. Electromyography (EMG) electrodes are generally used to detect the electric potentials generated by muscle cells when these cells are electrically or neurologically activated. Two kinds of electrodes are used for detecting the signals coming from the plant.

First one is the Patch EMG electrode and the other is conductive needle-type electrodes. Uni Patch Tyco EMG electrodes which are sensitive and circular in shape with diameter of 20 mm and are used to collect the output signal from the plant leaves are shown in Figure 3(a). Another kind of electrodes used is needle-type conductive electrodes as shown in Figure 3(b), and it is made of copper. The length of the electrode is 350 mm and the edge of the electrode is very sharp. The surface area of the leaf is nearly 2600 mm^2, and the area of electrode is nearly 314 mm^2. The ratio of leaf surface area to electrode area is nearly 8 : 1. Figure 4 shows the connection of these electrodes to the plant. EMG electrode is kept away at a distance of 250 mm from the needle-type electrode to observe the action potentials.

3. Experiments to Measure Electrical Signal from the Plant Leaf

3.1. Experiment I. The objective of this experiment is to verify the behavior of action potentials in the plant. NI LabVIEW is used for analyzing the signals coming from the plant.

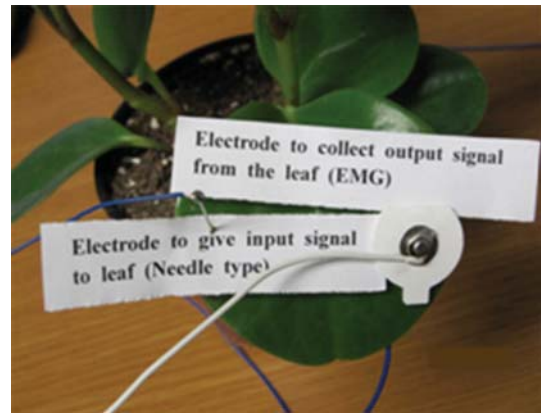


Figure 4: Insertion of EMG electrode and needle-type conductive electrode (by wounding the leaf).

3.1.1. Description of the Circuit. Totally four EMG electrodes and two needle-type conductive electrodes are used for the experiment. Three leaves of *descoingsii* x *haworthioides* plant are used in this experiment. Two leaves are used for the input (sinusoidal) signal from the function generator through two needle-type electrodes. The output signal is collected through two EMG electrodes for detecting the action potential signals from the plant and stored in data acquisition system. The other two EMG electrodes are used as reference electrodes connecting to each leaf. Figure 5 shows the hierarchy of the system for experiment I which shows the block diagram of the system components. The plant receives an input signal from the function generator, and the output signal from the plant is collected by using EMG electrodes and the data is stored in data acquisition system. The frequency of the function generator is chosen based on the action potential signal strength coming from the plant leaf. Figure 6 shows the experimental setup.

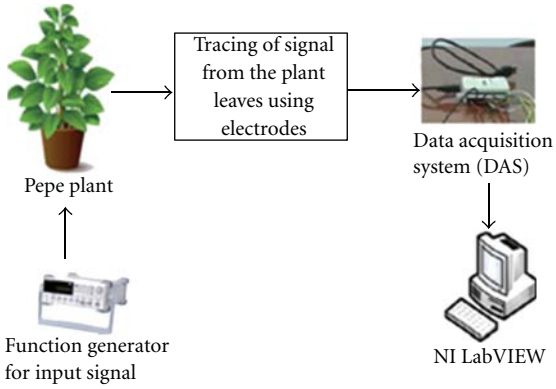


Figure 5: Hierarchy of system for experiment I.

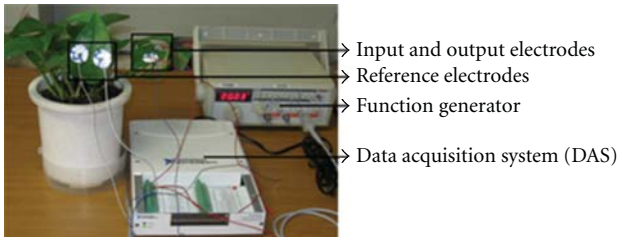


Figure 6: Apparatus for measuring the action potentials from the plant leaves.

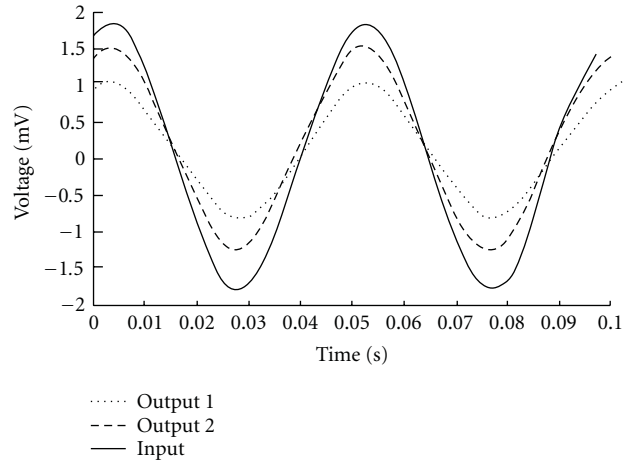


Figure 7: Output signals from the two leaves of the plant for a given sinusoidal input signal of same frequency.

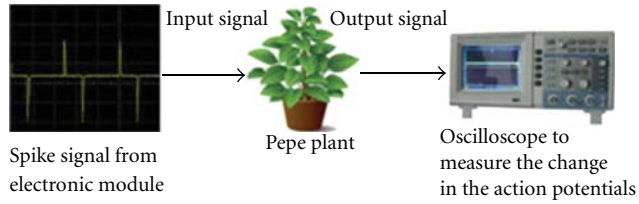


Figure 8: Hierarchy of system for experiment II.

3.1.2. Results and Discussion. The experiments are carried out with different frequencies ranging from 16 Khz to 24 Khz. The output from the EMG electrode is connected to the NI-DAS (data acquisition system). The analog output signal is viewed on the computer screen, and the same is converted to digital data and stored in the DAS. When the sinusoidal signal frequency is at 22 Khz in the function generator, the output signal from the plant has the same nature as that of input signal but with little variation in amplitude as shown in Figure 7. In Figure 7, output signal from the two leaves of the plant is shown for a given input signal of the same frequency. This gives information about the output signal from the plant for a given sinusoidal input signal. Other types of signals such as square and triangle signals from the function generator are also given as input signal to the plant leaf by varying the frequencies, but the output obtained for these types of signals are very weak, and there is no much variation in the amplitude. Hence sinusoidal signal is chosen for this experiment.

3.2. Experiment II. The objective of this experiment is to verify the response of the plant when exposed to different light modes. Figure 8 shows the hierarchy of the system for experiment II.

In this experiment only one leaf is used to measure the signal coming from the plant. One needle-type conductive electrode is used for giving input signal. Two EMG electrodes are used, one for collecting output signal and the other as a reference electrode. In Figure 9, only the input electrode and output electrode are shown, and the reference electrode connected to the ground is attached on the backside of the leaf. The glowing of LED is shown by a rectangular mark.

3.2.1. Description of the Circuit. Differential circuit is constructed by using NE555 timer to generate the spike signals. Figure 10 shows the circuit designed to get the spike signal as plant input signal so that change in the action potential signals from the plant leaf can be seen easily. These signals are very sharp and assist in detecting small changes in the amplitude of output signals. The voltage of input signal is 2.4 mV. The voltage of output signal collected from the plant leaf is 1.2 mV. The input frequency is 23.4 Khz, and the output frequency is 12.24 Khz. These readings are taken from the oscilloscope.

3.2.2. Procedure of the Experiment II. The input signal is given as spike signal from the electronic module shown in Figure 9, and the changes in action potential signals can be seen in oscilloscope.

The plant is exposed to three different light modes. The specification of the lamp used during experiments is GR2001 GRACE BIOLAMP, multifaceted reflector (MR16 HALOGEN LAMP). The brightness of the lamp can be adjusted. The maximum and minimum brightness of the lamp ranges between 560 lumens and 710 lumens, and medium brightness is 625 lumens. The mode of brightness of the table lamp is changed every 20 minutes. There is a

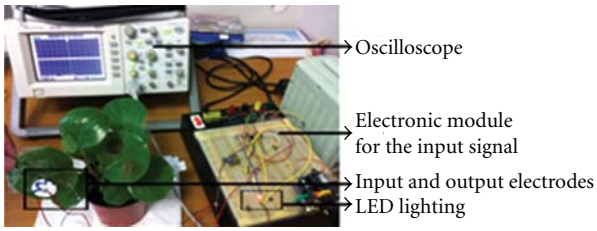


Figure 9: Apparatus for observing the change in action potential when plant is exposed to different light modes.

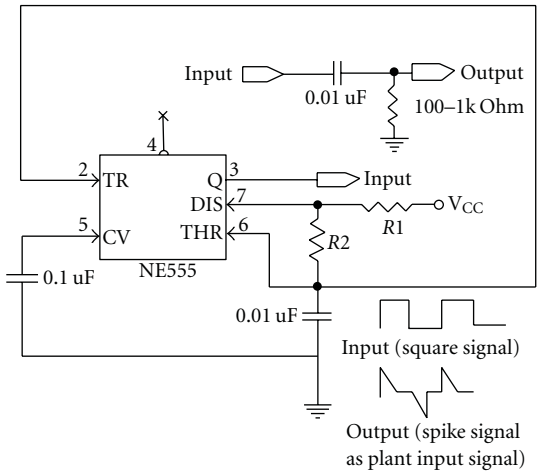


Figure 10: Circuit design for the plant input signal as spike signal.

change in frequency in the plant signal every 5 to 10 sec during each measurement mode. Experiments are carried out for each light intensity mode for about 300 seconds (no light to maximum light brightness condition) after waiting for 30–45 minutes to stabilize the lamp intensity. The change in frequency of the action potential signal is observed from oscilloscope every 5 seconds using stop watch.

3.2.3. Results and Discussion. The observed maximum frequency values of the action potential signals in all the three modes of light are stored in the microcontroller. There is a little noise in the signal due to environmental vibrations. These effects are neglected since all the experiments are carried out in the similar environmental conditions. Figure 11 shows frequency responses of the action potential signals from the plant leaf taking into account all the three modes of light intensities.

There is an overlap in frequency values when there is a change in light intensity from no light to maximum brightness mode. These overlapped frequencies are removed during those particular time intervals by using the control algorithm developed for moving the bio-machine. The bio-machine will move to the right, left or, straight based on the frequency values which are in close agreement with the values recorded in the microcontroller. Flow chart of the bio-machine movement is described in Figure 12.

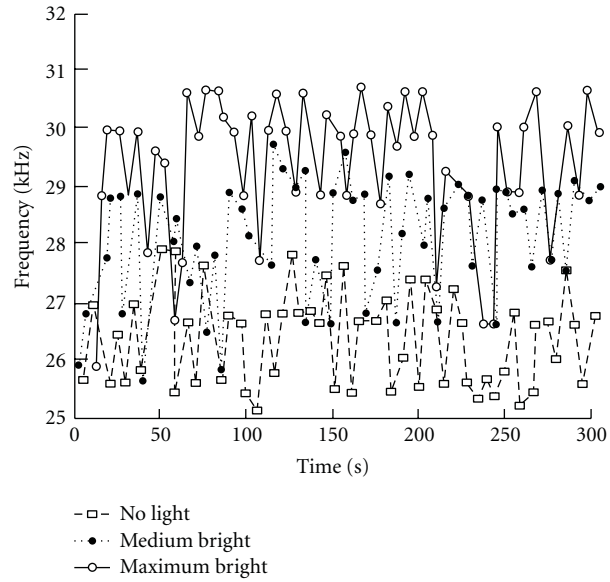


Figure 11: Frequency responses of the action potentials (APs) from the plant leaf at different time intervals when exposed to three modes of light intensities.

3.3. Converting Frequency to Voltage. By using the inverse fast Fourier transform (IFFT) in MATLAB (R2010b), the frequency values are converted to voltages as shown in Figure 13 for the three different light modes. The differences in the amplitudes can be clearly seen at around 150 seconds.

4. Working and Control of Bio-Machine

The change in frequencies when plant is exposed to different light conditions can be used as control signal for the bio-machine as shown in Figure 14. Inverse fast Fourier transform (IFFT) can be used to show dominant amplitudes in the three different light modes. These values are stored in microcontroller. Depending on the light modes, the bio-machine will operate. Frequency counters can be used to see the response of plant to different light conditions, and the frequency can be converted into voltage signal by using the IC, LM2907. The direction change in the bio-machine can be seen by varying the light modes which results in the change in frequency of the action potential signals from the plant.

5. Conclusion and Future Research

In this research a simple method of detecting plant signals is investigated and the method is verified experimentally. The change in frequency levels is observed when the plant is exposed to different light conditions. Applying these results, bio-machine is constructed by designing a circuit and interfaced to a microcontroller. Software is written based on the developed algorithm to move the bio-machine in a desired way. There is a little inconsistency in the movement of the bio-machine due to the random nature of the signals. However, the present work has paved the way for extensive research on the plant intelligence in response to external

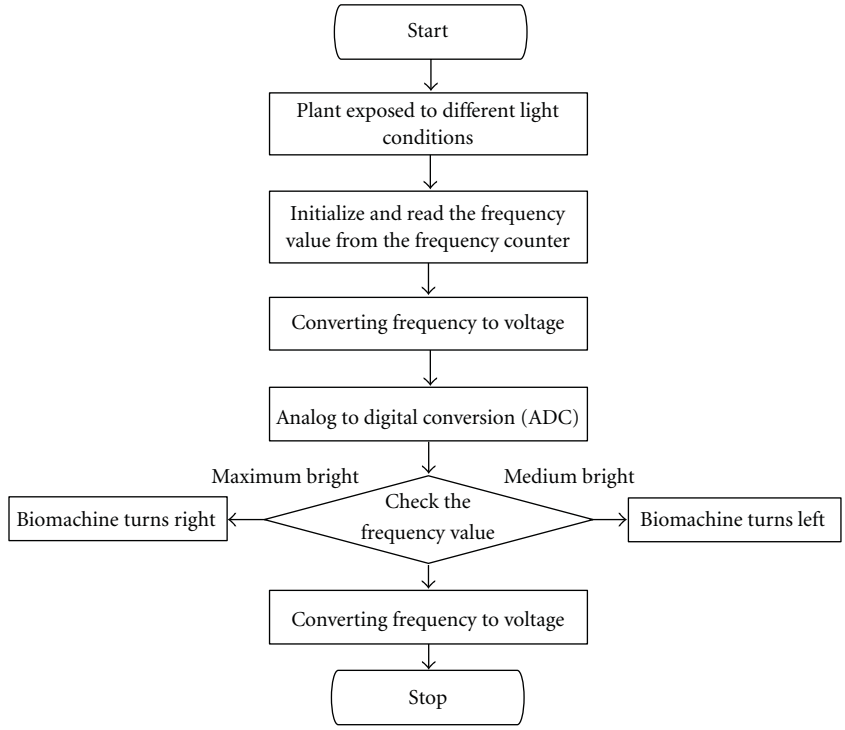


Figure 12: Flow chart for the movement of the bio-machine.

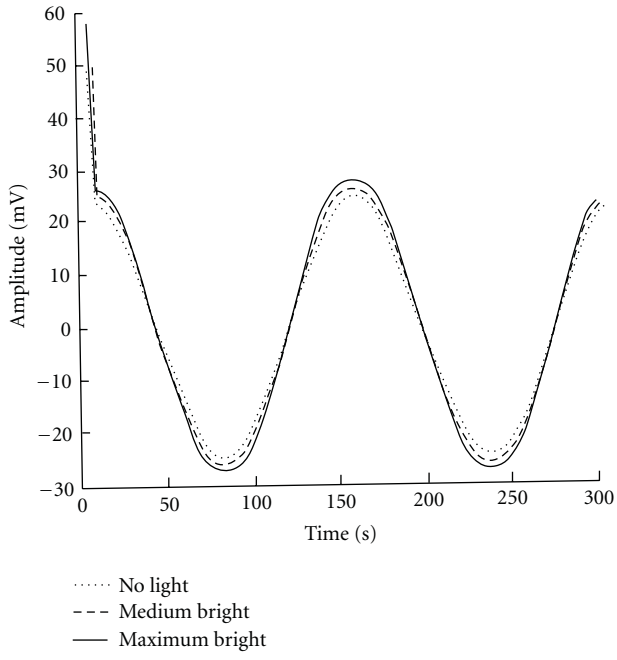


Figure 13: Real part of inverse fast Fourier transform (IFFT) for converting the frequency signal to amplitude (mV).

Figure 14: Prototype of bio-machine.

stimuli as it holds out the true potential for innumerable and very interesting applications. Green plants interfaced with a computer through data acquisition systems can be used as biosensors for monitoring the environment and to detect the effects of pollutants on the plants. This method can also be used for the study of plant electrophysiology. Future studies will be directed towards a better understanding of the plant action potential signals and its response to the other environmental conditions such as weather, presence of sunlight, temperature, use of pesticides and test for the better movement of the bio-machine in response to these environmental conditions.

Acknowledgments

The authors would like to thank the Korean Institute of Science and Technology (KIST) and Amrita Vishwa Vidyapeetham, Amrita School of Engineering, Amritapuri campus, for providing support to carry out the research and experiments.

References

[1] J. Burdon Sanderson, "Note on the electrical phenomena which accompany irritation of the leaf of Dionaea muscipula," *Royal Society of London Proceedings I*, vol. 21, pp. 495–496, 1872.

[2] C. Darwin, *Insectivorous Plants: John Murray*, 1875.

[3] J. Fromm, "Control of phloem unloading by action potentials in Mimosa," *Physiologia Plantarum*, vol. 83, no. 3, pp. 529–533, 1991.

[4] J. Fromm and T. Bauer, "Action potentials in maize sieve tubes change phloem translocation," *Journal of Experimental Botany*, vol. 45, no. 273, pp. 463–469, 1994.

[5] M. Malone and B. Stankovi, "Surface potentials and hydraulic signals in wheat leaves following localized wounding by heat," *Plant, Cell & Environment*, vol. 14, no. 4, pp. 431–436, 1991.

[6] D. Wildon, J. Thain, P. Minchin et al., "Electrical signalling and systematic proteinase inhibitor induction in the wounded plant," *Nature*, vol. 360, no. 6399, pp. 62–65, 1992.

[7] O. S. Ksenzhek and A. G. Volkov, *Plant Energetics*, Academic Press, 1998.

[8] S. J. C. Bose, *The Nervous Mechanism of Plants*, Longmans Green, London, UK, 1926.

[9] C. Backster, "Evidence of a primary perception in plant life," *International Journal of Parapsychology*, vol. 10, no. 4, pp. 329–348, 1968.

[10] S. Lautner, T. E. E. Grams, R. Matyssek, and J. Fromm, "Characteristics of electrical signals in poplar and responses in photosynthesis," *Plant Physiology*, vol. 138, no. 4, pp. 2200–2209, 2005.

[11] I. T. Baldwin, R. Halitschke, A. Paschold, C. C. Von Dahl, and C. A. Preston, "Volatile signaling in plant-plant interactions: "talking trees" in the genomics era," *Science*, vol. 311, no. 5762, pp. 812–815, 2006.

[12] A. G. Volkov and R. A. Haack, "Insect-induced bioelectrochemical signals in potato plants," *Bioelectrochemistry and Bioenergetics*, vol. 37, no. 1, pp. 55–60, 1995.

[13] A. G. Volkov, T. C. Dunkley, A. J. Labady, and C. L. Brown, "Phototropism and electrified interfaces in green plants," *Electrochimica Acta*, vol. 50, no. 21, pp. 4241–4247, 2005.

[14] L. Wang and Q. Li, "ARIMA signals processing of information fusion on the chrysanthemum," *Lecture Notes in Computer Science*, vol. 6319, no. 1, pp. 239–247, 2010.

[15] P. Datta and P. Palit, "Relationship between environmental factors and diurnal variation of bioelectric potentials of an intact jute plant," *Current Science*, vol. 87, no. 5, pp. 680–683, 2004.

[16] L. A. Gurovich and P. Hermosilla, "Electric signalling in fruit trees in response to water applications and light-darkness conditions," *Journal of Plant Physiology*, vol. 166, no. 3, pp. 290–300, 2009.

[17] J. D. Rhodes, J. F. Thain, and D. C. Wildon, "The pathway for systemic electrical signal conduction in the wounded tomato plant," *Planta*, vol. 200, no. 1, pp. 50–57, 1996.

[18] S. Pyatygin, V. Opritov, and V. Vodeneev, "Signaling role of action potential in higher plants," *Russian Journal of Plant Physiology*, vol. 55, no. 2, pp. 285–291, 2008.

[19] E. D. Brenner, R. Stahlberg, S. Mancuso, J. Vivanco, F. Baluska, and E. Van Volkenburgh, "Plant neurobiology: an integrated view of plant signaling," *Trends in Plant Science*, vol. 11, no. 8, pp. 413–419, 2006.

[20] M. R. Zimmermann, H. Maischak, A. Mithöfer, W. Boland, and H. H. Felle, "System potentials, a novel electrical long-distance apoplastic signal in plants, induced by wounding," *Plant Physiology*, vol. 149, no. 3, pp. 1593–1600, 2009.

[21] J. Fromm and H. Fei, "Electrical signaling and gas exchange in maize plants of drying soil," *Plant Science*, vol. 132, no. 2, pp. 203–213, 1998.

[22] T. E. E. Grams, C. Koziolek, S. Lautner, R. Matyssek, and J. Fromm, "Distinct roles of electric and hydraulic signals on the reaction of leaf gas exchange upon re-irrigation in Zea mays L," *Plant, Cell and Environment*, vol. 30, no. 1, pp. 79–84, 2007.

[23] A. Schaller and C. Oecking, "Modulation of plasma membrane H^+-ATPase activity differentially activates wound and pathogen defense responses in tomato plants," *The Plant Cell*, vol. 11, no. 2, pp. 263–272, 1999.

[24] F. Baluška, S. Mancuso, D. Volkmann, and P. Barlow, "Root apices as plant command centres: the unique "brain-like" status of the root apex transition zone," *Biologia*, vol. 59, supplement P, pp. 7–19, 2004.

[25] F. Baluska, D. Volkmann, and D. Menzel, "Plant synapses: actin-based domains for cell-to-cell communication," *Trends in Plant Science*, vol. 10, no. 3, pp. 106–111, 2005.

[26] B. Stanković and E. Davies, "The wound response in tomato involves rapid growth and electrical responses, systemically up-regulated transcription of proteinase inhibitor and calmodulin and down-regulated translation," *Plant and Cell Physiology*, vol. 39, no. 3, pp. 268–274, 1998.

[27] FLORANIUM, "LightArtVision," 2011, http://www.floranium.com/index.php.

[28] "The PLANTS PROJECT," DaisyTeam 2011, http://daisy.cti.gr/plants/.

[29] Ivanhenriques, "JuremaActionPlant," Exhibition Taming Technology, Florence (IT) 2011, http://ivanhenriques.wordpress.com/.

[30] A. G. Volkov, T. C. Dunkley, S. A. Morgan, D. Ruff, Y. L. Boyce, and A. J. Labady, "Bioelectrochemical signaling in green plants induced by photosensory systems," *Bioelectrochemistry*, vol. 63, no. 1-2, pp. 91–94, 2004.

[31] J. Fromm and S. Lautner, "Electrical signals and their physiological significance in plants," *Plant, Cell and Environment*, vol. 30, no. 3, pp. 249–257, 2007.

[32] A. Volkov and C. Brown, "Electrochemistry of plant life," *Plant Electrophysiology*, pp. 437–459, 2006.

[33] J. Mwesigwa, D. J. Collins, and A. G. Volkov, "Electrochemical signaling in green plants: effects of 2, 4-dinitrophenol on variation and action potentials in soybean," *Bioelectrochemistry and Bioenergetics*, vol. 51, no. 2, pp. 201–205, 2000.