ABSTRACT

VANIJJIRATTIKHAN, RANGSARIT. On Network-Based Control and Sensitivity Characterization of Mobile Robot in Intelligent Space. (Under the direction of Dr. Mo-Yuen Chow.)

This dissertation addresses the problem of path-tracking control of a mobile robot, also called an Unmanned Ground Vehicle (UGV), in Intelligent Space, where the controller is located on an entity different from the robot and communicates with the robot over a communication network. The involvement of a communication network leads us to the core of this research, the network time-delay factor. The existence of a network delay presents a challenging problem that might degrade the overall system performance and even destabilize the closed-loop control system.

The existing research area for the aforementioned scenario is called Network-based control system (NBC) mostly focused on a general linear system for which the controller must be redesigned so that the overall NBC system can work properly. Distinct from the existing research, and innovative in its own right, the research presented in this dissertation focuses on a specific nonlinear system, the remote UGV path-tracking. More specifically, we focus on the methods that allow the existing workable path-tracking controller to be reused in the NBC environment.

In this work, Accumulated effect parameter tuning method is firstly proposed to tune the geometrical path-tracking controller used in UGV before operating over communication network; then sensitivity analysis is introduced to consider how the system is sensitive to noise or perturbation so that the operating condition, such as UGV speed and path curvature, may be changed to limit the effect from noise or perturbation; afterwards, Feedback pre-

processor (FP) is proposed to alleviate the effect of network delay by using UGV position estimation through UGV kinematics model; along with FP, UGV response time is proposed to demonstrate the effect of different UGV characteristics on path-tracking performance; finally, the effect of using Gain scheduler (GS) with two-dimensional and one-dimensional gain table is investigated for the capability to alleviate the network delay on remote UGV path-tracking.

On Network-Based Control and Sensitivity Characterization
of Mobile Robot in Intelligent Space

by
Rangsarit Vanijjirattikhan

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Electrical Engineering

Raleigh, North Carolina

2008

APPROVED BY:

_____	_____
Dr. Fen Wu	Dr. Griff L. Bilbro
_____	_____
Dr. James J. Brickley, Jr.	Dr. Mo-Yuen Chow Chair of Advisory Committee

DEDICATION

To my parents, Phitaya and Sirima Vanijjirattikhan, and my wife, Sukwida Manorangsan, for their endless love, support, and encouragement

BIOGRAPHY

Rangsarit Vanijjirattikhan was born in Suphanburi, Thailand. He received his Bachelor of Engineering in Computer Engineering from Kasetsart University, Bangkok, Thailand in 1996. He received his Master of Science in Electrical Engineering from North Carolina State University in 2003. He is currently a Ph.D. candidate at North Carolina State University.

During his study at North Carolina State University, he worked as a teaching assistant for a Mechatronics course. He also worked as a research assistant at Advanced Diagnosis, Automation, and Control lab. His research area is in mobile robot path-tracking and Network-based control system. He is also interested in computer simulation and computer programming. His specialty lies in fast prototyping by integrating computer hardware and software to effectively deliver research results.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Mo-Yuen Chow for his tireless effort teaching me how to do research. Dr. Mo-Yuen Chow has been a great advisor with his care on many aspects of my student life. He also provided me an opportunity to come here to study in USA. I am grateful to Dr. Chow and his kindness of providing me 5 years full financial support. I also would like to thank Dr. James J. Brickley, Dr. Griff L. Bilbro, and Dr. Fen Wu for his time being my committee and useful comments.

I would like to thank Dr. Yodyium Tipsuwan for his friendship and his help during my first two years in the USA. I thank my colleagues at Advanced Diagnosis, Automation, and Control (ADAC) lab for their friendship and their help in several aspects of my study. I would like to make a special mention of Mr. Zheng Li for his insight on sensitivity research, Ms. Rachana A. Gupta for providing technical support at various stages, Mr. Manas Talukdar for proof reading my dissertation, and making very insightful suggestions, Mr. Yixin Cai for contributing to the development of parts of the ADAC iSpace simulation program that I have been using, Mr. Lei Wang , Mr. Unnati Ojha, Ms. Preetika Kulshrestha and Mr. Hong-Bo Li for several helpful comments on my oral exam presentation. I would like to thank my wife, Mrs. Sukwida Manorangsan, for being such a good friend and providing me supports on every aspect of my life. I would like to thank my mother-in-law, Mrs. Siriporn Manorangsan, for her help in taking care of my son.

Last but not least, I would like to thank my parents, Mr. Phitaya and Mrs. Sirima

Vanijjirattikhan, for their unconditional and endless love, support, encouragement, and for taking care of my son during my Ph.D. study.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER II

CHAPTER III

CHAPTER IV

CHAPTER V

LIST OF FIGURES

CHAPTER I

CHAPTER II

CHAPTER III

CHAPTER IV

CHAPTER V

APPENDIX A

APPENDIX B

APPENDIX C

CHAPTER I

INTRODUCTION

Automated Guided Vehicle (AGV) is usually referred to a mobile robot that can automatically move from one place to a desired place without the need for human intervention. There are several applications that emerge with the evolution of the AGV, such as automated delivery system, patrol and security, and house hold application. AGV has gained much attention due to the need for efficient transportation[1], reducing human workload, decreasing cost[2], and enhancing human life style[3]. Some examples of AGV in action are:

1. Fork Lift Vehicle (FLV) AGV from Egemin®[4] that can automatically move the pallet for material handling task in several industries such as automotive, paper & printing, and textile.
2. Mobile robot from KIVA Systems®[5] that can automatically move product inventory in the distribution center such as WalGreens' and Staple's in order to facilitate online order fulfilment.
3. ADAM AGV from RMT Robotics®[6] used in the tire industry to transfer tires between work stations.
4. TransCar AGV from Swisslog®[7] used in hospital that can navigate through multi-floors facilities for material delivery.
5. RoboCleaner from Karcher®, Roomba from iRobot®, and Tribolite from

Electrolux®[3], the autonomous vacuum cleaning robots that can do intelligence planning to traverse the floor area.

6. Automower from Husqvarna®, Robomower from Friendly Robotics®, and Lawnbott from Zucchetti®[3], the autonomous robot that can automatically mow the lawn and come back to the dock station for battery charging without human interaction.

There are several research areas related with the algorithms that control the AGV such as artificial intelligence, path-planning, path-tracking, and collision avoidance ([8] and references therein, [9-11]). As the main purpose of the AGV is to be autonomous, the controller that drives the robot is usually installed locally on the robot itself. This is distinct from the research area in our focus where the mobile robot is controlled by using communication network for data exchange in the control loop. The aforementioned research area can be a basis for a new area of applications because, in several situations of mobile robot path-tracking control, the best decisions can be made from a remote site. For example, in the situation where the AGV is tracking the path in a complex environment (such as a maze), the local sensors and controller on the AGV may not have enough information to guide the AGV to avoid obstruction and go to the destination. The sensors and controller installed somewhere else, such as at the ceiling of the room, can acquire more useful information and guide the robot more efficiently. Or, in the situation that the controller requires computing power such as interpreting human behavior by image processing, then the controller should locate on a computer with high computing power instead of on a small robot. The aforementioned situations are in line with a new application framework called Intelligent Space[12] in which intelligent service is delivered based on global information.

The research presented in this dissertation focuses on the problem of controlling a mobile robot to track a predefined path in which the exchanged data between the path-tracking controller and the mobile robot is conducted over a communication network. This chapter introduces the motivation of our research including the challenge and the literature survey of the existing researches in the same area. This chapter is organized as follows: Section I introduces the concept, structure, and advantage of an application area for remote UGV path-tracking called Intelligent Space; Section II introduces the network delay problem of Network-based control system; Section III provides a survey on the methods proposed for network delay alleviation in NBC system. At the end of this chapter, we provide an outline of the rest of our dissertation.

I. INTELLIGENT SPACE

The existence of computational device and computer systems provide plenty of useful applications such as robots in factory automation[13], computer numerical control milling machine in manufacturing process, automatic suspension control and anti-lock braking system in automobiles[14], mower robot in household application[3], etc. However, current applications of computational devices are mostly limited by their physical location, i.e. the computational device is embedded into a local object such as the robots or cars.

A rapidly evolving and innovative trend in engineering research based applications is to spread computational devices including sensors and actuators into a space and to provide various types of automated services (e.g., making the room "intelligent" for providing nursing services by robots). This space of distributed connection of multiple computational devices, sensors and actuators, or *Intelligent Space*(iSpace), *Smart Space*, *Interactive*

Workspace or *Intelligent Environment*[12,15-17], as it has been called in published literature, is a novel way of providing the aforementioned services using autonomous agents with little or no human control effort involved. We will refer to such a system as iSpace. With advances in technologies such as sensor network, network-based control, mechatronics, miniaturization of sensors and actuators (MEMS, NEMS), distributed control, robotics, etc., iSpace has been rapidly evolving.

To demonstrate some general aspects of the systems mentioned above, system definitions from published literature are described here. In [12], Hashimoto defines intelligent space (iSpace) as "a space where we can easily interact with computers and robots and get useful services from them". In [16], NIST (National Institute of Standards and Technology) defines Smart Space as "a work environment with embedded computers, information appliances, and multi-modal sensors allowing people to perform tasks efficiently by offering unprecedented levels of access to information and assistance from computers." And, in [17], Huang and Trivedi define Intelligent Environments as "systems that are aware of the spatial information and activities within them through sensors and interact with people in a natural and unobtrusive way." All of these definitions emphasize on a natural way of human-machine interaction and providing services according to human's need.

As an example corresponding to the concept, Fig. 1 shows a room with a number of CCD cameras installed to observe the behavior of human beings in the room. The data from the CCD cameras will be processed to extract the useful information in order to assign tasks for the robot to provide services to humans.

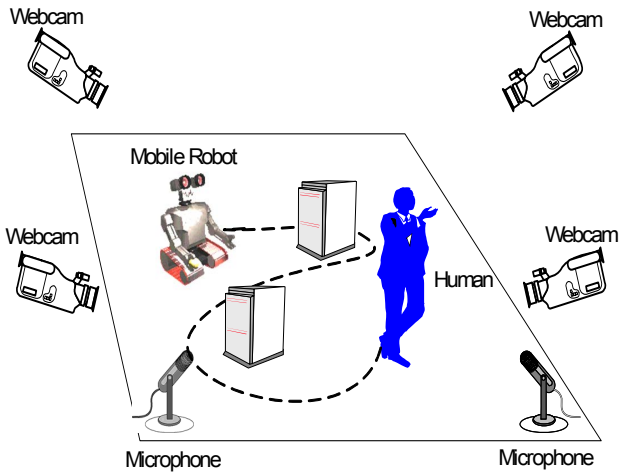

Fig. 1. Human-robot interaction in an iSpace.

This scenario can be applied to a hospital where a patient is having difficulty helping himself. If he is in iSpace, the patient can use verbal or posture command to ask the robots to help him get water and food, or allow the robots to take appropriate action in case of an emergency.

In this manuscript, we adopt the iSpace architecture from the viewpoint of Hashimoto [12] because the structure that had been laid out is simple and covers all the major functionalities of iSpace.

iSpace has 3 main types of components as shown in Fig. 2.

- Control Agent
- Communication Network
- Action Agent (such as robots performing physical services)

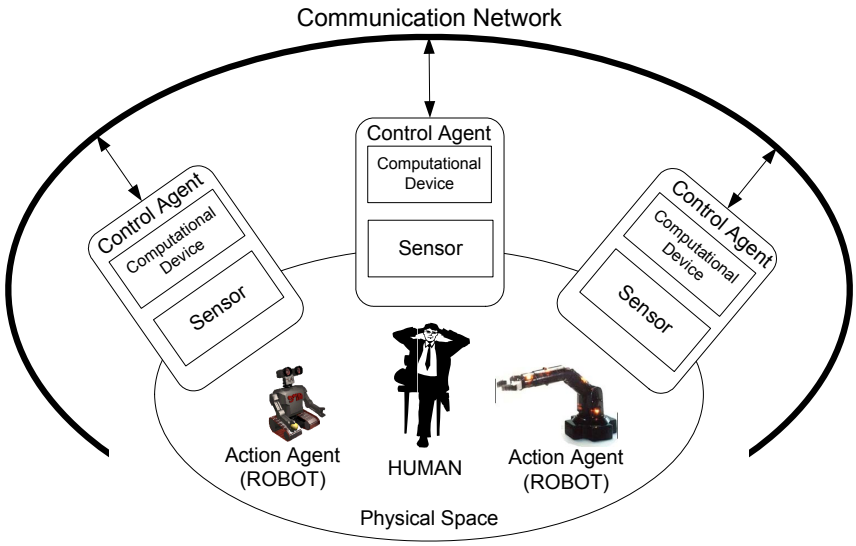

Fig. 2. Intelligent Space.

Control Agent composes of a computational device and sensors. The functionalities of Control Agent are

- Observing events in the space via sensors such as camera, microphone, ultrasonic sensors, etc.
- Processing data and transferring processed data to other Control Agents through computer network in a sensor-independent format for further processing
- Making decisions according to the event observed by itself or by other Control Agents on the network.
- Commanding the Action Agents, e.g. robots, to perform physical actions.

Communication Network (wired or wireless) is used to transfer data among Control Agents and Action Agents.

Action Agent in this manuscript is an actuator that is not connected directly with Control Agents but through the communication network. The main responsibility of the Action Agent

is to execute tasks to support people in the physical space.

In order to provide a service, Control Agents in iSpace make

(1) local decisions based on it's own sensor information or

(2) global decisions based on aggregated information from other Control Agents (collaborative decision and control) as shown in Fig. 3 and Fig. 4 respectively.

As a result, Control Agents can compute distributively when compared to an integrated computational device.

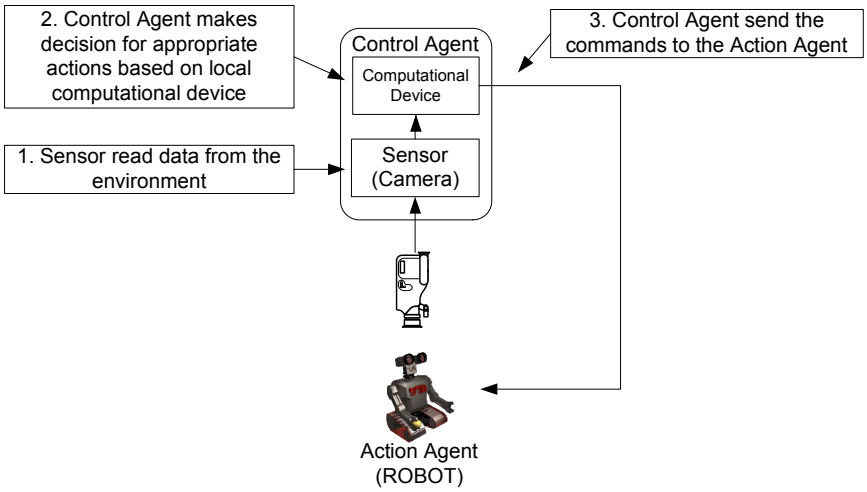

Fig. 3. Local decision making in Control Agent.

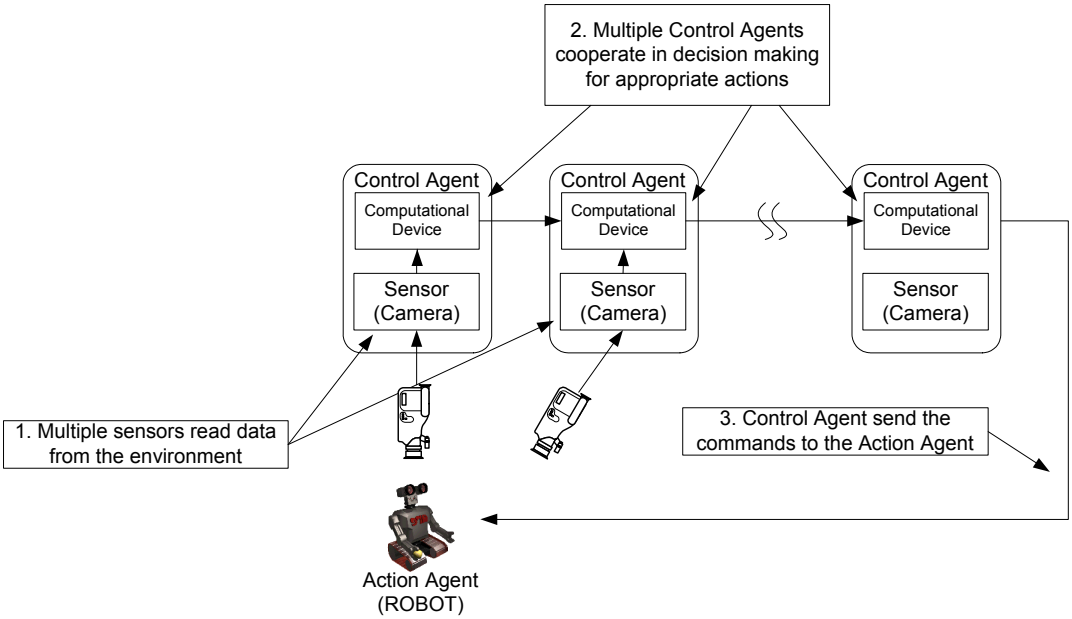

Fig. 4. Global decision making in Control Agents.

Many different research areas such as human-machine interface, human position prediction, face recognition, omnivision sensors, and robot navigation [17-20] are currently involved in investigating and developing iSpace. There are still many issues and open problems that need much more investigation. One important research topic that arises in iSpace is to solve the network delay problem that occurs when remote controllers (control agents) send control signals to the actuators (action agents) through a communication network.

In general, integration/collaboration among computational device, sensors, and actuators to offer a service starts from sensors getting data from the environment. Then, the data is transferred to the computational device for processing and collecting useful information. After that, an appropriate command, according to the information collected, will be sent to the actuators to perform a physical service. The procedure mentioned above requires a certain amount of time for computation and data communication. Network delay can occur while

command/reference signal is sent between computational device, sensors, and actuators. As a result, the actuators' performance (e.g., accuracy, stability, robustness) might degrade. One reason for the performance degradation is the time shift where the actuators receive a control command that is not appropriate for the current time period.

We can formulate this network delay issue in the iSpace as a closed-loop control system operated over a communication network. In the next section, we introduce the problem of control performance degradation that may be caused by network delay.

II. NETWORK-BASED CONTROL SYSTEM AND NETWORK DELAY PROBLEM

Control system can be classified as open-loop control system or closed-loop control system (feedback control system). In general, open-loop control system refers to the control system where the controller ignores the outcome of the controlled process. The controller only uses the signal from the reference input r to determine the control signal u. On the other hand, closed-loop control system or feedback control system refers to the control system that the controller can observe the result of the controlled process (controlled variable, y) in addition to only the objective (reference input, r) available in open-loop control system.

For conventional feedback control system, data transfer between controller and controlled process is direct and immediate. But, in some cases, the connection between controller and controlled process is implemented by a communication network. This can introduce several advantages to the control applications such as reducing investment and maintenance cost for wiring complexity (e.g., in factory automation), enabling teleoperation, and rendering new control concepts and applications [12,16,21-23].

The application of communication networks can vary from short distance data

communication within a single computer to global and massive information sharing over the Internet. Communication networks also take part in control system applications such as the control network in automobiles, teleoperation of robot arm manipulators, and distributed control of multiple robots [21,24,25]. This network-based control (NBC) system, networked control system (NCS), or distributed control system [26-28] is an emerging topic that has gained a lot of attention during the last decade.

There are several structures for the NBC system implementation. Denoting controller by C_n, sensor by S_n, and actuator by A_n, $n = 1, 2, \ldots, N$, a control system with communication network can be configured in a Direct Structure as shown in Fig. 5, or in a Hierarchical Structure as shown in Fig. 6.

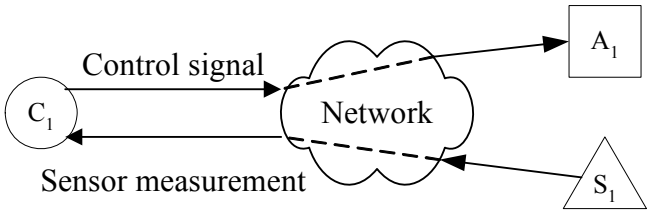

Fig. 5. Simplified diagram of Network-based control system, Direct Structure.

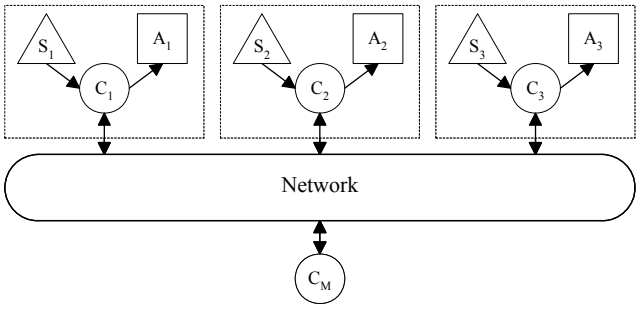

Fig. 6. Distributed control system, Hierarchical Structure.

For the Hierarchical Structure (as shown in Fig. 6), we can see that the controller shares it's responsibilities with the central controller, C_M and the local controllers C_1, C_2, C_3 in each

of the subsystems. An example of the cooperation is that C_m provides reference inputs to C_1, C_2, and C_3 while C_1, C_2, and C_3 control their own actuator according to the received reference input. Because the components of the control system in both Fig. 5 and Fig. 6 are physically decentralized, we also call these kind of systems as distributed control system [29].

In this dissertation, we will focus only on the direct structure which can be represented by the block diagram shown in Fig. 7.

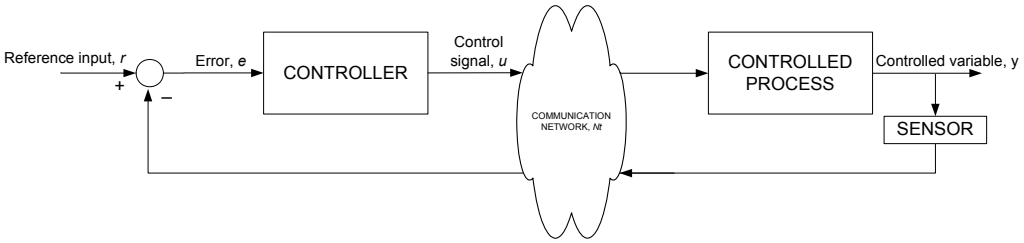

Fig. 7. Network-based control system.

The sequence of operation for network-based control (NBC) system starts from the controller generating control signal (u), from the error (e) between reference input (r) and controlled variable (y) (Fig. 7). The control signal u is sent through the communication network N_t to actuate the controlled process or actuator. According to the system dynamics, the output of controlled process is generated, measured by sensor, and then, sent back to the controller to complete the process cycle. The controller then starts to generate a control signal and sends it to the actuator for the next cycle. Studies show that the network delay induced by the communication network can degrade the performance and reduce the stability region of the NBC system [28,30-38]. The following example is used to illustrate the performance degradation issue in NBC.

We simulate a NBC system composed of an actuator/plant, a controller and communication delays as shown in Fig. 8. The plant is a third-order system described by a transfer function $G(s) = \frac{30}{s(s+3)(s+6)}$. The controller is a proportional controller with gain of one. The reference signal of the feedback control loop is provided by a step function.

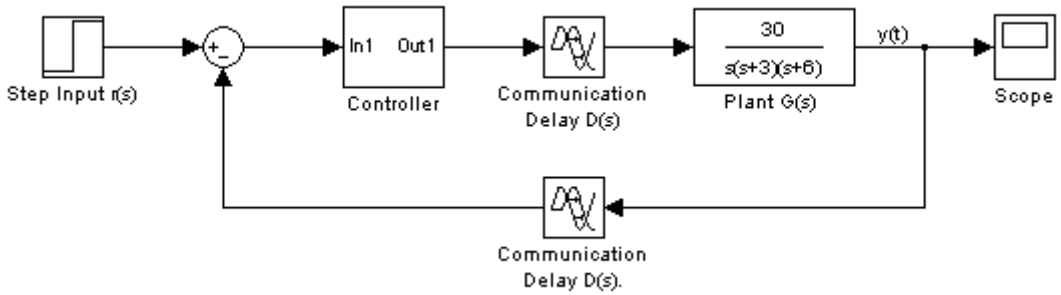

Fig. 8. An example of feedback control system with network delay.

The simulation results of the step response with various constant delays are depicted in Fig. 9. These results demonstrate that increasing delay time also increases the system overshoot and settling time. Hence, the system performance is degraded as the delay time is larger. If delay time is larger than a certain limit, the system can even be unstabilized.

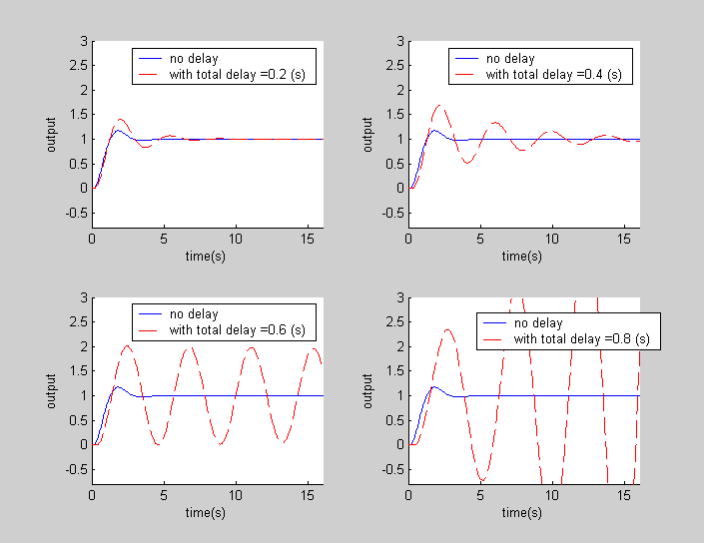

Fig. 9. Step responses of feedback control system with various network delay.

The simulation results in Fig. 9 show that it is important to have the effect of network delay considered when designing a NBC system. In the next section, a survey on existing literature that had been proposed to alleviate the effect of network delay is presented.

III. SURVEY ON NBC SYSTEM

Control applications using communication networks can reduce investment and maintenance cost for wiring complexity (e.g., in factory automation), and render new control concepts and applications such as iSpace[12,18,39-41]. One major area of application of a control system with network is teleoperation, such as the Jason project[42] with an undersea robot exploring the famous sunken ship Titanic, the Mercury project[23] with the first on-line robotics arm to explore the terrestrial surfaces, the Xavier project[43] with the first on-line mobile robot moving to a destination preset by the remote user, the USAR (Urban Search And Rescue) mobile robots [44] with the ability to survey the environment that is dangerous for human, the daVinci surgical system[21] with the tools that allow surgeons to perform

minimal invasive surgery, and NASA's Mars Exploration Rovers[22] with the ability to survey Martian terrain.

Among all the control systems with network, one major challenge is the existence of a network delay that might degrade the overall system performance and even destabilize the closed-loop control system in the network.

There exist several approaches to alleviate the effect of delay time. As in teleoperation where the human operator can use a master robot to remotely control a slave robot located far away, the simplest method is to use human intuitive to learn about the effect of network delay and perform "move and wait", in which the operator slightly moves the master robot at a time [45]. Then, the operator waits to observe the slave robot to see if the movement is under control. In this way, the operator can get some knowledge of the delay effect and be able to control the master robot to the extent that the slave robot is still working as expected. Another method to deal with the network delay is "teleprogramming" where there exists an extra controller at the remote plant as shown by the hierarchical structure in Fig. 6 [46]. The central controller (human or a program) needs to send only a command or a task to the remote plant. Then, that extra controller will take responsibility to finish the task and network delay will not have any effect on this local operation. This method presents a solution of the communication delay problem in a system in which the delay does not need to be directly considered. As a result, the system tends to be less flexible and less interactive compared to the system that directly uses the information of the delay time and has the feedback loop span over the communication network. This aspect is the primary focus of the research presented in this dissertation.

One well-known direction in dealing with communication delay in NBC is by using passivity theorem. Along this line, Anderson and Spong [31] used the power variable in force feedback teleoperation to successfully stabilize the NBC system in passive environment under an arbitrary size of the delay. Niemeyer and Slotine [47] introduced the wave variable instead of power variable for information exchange in force feedback teleoperation. In addition to robustness and system stability to arbitrary network delay, wave variable has the benefit to reduce the complexity of the system implementation. In [33,48], Hannaford and Ryu harnessed the concept of passivity to enhance the haptic interface system which has the information flow in two directions, like force feedback teleoperation system. They used a real-time energy measurement module and adaptive energy dissipative element in order to interactively control the net energy of the system so that the stability of the system is achieved with less conservative than a rigid passivity based design.

Another direction to address the NBC problem is focusing at the communication network to derive a suitable network characteristic such as the communication protocol, scheduling algorithm, transmission rate, and bandwidth. In [32,49] Halevi and Ray proposed an analytical technique to test the stability of a specific NBC system whose communication delay can be assumed periodic. Their approach can be applied to a token passing network, such as token ring. They also proposed a time skew varying method to reduce the vacant sampling and message rejection of the data packet transmitted between the controller and the plant through a communication network. This method will help the controller to have higher probability of getting the most updated data from the sensor and eventually improve the system performance. In [30], Walsh et al. proposed a novel control network protocol, try-

once-discard (TOD), for multiple-input-multiple-output (MIMO) NBC system. TOD allows the node that has the highest priority to get the transmission while ensuring the maximum allowable transfer interval of every node to guarantee stability. In [34], Zhang et al. studied the relationship between the network round-trip-time delay (RTT), controller sampling period, and the stability of a system. They proposed an analytical method to find a stability region given controller sampling period and RTT for a specific case of the system. Their results also include the stability condition of a NBC hybrid system, system with packet dropouts, and multiple-packet transmission. In [28] Lian et al., proposed a guideline to find a relationship between the control quality of performance (QoP) and the network and control parameters. They provided a method to choose a proper sampling period so that the rate of the data exchange between the controller and the plant is high enough to maintain an acceptable performance but not too high until the network is congested.

Another direction to deal with network delay in NBC system is by using techniques based on soft computing such as Neural network and Fuzzy logic. In [38] Almutairi et al. uses fuzzy logic compensation method to compensate network delay effect by externally updated PI controller gain with respect to the system output error caused by network delay. In [50], Lee et al. proposed the remote fuzzy logic controller method in which a typical PID controller is replaced by a fuzzy logic controller. For both methods, the expert knowledge can be embedded within the fuzzy rules to calculate the control signal based on the system error in order to alleviate the effect of delay. On the other hand, Neural network (NN) does not need the expert knowledge but uses learning capability to cope with the delay effect. In [51], Huang and Lewis uses NN to extend the concept of Smith predictor to the nonlinear

system control under constant delay. Smith predictor is a famous method to control a system with delay using an accurate model-based prediction [52]. Huang and Lewis propose to use recurrent neural network to compensate the nonlinear effect of the plant so that the plant can be treated as a linear system to be controlled using Smith predictor.

Several researchers approached the network delay problem by considering model-based predictor. In [53], Luck and Ray use observer-based compensator to alleviate the effect of network delay. They use observer to determine the state of the plant from the plant output. Then, based on the plant model, the compensator can estimate the current plant state from the delayed plant state so that the controller will get the most updated state to calculate the control signal. In [54], Chan and Ozguner used the estimated state of the plant based on the transmission queue length information embedded in the data packet before sending it from the plant. The queue length can determine how many time periods have passed after the plant output is measured. Then, a model-based predictor can predict the state of the plant so that the control signal can be prepared accordingly. In [37], Montestruque and Antsaklis use model-based prediction to predict the future state of the linear plant. A stability condition is also provided so that the maximum period of two consecutive data transmissions can be determined to reduce the network load while the system remains stable.

The stability of the NBC system is also studied using the counterpart of the Lyapunov stability condition, Lyapunov-Krasovskii and Razumikhin stability condition. There is emerging literature focussing in this direction such as [36,55] and the references therein. For a linear NBC system in general, the stability can be tested by formulating the L-K or Razumikhin stability condition into convex optimization problem. The problem is

represented by Linear Matrix Inequalities which can be efficiently solved by existing algorithm [56].

Optimal stochastic control and robust control have also been reported to have the capability to alleviate the network delay problem. In [35], Nilsson proposed an optimal stochastic control method to control a NBC system in which the random round-trip-time delay is less than the sensor sampling period. This method can derive an optimal control signal subjected to minimizing a predefined cost function. In [57], Shousong and Qixin formulate a NBC system so that the optimal stochastic control framework can be applied to the NBC system where the delay time is larger than the controller sampling period. In [58,59], robust control framework is used to alleviate the effect of delay time. Random delay is treated as bounded system uncertainties and the robust controller is designed for all system variation under uncertainties to ensure a predefined limit of the error amplification (i.e., the plant output can track the reference signal within a bound). In [60], Kyoo Kim et al. formulate a NBC system into several patterns according to the history of the delay. Several H_∞ controllers are designed to form a switched H_∞ controller in order to ensure a bounded error amplification corresponding to the pattern of the delay history for each time instant.

The aforementioned researches mostly focus on a general linear system for which the controller must be redesigned so that the overall NBC system can work properly. Distinct from the existing research, and innovative in its own right, the research presented in this dissertation focuses on a specific nonlinear system, the remote UGV path-tracking. More specifically, we focus on the methods that allow the existing workable path-tracking controller to be reused in NBC environment. Our methods use additional modules embedded

into the existing system so that the outputs from the UGV is preprocessed before using them in the control signal calculation and the control signals are adjusted appropriately for current network traffic condition and path-tracking environment.

The succeeding chapters are organized as follows. In chapter II, we focus on the parameters tuning of path-tracking algorithm as it is important that the path-tracking parameters should be finely tuned before operating in NBC environment. Chapter III focuses on how the path-tracking error is sensitive to the UGV speed, path curvature, and magnitude of network delay so that we can carefully compensate when those signals have noise or perturbation. Chapter IV focuses on the Feedback Preprocessor module to preprocess the sampled data from the UGV before using them in path-tracking controller in order to alleviate the effect of network delay. Chapter IV introduces the concept of UGV response time to predict the level of performance degradation caused by network delay. Finally, Chapter V focuses on applying Gain Scheduler Middleware with a variation of gain table to alleviate the effect of network delay in Intelligent Space by externally adjusting the path-tracking controller gain.

REFERENCES

[1] J. Zhang, P. A. Ioannou, and A. Chassiakos, "Automated container transport system between inland port and terminals," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 16, pp. 95-118, 2006.

[2] P. A. Ioannou*, et al.*, "Advanced Material Handling: Automated Guided Vehicles in Agile Ports," Center for Advanced Transportation Technologies, Univ. Southern California, Los Angeles, 2000.

[3] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing [Applications of control]," *IEEE Control Systems Magazine*, vol. 27, pp. 20-96, 2007.

[4] http://www.egeminusa.com/.

[5] http://www.kivasystems.com/.

[6] http://www.adam-i-agv.com/.

[7] http://www.swisslog.com/.

[8] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, pp. 677-709, 2006.

[9] K. Yoshizawa*, et al.*, "Path tracking control of mobile robots using a quadratic curve," Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 1996.

[10] O. Amidi, "Integrated Mobile Robot Control, tech. report CMU-RI-TR-90-17," Robotics Institute, Carnegie Mellon University 1990.

[11] J. Wit, C. D. Crane III, and D. Armstrong, "Autonomous ground vehicle path tracking," *Journal of Robotic Systems*, vol. 21, pp. 439-449, 2004.

[12] H. Hashimoto, "Intelligent space - How to make spaces intelligent by using DIND?," Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia, 2002.

[13] M. Mes, M. van der Heijden, and J. van Hillegersberg, "Design choices for agent-based control of AGVs in the dough making process," *Decision Support Systems*, vol. 44, pp. 983-999, 2008.

[14] R. Isermann, R. Schwarz, and S. Stolzl, "Fault-tolerant drive-by-wire systems," *IEEE Control Systems Magazine*, vol. 22, pp. 64-81, 2002.

[15] B. F. Johanson, A.; Winograd, T., "The Interactive Workspaces project: experiences with ubiquitous computing rooms," *Pervasive Computing, IEEE*, vol. 1, pp. 67-74, 2002.

[16] L. S. Rosenthal, V., "NIST Smart Space: pervasive computing initiative," Proeedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.

[17] K. S. Huang and M. M. Trivedi, "Networked omnivision arrays for intelligent environment," Proceedings of Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation IV, San Diego, CA, 2001.

[18] P. T. Szemes, T. Sasaki, and H. Hashimoto, "Mobile agent in the intelligent space which can learn human walking behavior," Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2005.

[19] S. K. Das*, et al.*, "The role of prediction algorithms in the MavHome smart home architecture," *IEEE Wireless Communications*, vol. 9, pp. 77-84, 2002.

[20] K. Morioka*, et al.*, "Robust tracking of multiple objects using color histogram in intelligent environment," Proceedings of the 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2003.

[21] J. Leven*, et al.*, "DaVinci Canvas: A Telerobotic Surgical System with Integrated, Robot-Assisted, Laparoscopic Ultrasound Capability," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2005, pp. 811-818.

[22] J. J. Biesiadecki, P. C. Leger, and M. W. Maimone, "Tradeoffs Between Directed and Autonomous Driving on the Mars Exploration Rovers," *The International Journal of Robotics Research*, vol. 26, pp. 91-104, 2007.

[23] K. Goldberg and R. Siegwart, *Beyond webcams : an introduction to online robots*. Cambridge, MA: MIT Press, 2002.

[24] N. Navet*, et al.*, "Trends in Automotive Communication Systems," *Proceedings of the IEEE*, vol. 93, pp. 1204-1223, 2005.

[25] A. Al-Jumaily and S. Kozak, "Behavior based multi robot cooperation by target/task negotiation," Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, 2004.

[26] Y. Tipsuwan and M.-Y. Chow, "Control Methodologies in Networked Control Systems," *Control Engineering Practice*, vol. 11, pp. 1099-1111, 2003.

[27] G. C. Walsh and H. Ye, "Scheduling of networked control systems," in *IEEE Control Systems Magazine*, vol. 21, 2001, pp. 57-65.

[28] F.-L. Lian, J. Moyne, and D. Tilbury, "Network design consideration for distributed control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, pp. 297-307, 2002.

[29] R. Uusijävri and M. Törngren, "Introducing Distributed Control in Mobile Machines Based on Hydraulic Actuators," *Journal of Mechatronics*, vol. 4, pp. 139-157, 1994.

[30] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, pp. 438-446, 2002.

[31] R. J. Anderson and M. W. Spong, "Bilateral control of teleoperators with time delay.," *IEEE Transactions on Automatic Control*, vol. 34, pp. 494-501, 1989.

[32] Y. Halevi and A. Ray, "Integrated communication and control systems : Part I - Analysis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 367-373, 1988.

[33] B. Hannaford and J.-H. Ryu, "Time-domain passivity control of haptic interfaces," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 1-10, 2002.

[34] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, pp. 84-99, 2001.

[35] J. Nilsson, "Real-time control systems with delays," in *Department of Automatic Control*. Lund, Sweden: Lund Institute of Technology, 1998, pp. 138.

[36] K. Gu, V. Kharitonov, and J. Chen, *Stability of time-delay systems*. Boston [Mass.]: Birkhèauser, 2003.

[37] L. A. Montestruque and P. J. Antsaklis, "On the model-based control of networked systems," *Automatica*, vol. 39, pp. 1837-1843, 2003.

[38] N. B. Almutairi and M.-Y. Chow, "Stabilization of Networked PI Control System Using Fuzzy Logic Modulation," Proceedings of the American Control Conference, Denver, Colorado, 2003.

[39] W.-L. Leung*, et al.*, "Intelligent space with time sensitive applications," Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, California USA, 2005.

[40] J.-H. Lee and H. Hashimoto, "Intelligent Space - Its concept and contents," *Advanced Robotics Journal*, vol. 16, pp. 265-280, 2002.

[41] R. Vanijjirattikhan*, et al.*, "Mobile Agent Gain Scheduler Control in Inter-Continental Intelligent Space," Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA05), Barcelona, Spain, 2005.

[42] http://www.jasonproject.org/.

[43] R. Simmons, *et al.*, "Lessons learned from Xavier," *IEEE Robotics & Automation Magazine*, vol. 7, pp. 33-39, 2000.

[44] R. R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 34, pp. 138-153, 2004.

[45] T. B. Sheridan, "Space teleoperation through time delay: review and prognosis," *Robotics and Automation, IEEE Transactions on*, vol. 9, pp. 592-606, 1993.

[46] J. Funda, T. S. Lindsay, and R. P. Paul, "Teleprogramming: toward delay-invariant remote manipulation," *Presence: Teleoperators and Virtual Environments*, vol. 1, pp. 29-44, 1992.

[47] G. Niemeyer and J.-J. E. Slotine, "Telemanipulation with Time Delays," *The International Journal of Robotics Research*, vol. 23, pp. 873-890, 2004.

[48] J.-H. Ryu, D.-S. Kwon, and B. Hannaford, "Stable teleoperation with time-domain passivity control," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 365-373, 2004.

[49] A. Ray and Y. Halevi, "Intergrated communication and control systems: Part II - Design considerations," *Journal of Dynamic Systems, Measurement, and Control*, vol. 110, pp. 374-381, 1988.

[50] S. Lee, S. H. Lee, and K. C. Lee, "Remote fuzzy logic control for networked control system," Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society (IECON '01), 2001.

[51] J.-Q. Huang and F. L. Lewis, "Neural-network predictive control for nonlinear dynamic systems with time-delay," *IEEE Transactions on Neural Networks*, vol. 14, pp. 377-389, 2003.

[52] O. J. M. Smith, "A controller to overcome dead-time," *Instrument Society of America Journal*, vol. 6, pp. 28-33, 1959.

[53] R. Luck and A. Ray, "An observer-based compensator for distributed delays," *Automatica*, vol. 26, pp. 903-908, 1990.

[54] H. Chan and U. Ozguner, "Closed-loop control of systems over a communications network with queues," *International Journal of Control*, vol. 62, pp. 493-510, 1995.

[55] E. Fridman, "New Lyapunov-Krasovskii functionals for stability of linear retarded and neutral type systems," *Systems & Control Letters*, vol. 43, pp. 309-319, 2001.

[56] S. P. Boyd, *Linear matrix inequalities in system and control theory*. Philadelphia: Society for Industrial and Applied Mathematics, 1994.

[57] H. Shousong and Z. Qixin, "Stochastic optimal control and analysis of stability of networked control systems with long delay," *Automatica*, vol. 39, pp. 1877-1884, 2003.

[58] G. M. H. Leung, B. A. Francis, and J. Apkarian, "Bilateral controller for teleoperators with time delay via mu-synthesis," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 105-116, 1995.

[59] F. Goktas, J. M. Smith, and R. Bajcsy, "mu-synthesis for distributed control systems with network-induced delays," Proceedings of the 35th IEEE Decision and Control, 1996.

[60] D. Kyoo Kim, P. G. P. Park, and J. W. Ko, "Output-feedback H-infinity control of systems over communication networks using a deterministic switching system approach," *Automatica*, vol. 40, pp. 1205-1212, 2004.

CHAPTER II

ACCUMULATED EFFECT PARAMETER TUNING METHOD FOR GEOMETRICAL PATH TRACKING OF WHEELED MOBILE ROBOTS

Rangsarit Vanijjirattikhan, Manas Talukdar, Mo-Yuen Chow

rvanijj, mtalukd, chow@ncsu.edu

Advanced Diagnosis Automation and Control Lab

Department of Electrical and Computer Engineering

North Carolina State University, Raleigh NC 27695, USA

This chapter is published in the proceedings of the 2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2007), Zürich, Switzerland, 2007.

ACCUMULATED EFFECT PARAMETER TUNING METHOD FOR GEOMETRICAL PATH TRACKING OF WHEELED MOBILE ROBOTS

Abstract - This paper proposes a novel method for parameter tuning in geometrical path tracking algorithms for wheeled mobile robots. Geometrical path tracking has the advantage in its ease of implementation. However, before using such an algorithm, several critical parameters need to be found experimentally. Such a procedure is quite time consuming and unreliable. Moreover, there is no certainty that the parameters found using this method will hold for different kinds of paths. The research presented in this paper addresses this particular issue. A tuning method called Accumulated Effect (AE) parameter tuning method is proposed for determining the appropriate parameter values which will hold for different paths under different conditions. This method is based on the steady state of a mobile robot, tracking a circular path. The effectiveness of this method has been demonstrated using simulation results for three different path tracking algorithms.

Keywords: *mobile robot, parameter tuning, path tracking*

I. INTRODUCTION

Path tracking of wheeled mobile robot has gained substantial research attention in last decade due to extensive applications of wheeled mobile robot such as automatic inventory system in factory and warehouse, innovative applications such as Intelligent Space [1, 2] and

Network-based control applications [3, 4].

There are three basic problems related with the path tracking of wheeled mobile robot. We can categorize the existing literature based on these three problems:

- The first problem is *path following* where there exists a path known by the controller and the mobile robot needs to follow the path.
- The second problem is *trajectory tracking* where there exists a virtual reference robot moving along the desired path and the actual mobile robot needs to be controlled to converge to the reference robot.
- The third problem is *set point stabilization* where there exists a final posture where the robot at an initial posture is required to converge to.

Ideally, a path tracking algorithm should address all of these problems. Most existing algorithms address one or a combination of these problems. Some existing works are geometry based path tracking controller proposed by Amidi [5], Yoshizawa [6], and Wit [7], which can address the path following problem and the trajectory tracking problem. The kinematics model based stable non-linear path tracking controller proposed by Kanayama et al. [8] and adaptive fuzzy logic-based controller by Das and Kar [9], addresses the trajectory tracking problem. Globally stabilizing time-varying feedback approach proposed by Samson [10] and dynamic feedback linearization approach proposed by Oriolo et al. [11], addresses both the trajectory tracking and set point stabilization. Robust adaptive controller based on neural network and backstepping paradigm by Fierro and Lewis [12], addresses all of the problems. Each of these approaches has their own nuances. For example, [8, 10, 11] consider only the kinematics model to derive stabilizing nonlinear controller. [9, 12] considers both

the kinematics and dynamics for controller design. [9, 12] also has the capability to adapt the controller based on model uncertainty. [5, 6] considers only the position error between the robot and the path/trajectory while the other algorithms also consider the heading error.

In this paper, we focus on the path following problem implemented by geometrical algorithms [5-7]. These algorithms are relatively easy to implement after the information of the path is known since there is no need to generate the trajectory of the virtual reference robot as aforementioned in the trajectory tracking problem. However, using these geometrical algorithms require tuning of some parameters which are typically done by experiment and can be time consuming. Therefore, we propose a parameter tuning method called *Accumulated effect (AE)* parameter tuning approach based on the numerical result of the steady state, while the robot tracks a constant circular path. A bound of path deviation error is used in the tuning process so that the robot using the tuned parameter has path-deviation error approximately within the assigned bound.

The parameter tuning for geometrical path tracking is a vital issue which merits meticulous research. There is an existing work [13] that addresses the look-ahead distance for a stable pure pursuit path tracking system. However, there is no analysis on the path tracking performance. Some other literature [9, 12] address the issue of the unknown robot dynamics parameter, but up to our awareness, there has been no specific detailed study of geometrical path tracking parameter tuning.

This paper is organized as follows: in section II, the system setup is described, section III discusses the parameter tuning method of geometrical path tracking algorithm, section IV discusses the simulation results, and the paper is concluded in section V.

II. SYSTEM SETUP

In this section, we give an overview of the unmanned ground vehicle (UGV) path tracking system used in this research. The focus here will be on path following problem where the path can be varied according to the environment and the information of the path is known by the path tracking controller (e.g., UGV for services in hospitals, offices, homes, or industrial plants [2]).

A. Uumanned ground vehicle model

In this research, we considered the path tracking of a differential drive mobile robot with two driving wheel and one or more caster wheels. The UGV can turn by driving the wheels with two different speeds. The center of turn is treated as the point-wise position of the UGV for path tracking error calculation. The center of turn can be different from the center of gravity (CG) because the position of the wheel may be away from the most density part of the mass.

The model of the UGV is used to describe the behavior of the UGV according to the control signal which is reference speed and turn rate, $\mathbf{u} = [v_{ref}\ \omega_{ref}]^T$. This UGV model is composed of the UGV kinematics & dynamics, the motor dynamics, and the motor speed controllers. The simulation in this paper is based on the nonholonomic UGV kinematics & dynamics described in [14] and the DC motor and gear train model described in [15].

B. Path tracking algorithms

In this paper, we focus on three geometrical path tracking algorithms, Vector Pursuit (VP) [7], Quadratic Curve (QC) Path Tracking [6], and Pure Pursuit (PP) [5]. These algorithms can generate control signals such as the UGV speed and turning rate in order to

drive the UGV along a predefined path.

In order to track the path, the UGV needs to target one reference point on the path at a time and steer towards that point. The reference point $\mathbf{x}_{ref}(t) = [x_{ref}\ y_{ref}]^T$ will lead the UGV and keep a distance, called *look-ahead distance* (d_0), ahead of the UGV as shown in Fig. 1. As the UGV follows the reference point and the point moves along the path, the UGV eventually tracks the predefined path.

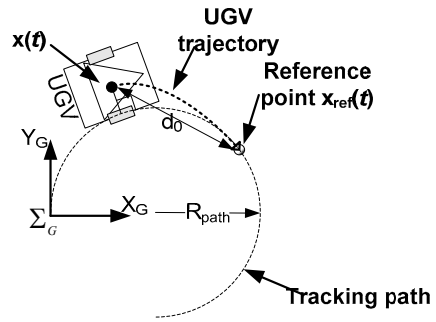

Fig. 1. UGV path tracking using geometrical algorithm.

d_0 can be a fixed constant as in PP and VP or a function of the tracking condition as in QC. In QC, the look-ahead distance is adjusted based on the curvature of the path, using the quadratic coefficient (A), as calculated by Eqn. (1) where d_{max} and β are constants, A is equal to $1/2R_{UGV}$.

$$d_0 = d_{\max}/(1+\beta|A|) \tag{1}$$

For all three algorithms, after the path tracking controller knows where the reference point in each calculation step is, the UGV needs to find the speed gain K and the circular radius of the UGV trajectory R_{UGV} in order to calculate v_{ref} and ω_{ref} according to Eqn. (2)-(3)

$$v_{ref} = K \tag{2}$$

$$\omega_{ref} = K / R_{UGV} \tag{3}$$

where v_{ref} is the speed of the UGV in m/s and ω_{ref} is the turn rate of the UGV describing how the heading angle changes in rad/s. By using the control signal from Eqn. (2)-(3), the UGV will run along a circle path radius R_{UGV} with the speed K m/s. In VP and PP, v_{ref} is typically kept constant. However, in QC, v_{ref} is changed according to the path curvature as shown in Eqn. (4)

$$v_{ref} = sign(e_x)\alpha / (1 + |A|) \tag{4}$$

where α is a constant, $sign(e_x)$ is 1 if the reference point is ahead of the UGV and -1 if it is behind the UGV. For the detailed calculation of K and R_{UGV} for VP, QC, and PP, please refer to [5-7].

III. PARAMETER TUNNING FOR GEOMETRICAL PATH TRACKING ALGORITHM

There are several constant values that appear in the path tracking algorithms such as d_0 and K in VP and PP and d_{max}, β and α in QC. Note that d_{max} and β in QC is used to calculated d_0 which is required to find the reference point the same way as in PP and VP. All of these constants are the parameters that must be carefully tuned because of their critical effect on path tracking performance. For example, if d_0 is too large the robot can take a shortcut while tracking the path and cause a large error. If d_0 is too small, the robot may oscillate around the path and take a long time to reach the destination.

Tuning method provided in this section will be a guideline for the eligible parameters to

be easily selected. At the end of this section, we will provide a data table for parameter lookup based on the characteristics of the robot such as the speed v_{ref} and the controller sampling period *T*. This can simplify the implementation process of the path tracking algorithm. In this paper, we first discuss tuning d_0, and then tuning d_{max}, β.

A. Look-ahead distance (d_0) tuning

Tuning for a suitable look-ahead distance d_0 for using in the path tracking algorithm is not an obvious task since there are a lot of factors we need to consider such as the initial position and heading of the UGV, the speed of the UGV, the sampling rate of the UGV, and the shape of the path. Ideally we need to define a mapping function between d_0 and the path tracking performance so that we can adjust d_0 to tune for the best path tracking performance using the least possible information.

Our method for tuning d_0 is called *Accumulated effect (AE) parameter tuning* approach. The approach is based on an observation of the UGV while tracking a fixed radius circular path. Given enough time, the UGV tends to converge to a *steady state* where the distance between the UGV and the path is constant. We call this distance as the *path-deviation error at steady state* or steady state error where the distance between the robot and the path is constant as shown in Fig. 2.

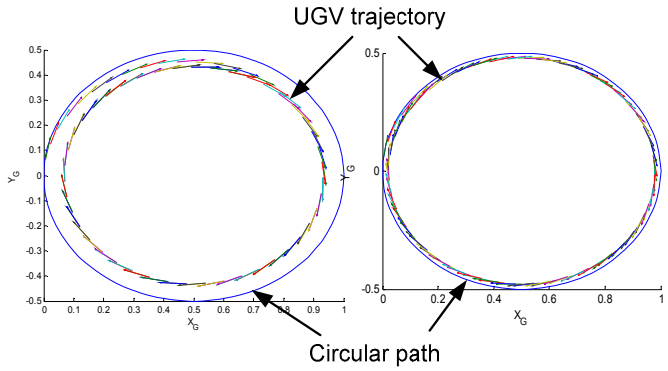

Fig. 2. UGV trajectory while using QC to track a constant curvature path with $d_0 = 0.5$ m (left) and d_0=0.4m (right).

Fig. 2 shows the trajectories of the same UGV while using QC to track the path with constant $d_0 = 0.5$ and 0.4 m. The UGV starts from the origin and successively track the circular path as showed by the arrows. We can observe the difference in the path-deviation error with different values of d_0. Our AE method provides a way to find the steady state error of the UGV path tracking. Given the *path-deviation tolerance* $\varepsilon_e \in \mathbf{R}$ as the maximum bound of the error, we can adjust d_0 so that the error at steady state is less than ε_e. If there are several values of d_0 causing the steady state error less than ε_e, we will select the value of d_0 that makes the UGV converge to the steady state the fastest, since this value of d_0 would imply a good response of the UGV to regulate the path-deviation error. As a result, this AE approach can provide a data table so that we can look up for an appropriate d_0 according to the UGV speed, the sampling period, and the minimum radius of the tracking path.

We can describe our approach in the following steps:

(i) Define discrete-time state transition function of the UGV

In our AE approach, the steady state error of the UGV path tracking can be found by

successively calculating the next state of the UGV from the current state. The simulation can be used to acquire the state of the UGV at any given time. However, the result would depend highly on the detail specification of the UGV such as the mass, dimension, and motor parameters, making it difficult to draw a general conclusion. Therefore, we propose a discrete time state transition equation with the assumption that the turn radius and the exact distance the UGV traveled in each sampling period is known to represent the UGV in general. The proposed discrete-time state transition function of the UGV is shown in Eqn. (5)

$$\mathbf{x}_b(t_{i+1}) = \mathbf{f}_b\left(\mathbf{x}_b(t_i), d_0, R_{path}, s_{UGV}\right) \tag{5}$$

where d_0 is the look-ahead distance, R_{path} is the circular path radius having positive value when the path center is on the left of the robot, s_{UGV} is the distance that the robot travels within one sampling period, $\mathbf{f}_b(\cdot)$ is the state transition function elaborated in the appendix, and $\mathbf{x}_b$ is a transformed state of the UGV. We use the transformed state $\mathbf{x}_b$ here to reduce the information we need to consider. Since we focus on the path-deviation error to be calculated in each step, we don't need to know the exact position of the UGV. Therefore, we can represent the state of the UGV, by the distance between the UGV and the path, and the heading angle of the UGV referenced with a perpendicular line from the path denoted by $\mathbf{x}_b = \begin{bmatrix} x_b & \phi_b \end{bmatrix}^T$, as shown in Fig. 3. This representation of state can help us easily keep track of the path-deviation error as the error is equal to x_b.

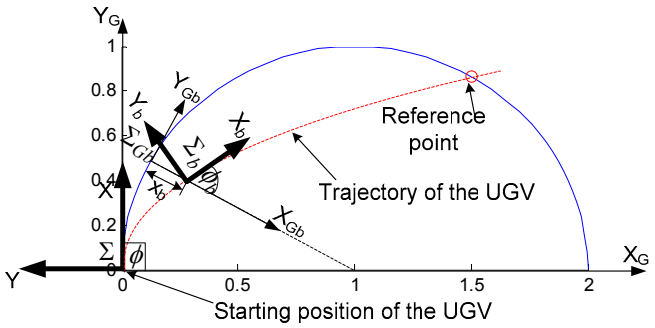

Fig. 3. UGV trajectory to illustrate the state representation.

Another benefit of new state representation is that it is easier to calculate the state of the UGV for the next period because the calculation is done in the adjusted world coordinate Σ_{Gb}, described by (X_{Gb}, Y_{Gb}) where the position of the UGV on Y_{Gb} axis is always zero.

(ii) Find steady state error based on the system parameters

We can observe from Fig. 2 that there is a situation where the trajectory of the UGV converges to the state that has constant path-deviation error. We call this constant value of the error as the steady state error. Our goal is to find this steady state error according to s_{UGV}, R_{path}, and d_0. Steady state error is denoted as x_b^* in $\mathbf{x}_b^* = \begin{bmatrix} x_b^* & \phi_b^* \end{bmatrix}^T$ which can be described by the proposed state transition equation as shown in Eqn. (6).

$$\mathbf{x}_b^* = \mathbf{f}_b\left(\mathbf{x}_b^*, d_0, R_{path}, s_{UGV}\right) \tag{6}$$

We can numerically find $\mathbf{x}_b^*$ by using the state transformation equation to successively find the next state from the current state. When the next state is equal to the current state, then we obtain the steady state $\mathbf{x}_b^*$. In practical scenario, $\mathbf{x}_b^*$ is found by checking for the difference between the current state and the next state to be within a small bound for several

periods. Associated with $\mathbf{x}_b^*$ are d_0, R_{path}, and s_{UGV}. A smaller value of steady state error contributed by d_0 indicates a good tracking performance which implies that the path-deviation error is small. We select d_0 that makes the steady state error stay within a small bound of ε_e. We denote this optimal value of d_0 by d_0^*. In addition, to ensure that the robot will converge to the path fast and has a small path-deviation error, d_0^* is selected so that the UGV takes the least state transformation steps to converge to steady state.

(iii) Collect the optimal d_0 and generate a lookup table

For each path tracking algorithm, we collect the data of d_0^* for several values of R_{path} and s_{UGV} as shown in Table 1-Table 3 given ε_e=0.05 m. Note that s_{UGV} is represented by KT for VP & PP and is represented by αT for QC. From the tables, if we know α or K used in the path tracking algorithm, the radius of the path R_{path}, and the sampling period T, we can determine the optimal look-ahead distance d_0^* for the path tracking algorithm. If the path to be tracked comprises of several circular curves with different radius, d_0^* will be looked up from the smallest R_{path} because d_0^* should be suitable for the worst case since the UGV has the most difficulty tracking the circular path with smallest radius.

Table 1. d_0^* for VP path tracking.

	d^*_0 (m)			
	R_{path}=0.15m	R_{path}=0.60m	R_{path}=1.20m	R_{path}=1.80m
KT=0.01	0.011192	0.012185	0.015894	0.012715
KT=0.05	0.051954	0.051788	0.055629	0.058808
KT=0.10	0.106358	0.103709	0.103576	0.107285
KT=0.15	0.149007	0.156556	0.159205	0.155364
KT=0.20	N/A	0.208212	0.206887	0.209801
KT=0.25	N/A	0.25947	0.254834	0.264238
KT=0.30	N/A	0.312318	0.311523	0.310728

Table 2. d_0^* for QC path tracking.

	d*$_0$ (m)			
	R_{path}=0.15m	R_{path}=0.60m	R_{path}=1.20m	R_{path}=1.80m
αT=0.01	0.010629	0.015762	0.019338	0.019868
αT=0.05	0.016523	0.036821	0.047947	0.048079
αT=0.10	0.033477	0.072053	0.093245	0.099735
αT=0.15	0.052483	0.112053	0.142517	0.155762
αT=0.20	0.077119	0.15298	0.188609	0.208212
αT=0.25	0.096589	0.193113	0.24106	0.263046
αT=0.30	0.118642	0.243841	0.290331	0.311523

Table 3. d_0^* for PP path tracking.

	d*$_0$ (m)			
	R_{path}=0.15m	R_{path}=0.60m	R_{path}=1.20m	R_{path}=1.80m
KT=0.01	0.01298	0.01298	0.015894	0.011921
KT=0.05	0.062252	0.064636	0.063576	0.069934
KT=0.10	0.126623	0.125695	0.129801	0.131126
KT=0.15	0.149007	0.190728	0.185695	0.193907
KT=0.20	N/A	0.251921	0.250596	0.254305
KT=0.25	N/A	0.311391	0.313907	0.316689
KT=0.30	N/A	0.375762	0.379073	0.375894

B. Find d_{max} and β for QC algorithm

According to QC algorithm, d_0^* at different R_{path} is calculated from quadratic curve coefficient, A, as shown in Eqn. (1) where the relationship between A and R_{path} can be derived as $A = 1/(2R_{path})$ when the UGV moves with $R_{UGV} = R_{path}$. Therefore, instead of looking up d_0^* from R_{path} and αT, QC needs to look up d_{max} and β for each αT. We can find d_{max} and β for each αT by using the information in Table 2 and nonlinear curve fitting based on Eqn. (1). An example for finding d_{max} and β for one value of αT from each row of Table 2 is shown in Fig. 4. Then, we find a pair of d_{max} and β for each value of αT. The resulting

d_{max} and β for each value of αT are shown in Table 4.

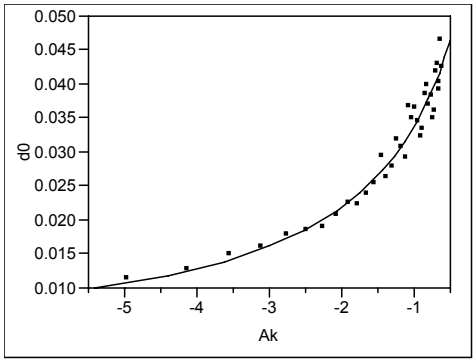

Fig. 4. Nonlinear curve fit.

Table 4. Tuned d_{max} and β.

	d_{max}	β	d_{min}
αT=0.01	0.021335	0.350524	0.010629
αT=0.05	0.065985	0.926369	0.016523
αT=0.10	0.127774	0.887889	0.033477
αT=0.15	0.187675	0.786966	0.052483
αT=0.20	0.247314	0.750866	0.077119
αT=0.25	0.305059	0.666884	0.096589
αT=0.30	0.36021	0.588742	0.118642

Note that, even QC can adjust d_0 according to several values of path curvature; our tuned d_0 in

Table 2 has a lower bound according to the smallest path radius in consideration. While using QC path tracking, we need to ensure that the value of d_0 is not less than the minimum value d_{min} because using the value of d_0 less than the tuned minimum value can cause unpredictable result.

C. Effect of UGV dynamics on table lookup

So far, we assume that the UGV can regulate the speed to be close to the reference speed v_{ref}, and $s_{UGV}=v_{ref}T$. However, in some cases, the dynamics of the UGV has substantial effect, and causes the speed of the wheel to take some time period to change from the current

value to a reference value. Consider the case that the wheel of the UGV is desired to run at v_{ref} m/s (= v_{ref}/r_{wheel} rad/s) between time period [0, T) and stop beyond that as shown in Fig. 5 (left). For the ideal case, the speed of the wheel can drop down to zero in no time. However, in a practical scenario, the wheel may take some time to slow down and the distance that the UGV travels within one period is not $v_{ref}T$ but includes all of the extra shaded area in Fig. 5 (right). Because the extra distance the UGV travels can cause excessive error, d_0^* should be selected according to that extra distance or KT and αT equal to the entire shaded area, as in Fig. 5 (right). Therefore, it is recommended to acquire the information as to how fast the actual UGV of interest can change its speed (e.g. from a step response) before finding KT and αT for table lookup.

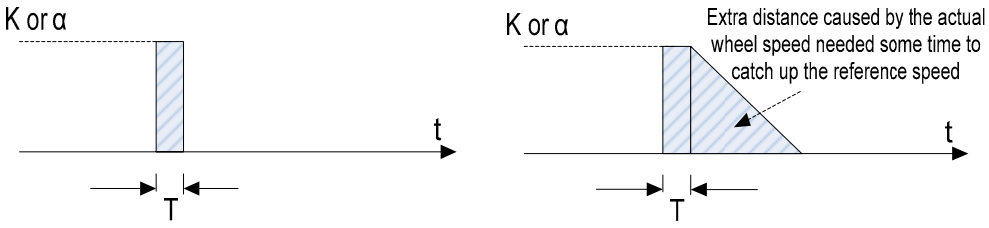

Fig. 5. Area showing the distance the UGV moves caused by exerting the reference signal for one sampling step (left), Area showing the distance the UGV moves affected by the UGV dynamics (right).

IV. SIMULATION RESULTS

A. Simulation setup

We use the model of the UGV as described in section II with the parameters in

Table 5 to simulate the movement of a small UGV named Ana2 while tracking a predefined path as shown in Fig. 6.

Table 5. UGV parameters.

Parameters	values
Distance between the driving wheels and the axis of symmetry (b_1)	0.0865 m
Distance between the center of turn and center of gravity (b_2)	-0.09 m
Radius of the wheel (r)	0.036 m
Mass of each wheel and rotor (m_w)	0.025 kg
Moment of the UGV about the center of turn (I_c)	2.434e-02
Moment of inertia of each wheel about the wheel diameter (I_m)	3.375e-06
Moment of inertia of the wheel about the wheel axis (I_W)	1.62e-05 kgm^2
Damping coefficient of the motor (b_{motor})	2e-07
Moment of inertia of the rotor plus the gears' (J_{motor})	1.5e-07 kgm^2
Motor gear ratio (N_a)	333.6375
Motor inductance (L_{motor})	2.0e-03 H
Motor armature resistance (R_{motor})	7.4 ohms

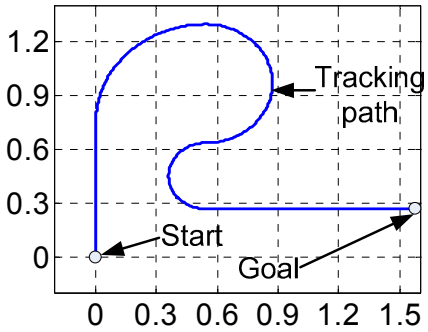

Fig. 6. Tracking path used in simulation.

For each run, we stop the simulation when the robot reaches the goal or the simulation time reaches 120 sec. In the simulation, using VP and PP, the path is tracked with constant speed 0.09 m/s, while QC tracks with maximum speed 0.126 m/s. The sampling time is 0.1 s. However, the step response of the UGV behaves like a first order system with time constant = 0.2756 s. The tuned parameters from our proposed method is d_0=0.0354 m for VP (given k=2.5 [7]), d_{max}=0.0630 m for QC, and d_0 = 0.0423 for PP.

The criterion to compare the result comes from the cost function that can measure how

large the path-deviation error is and how fast the robot reaches its destination. We will use a weighted cost function from the integration of the path-deviation error and the tracking speed of UGV defined by

$$J_1 = \int_{t_0}^{t_f} \sqrt{\left(x(t) - x_p(s_t)\right)^2 + \left(y(t) - y_p(s_t)\right)^2}\, dt \tag{7}$$

$$J_2 = -{}^{d_{t_f}}\!/_{t_f - t_0} \tag{8}$$

(x_p,y_p) is a function describing the coordinate of the path closest to the robot at time t, (x, y) is the current position of the mobile robot, t_0 is the time we start the simulation, t_f is the time we stop the simulation, d_{tf} is the distance along the path from the starting point to the projected point of the UGV on the path at $t=t_f$. The weighted cost function is defined by

$$J = w_1 J_1 + w_2 J_2 + w_3 \tag{9}$$

In a practical scenario, these weights are found by comparing several outcomes of UGV path tracking. The eligible weights would generate the cost that conforms to human heuristic sense. One suitable set of the weights that we found are $w_1 = 5.555$, $w_2 = 61.455$, and $w_3 = 5.109$.

B. Result and discussion

The path tracking parameter tuning method can be evaluated by considering the simulation result. We ran the simulation for the three path tracking algorithms VP, QC, and PP. Then, we varied the look-ahead distance used in the algorithm (d_0 for VP and PP, d_{max} for QC) and observed the path tracking performance as shown in Fig. 7.

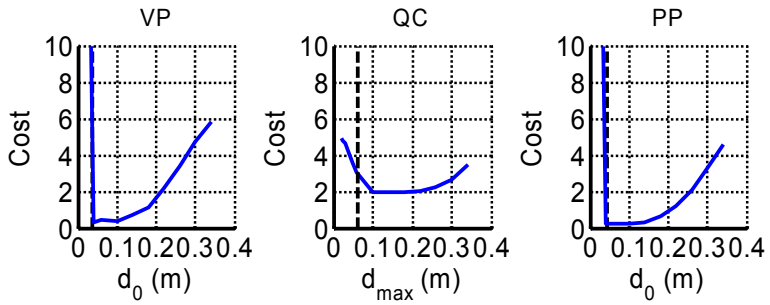

Fig. 7. Path tracking cost according to different values of look-ahead distance (the dashed line is the tuned look-ahead distance).

The dashed line in Fig. 7 is corresponding to the value of the tuned look-ahead distance. Note that the cost while using tuned look-ahead distance is near to the smallest cost value. Only in case of QC, the tuned parameter is substantially less than the actual optimal d_{max}.

Another set of simulation was run, with noise to simulate the position measurement error of the UGV. The additive noise is the uniform random distribution with the range [-.008, .008] m for the UGV coordinate in x and y axis and [-3, 3] degree for the heading angle of the UGV. This noise can occur in the actual system where the UGV position is acquired from a camera image with quantization error imposed by the image pixel. The cost of the result is plotted as shown in Fig. 8.

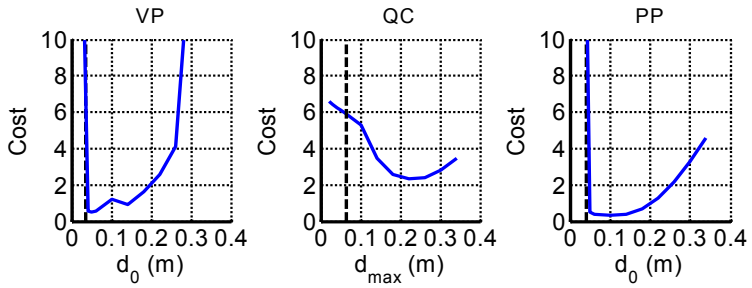

Fig. 8. Path tracking cost according to different values of look-ahead distance with added noise.

As shown in Fig. 8, we can observe that there is a small change on the shape of the graph for VP and PP as the minimum cost moves slightly to the right. For QC, the minimum cost

moves to the right more than the other two algorithms. The results in Fig. 7 and Fig. 8 show that our tuned look-ahead distance provides a lower bound of the value corresponding to the minimum path tracking cost. In practical scenario, for VP and PP, we should use the look-ahead distance slightly greater than the tuned value to reduce the sensitivity of the cost according to the noise. For QC, we might need to do some experiment with several larger values of look-ahead distances in order to find the one corresponding to the lowest cost.

V. CONCLUSION

In this paper, we have proposed a parameter tuning technique for geometrical mobile robot path tracking. Three geometrical path tracking algorithms, vector pursuit, quadratic curve path tracking and pure pursuit were considered. The result is a lookup table from which a suitable value of look-ahead distance can be easily found. The effectiveness of the tuning method is demonstrated by the results from simulation which consider the dynamics of the mobile robot and the dynamics of the motor. The results show that the tuned parameter corresponds closely to the optimal path tracking performance for VP and PP. Further fine tuning can be achieved by considering the tuned parameter as the lower bound to find the look-ahead distance corresponding to the lowest cost.

APPENDIX

Implementation of the UGV state transition equation

The state equation function in Eqn. (5) can be implemented by breaking into several steps as follows:

1) *Find* the *reference point* (x_{ref},y_{ref}) referenced with the adjusted global coordinated $\Sigma_{G,b}$

from the current state by

$$x_{ref} = -\left(d_0^2 - x_b^2\right) \Big/ \left(2R_{path} + 2x_b\right) \tag{10}$$

$$y_{ref} = \sqrt{R_{path}^2 - (x_{ref} + R_{path})^2} \tag{11}$$

Note that d_0 must be greater than the distance between the UGV and the path in order for (x_{ref},y_{ref}) to be real number. If d_0 is too small, that value of d_0 will be rejected because it cannot handle excessive error.

2) *Find the circular UGV trajectory* R_{UGV} based on the path tracking algorithm of our choice. Since the UGV uses differential drive to turn, the UGV will have circular trajectory with constant radius R_{UGV} **.**

3) *Find the position of the UGV for the next sampling period.* The position difference of the UGV between each consecutive period on the UGV coordinate can be described by Eqn.(12)-(14).

$$\Delta\phi = s_{UGV} / R_{UGV} \tag{12}$$

$$\Delta x = R_{UGV} \times \sin(\Delta\phi) \tag{13}$$

$$\Delta y = R_{UGV} - R_{UGV} \times \cos(\Delta\phi) \tag{14}$$

Then, by using coordinate transformation we can find the updated position of the UGV ($\mathbf{x} = \left[x_{temp} \;\; y_{temp} \;\; \phi_{temp}\right]^T$) on the adjusted global coordinate $\Sigma_{G,b}$ as follows:

$$x_{temp} = x_b + \Delta x \cos(\phi_b) - \Delta y \sin(\phi_b) \tag{15}$$

$$y_{temp} = \Delta x \sin(\phi_b) + \Delta y \cos(\phi_b) \tag{16}$$

$$\phi_{temp} = \phi_b + \Delta\phi \tag{17}$$

where ϕ_{temp} is the heading angle of the UGV.

4) *Find the UGV state* $\mathbf{x}_b$ *from the UGV position.* In this step, we need to find the UGV state for the next step $\mathbf{x}_b(t_{i+1}) = \left[x_b(t_{i+1}) \ \phi_b(t_{i+1}) \right]^T$ from the position of the UGV based on $\Sigma_{G,b}$. It is similar to changing the coordinate $\Sigma_{G,b}$ by moving the origin to the projected point of the UGV on the path and having Y_{Gb} align with the path. We calculate $x_b(t_{i+1})$ based on how far the UGV away from the circular path and calculate $\phi_b(t_{i+1})$ based on the heading angle of the UGV referenced to the new $\Sigma_{G,b}$ as follows:

$$x_b(t_{i+1}) = \text{sgn}(R_{path}) \sqrt{\left(R_{path} + x_{temp} \right)^2 + y_{temp}^2} - R_{path} \tag{18}$$

$$\begin{aligned} \phi_b(t_{i+1}) = \phi_{temp} + \text{atan2}\Big(& -\text{sgn}(R_{path}) y_{temp} \\ & , -\text{sgn}(R_{path})(-R_{path} - x_{temp}) \Big). \end{aligned} \tag{19}$$

Finally, we can combine all the steps in this appendix to define $\mathbf{f}_b\left(\mathbf{x}_b, d_0, R_{path}, s_{UGV} \right)$ as the state transition equation of the UGV path tracking.

ACKNOWLEDGMENT

This research was partially sponsored by the National Science Foundation (NSF) through Grant No. 046852, “Intelligent Human-Machine Interface & Control for Highly Automated Chemical Screening Processes.”

REFERENCES

[1] W.-L. Leung, R. Vanijjirattikhan, Z. Li, L. Xu, T. Richards, B. Ayhan, and M.-Y. Chow, "Intelligent space with time sensitive applications," IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, California USA, 2005.

[2] H. Hashimoto, "Intelligent space - How to make spaces intelligent by using DIND?," Proceedings of 2002 the IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia, 2002.

[3] Y. Tipsuwan and M.-Y. Chow, "Gain scheduler middleware: A methodology to enable existing controllers for networked control and teleoperation: PART II: teleoperations," *IEEE Transactions on Industrial Electronics*, vol. 51, 2004.

[4] R. Vanijjirattikhan, M.-Y. Chow, P. Szemes, and H. Hashimoto, "Mobile Agent Gain Scheduler Control in Inter-Continental Intelligent Space," The 2005 IEEE International Conference on Robotics and Automation (ICRA05), Barcelona, Spain, 2005.

[5] O. Amidi, "Integrated Mobile Robot Control, tech. report CMU-RI-TR-90-17," Robotics Institute, Carnegie Mellon University 1990.

[6] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path tracking control of mobile robots using a quadratic curve," Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 1996.

[7] J. Wit, C. D. Crane III, and D. Armstrong, "Autonomous ground vehicle path tracking," *Journal of Robotic Systems*, vol. 21, pp. 439-449, 2004.

[8] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," IEEE International Conference on Robotics and Automation, 1990.

[9] T. Das and I. N. Kar, "Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots," *IEEE Transactions on Control Systems Technology*, vol. 14, pp. 501-510, 2006.

[10] C. Samson, "Control of chained systems application to path following and time-varying point-stabilization of mobile robots," *IEEE Transactions on Automatic Control*, vol. 40, pp. 64-77, 1995.

[11] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, pp. 835-852, 2002.

[12] R. Fierro and F. L. Lewis, "Control of a nonholomic mobile robot: Backstepping kinematics into dynamics," *Journal of Robotic Systems*, vol. 14, pp. 149-163, 1997.

[13] A. Ollero and G. Heredia, "Stability analysis of mobile robot path tracking," Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95), Pittsburgh, PA, 1995.

[14] Y. Yamamoto, "Control and Coordination of Locomotion and Manipulation of a Wheeled Mobile Manipulator," University of Pennsylvania, 1994.

[15] J. J. D'Azzo and C. H. Houpis, *Linear control system analysis and design : conventional and modern*, 4th ed. New York:McGraw-Hill, 1995.

CHAPTER III

THE SENSITIVITY ISSUE OF MOBILE ROBOT PATH TRACKING PROBLEM: A DISCUSSION OF NETWORK-BASED CONTROL SYSTEM UNDER NETWORK DELAY CONSTRAINTS

Rangsarit Vanijjirattikhan, Zheng Li, Mo-Yuen Chow

rvanijj, zli7, chow@ncsu.edu

Advanced Diagnosis Automation and Control Lab

Department of Electrical and Computer Engineering

North Carolina State University, Raleigh NC 27695, USA

THE SENSITIVITY ISSUE OF MOBILE ROBOT PATH-TRACKING PROBLEM: A DISCUSSION OF NETWORK-BASED CONTROL SYSTEM UNDER NETWORK DELAY CONSTRAINTS

Abstract - Sensitivity analysis (SA) is a method to show how the outputs of a model changed with the variations of its parameters. In this chapter, we use this method to examine the Network-based Control System (NCS) which combines communication networks and control systems. A network-based mobile robot path-tracking problem is formulated and analytic sensitivity is obtained through mathematical formulation. This approach provides us guidelines on how to choose feasible/optimal operating conditions for NCS with the concern of the change system parameters.

Keywords: *Sensitivity Analysis, Network-based Control System, Mobile Robot, Path-tracking, Network delay*

I. INTRODUCTION

Sensitivity analysis (SA) studies the relationship between information flowing in and out of the model and shows how a model's performance changes with variations in its parameters [1]. According to the performance changes, the sensitivity values can identify which parameters are the most important or most likely to affect the performance. Consequently, we can focus on tuning or controlling the values of these critical parameters to effectively improve the system performance. Sensitivity can be calculated as a function of time or a

function of operating conditions changing with time. In real time control systems, we can use this information to estimate or predict the system behaviour under various perturbations in order to adjust the system parameters or re-allocate resources to maximize the overall system performance.

SA has been used in different research areas and projects. To name a few: A regression-based sensitivity analysis was used in the risk assessment of the nuclear waste disposal which is modelled in an uncertainty framework [1]. In [1], SA was used in the field of solid state physics, investigating the stability properties of an inverse problem. [2] used SA to measure the robustness of a fault detection/diagnosis artificial neural network. Inspired from many of the SA applications, this chapter investigates the use of SA in the analysis of remote intelligent humanitarian robotic control system, such as Intelligent Space (iSpace) [3], which is a large scale mechatronics system, as shown in Fig. 1. The information from the distributed sensors in the iSpace is fused via wireless communication environment, analyzed and sent to the controller to provide appropriate control to the robots in the space. For example, when a patient in a nursing home wants to drink water, the iSpace senses the request from the patient (e.g., voice recognition from the patient), communicates with a mobile robot, and navigates the robot to carry the water, go through the corridor and deliver the water to the patient. These kinds of applications belong to a remotely controlled mobile robot path-tracking problem in network-based control systems.

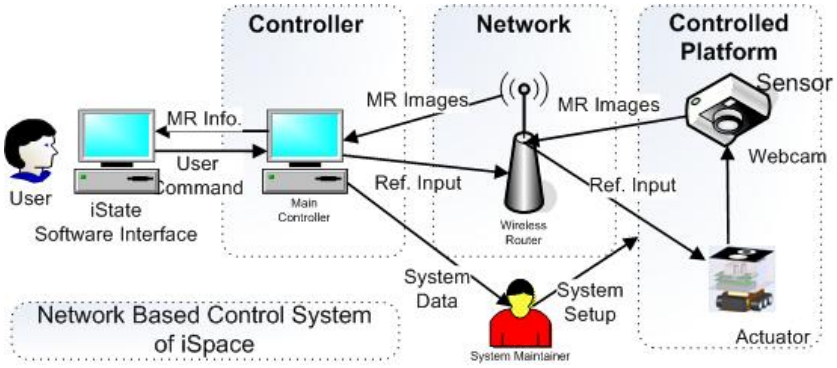

Fig. 1. The overall structure of an iSpace.

Network-based Control (NBC) technologies have been investigated and developed to combine communication networks with control systems to form Network Control System (NCS) (e.g., control network in automobiles, the teleoperation and coordination of distributed Unmanned Ground Vehicles (mobile robot) and teleoperation of robot arm manipulators [4]). The components in a NCS (physically located far away from each others in most cases) can communicate with each other through networks. Fig. 2 shows a typical NCS in which the controller controls the plant through a communication network. The control signals and feedback measurements from the plant are all transmitted via a network. The control applications using NCS can reduce investment and maintenance cost for wiring complexity (e.g., in factory automation), enable teleoperation, and enable new control concepts and applications.

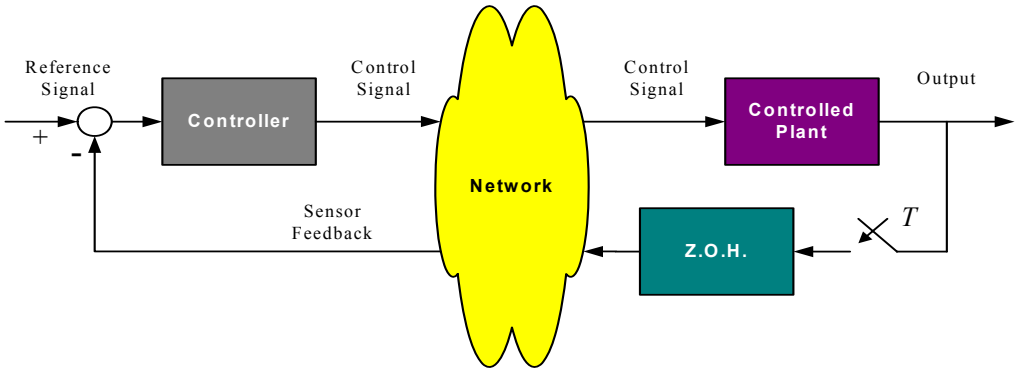

Fig. 2. Network-based control system.

In the NBC path-tracking problem using Internet Protocol (IP), due to the stochastic nature of the IP network delay, with the same control strategy, the path-tracking error can be significantly different under different environmental and the robot operating conditions. However, the effect on tracking error under different environmental and robot operating conditions in presence of the network delay has not been discussed before. In this chapter, we will use the sensitivity analysis method to investigate and obtain quantitative measures on how the robot tracking error will be under different environments and robot operating conditions, such as network delay, path curvature and the operating robot speed. With this quantitative information, we can focus on the investigation and tune several parameters to effectively improve the overall system performance.

The chapter is organized as follows. In section II, we introduce two types of sensitivity functions. In section III, we mathematically formulate the remote mobile robot path-tracking problem. In section IV, the cost and analytical sensitivity functions for remote mobile robot path-tracking are mathematically derived and an application of sensitivity data is discussed Finally, in section V, we conclude the results of this study.

II. SENSITIVITY OF THE DYNAMICS SYSTEM

This section briefly describe two types of sensitivity analysis [5].

A. Analytical sensitivity

Analytical sensitivity functions are used when the system under study is relatively simple, well defined and mathematically well behaved. Analytical sensitivity functions generally take the form of partial derivatives. The derivatives have the system parameters as their inputs and can be plotted as the functions of these variables. We could easily analyze the system according to the derivative functions.

A simple analytical sensitivity S of the function F to variation in the parameter α is given by

$$S_{\alpha}^{F} = \left.\frac{\partial F}{\partial \alpha}\right|_{NOP}, \tag{1}$$

where $\left.\frac{\partial F}{\partial \alpha}\right|_{NOP}$ denotes the partial derivative evaluated at the normal operating point (NOP) where all the parameters are at their nominal values.

B. Empirical sensitivity

Empirical sensitivity functions are often point evaluations of a system's sensitivity to a given parameter(s) when other parameters are known and at fixed values. Since many engineering systems are not simple enough to be mathematically well defined, their partial derivative can not be easily described. Consequently, the sensitivity functions are derived empirically. Various methods are deployed to vary system parameters across the domain of environment and operating conditions of interest in order to obtain meaningful and useful

information on how the parameters affect the system output/performance. Experiments are performed and appropriate input-output data are collected by perturbing the system operating condition from its nominal operating condition.

With the empirical sensitivity functions, the sensitivity determined is only valid near the nominal operating condition. When there is an operating range instead of an operating point, we need to select appropriate operating points over the entire operating range and perform similar experiments on all the operating points. The operating points are selected so that the sensitivity surface constructed by interpolating the sensitivity at each operating points are representatives of the overall system sensitivity under different environments and operating conditions One concern with the empirical sensitivity is that the number of experiment increases geometrically with the number of parameters being studied. Empirical sensitivity can be acquired from the following equation:

$$\frac{\partial F}{\partial \alpha} = \lim_{h \to 0} \frac{F(\alpha + h) - F(\alpha)}{h} \tag{2}$$

where α is the parameter that we perturb to observe how cost function F sensitive to.

III. REMOTE MOBILE ROBOT PATH-TRACKING PROBLEM

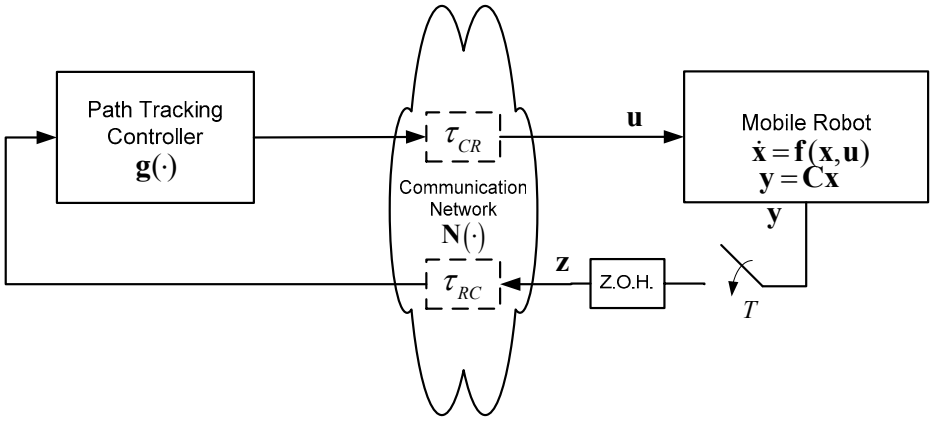

Fig. 3. Mobile robot path-tracking network-based control system.

The overall system of the robot (differential drive robot is used in this chapter) path-tracking control over a communication network is depicted in Fig. 3. The path information and the position of the robot are assumed known or measurable to the controller. This information is used to calculate the control signals to drive the wheels of the robot to track the path.

A. Mathematic model of the remote mobile robot path-tracking problem

As shown in Fig. 3, the system is composed of a mobile robot, a communication network $\mathbf{N}(\cdot)$ with the delay time τ_{CR} and τ_{RC}, and a path-tracking controller $\mathbf{g}(\cdot)$. The communication network considered here is a packet switching network where the data is transferred in chunk from time to time. The communication network separates the system into a remote plant site and a central controller site. The robot on the remote plant site can be modeled by state space description $\mathbf{f}(\cdot)$ as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}, \mathbf{u}\right) \tag{3}$$

where $\mathbf{x} = \left[x\, y\, \theta_r\, \theta_l\, \dot{\theta}_r\, \dot{\theta}_l\, i_{a_r}\, i_{a_l}\, z_r\, z_l\right]^T$ is the robot states; $\mathbf{u} = \left[v_{ref}, \omega_{ref}\right]^T$ is the control signal from the path-tracking controller. Note that (x,y) is the robot coordinate, θ and $\dot{\theta}$ are the angular displacement and angular velocity of the wheel, i_a is the state of the motor electrical dynamics, z is the state of the integrator in PI speed controller. *r* and *l* denote the right and the left wheel, respectively. For the control signal, it composes of reference speed (v_{ref}) and reference turn rate (ω_{ref}) for the mobile robot. The output signal of the robot is a subset of its states denoted by $\mathbf{y} = \left[x\, y\, \phi\right]^T$ where ϕ is the heading angle of the robot which can be calculated from θ_r and θ_l.

Before sending the output signal of the robot through the communication network to the path-tracking controller, the output signal is sampled and held over every constant period *T*. Denoted as **z**, the sampled and hold output signal is sent to the path-tracking controller and used as the input for calculating control signals. Then, the control signal is sent back to the robot through the communication network. The calculation of the control signal can be described by Eqn. (4).

$$\mathbf{u}(t^+) = \mathbf{g}\left(\mathbf{y}(t - \tau^{k_i})\right), \;\; t \in \left\{k_i T + \tau^{k_i}, i = 1, 2, 3, ...\right\} \tag{4}$$

where $k_i T$ is the time instant that the robot output signal is sampled, $\tau^{k_i} = \tau_{RC}^{k_i} + \tau_{CR}^{k_i}$ is the round-trip-time delay associated with sampling index k_i, $\left\{k_1, k_2, k_3, ...\right\} \subset \left\{0, 1, 2, 3, ...\right\}$. T is assumed to be substantially small compared with τ^{k_i}. Note that this equation describes the control signal as perceived by the robot. Since the robot output signal is delayed by $\tau_{RC}^{k_i}$

before reaching the controller and the control signal is delayed by $\tau_{CR}^{k_i}$ before reaching the robot, the robot will get the control signal based on the output of the robot at time $t-\tau_{RC}^{k_i}-\tau_{CR}^{k_i}=t-\tau^{k_i}$ in the past. Since we use packet switching network in this model, the robot will receive updated control signal only at a certain time instant ($k_iT+\tau^{k_i}$) when a new data packet has arrived. Then, robot will use the arrived control signal until the next data packet comes. In this chapter, we focus on the system that **z** will be sent out to the path-tracking controller only after **u,** corresponding to the previous **z,** had arrived at the robot. Therefore, we enforce a constraint that $k_{i+1}T > k_iT+\tau^{k_i}$. Hence, $\left[k_iT, k_iT+\tau^{k_i}\right)\cap\left[k_{i+1}T, k_{i+1}T+\tau^{k_{i+1}}\right)=\varnothing$ and $\cup_{i=1}^{\infty}\left[k_iT+\tau^{k_i}, k_{i+1}T+\tau^{k_{i+1}}\right)=\left[t_0,\infty\right), t_0\geq 0$.

B. Network delay effect on the remote mobile robot path-tracking

A sequence of the information exchanges between the mobile robot and the controller is depicted in Fig. 4. The position of the mobile robot will be measured and sent out.

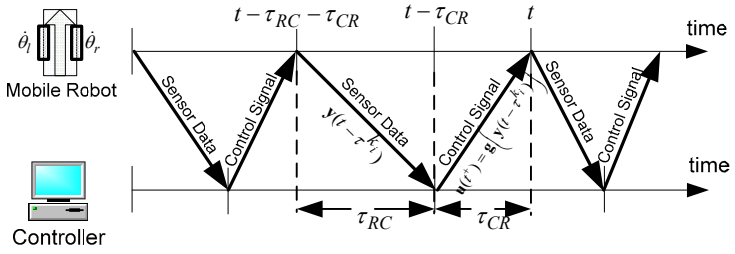

Fig. 4. Information exchange between the mobile robot and the central controller.

As shown in Fig. 4, the control signal comes from the controller to the mobile robot whenever the controller receives the output signal from the robot. The output signal is delayed by τ_{RC} while it is being sent through the communication network to the controller.

The control signal is delayed by τ_{CR} while it is being sent through the communication network to the robot. Therefore, the control signal is calculated based on the position of the mobile robot in the last τ sec as shown by the term $\mathbf{y}(t-\tau^{k_i})$ in Eqn. (4). This τ time gap makes the controlled signal always out of date of the current mobile robot state and increases the path-tracking error, or even fails to track the path in a certain time. Note that we have developed a feedback pre-processor (similar to a Smith predictor) to compensate this delay out of sync problem [6]. However, in this chapter, we are mainly interested in the sensitivity of our system performance according to system parameters such as communication delay, tracking path curvature, and robot speed. Thus, we assume that there is no feedback preprocessor to compensate the effect delay.

When there is no network delay and the controller locally controls the mobile robot, the controller generates control signal every sampling period *T* which is usually much smaller than the network delay τ. Fig. 5 shows that the mobile robot can track the path well when it is locally controlled by the path-tracking controller.

On the other hand when there is communication delay in the control loop, the delay time from the controller to the mobile robot is τ_{CR}, the delay time from the robot to the controller is τ_{RC}, and the round-trip-time delay is $\tau = \tau_{CR} + \tau_{RC}$. Because of the delay, the control signal will arrive at the robot totally τ sec later after the measured state. This τ time gap makes the controlled signal always out of date of the current mobile robot state and decreases the efficiency of the path-tracking algorithm, or even can not track back to the path in a certain time, which can be regarded as unstable. In Fig. 6, with two different delays $\tau_1 > \tau_2$, path-

tracking error from the case 1 (regular line) is larger then that from case 2 (dashed line). This leads us to believe that the path-tracking error or other cost functions can be quantitatively related with the system parameters and it would be important to study the how the change these parameters affect the change in path-tracking error.

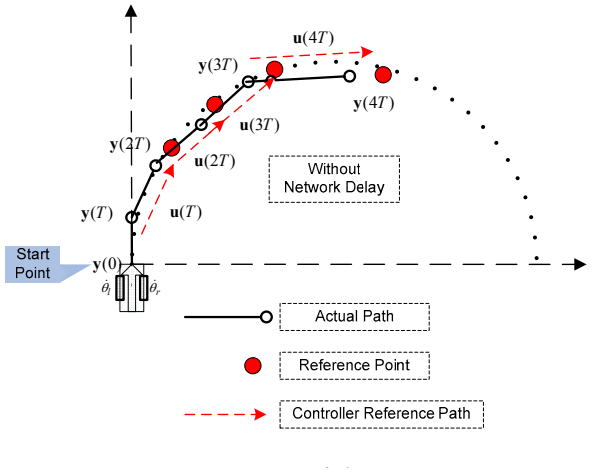

Fig. 5. Path-tracking error without network delay.

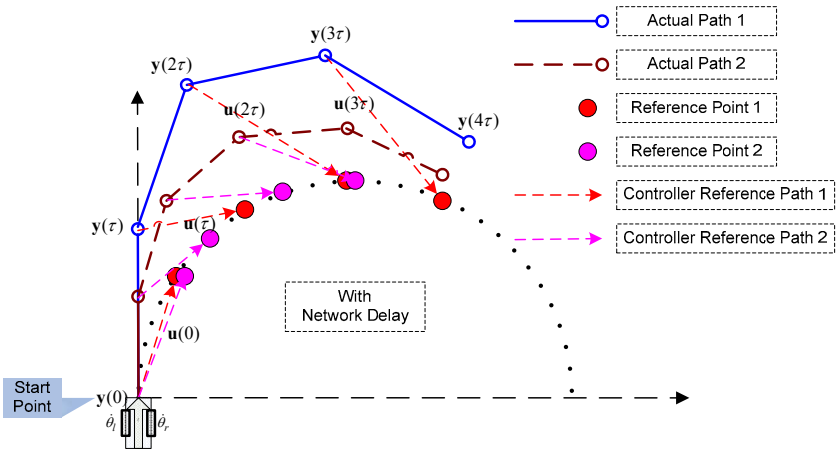

Fig. 6. Path-tracking error with network delay τ.

IV. SENSITIVITY OF REMOTE MOBILE ROBOT PATH-TRACKING

A. Path-tracking cost functions

In this chapter, we investigate the sensitivities of the path-tracking cost with respect to three system parameters, time delay τ, path curvature κ, and robot speed v. We selected these three parameters to study because they directly affect the shortest distance between the robot trajectory and the tracking path which constitutes the path-tracking cost. They are also subjected to the change of system parameters. For example, randomness of the communication delay can perturb τ to be a value slightly different from the average value; distortion of the view (while using camera and image processing to acquire robot position) may cause a difference between path curvature κ known by the path-tracking controller and the actual path on the floor; the robot actual speed may be perturbed from a reference speed from time to time because the motor speed controller requires a settling time period to regulate motor speed. In this section, we find analytical sensitivity of the mobile robot path-tracking cost so that we will know how much the path-tracking cost is affected by the aforementioned change of system parameters.

The path-tracking cost J is defined as an area formed by the shortest distance between the robot and the tracking path as shown by the shaded area in Fig. 7.

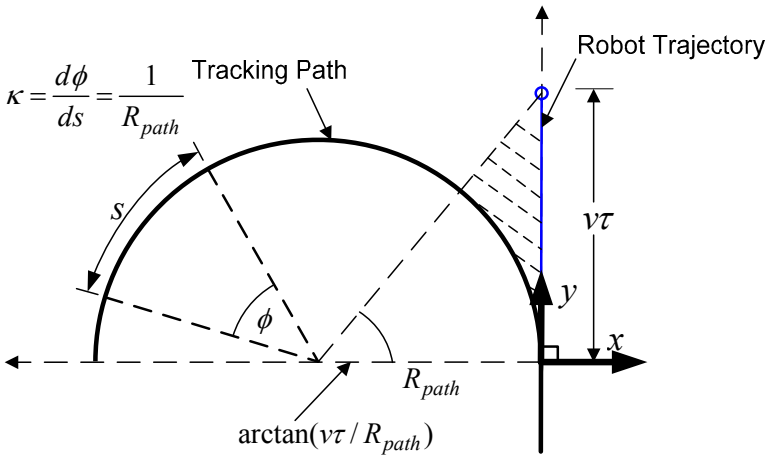

Fig. 7. Path-tracking cost J (gray area) formed by robot trajectory.

Fig. 7 illustrates the tracking path trajectory with curvatures κ and a mobile robot trajectory. The mobile robot's initial position is at (0, 0), heading toward along the y+ axis. The path is a half circle which leads the left with radius R_{path} satisfied $\kappa = \frac{d\phi}{ds} = \frac{1}{R_{path}}$, where ϕ and s are the angle and the length of the same arc on the circle. As the robot tracks the path, the shortest distance between the robot and the path forms an area which represents the path-tracking cost J.

To calculate J, the tracking path must be well described beforehand. In this chapter, the tracking path is described by $\mathbf{x}_p(d) = [x_p(d)\ y_p(d)]^T$ where d is the distance along the path from the starting point. For example, we can define a half circle path with radius R_{path} by

$$\begin{aligned} x_p &= R_{path} \times \left(\cos(d / R_{path}) - 1\right); 0 \le d \le R_{path} \times \pi \\ y_p &= R_{path} \times \left(\sin(d / R_{path})\right); 0 \le d \le R_{path} \times \pi \,. \end{aligned} \tag{5}$$

B. Analytic sensitivity of the mobile robot path-tracking

In this section, we focus on analytic sensitivity of the mobile robot path-tracking cost accumulated over one RTT period. The considered period starts from the robot receiving a control signal from the controller until it receives the next control signal. The advantage of considering only one sampling period is to reduce complexity when deriving analytic sensitivity. The mobile robot path-tracking scenario that we had selected for sensitivity analysis is illustrated Fig. 7 where the robot is moving from a straight path to a constant curvature path. Assume that the robot receives the control signal when the robot is at the beginning of the constant curvature path. The robot will receive the next control signal after τ sec. Because the effect of delay and the robot was tracking a straight path before reaching the constant curvature path, the robot got the control signal that makes it move straight forward. This scenario represents the effect of time delay while the robot is moving from one segment of the path to another segment which is a major cause of remote mobile robot path-tracking error.

As we focus on three system parameters, time delay τ, path curvature κ, and robot speed v for the acquiring of path-tracking cost sensitivities, we denote these parameters as $\boldsymbol{\Gamma} = [\gamma_1 \cdots \gamma_n]^T = [\tau \ \kappa \ v]^T$. A specific set of $\boldsymbol{\Gamma}$ represents an operating condition in which the sensitivity value is acquired. The acquired sensitivity value can be varied according to different operation condition. The sensitivity of remote mobile robot path-tracking cost J according to a parameter γ_i while the system is operating using system parameters $\boldsymbol{\Gamma}$ is denoted as

$$S_{\gamma_i}^{J} = \left.\frac{\partial J}{\partial \gamma_i}\right|_{\boldsymbol{\Gamma}}. \tag{6}$$

(i) Analytical cost function

As defined, cost function J is the area between the tracking path and the robot trajectory, which is the shaded area in Fig. 7. This area can be calculated by:

$$\begin{aligned} J &= \frac{1}{2} v\tau R_{path} - \frac{1}{2} R_{path}^{2} \arctan(v\tau / R_{path}) \\ &= \frac{1}{2}\frac{v\tau}{\kappa} - \frac{1}{2}\frac{1}{\kappa^{2}} \arctan(v\tau\kappa) \end{aligned} \tag{7}$$

Eqn. (7) roughly indicates that the cost function increases with the increase of mobile robot speed and delay time and the decrease of the tracking path curvature.

(ii) Analytic sensitivity

The sensitivities of the cost function on τ, v and κ can be derived according to Eqn. (6) and Eqn. (7) as:

$$S_{\tau}^{J} = \frac{\partial J}{\partial \tau} = \frac{1}{2}\frac{v}{\kappa}\left[1 - \frac{1}{1 + (v\tau\kappa)^{2}}\right] \tag{8}$$

$$S_{\kappa}^{J} = \frac{\partial J}{\partial \kappa} = -\frac{1}{2}\frac{v\tau}{\kappa^{2}}\left[1 + \frac{1}{1 + (v\tau\kappa)^{2}}\right] + \frac{\arctan(v\tau\kappa)}{\kappa^{3}} \tag{9}$$

$$S_{v}^{J} = \frac{\partial J}{\partial v} = \frac{1}{2}\frac{\tau}{\kappa}\left[1 - \frac{1}{1 + (v\tau\kappa)^{2}}\right]. \tag{10}$$

Note that the sensitivity values are different according to each operation condition $\boldsymbol{\Gamma} = [\tau \ \kappa \ v]^{T}$. We can calculate the sensitivity value at different operating conditions as shown

in Fig. 8-Fig. 10.

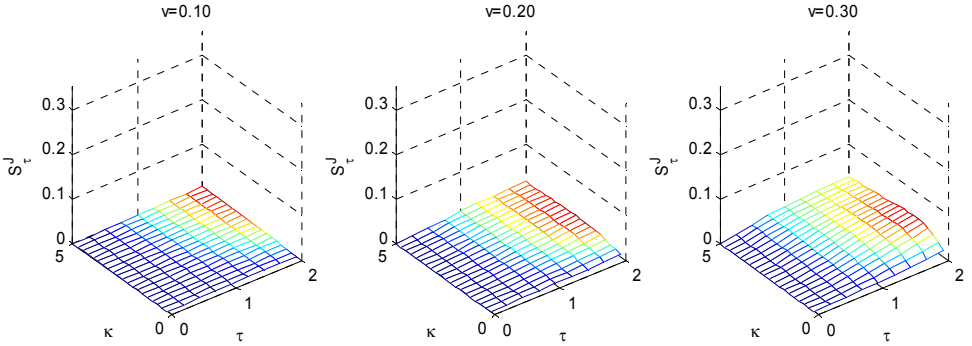

Fig. 8. Path-tracking cost sensitivity respected to round trip time delay, s_τ^J.

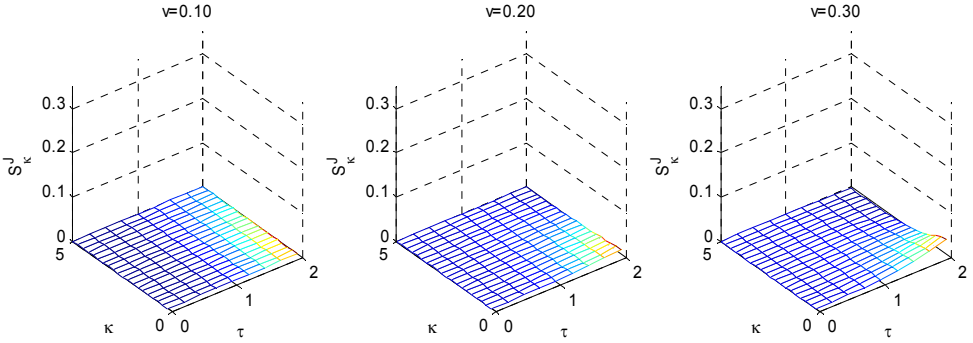

Fig. 9. Path-tracking cost sensitivity respected to path curvature, s_κ^J.

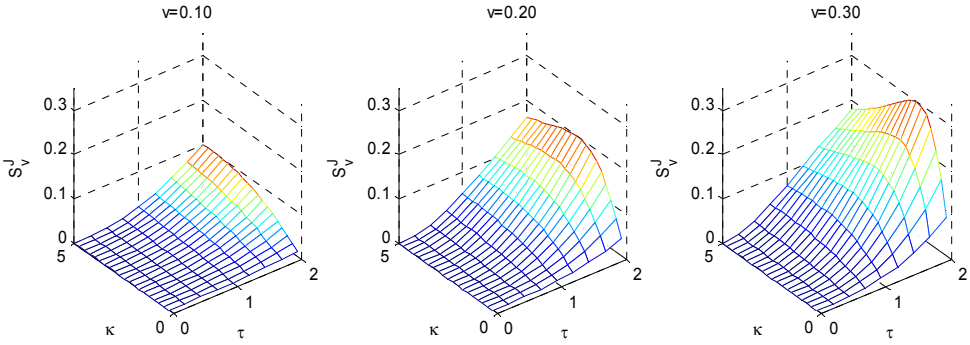

Fig. 10. Path-tracking cost sensitivity respected to robot speed, s_v^J.

These plots show the path-tracking cost sensitivity respected to τ, κ, and v. The range of operating condition we plot here is $\tau \in [0, 2]$, $\kappa \in [0, 5]$, and $v \in [0.1, 0.3]$. Since there are three parameters defining the operating condition and we can only vary the first and second

parameters on x and y axis of the 3D sensitivity surfaces plot, we then use several surface plots to represent the variation of the third parameter.

The surface graphs in Fig. 8 show S_τ^J, the path-tracking cost sensitivity according to τ. These graphs tell us that the system is more sensitive to the variation of time delay when τ, κ, and v is larger. However, when τ, κ, and v are large as shown by Fig. 8 on the right, larger κ can decrease sensitivity as the rate of change of path-tracking cost is decreased when the robot in Fig. 7 moves away from the path. This same trend goes with Fig. 10 where the path-tracking cost sensitivity according to v (S_v^J) is illustrated. For Fig. 9 , the path-tracking cost sensitivity respected to κ (S_κ^J) has different trend. The sensitivity is increased as κ is decreased and τ, v are increased which illustrates that the variation of κ has more effect when the tracking path is straight and has less effect when the tracking path has a high curvature.

(iii) Application of sensitivity data

An application of using sensitivity data is to determine an operating environment parameters (τ, κ, and v) that is proper for a level of the change of a system parameter. Since S_τ^J, S_κ^J, and S_v^J describes the relationship between the change of path-tracking cost (ΔJ) and the change of parameters ($\Delta\tau$, $\Delta\kappa$, or Δv), we can use S_τ^J, S_κ^J, and S_v^J to estimate how the path-tracking cost changes if one of the system parameter is changed. We can also use S_τ^J, S_κ^J, and S_v^J the other way around. We can estimate the change of parameter ($\Delta\tau$, $\Delta\kappa$, or Δv) if we know a small change on the path-tracking cost (ΔJ). Therefore, if we set

the magnitude of ΔJ allowed to happen to the system, we can specify the boundary magnitude of the parameter change that can be tolerated by the system for each operating condition from Eqn. (11). Then, we can use this data to find the operating condition that is proper for a given boundary magnitude of the parameter change.

$$\Delta\tau \cong \frac{\partial\tau}{\partial J}\times\Delta J = \frac{\Delta J}{S_\tau^J}$$

$$\Delta\kappa \cong \frac{\partial\kappa}{\partial J}\times\Delta J = \frac{\Delta J}{S_\kappa^J} \qquad (11)$$

$$\Delta v \cong \frac{\partial v}{\partial J}\times\Delta J = \frac{\Delta J}{S_v^J}$$

In order to find $\Delta\tau$, $\Delta\kappa$, and Δv, we need to select one value of ΔJ as the magnitude of the path-tracking cost that is allowed to be changed. ΔJ here should be a small value that has a meaningful representation. We can consider a scenario happened as in Fig. 7 where a mobile robot tracking the path with speed 0.1 m/s is affected by RTT τ=0.1 sec. Then, the robot will move forward for $0.1\times 0.1 = 0.01$ m during the period that the robot receives consecutive control signals. Suppose that the change of parameters (either $\Delta\tau$, $\Delta\kappa$, or Δv) cause the increasing of the path-tracking error for 0.01 m along the 0.01 m trajectory. Therefore, ΔJ for this scenario can be calculated as $\Delta J = 0.01\times 0.01 = 10^{-4}$. After we choose a value of ΔJ, we can find $\Delta\tau$, $\Delta\kappa$, and Δv for each operating condition as shown in Table 1-Table 3.

Table 1. $\Delta\tau$, $\Delta\kappa$, and Δv when v=0.1 m/s.

$\Delta\tau$ / $\Delta\kappa$ / Δv		κ(1/m)				
		1.0	2.0	3.0	4.0	5.0
τ (sec)	0.1	20.0020 600.1080 20.0020	10.0040 600.4321 10.0040	6.6727 600.9725 6.6727	5.0080 601.7297 5.0080	4.0100 602.7041 4.0100
	0.2	5.0020 75.0540 2.5010	2.5040 75.2162 1.2520	1.6727 75.4871 0.8363	1.2580 75.8674 0.6290	1.0100 76.3583 0.5050
	0.3	2.2242 22.2582 0.7414	1.1151 22.3665 0.3717	0.7467 22.5478 0.2489	0.5636 22.8033 0.1879	0.4544 23.1347 0.1515
	0.4	1.2520 9.4020 0.3130	0.6290 9.4834 0.1572	0.4227 9.6201 0.1057	0.3205 9.8138 0.0801	0.2600 10.0667 0.0650
	0.5	0.8020 4.8216 0.1604	0.4040 4.8869 0.0808	0.2727 4.9971 0.0545	0.2080 5.1542 0.0416	0.1700 5.3611 0.0340

Table 2. $\Delta\tau$, $\Delta\kappa$, and Δv when v=0.2 m/s.

$\Delta\tau$ / $\Delta\kappa$ / Δv		κ(1/m)				
		1.0	2.0	3.0	4.0	5.0
τ (sec)	0.1	2.5010 75.0540 5.0020	1.2520 75.2162 2.5040	0.8363 75.4871 1.6727	0.6290 75.8674 1.2580	0.5050 76.3583 1.0100
	0.2	0.6260 9.4020 0.6260	0.3145 9.4834 0.3145	0.2113 9.6201 0.2113	0.1602 9.8138 0.1602	0.1300 10.0667 0.1300
	0.3	0.2788 2.7958 0.1859	0.1409 2.8504 0.0939	0.0956 2.9430 0.0637	0.0734 3.0761 0.0490	0.0606 3.2534 0.0404
	0.4	0.1573 1.1854 0.0786	0.0801 1.2267 0.0401	0.0551 1.2977 0.0275	0.0431 1.4020 0.0215	0.0362 1.5446 0.0181
	0.5	0.1010 0.6109 0.0404	0.0520 0.6443 0.0208	0.0363 0.7027 0.0145	0.0290 0.7908 0.0116	0.0250 0.9159 0.0100

Table 3. $\Delta\tau$, $\Delta\kappa$, and Δv, when v=0.3 m/s.

$\Delta\tau$ $\Delta\kappa$ Δv		κ(1/m)				
		1.0	2.0	3.0	4.0	5.0
τ (sec)	0.1	0.7414 22.2582 2.2242	0.3717 22.3665 1.1151	0.2489 22.5478 0.7467	0.1879 22.8033 0.5636	0.1515 23.1347 0.4544
	0.2	0.1859 2.7958 0.2788	0.0939 2.8504 0.1409	0.0637 2.9430 0.0956	0.0490 3.0761 0.0734	0.0404 3.2534 0.0606
	0.3	0.0830 0.8351 0.0830	0.0425 0.8720 0.0425	0.0294 0.9360 0.0294	0.0232 1.0311 0.0232	0.0198 1.1635 0.0198
	0.4	0.0470 0.3563 0.0352	0.0245 0.3845 0.0184	0.0174 0.4350 0.0131	0.0142 0.5136 0.0107	0.0126 0.6303 0.0094
	0.5	0.0303 0.1851 0.0182	0.0161 0.2082 0.0097	0.0119 0.2513 0.0071	0.0101 0.3227 0.0060	0.0093 0.4386 0.0056

These tables show a comparison for the value of $\Delta\tau$, $\Delta\kappa$, and Δv in each operating condition. For example, Table 2 when τ=0.2, κ=3.0, and v=0.2, we got $\Delta\tau$=0.0956, $\Delta\kappa$=2.9430, Δv=0.0637, shows the level of the change on the parameters that induce the same effect in increasing the path-tracking cost for $\Delta J=10^{-4}$. As τ is increased by 0.0956 sec (while the other parameters remain constant), it will produce the same result as increasing κ by 2.9430 m^{-1} or increasing v by 0.0637 m/s. Assumed that ΔJ should not be more that 10^{-4}, we should operate the system in the condition that τ, κ, and v have the parameter change less than 0.0956, 2.9430, and 0.0637 respectively. To consider if the system is too sensitive to any parameter or not, we need to consider the parameter change that happen in the actual system. We should consider operating the system at some other different operating condition if the level of the parameter change is larger than the value of the calculated $\Delta\tau$, $\Delta\kappa$, or Δv for current operating condition. For example, we can consider changing the robot

speed such as using a lower speed or designing a new tracking path with different path curvature so that the mobile robot operates in a different operating condition that is less sensitive to the system parameter change.

V. CONCLUSION

This chapter proposes an analytical sensitivity analysis of a NBC mobile robot path-tracking system according to operating environment. The analytical results are obtained from a mathematical formulation of a path-tracking scenario. The sensitivity data can be used to determine how the path-tracking cost is sensitive to each system parameter and which parameter must be carefully concerned. The sensitivity data also provides us a guideline on how to choose an operation condition for network control system (NCS) problem given a magnitude of the change of the system parameter.

REFERENCES

[1] A. Saltelli, K. Chan, and E. M. Scott, *Sensitivity analysis*. Chichester ; New York: Wiley, 2000.

[2] M.-Y. Chow and S. O. Yee, "Robustness Test of an Incipient Fault Detector Artificial Neural Network," Proceeding of the 1991 International Joint Conference on Neural Networks, Seattle, Washington, 1991.

[3] W.-L. Leung, R. Vanijjirattikhan, Z. Li, L. Xu, T. Richards, B. Ayhan, and M.-Y. Chow, "Intelligent space with time sensitive applications," Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, California USA, 2005.

[4] Y. Tipsuwan and M.-Y. Chow, "Control Methodologies in Networked Control Systems," *Control Engineering Practice*, vol. 11, pp. 1099-1111, 2003.

[5] W. J. Karnavas, P. J. Sanchez, and A. T. Bahill, "Sensitivity analyses of continuous and discrete systems in the time and frequency domains," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, pp. 488-501, 1993.

[6] R. Vanijjirattikhan, M.-Y. Chow, and Y. Tipsuwan, "Feedback Preprocessed Unmanned Ground Vehicle Network-Based Controller Characterization," Proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society (IECON '04), Busan, Korea, 2004.

CHAPTER IV

FEEDBACK PREPROCESSED UNMANNED GROUND VEHICLE NETWORK-BASED CONTROLLER CHARACTERIZATION

Rangsarit Vanijjirattikhan, Mo-Yuen Chow

rvanijj, chow@ncsu.edu

Advanced Diagnosis Automation and Control Lab

Department of Electrical and Computer Engineering

North Carolina State University, Raleigh NC 27695, USA

This chapter is partially published in the proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society (IECON '04), Busan, Korea, 2004.

FEEDBACK PREPROCESSED UNMANNED GROUND VEHICLE NETWORK-BASED CONTROLLER CHARACTERIZATION

Abstract **-** This chapter characterizes the performance of a Feedback Preprocessor (FP) that is used to alleviate the effect of the delay time on an Unmanned Ground Vehicle (UGV) path tracking problem in Network-Based Control (NBC) environment. FP estimates the UGV state in order to compensate for the effect of a constant time delay induced by the network. FP technique can be implemented as an external module appending to an existing nonlinear control system without having to redesign the controller. This technique uses the UGV kinematics to estimate the future position of the UGV under network-based control. The overall NBC system performance is investigated via simulation in terms of UGV characteristics and network time delay magnitudes. The approach and results from this paper can be used to estimate the performance of a similar UGV operating in a NBC environment. An intermediate parameter, *UGV response time*, may be used to relate the result in this chapter to other UGV in general. This chapter is based on one of our papers published in the 30th Annual Conference of the IEEE Industrial Electronics Society (IECON '04) with a different set of the simulation result. The simulation result in this chapter is regenerated for a small-sized UGV that we focus on through out the dissertation.

I. INTRODUCTION

The application of communication networks can vary from short distance data communication within a single computer to global and massive information sharing over the Internet. Communication networks also take part in control system applications such as the control network in automobiles, teleoperation of robot arm manipulators, and distributed control of multiple robots. This network-based control (NBC) system, networked control system (NCS), or distributed control system [1-3] is an emerging topic that has gained much attention during the last decade.

Control applications using communication networks can reduce investment and maintenance cost for wiring complexity (e.g., in factory automation), enable teleoperations, and render new control concepts and applications such as iSpace [4]. Among all the Network-Based Control systems, one major challenge is the existence of a network delay that might degrade the overall system performance and even destabilize the closed-loop control system in the network. Several techniques have been proposed to reduce the effect of network delay [1, 5, 6], most of which are only appropriate for linear systems or some specific forms of nonlinear systems.

One major area of application in NBC is teleoperation, such as the Jason project [7] with an undersea robot exploring the famous sunken ship Titanic, the Mercury project [8] with the first on-line robotics arm to explore the terrestrial surfaces, and the Xavier project [9] with the first on-line mobile robot moving to a destination preset by the remote user. The history of the teleoperation systems has emerged along with the existence of the communication network. Among several existing techniques that deal with the time delay problem in

teleoperation, *Gain Scheduler Middleware* (GSM) is a distinguished one that can improve system performance with less modification on the system. Tipsuwan and Chow have proposed the use of GSM to compensate for the effects of communication network delay in the Network-Based Control (NBC) path tracking problem on Unmanned Ground Vehicle (UGV) in [5]. In this paper, we focus on the system estimation module in the GSM, the Feedback Preprocessor (FP), and investigate the effects of different time delay magnitudes and UGV characteristics on the overall teleoperation performance.

FP approach is based on the concept introduced in the famous Smith predictor [10] that has been widely accepted for dealing with processes having dead-time [11, 12]. The concept of Smith predictor has been adopted and applied with several other techniques such as wave variable, Kalman filter, and passivity theory to reduce the effect of time delay [13, 14]. Similar to the Smith predictor approach, FP approach uses an estimated plant state to compensate for the effect of time delay. However, the structure of FP is more direct and simple. FP approach can be augmented to an existing controller to compensate for the network delay effect, even though the controller had not been designed for the NBC environment. This paper will investigate the effect of a Feedback Preprocessor (FP) module from the GSM to alleviate the effect of constant delay time in a NBC path tracking problem on the UGV. Preliminary results on this problem with different UGV characteristics under various network delay magnitude will be described and discussed.

This paper is organized as follows: in section II, the problem is formulated, section III describes the model of FP and the UGV state estimation mathematically, section IV discusses simulation results, and the paper is concluded in section V.

II. SYSTEM DESCRIPTION

UGV tracking paths can be categorized into static and dynamic paths. For static path, the guidelines are statically installed on the landmark. For dynamic path, the path is dynamically programmed based on the perception system such as real-time image processor, range sensors, sonars, or dead reckoning techniques. This paper will focus on a UGV tracking a dynamic path where the information of the path and the position of the UGV are assumed known by the controller. This information is used to calculate the control signals to drive the wheel of the UGV to track the path. The overall system of the UGV path tracking control over a communication network is depicted in Fig. 1.

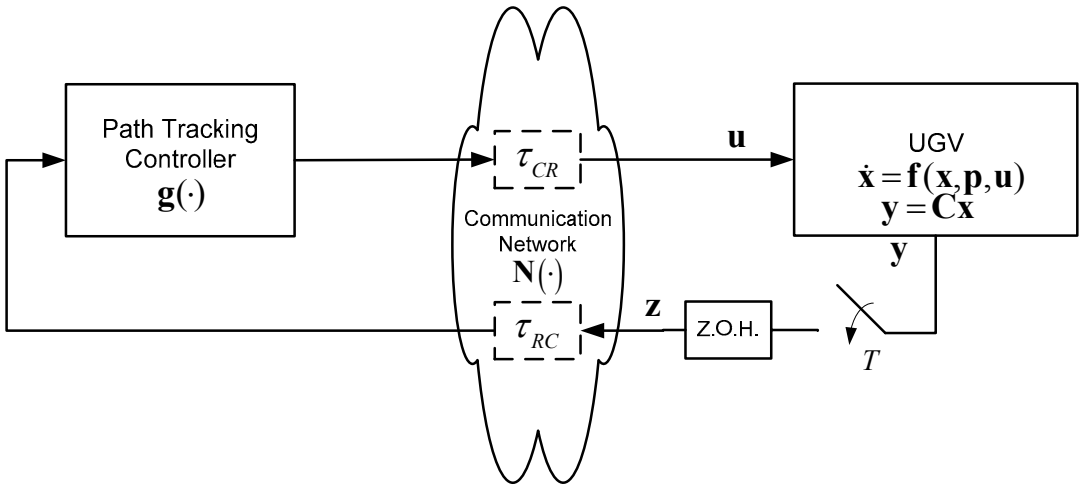

Fig. 1. UGV path tracking network-based control system.

As shown in Fig. 1, the system is composed of the UGV, the communication network $\mathbf{N}(\cdot)$ with the delay time τ_{CR} and τ_{RC}, and the path tracking controller $\mathbf{g}(\cdot)$. The communication network considered here is a packet switching network where the data is transferred in chunk from time to time. The communication network separates the system

into a remote plant site and a central controller site. The UGV on the remote plant site can be modeled by state space description $\mathbf{f}(\cdot)$ as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left(\mathbf{x}, \mathbf{p}, \mathbf{u}\right) \tag{1}$$

where $\mathbf{x} = \left[x_1, \cdots, x_{n_1}\right]^{\mathrm{T}} \in \mathbf{R}^{n_1}$ is the UGV states; $\mathbf{p} = \left[p, \cdots, p_{n_2}\right]^{\mathrm{T}} \in \mathbf{R}^{n_2}$ is the system parameters. The UGV receives the control signal from the path tracking controller denoted by $\mathbf{u} = \left[u_1, \cdots, u_{n_3}\right]^{\mathrm{T}} \in \mathbf{R}^{n_3}$. The output signal of the UGV is a subset of its states denoted by $\mathbf{y} = \mathbf{C}\mathbf{x}$.

Before sending the output signal of the UGV through the communication network to the path tracking controller, the output signal is sampled and held every constant period *T*. Denoted as **z**, the sample and hold UGV output signal is sent to the path tracking controller and used as the input for calculating control signals. Then, the control signal is sent back to the UGV also through the communication network. The calculation of the control signal can be described by Eqn. (2).

$$\mathbf{u}(t^+) = \mathbf{g}\left(\mathbf{y}(t - \tau^{k_i})\right), \;\; t \in \left\{k_i T + \tau^{k_i}, i = 1, 2, 3, \ldots\right\} \tag{2}$$

where $k_i T$ is the time instant that the UGV output signal is sampled, $\tau^{k_i} = \tau_{RC}^{k_i} + \tau_{CR}^{k_i}$ is the round-trip-time delay associated with sampling index k_i, $\left\{k_1, k_2, k_3, \ldots\right\} \subset \left\{0, 1, 2, 3, \ldots\right\}$. T is assumed to be substantially small compared with τ^{k_i}. Note that this equation describes the control signal as perceived by the UGV. Since the UGV output signal is delayed by $\tau_{RC}^{k_i}$ before reaching the controller and the control signal is delayed by $\tau_{CR}^{k_i}$ before reaching the

UGV, the UGV will get the control signal based on the output of the UGV at time $t-\tau_{RC}^{k_i}-\tau_{CR}^{k_i}=t-\tau^{k_i}$ in the past. Since we use packet switching network in this model, the UGV will receive updated control signal only at a certain time instant ($k_iT+\tau^{k_i}$) when a new data packet has arrived. Then, UGV will use the arrived control signal until the next data packet comes. In this paper, we focus on the system that **z** will be sent out to the path tracking controller only after **u,** corresponding to the previous **z,** had arrived. Therefore, we enforce a constraint that $k_{i+1}T > k_iT+\tau^{k_i}$. Hence, $\left[k_iT, k_iT+\tau^{k_i}\right)\cap\left[k_{i+1}T, k_{i+1}T+\tau^{k_{i+1}}\right)=\varnothing$ and $\cup_{i=1}^{\infty}\left[k_iT+\tau^{k_i}, k_{i+1}T+\tau^{k_{i+1}}\right)=\left[t_0,\infty\right), t_0\geq 0$. The following sections describe each individual module in the UGV path tracking network-based control system.

A. Unmanned Ground Vehicle (UGV)

The UGV is composed of a chassis, two driving wheels, a free running support wheel, two motors, and two proportional-integral (PI) speed controllers. The reference UGV speed and reference turn rate (how fast the UGV turns) from the path tracking controller will be transformed to motor reference wheel speeds for PI speed controllers. Based on reference wheel speeds, PI speed controllers will provide electricity to drive the motors which generate torques to drive the wheels of the UGV. The simulator in this paper uses the UGV kinetics model to describe the behavior of UGV according to the driving torques. The UGV kinetics is described by the mathematical model of the non-holonomic two-driving wheeled mobile robot (illustrated in Fig. 2) from [15]. The model is restated in appendix A for convenient reference.

As shown in Fig. 2, the center of gravity (CG) of the UGV does not need to be the same

point as the center of turn CT. The wheels with the radius *r* and the chassis have their own mass and moment of inertia.

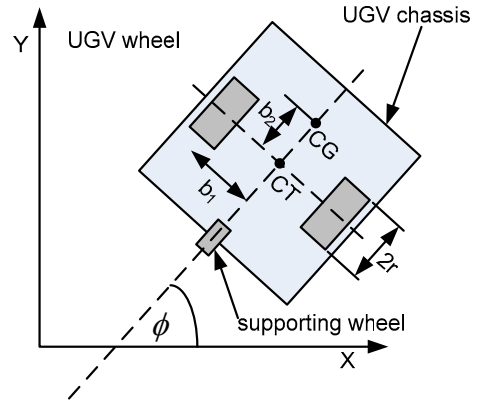

Fig. 2. Unmanned Ground Vehicle (UGV).

Combining the UGV kinetics model, the DC motor model, and the PI controller gives the UGV model that describes the behaviour of the UGV according to the reference speed and turn rate as shown in Eqn. (1). The state of the UGV composed of the state of UGV kinetics, dc motors, and PI controllers is described by $\mathbf{x} = \left[x\, y\, \theta_r\, \theta_l\, \dot{\theta}_r\, \dot{\theta}_l\, i_{a_r}\, i_{a_l}\, z_r\, z_l \right]^T$ where (x,y) is the UGV coordinate, θ and $\dot{\theta}$ are the angular displacement and angular velocity of the wheel, i_a is the state of the motor electrical dynamics, z is the state of the integrator in PI speed controller. *r* and *l* denote the right and the left wheel, respectively.

B. Path Tracking Controller

The path tracking controller ($\mathbf{g}(\cdot)$) generates the reference speed and reference turn rate, $\mathbf{u} = \left[v_{ref}, \omega_{ref} \right]^T$, for the UGV to track the path. Among the published path tracking algorithms, *pure pursuit* [16] is a well-known and one of the most efficient algorithm. Based on a simple geometric consideration, pure pursuit tracks the path by repeatedly fitting a

circular arc trajectory for the vehicle from its current position to a reference point on the path. As the vehicle moves forward, the reference point also moves forward along the path. This algorithm will generate the reference speed for both wheels of the vehicle to track the fitted arcs and will eventually drive the vehicle to track the path. However, the pure pursuit method does not specify "*how*" to choose the reference point on the path. Therefore, we use an alternative algorithm called *quadratic curve* algorithm [17]. This algorithm is slightly modified from the pure pursuit algorithm, in which it also provides a mean to choose an appropriate reference point **r** such that the UGV can effectively track a predefined path. The conceptual diagram of the quadratic curve algorithm is depicted in Fig. 3.

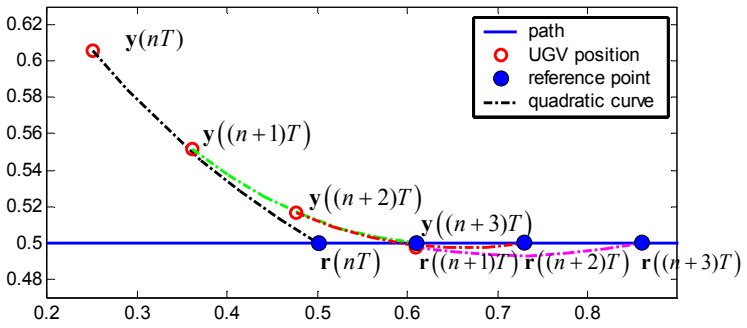

Fig. 3. Conceptual diagram for quadratic curve algorithm.

The UGV position is represented by a set of coordinate and a heading angle, $\mathbf{y} = [x\ y\ \phi]^T$. As shown in Fig. 3, the algorithm will drive the UGV position to track a parabolic curve. The curve is repeatedly drawn from the position of the UGV to the tracking path. The algorithm will drive the vehicle along the curve and eventually track the desired path [17].

The path tracking controller is embedded with a navigation module for providing an updated reference point $\mathbf{r} = [x_{ref}\ y_{ref}]^T$ for the UGV to follow. The path description is

assumed known to the navigation module and is described by

$$\mathbf{P}(d) \quad = [x_p(d) \;\; y_p(d)]^T = \mathbf{x}_p(d) \tag{3}$$

where $x_p(d)$ and $y_p(d)$ are the mathematical functions parameterised by d to describe the path. For example, a 90 degree turn path can be described by

$$x_p(d) \quad = \begin{cases} 0; & d \le 0.5 \\ d - 0.5; & 0.5 < d \le 1.5 \end{cases} \tag{4}$$

$$y_p(d) \quad = \begin{cases} d; & d \le 0.5 \\ 0.5; & 0.5 < d \le 1.5 \end{cases} \tag{5}$$

Given $\mathbf{x}_p(d)$, the navigation module acquires a new reference point for the UGV by looking for a point on the path that keeps a certain distance d_0 away from the UGV. Since d_0 can be determined from the quadratic curve algorithm, the new reference point is acquired by solving the following equation:

$$\sqrt{(x - x_p(d))^2 + (y - y_p(d))^2} = d_0 . \tag{6}$$

where d is the variable to be found and the reference point is $\mathbf{r} = [x_p(d) \; y_p(d)]^T$.

Because the reference point should only move forward along the path to guide the UGV to the destination, we enforce a constraint that our reference point will not move backward along the path. In the case that the UGV wanders away from the path or move backward, we will use the previous reference point against the new reference point acquired from Eqn. (6) that has a smaller value of d. Moreover, in addition to the original algorithm, we enforce a constraint to limit the minimum value of d_0 so that $d_0 \ge d_{\min}$. This can eliminate the situation where d_0 is too small causing the UGV to highly oscillate around the path.

III. FEEDBACK PREPROCESSOR

As depicted in Fig. 1, the controller and the UGV exchange data through the communication network, which is the main factor constituting signal delay. Fig. 4 illustrates a timing diagram of signals exchanged between the controller and the UGV.

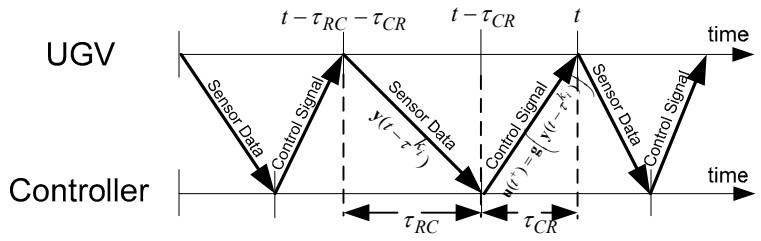

Fig. 4. Timing diagram for data transmission between UGV and the controller.

At time $t \in \left\{k_i T + \tau^{k_i}, i = 1,2,3,...\right\}$ the UGV receives the control signal $\mathbf{u}(t^+) = \mathbf{g}\left(\mathbf{y}(t-\tau^{k_i})\right)$ from the controller (as described in Eqn. (2)). The position $\mathbf{y}$ of the UGV at time $t-\tau^{k_i}$ is employed in the calculation. Consequently, the UGV receives the control signal that is proper for the previous position of the UGV at time $t-\tau^{k_i}$. This is one of the reasons why the UGV path tracking performance is worse when the control signal and the sensor data are sent through a communication network.

Feedback Preprocessor (FP) can help to alleviate this time delay influence on the overall system performance by providing the controller with an estimated value of the UGV's current position $\hat{\mathbf{y}}$. Considering the same timing diagram as described previously, at time *t* the UGV will receives the control signal $\mathbf{u}(t)$ that was calculated from $\hat{\mathbf{y}}(t) \cong \mathbf{y}(t)$ instead of $\mathbf{y}(t-\tau^{k_i})$. $\hat{\mathbf{y}}(t)$ predicts $\mathbf{y}(t)$ as shown in Eqn. (7):

$$\hat{\mathbf{y}}(t^{+}) = \boldsymbol{\psi}\left(\mathbf{y}(t-\tau^{k_i}), \mathbf{u}(t-\tau^{k_i}), \hat{\tau}^{k_i}\right) \;,\; t \in \left\{k_i T + \tau^{k_i}, i = 1,2,3,\ldots\right\} \tag{7}$$

This relates to the case where the UGV receives the control signal $\mathbf{u}(t)$ relative to the current UGV position $\mathbf{y}(t)$ which virtually eliminates the effect of the network delay. FP can be embedded into the UGV path tracking NBC system in Fig. 1 as a result shown in Fig. 5.

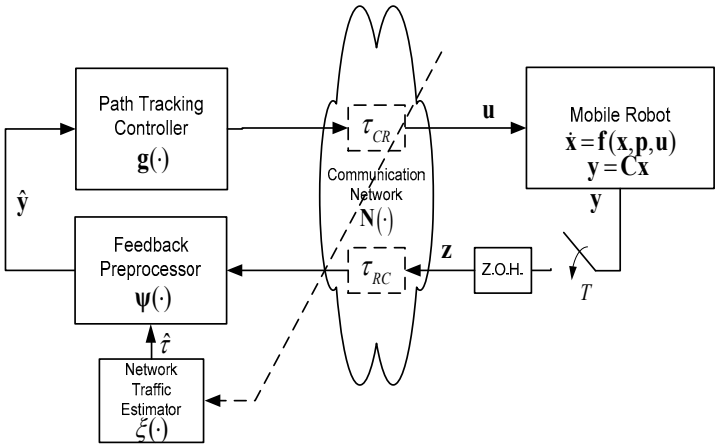

Fig. 5. UGV path tracking NBC system applied with Feedback Preprocessor.

Fig. 5 illustrates the overall system embedded with a FP. In order to predict the current UGV position $\mathbf{y}(t)$, FP requires three parameters, the control signal $\mathbf{u}(t-\tau^{k_i})$, the position of the UGV in the past $\mathbf{y}(t-\tau^{k_i})$, and the estimated round-trip-time delay $\hat{\tau}^{k_i}$. $\hat{\tau}^{k_i}$ can be estimated by the Network Traffic Estimator module $\xi(\cdot)$ from the history of data packet sent across the network (e.g. use an average value of previous delays). Because the functionality of FP is separated from the controller, one can implement FP without modifying the existing controller to compensate for the network delay effects.

FP uses kinematics model of the UGV to solve for an estimated UGV position as follows:

(1) Relocate terms and apply an integration

The kinematics model of the UGV describing how the UGV move according to the reference speed (v_{ref}) and reference turn rate (ω_{ref}) is described in Eqn. (8).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} v_{ref} \cos\phi(t) \\ v_{ref} \sin\phi(t) \\ \omega_{ref} \end{bmatrix} \tag{8}$$

where ϕ is the heading angle of the UGV. Note that the kinematics model concerns only the movement of the UGV and does not take into account the force and torque that drives the UGV. The UGV is assumed to be able to run accurately at the reference speed and reference turn rate. In the case that the actual speed and turn rate are known by the controller, they should be used in this model instead of reference values.

From the kinematics model, we can substitute $\begin{bmatrix} \dot{x} & \dot{y} & \dot{\phi} \end{bmatrix}^T$ by $\begin{bmatrix} \frac{dx}{dt} & \frac{dy}{dt} & \frac{d\phi}{dt} \end{bmatrix}^T$ and relocate terms. Then, we integrate both side of the equation as follows:

$$\int \begin{bmatrix} dx \\ dy \\ d\phi \end{bmatrix} = \int \begin{bmatrix} v_{ref} \cos\phi(t) \\ v_{ref} \sin\phi(t) \\ \omega_{ref} \end{bmatrix} dt \tag{9}$$

(2) Solve for estimated UGV position

Since $\mathbf{y}(t) = \begin{bmatrix} x & y & \phi \end{bmatrix}^T$, $\hat{\mathbf{y}}(t) = \begin{bmatrix} \hat{x} & \hat{y} & \hat{\phi} \end{bmatrix}^T$, and a constant control signal $\mathbf{u}(t-\tau^{k_i}) = \begin{bmatrix} v_{ref}(t-\tau^{k_i}) & \omega_{ref}(t-\tau^{k_i}) \end{bmatrix}^T$ is used by the UGV during period $\left[t-\tau^{k_i}, t \right)$, we can rewrite Eqn. (9) as:

$$\mathbf{y}(t) \cong \hat{\mathbf{y}}(t) \triangleq \mathbf{y}(t-\tau^{k_i}) + \int_{t-\hat{\tau}^{k_i}}^{t} \begin{bmatrix} v_{ref}(t-\tau^{k_i})\cos\phi(\zeta) \\ v_{ref}(t-\tau^{k_i})\sin\phi(\zeta) \\ \omega_{ref}(t-\tau^{k_i}) \end{bmatrix} d\zeta \tag{10}$$

Then, we can solve for $\hat{\phi}(t)$ given that τ^{k_i} is estimated by $\hat{\tau}^{k_i}$. We need to estimate the value of τ^{k_i} here because FP calculates $\hat{\mathbf{y}}(t)$ before the actual value of τ^{k_i} exists. After we solve for $\hat{\phi}(t)$, we need to replace $\hat{\phi}(t)$ back into Eqn. (10) and solve for $\hat{x}(t)$ and $\hat{y}(t)$ as the solution shown below.

$$\begin{aligned} \hat{\phi}(t) \quad &= \phi(t-\tau^{k_i}) + \omega_{ref}(t-\tau^{k_i}) \times \hat{\tau}^{k_i} \\ \hat{x}(t) &= x(t-\tau^{k_i}) + \frac{v_{ref}(t-\tau^{k_i})}{\omega_{ref}(t-\tau^{k_i})}\left(\sin\left(\phi(t-\tau^{k_i}) + \omega(t-\tau^{k_i}) \times \hat{\tau}^{k_i}\right) - \sin\phi(t-\tau^{k_i})\right) \\ \hat{y}(t) &= y(t-\tau^{k_i}) + \frac{v_{ref}(t-\tau^{k_i})}{\omega_{ref}(t-\tau^{k_i})}\left(\cos\phi(t-\tau^{k_i}) - \cos\left(\phi(t-\tau^{k_i}) + \omega(t-\tau^{k_i}) \times \hat{\tau}^{k_i}\right)\right) \end{aligned} \tag{11}$$

In the next section, we show the simulation result of the system in Fig. 5 for the round-trip-time, $\tau_{RC} + \tau_{CR}$, between 0.0 and 0.8 sec which should the range of intercontinental internet delay. The effects of using different UGVs with different masses and motor electromotive force constants are also illustrated.

IV. SIMULATION RESULTS AND ANALYSIS

This section shows the simulation result of the UGV tracking performance. The results will be analyzed with respect to representative parameters of the UGV and communication delay. We focus on the simulation result when the overall system operated:

(1) without the Feedback Preprocessor (FP), and

(2) with the Feedback Preprocessor (FP).

In each case, we investigated the time delay effects on UGV path tracking performance

with different masses (M), motor electromotive force constants (K_m), and round-trip-time delays (τ). Mass M reflects different sizes and/or payload of the UGV. M under consideration is in the range of [1.5, 15.5] kg, not including the wheels. Motor electromotive force constant K_m of the dc motors that are used to propel the UGV reflects how the responsiveness of the UGV with respect to its speed control. K_m under consideration is in the range of [0.25e-3, 2.00e-3] N·m/Amp. Constant round-trip-time delay τ reflects the time required for data communication between the UGV and the controller. τ under consideration is in the range of [0.0, 0.8] sec (since most Internet time delay is within this range, but is not constant).

The UGV in this study is a small size mobile robot with parameters as listed in Table 1 , Table 2, and Table 3.

Table 1. UGV chassis parameters.

Parameters	**values**
Distance between the driving wheels and the axis of symmetry (b_1)	0.0865 m
Radius of the wheel (r)	0.036 m
Mass of each wheel and rotor (m_w)	0.025 kg
Moment of inertia of each wheel about the wheel diameter (I_m)	3.375e-06 $kg \cdot m^2$
Moment of inertia of the wheel about the wheel axis (I_w)	1.620e-05 $kg \cdot m^2$

Table 2. Motor parameters.

Parameters	**values**
Electric resistance (R_{motor})	7.4 ohms
Electric inductance (L_{motor})	2e-3 H
Damper (b_{motor})	2e-7 $N \cdot m \cdot s$
Moment of inertia of the rotor (J_{motor})	1.5e-7 $kg \cdot m^2$
Gear ratio (N_{motor})	333.6375

Table 3. Quadratic curve path tracking parameters.

Parameters	values
Maximum look ahead distance (d_{max})	0.2473 m
Minimum look ahead distance (d_{min})	0.0771 m
Look ahead distance decreasing factor (β)	0.7509
Maximum speed (α)	0.156 m/s

A straight path with a 90 degree turn, as shown in Fig. 6, is used as the test path for our investigation. This test path is chosen because it resembles the step input function test for a classical single-input single output system so that we can investigate the "step response" of the UGV subjected to time delay.

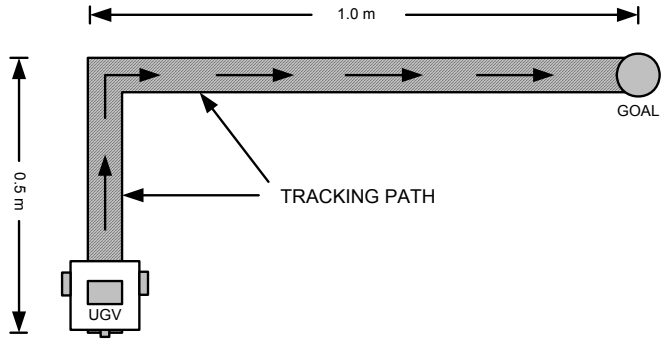

Fig. 6. The test path used for the time-delay effect analysis.

Two cost functions are used to evaluate the performance of the overall network-based control system. The first cost function, J_1, indicates the deviation of the UGV from the path:

$$J_1 = \int_{t_0}^{t_f} D(x(t), y(t))dt \tag{12}$$

where D is the shortest distance between the position of the UGV $(x(t), y(t))$ and the tracking path as depicted in Fig. 7. t_0 is the initial time preset as $t_0 = 0$, and t_f is the final time when the UGV reaches the destination. The second cost function, J_2, is the traveling time for the UGV to reach the destination:

$$J_2 = \ t_f - t_0 = t_f \tag{13}$$

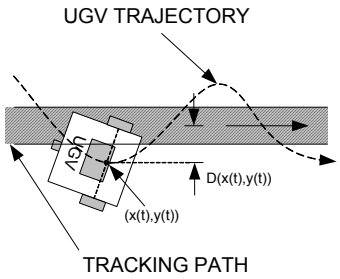

Fig. 7. The shortest distance between the UGV and the path.

A. Simulation Results

This section compares the results of the UGV under NBC path tracking simulation between with and without using the Feedback Preprocessor.

(1) Without Feedback Preprocessor (FP)

Fig. 8 shows the simulation result of the overall system without FP according to a range of the communication round-trip-time delay τ.

As the layered surface graphs of cost functions are higher with a larger τ, Fig. 8 indicates that J_1 and J_2 increase with increased τ, increased M, and decreased K_m. Both J_1 and J_2 have a similar shape due to the inherent definition of these cost functions. Since the UGV travels a longer distance, the deviation from the path J_1 will be larger when the traveling time J_2 is larger. The UGV trajectories according to several cases are also shown to indicate how the UGV behaves due to the changing of τ, M, and K_m. Conforming with Eqn. (12) and (13), the higher J_1 and J_2 indicate the higher overshoot and oscillation amplitude of the UGV trajectory.

(2) With Feedback Preprocessor

Fig. 9 shows the simulation of the overall system when operated with a feedback preprocessor.

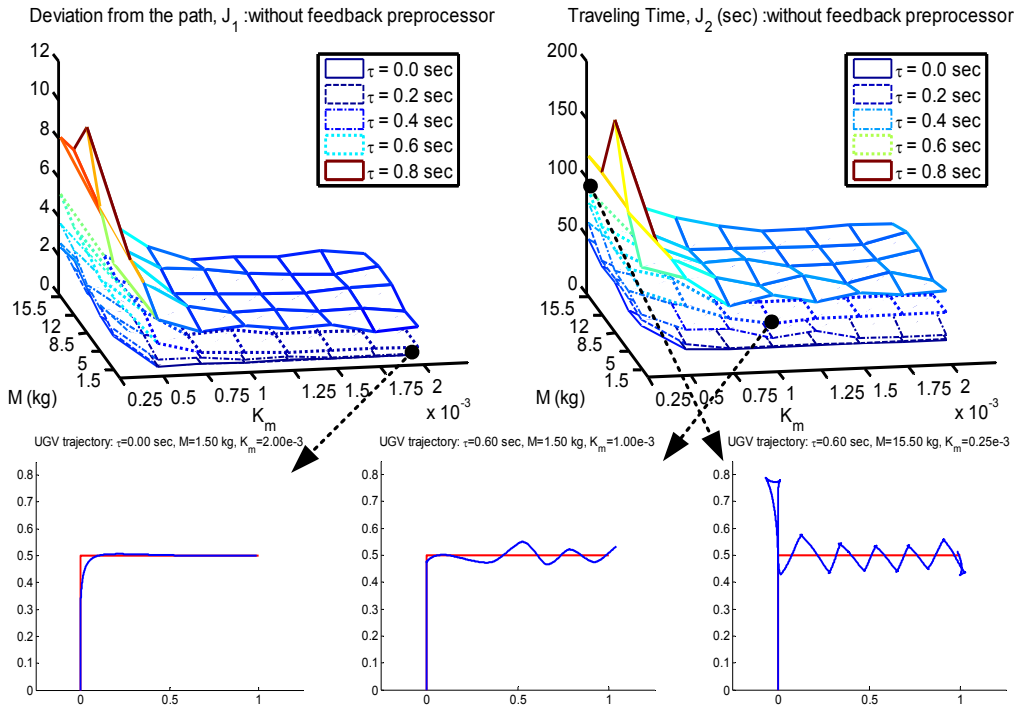

Fig. 8. Simulation Result, J_1 and J_2 (without Feedback Preprocessor).

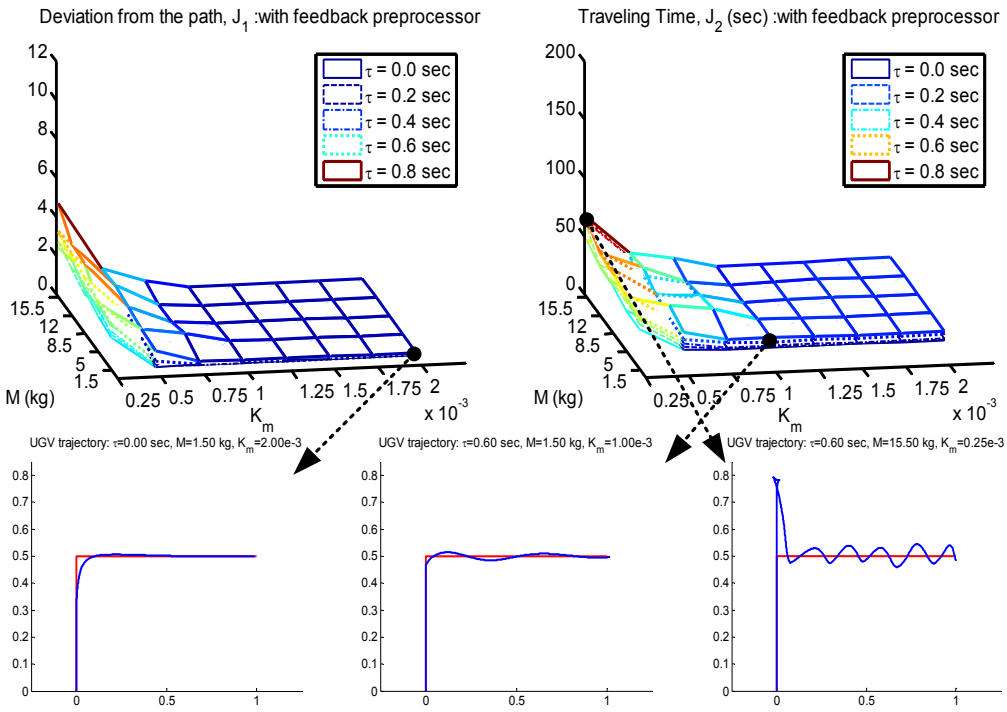

Fig. 9. Simulation Result, J_1 and J_2 (with Feedback Preprocessor).

One can observe from Fig. 9 that τ has less effect on J_1 and J_2 compared with the same situation in Fig. 8. This is deduced from the surface graphs that remain almost in the same level as τ increased. When the UGV mass is small and the motor electromotive force constant is large, the model works extremely well. This confirms the effectiveness of FP to alleviate the effect of time delay in the UGV path tracking problem. The UGV trajectories are also shown with less overshoot and oscillation.

B. Performance Analysis with Respect to UGV Characteristics

Since UGVs can have different characteristics (e.g., wheel radius, distance between the wheels, mass, moment of inertia, motor characteristics, motor power, etc.), it is difficult to determine the factors that affect the NBC path tracking performance. In order to investigate the path tracking problem, we use the *UGV response time* to characterize the UGV.

In this paper, we define the UGV response time T_{UGV} as *"the time required for the UGV angular wheel speed to reach 50% of the angular step reference speed"* as shown in Eqn. (14).

$$\boldsymbol{\omega}(T_{UGV}) = 0.5 \times \boldsymbol{\omega}_R \tag{14}$$

The mapping from UGV characteristics to T_{UGV} can be determined from a typical UGV step response of the UGV wheel speed as shown in Fig. 10.

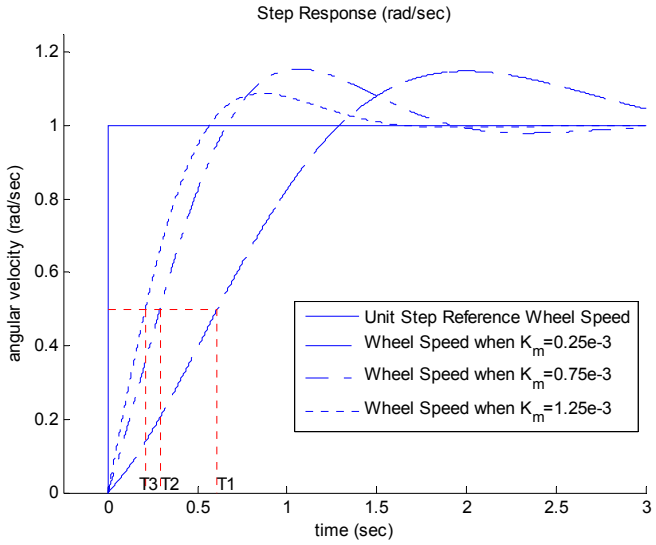

Fig. 10. Step response of UGV wheel speed with response time T_1, T_2, T_3 according to mass M = 1.5 kg and various motor electromotive force constants K_{m}.

Fig. 10 shows the step responses of the UGV with predefined parameters as shown in Table 1-3 and various values of motor electromotive force constant K_m. One can easily plot this step response graph from an experiment by applying desired wheel speed as the reference input to the UGV and measure the actual wheel speed. The time required for the UGV to start moving and reach 50% of the reference speed is the UGV response time. Fig. 11 shows T_{UGV} with respect to different UGV masses (M) and different K_{m}.

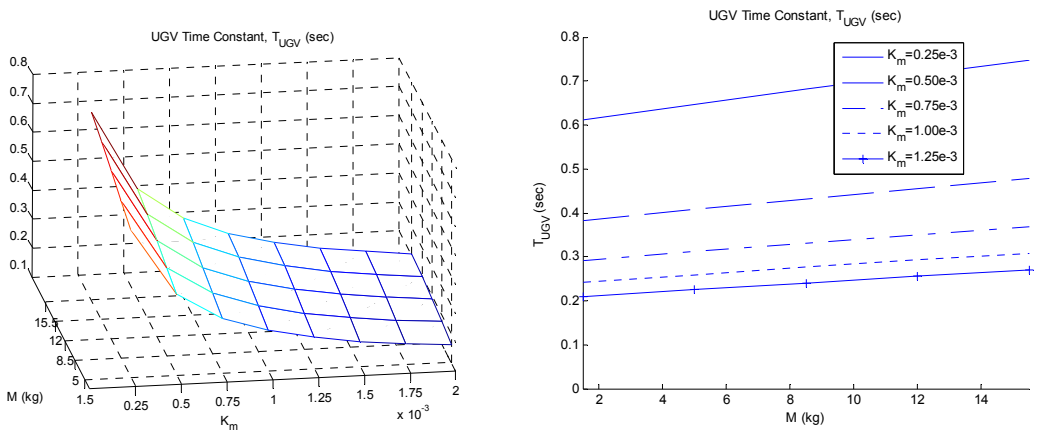

Fig. 11. UGV response time with various M and K_{m}.

Fig. 11 clearly indicates that T_{UGV} is proportional to the mass M and inversely proportional to the motor electromotive force constant K_m. The results agree with heuristics that the heavier the UGV is, the more time it requires to pick up speed; in addition, the more powerful the motor is, the faster the UGV can pick up speed.

By mapping from M and K_m to T_{UGV}, the simulation result in Fig. 8 and Fig. 9 can be plotted based on T_{UGV} and τ as depicted in Fig. 12. The surface plots in Fig. 12 shows the mapping between T_{UGV} and the average cost corresponding to T_{UGV} and τ.

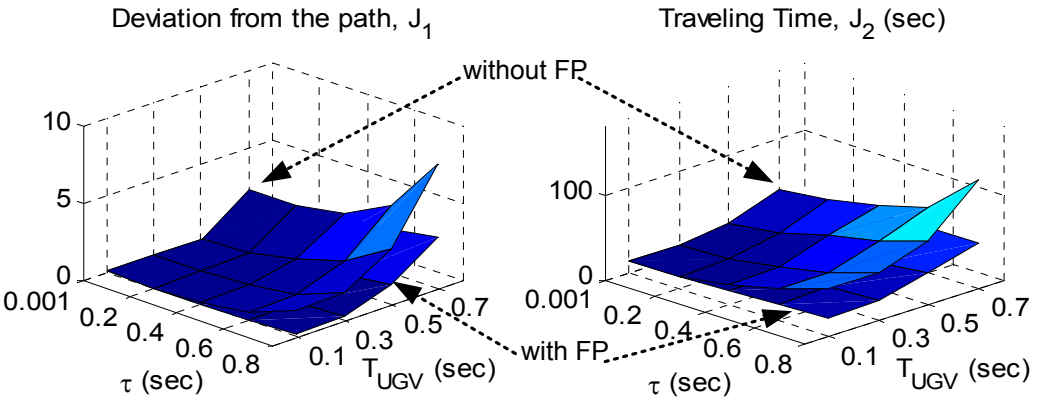

Fig. 12. J_1 and J_2 with respect to T_{UGV} and τ.

One can notice from Fig. 12 that, when the path tracking controller operate without FP,

the cost function J_1 and J_2 increase as T_{UGV} and τ increase; when operated with FP, the cost is only slightly changed due to T_{UGV}. The contour plots illustrated in Fig. 13 shows a more obvious comparison.

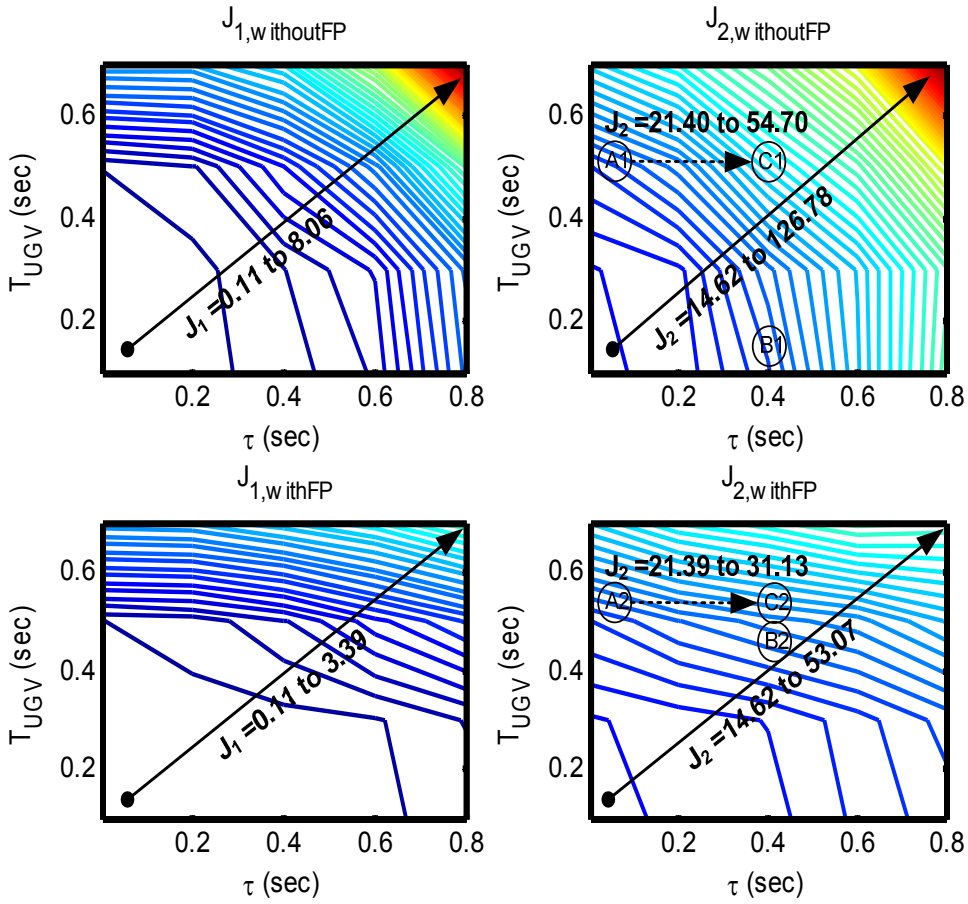

Fig. 13. Contour plots of J_1 and J_2 with respect to T_{UGV} and τ.

The contour plots in Fig. 13 clearly illustrate the effect of the cost function according to the UGV response time T_{UGV} and the network delay time τ. The contour lines show the parameter regions that have the same cost value. The plots provide additional information especially for FP case indicating how different the cost functions increase with increasing T_{UGV} and τ; the delay time τ clearly has less effect on the cost functions as J_1 and J_2 increase

slower according to the increasing τ compared with the without FP case.

Fig. 13 also provides a relationship between T_{UGV}, τ and J_i. This relationship can be used as a guideline to estimate the behaviour of a UGV under NBC path tracking system. For example, when the operating point is changed from τ=0.0 to 0.4 sec, a UGV with T_{UGV}=0.5 sec operating without FP at point A_1 must be considered to be modified (e.g., decrease the load, increase the motor performance) such that T_{UGV} is decreased from 0.5 to 0.1 sec to operate at point B_1 in order to maintain the performance; otherwise the system will operate at point C_1 and the cost J_2 will be increased from 21.40 to 43.50 sec in which the UGV spends more than twice of the original tracking time. When using FP at τ=0.4 sec, the UGV operates at point C_2. The UGV uses the tracking time closed to the no delay system case at point B_2 without any modification on T_{UGV}.

V. CONCLUSION

This paper has investigated the effects of using a Feedback Preprocessor (FP) based on the concept of the Smith predictor in a NBC path tracking problem on a UGV. The problem is formulated mathematically along with the structure and the description of FP. FP estimates the UGV state in the future to be used by the path-tracking controller in order to compensate the effect of time delay induced by the network. FP augments an existing controller, designed to operate in delay-free condition, to externally compensate the time delay effect for Network-Based Control applications without redesigning the controller.

Simulation results have affirmed the effectiveness of using FP to reduce the time delay effect on the NBC path tracking problem. The approach and result in this paper can be used to estimate the performance of any similar UGV operating in a NBC environment. An

intermediate parameter, *UGV response time*, may be used to relate the result in this paper to other UGVs in general. It provides a means to estimate the performance of the UGV path-tracking performed under different UGV characteristics such as different UGV's weights and motor electromotive force constants.

VI. ACKNOWLEDGMENT

The authors would like to thank Danny Leung for his reviews and several helpful suggestions.

REFERENCES

[1] M.-Y. Chow and Y. Tipsuwan, "Network-based control systems: a tutorial," *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society (IECON '01)*, Denver, CO, 2001, vol. 3, pp. 1593 -1602.

[2] G. C. H. Y. Walsh, "Scheduling of networked control systems ," *IEEE Control Systems Magazine*, vol. 21, pp. 57-65, 2001.

[3] F.-L. Lian, J. Moyne, and D. Tilbury, "Network design consideration for distributed control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, No. 2, pp. 297-307, 2002.

[4] H. Hashimoto, "Intelligent space - How to make spaces intelligent by using DIND?," *Proceedings of 2002 the IEEE International Conference on Systems, Man and Cybernetics*, Yasmine Hammamet, Tunisia, Oct 6-9 2002, vol. 1, pp. 14-19.

[5] Y. Tipsuwan and M.-Y. Chow, "Gain Scheduling Middleware for Networked Mobile Robot Control," Proceedings of American Control Conference 2004 (ACC2004), Boston, MA, Jun 30 - Jul 2 2004.

[6] Y. Tipsuwan and M.-Y. Chow, "Control Methodologies in Networked Control Systems," *Control Engineering Practice*, vol. 11, pp. 1099-1111, 2003.

[7] Jason Project. http://www.jasonproject.org/.

[8] K. Goldberg and R. Siegwart, *Beyond webcams : an introduction to online robots*. Cambridge, MA: MIT Press, 2002.

[9] R. Simmons, J. L. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan, "Lessons learned from Xavier," *IEEE Robotics & Automation Magazine*, vol. 7, No. 2, pp. 33-39, 2000.

[10] O. J. M. Smith, "A controller to overcome dead-time," *Instrument Society of America Journal*, vol. 6, No. 2, pp. 28-33, 1959.

[11] K. J. Astrom, C. C. Hang, and B. C. Lim, "A new Smith predictor for controlling a process with an integrator and long dead-time," *IEEE Transactions on Automatic Control*, vol. 39, No. 2, pp. 343-345, 1994.

[12] J.-Q. Huang and F. L. Lewis, "Neural-network predictive control for nonlinear dynamic systems with time-delay," *IEEE Transactions on Neural Networks*, vol. 14, No. 2, pp. 377-389, 2003.

[13] G. Niemeyer and J.-J. E. Slotline, "Stable adaptive teleoperation," *IEEE Journal of Oceanic Engineering*, vol. 16, No. 1, pp. 152-162, 1991.

[14] S. Munir and W. J. Book, "Internet-based teleoperation using wave variables with prediction," *IEEE/ASME Transactions on Mechatronics*, vol. 7, No. 2, pp. 124-133, 2002.

[15] X. Yun and Y. Yamamoto, "Internal dynamics of a wheeled mobile robot," *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, 1993, vol. 2, pp. 1288-1294 vol.2.

[16] A. Ollero and G. Heredia, "Stability analysis of mobile robot path tracking," *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95)*, Pittsburgh, PA, 5-9 Aug 1995 1995, vol. 3, pp. 461-466.

[17] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path tracking control of mobile robots using a quadratic curve," *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, 1996, pp. 58-63.

CHAPTER V

MOBILE AGENT GAIN SCHEDULER CONTROL IN INTELLIGENT SPACE

Rangsarit Vanijjirattikhan, Mo-Yuen Chow

rvanijj,chow@ncsu.edu

Advanced Diagnosis Automation and Control Lab

Department of Electrical and Computer Engineering

North Carolina State University, Raleigh NC 27695, USA

The material in this topic is partially published in the proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA05), Barcelona, Spain, April 18-22, 2005.

MOBILE AGENT GAIN SCHEDULER CONTROL IN INTELLIGENT SPACE

Abstract **–** Intelligent Space (iSpace) is a space (room, corridor, or street), with distributed sensory and mobile agents, that are capable of providing intelligent services. In this chapter, the iSpace prototyping project at North Carolina State University is introduced. The current infrastructure of the project has the capability to command the mobile agent to automatically move payload from current position to a desired position. The research topic to be focused on is the remote mobile robot path-tracking problem where the mobile agent is controlled over communication network. The presence of a network time-delay, more often than not, compounds to the challenging nature of the topic. A network-based control technique called Gain Scheduler Middleware is employed in the control loop to alleviate the effect of time delay. A high fidelity simulation model is proposed in order to simulate the iSpace project at NCSU. The simulation results of the mobile robot path-tracking is used to demonstrate the effectiveness of using Gain Scheduler Middleware to compensate the time delay effect on remote mobile robot path-tracking control over the Internet. Variations of the GSM method with one-dimensional and two-dimensional gain tables are compared in order to show the effectiveness and limitations of the method.

keywords: *Intelligent Space, Networked Control System, Tracking Control, Gain Scheduler Middleware*

I. INTRODUCTION

Intelligent Space (iSpace) [1] is a relatively new concept to effectively use distributed sensors, actuators, robots, computing processors, and information technology over a physically connected space or spaces connected using communication networks, to provide intelligent services to human beings. The space can be a room, a corridor, a hospital, an office, or even a planet. iSpace fuses *global information* within the space of interest to effectively and efficiently make intelligent decisions. For example, when Captain Picard in StarTrek says "coffee", the voice recognition module in iSpace would send a command to the central controller, where it computes and processes any necessary procedures to send a mobile robot to bring a cup of coffee from the coffee maker to Captain Picard. When iSpace research development matures, this "Enterprise dream" will soon become a reality. Another example is in a manufacturing plant where action agents are needed to automatically convey payload from point A to point B as shown in Fig. 1. The space is continuously surveyed and monitored by distributed sensors such as video cameras, acoustic sensors, biosensors, etc. The distributed sensors will continuously send information (e.g., video camera images) to control agents which will analyze data, extract features, and fuse the measured information from the images and the information stored in database (e.g., equipment location, feasible pathway) to extract the current operating conditions. Then the control agents will do a real-time path planning and send the reference signal for the action agent (e.g., a mobile robot) to carry a payload from point A to point B.

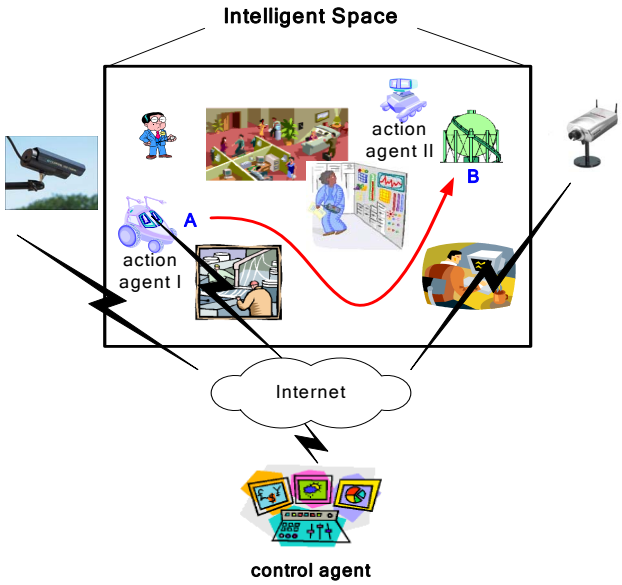

Fig. 1. Intelligent space in manufacturing plant.

Advanced Diagnosis, Automation, and Control (ADAC) Lab in the Department of Electrical and Computer Engineering at North Carolina State University has developed an iSpace prototyping project "Ana2 plays fetch in iSpace" as shown in Fig. 2. In this project, a space with obstacles is continuously monitored by a sensor (webcam in this case). A mobile robot named Ana2 is sent a control signal to automatically and autonomously move as quickly as possible, while avoiding collisions, from its current location to a specific destination chosen by a user remotely. This project can be applied in industrial automation where the mobile robot is needed to automatically carry payload from one location to another location without the need for constant human monitoring and control.

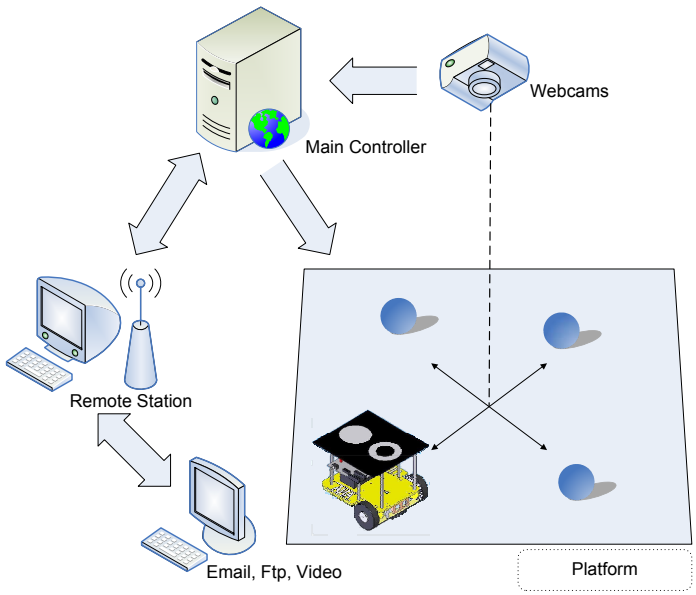

Fig. 2. iSpace at NCSU prototype configuration.

There exists a time delay within the control loop of iSpace at NCSU caused by computational devices, communication over the network, image acquisition, and image processing. Time delay becomes an important issue once we start to address real-time control. In general, time delay issues can cause network-based performance degradation and instability of the closed-loop system. Among the delays, the computation delay is usually bounded and well behaved so that we can design controllers to compensate for the computational delay accordingly. The communication delay, on the other hand, is more challenging to handle as it is random and unbounded.

In [2], we apply a technique proposed in [3] called Gain Scheduler Middleware to alleviate the delay time problem in iSpace. The results show that the mobile robot can successfully track the path from the starting point to the destination even if only a scalar gain is employed in gain scheduler module. In this chapter, we extend the previous study by using

a variation of one dimensional and two dimensional gain tables in the gain scheduler module instead of using a scalar gain. We investigate the advantage and limitation of using these gain tables about how they alleviate the time delay problem. This chapter also proposes a high fidelity simulation program for iSpace at NCSU for generating simulation results. The mass and moment of inertia of the robot are considered in the simulation for obtaining a result which will be very close to the actual real world experience.

This chapter is organized as follows: In section II, the Intelligent Space project at NCSU is explained. In section III, the overall system is described including the model of the mobile robot so called Unmanned Ground Vehicle and the Quadratic curve path-tracking controller. Section IV describes the Gain Scheduler Middleware to be applied to the remote UGV path-tracking. Section V illustrates the simulation result and provides discussion. The chapter is concluded in section VI.

II. INTELLIGENT SPACE AT NCSU

The overall structure of iSpace at NCSU is depicted in Fig. 3, which shows the information flow among all the components of iSpace, including actuators, sensors, the main controller (a control agent), and the network communication structure.

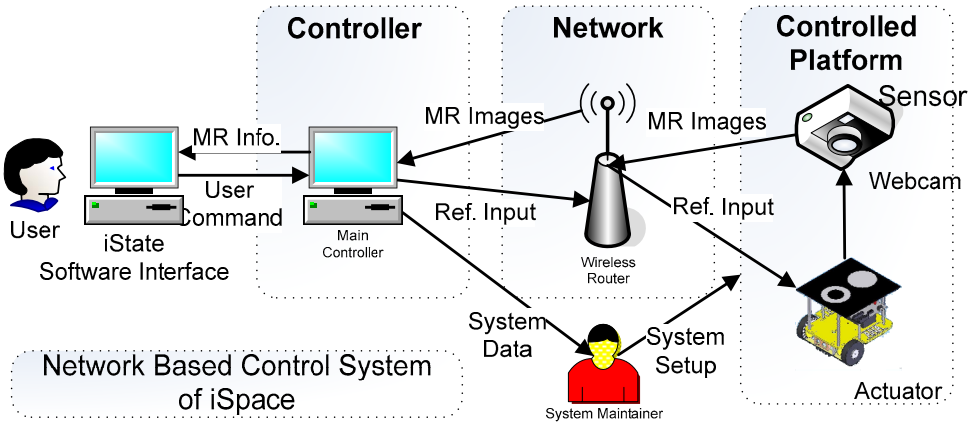

Fig. 3. iSpace structure diagram.

The project involves hardware, software, and communication network modules. The software portion of the project includes the image acquisition, image processing, path generation, and path tracking controller modules. The hardware portion includes sensors (webcam), and actuators (motors on Ana2). The communication network portion includes wired and wireless IP connection between the microcontroller board on Ana2 and the remote computer controller. This project has realized and developed an iSpace infrastructure to investigate research related to time sensitive network-based control and teleoperation [4-5]. It can be used to demonstrate how iSpace can make superior decisions based on global information from distributed sensors to control the actuators (Ana2) to complete a given task. This system is designed such that "iSpace at NCSU" may be controlled from anywhere in the world. The iSpace project is therefore transformed from a locally connected space to a virtually connected space.

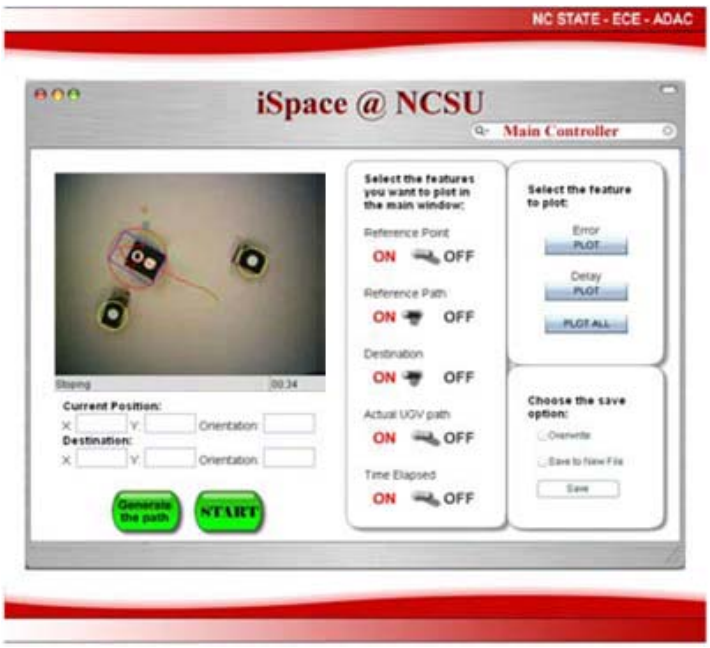

Fig. 4. Main controller GUI program.

To demonstrate the effects of time-sensitive applications in iSpace, the project uses the game of a dog playing fetch with its owner. The goal of the game is to intelligently monitor an area so that a mobile robot can be commanded to go from its current location to a specific destination as quickly as possible while avoiding collisions. The user can specify the mobile robot destination with a mouse click on the map displayed on the Main Controller GUI Program, as shown in Fig. 4. To accomplish this goal, this project utilized image processing techniques, as well as path generation and path tracking algorithms to command the mobile robot to go from a starting point to a destination point while avoiding collisions with obstacles. In this chapter, we focus on the problem of controlling the mobile robot tracking a predefined path when there is communication delay in the control loop. This is an important part in realizing iSpace since the delay time can have an adverse effect on path-tracking performance. We consider that there are two parts of the communication delay, the delay

when the data is sent from central controller to the remote plant (mobile robot) τ_{CR} and the delay when the data is sent from the remote plant to central controller τ_{RC}. τ_{RC} includes the delay from image acquisition and image processing in addition to the internet delay while τ_{CR} only includes the internet delay when the data is sent from the central controller to the robot. Here, we assume that the path to be tracked is passed to the path-tracking controller from a path generation module. We apply a variation of Gain scheduler middleware method proposed in [3] to alleviate the effect of time delay.

III. GAIN SCHEDULER MIDDLEWARE

In recent years, there has been a significantly amount of emphasis and effort on developing middleware. Middleware can be viewed as a technology to seamlessly link applications and users over a network while making the network transparent to the users. A Gain Scheduler Middleware (GSM) has been developed by Tipsuwan and Chow [3-6] to alleviate the network time delay effect on network-based control systems. The GSM handles all network connections between the controller and the remote system to be controlled over a network. These include network operations such as sending and receiving packets, bandwidth and resource allocation, network traffic monitoring, etc.

This section will briefly explain the features and properties of GSM where the system dynamics of a remote system (to be controlled) is described as:

$$\dot{\mathbf{x}}_R = \mathbf{f}_R(\mathbf{x}_R, \mathbf{p}_R, \mathbf{u}_R) \tag{1}$$

$$\mathbf{y}_R = \mathbf{h}_R(\mathbf{x}_R, \mathbf{p}_R, \mathbf{u}_R) \tag{2}$$

and a general form of the controller not including GSM can be described by:

$$\mathbf{u}_C = \mathbf{g}_C(\mathbf{y}_R, \mathbf{p}_C), \tag{3}$$

where $\mathbf{x}_R \in \mathbf{R}^n$ is the state variable of remote system, $\mathbf{y}_R \in \mathbf{R}^m$ is the remote system output, $\mathbf{u}_C \in \mathbf{R}^z$ is the control signal from the central controller, $\mathbf{u}_R \in \mathbf{R}^z$ is the delayed control signal when arrived at the remote system, $\mathbf{p}_R \in \mathbf{R}^w$ is the remote system parameters, $\mathbf{p}_C \in \mathbf{R}^r$ is the controller gains. A method to compensate network delay effects is to adapt the controller gain $\mathbf{p}_C$ by $\gamma \in \mathbf{R}^+$ using a gain scheduling approach. This paper introduces an external gain scheduling method to enable existing controllers for networked control with variable time delay. The main concept of this approach is to find a $\beta \in \mathbf{R}^+$ gain such that:

$$\beta \mathbf{u}_C = \beta \mathbf{g}_C(\mathbf{y}_R, \mathbf{p}_R) \cong \mathbf{g}_C(\mathbf{y}_R, \gamma \mathbf{p}_R). \tag{4}$$

The β gain adjusting the controller output $\mathbf{u}_C$ from the outside is equivalent to adjusting the controller gain $\mathbf{p}_C$ by the gain γ from the inside of the controller [7]. This β gain can be determined from different optimal objectives, and be applied on $\mathbf{u}_C$ to compensate network delays with respect to $\mathbf{x}_R$, $\mathbf{u}_C$, $\mathbf{p}_C$ and $\mathbf{q}$ where $\mathbf{q} \in \mathbf{R}^d$ is the network variable representing network traffic conditions. Network variables represented in $\mathbf{q}$ can be statistics of network delays such as mean delay, delay variance, loss rate, and other network QoS (Quality of Service) variables. A schematic diagram of the GSM is shown in Fig. 5.

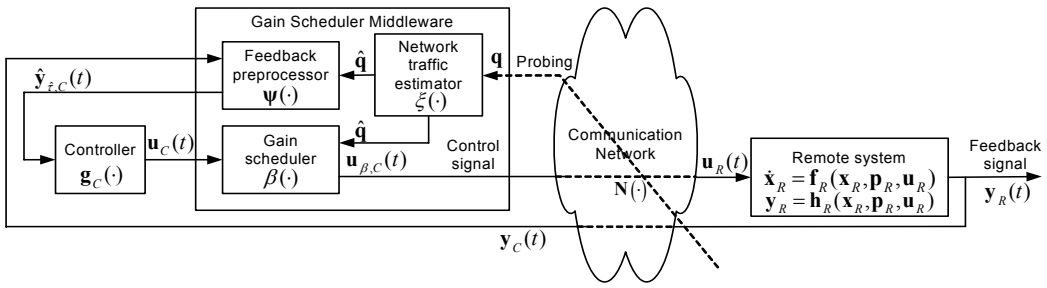

Fig. 5. A schematic diagram of Gain Scheduler Middleware (GSM).

The components of the GSM shown in Fig. 5 are Network traffic estimator, Feedback pre-processor, and Gain scheduler.

(1) Network traffic estimator (NTE). The network traffic estimator estimates the current network traffic conditions, which are characterized by the network variable $\mathbf{q}$. Various data in $\mathbf{q}$ such as mean delay and loss rate are then utilized by feedback preprocessor and gain scheduler. The network traffic estimator monitors the network conditions by sending probing packets to the remote system periodically (e.g., once a second). The estimator then characterizes the network conditions by updating the estimated network variable $\hat{\mathbf{q}}$ based on the monitored probing packet roundtrip measurements.

(2) Feedback pre-processor (FP). The feedback preprocessor pre-processes the feedback data $\mathbf{y}_C(t)$ such as motor speed and current from the remote system before forwarding the pre-processed signal $\hat{\mathbf{y}}_{\hat{\tau},C}(t)$ to the controller. Preprocessing in this case can be, for example, filtering noises in the feedback data, or prediction of remote system states. Necessity of these operations depends on the gain scheduling algorithm used in the gain scheduler.

(3) Gain scheduler (GS). By using an external gain scheduling algorithm, the gain scheduler unit modifies the controller output $\mathbf{u}_C(t)$ to $\mathbf{u}_{\beta,C}(t)$ which provides an optimal performance

for the current network conditions characterized by **q** [3]. The algorithm to modify the controller output depends on the overall system configuration of the controller and the remote system. The overall GSM operations for networked control and teleoperation can be summarized in Fig. 6.

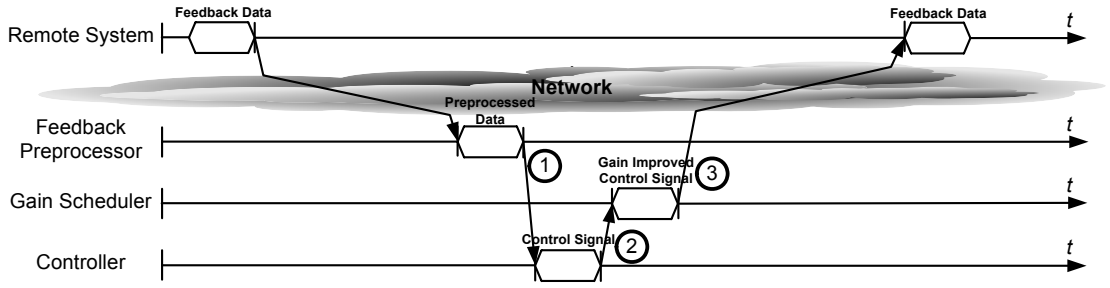

Fig. 6 Time diagram of the GSM operations.

1. Feedback preprocessor waits for feedback data from the remote system. Once the feedback data arrives, the preprocessor processes the data using the current values of network variables and passes the preprocessed data to the controller.
2. The controller computes the control signals and sends them to the gain scheduler.
3. The gain scheduler modifies the controller output based on the current values of network variables and sends the gain improved control signals to the remote system.

IV. GSM FOR REMOTE MOBILE ROBOT PATH-TRACKING

The conceptual diagram of the remote mobile robot path-tracking with GSM is depicted in Fig. 7.

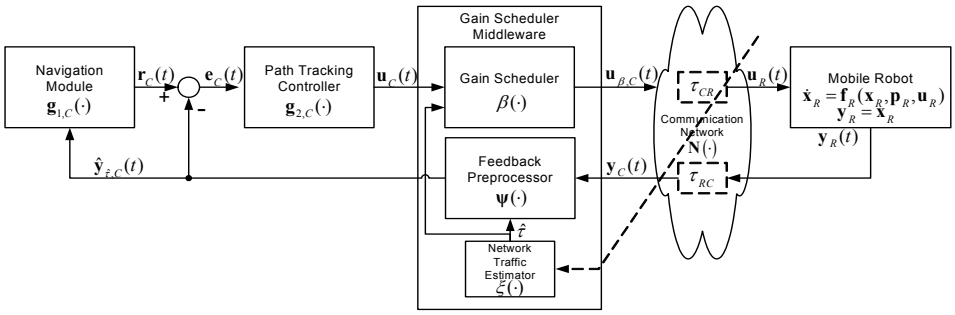

Fig. 7. Remote mobile robot path-tracking model applied with GSM.

The diagram shows the dataflow between the navigation module, path-tracking controller, GSM, and the mobile robot. If we consider the data flow started from the data sampled from the mobile robot $\mathbf{y}_R(t)$, the FP in GSM receives the state of the robot in the past $\mathbf{y}_C(t) = \mathbf{y}_R(t - \tau_{RC})$ and the estimated round-trip-time delay $\hat{\tau} \cong \tau_{RC} + \tau_{CR}$ from NTE in order to estimate the robot state (position) in the future, $\hat{\mathbf{y}}_{\hat{\tau},C}(t)$. The navigation module having the tracking path generated by artificial potential functions [8] (e.g., obstacle avoidance) calculates the reference point $\mathbf{r}_C(t)$ from the current estimated position of the robot. Path-tracking module then generates the reference wheel speeds $\mathbf{u}_C(t)$ for driving the mobile robot from $\mathbf{y}_C(t)$ to $\mathbf{r}_C(t)$. $\mathbf{u}_C(t)$ will be updated by the GS module to be $\mathbf{u}_{\beta,C}(t)$ and then be transmitted to drive the mobile robot.

As mentioned in the previous section, the NTE estimates the network traffic condition, the FP preprocesses the feedback data, and the GS modifies the control signal according to the current network condition. For the robot-path-tracking problem:

1. The *Network Traffic Estimator* (NTE) predicts the Round-Trip-Time (RTT) delay defined as:

$$\tau = \tau_{CR} + \tau_{RC} \tag{5}$$

As proposed in [3], τ is estimated from mean (μ) and median (η) of the RTT data logged in the NTE based on a generalized exponential distribution. The estimated RTT ($\hat{\tau}$) is calculated by:

$$\hat{\tau} = \max\{\eta, \mu\} \tag{6}$$

2. The *Feedback Preprocessor* (FP) estimates the robot position when the control signal arrives at the remote plant. The effect of the FP on robot with delay time is illustrated in Fig. 8.

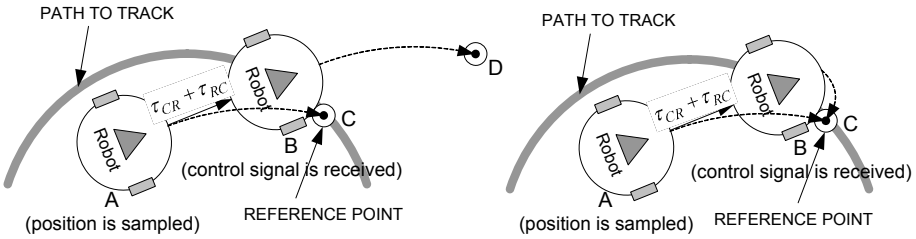

Fig. 8. The effect of the delay time on path-tracking mobile robot without FP (left) and with FP (right).

Fig. 8 (left) shows how the network time delay affects the path-tracking mobile robot. When the robot position is at *A*, the position of the robot is sampled. Without network delay, the robot should receive the feedback control signals almost instantaneously. In such case, the control signal should drive the robot to *C* to track the reference point on the path. However, when using the IP network as a signal transmission media, the control signal sent to the robot is delayed by $\tau_{CR} + \tau_{RC}$ sec. By that time, the robot has already moved to the position *B* and will be driven to position *D* because of the delayed control signal. To avoid this situation, a FP is used to reduce the effect of time delay by estimating the position of the

robot when the robot receives the control signals at *B*. The control signal is then calculated based on the estimated position of the robot in the future which will be appropriate for the robot while receiving the control signal as shown in Fig. 8 (right). More discussion in the FP can be found in chapter 4.

3. The *Gain Scheduler* (GS) adjusts the speed gain K of the path-tracking controller according to the network traffic, the robot speed, and the path curvature for an optimal performance. Higher speed gain tends to be more appropriate if the robot is tracking a straight path for faster speed (thus short travelling time) while it may cause more path-tracking error if the robot is tracking the curve path. The GS is developed to provide an optimally adjusted gain $K_{new} = \gamma K$ at any given time based on these varying environmental factors as shown in Eqn. (4).

From Eqn. (4), β is the gain calculated in real-time for the GS module to modify K of the path-tracking controller. By multiplying $\mathbf{u}_C$ with β, it is equivalent to multiplying K with γ for the path-tracking controller. Since K is proportional to the speed of the robot tracking the path, β should be calculated to get the optimal tracking performance by minimizing the path-tracking error and maximizing speed. With hard real-time control constraints, it will be difficult to calculate β to get the optimal tracking performance. Alternatively, we will use a sub-optimal gain to approximate the optimal gain selection. This is equivalent to selecting the gains K_{new} that minimizes the travelling time and also allows the deviation error of the robot to occur only in a certain bound. The detail of the implementation of this GS module can be found in appendix C.

V. SIMULATION RESULTS AND DISCUSSION

This section illustrates the effectiveness of GSM by simulation results. The simulation program is developed by using Matlab/Simulink® based on the model of the Unmanned ground vehicle (UGV) in the appendix A. The simulation results demonstrate the performance of the remote UGV path-tracking for both with and without GSM. The performance can be compared by using the integration of path-tracking error and the speed of the UGV.

We also investigate a variation of gain table generation by using two dimensional gain table and one dimensional gain table in this section. Using 2-D gain table provides more degree of freedom for the UGV to adjust its speed gain than 1-D gain table. However, the implementation of 1-D would be simpler. Taking into account this factor, our simulation generates several remote path-tracking results using 1-D gain table in order to compare with 2-D gain table.

The simulation result in this section is related to a small size UGV whose physical parameters and motor parameters are listed here. The gain tables used in the simulation are also illustrated here by graphs to show how the speed of the UGV would be adjusted according to the remote UGV path-tracking conditions. We present the simulation parameters, the gain tables, and then the simulation results and discussion successively.

A. Simulation parameters

The simulation was done for a small size UGV whose parameters listed in Table 1. These parameters describe physical quantities of the UGV model mentioned in appendix A. Since the UGV is driven by two DC motors, the characteristics of those motors would also be

important in the simulation. The parameters of those DC motors are listed in

Table 2. The DC motor speeds for tracking the path are determined by the path-tracking algorithm, the path-tracking parameters of Quadratic curve path-tracking algorithm are listed in Table 3.

Table 1. UGV chassis parameters.

Parameters	**Values**
Distance between the wheel and the line of symmetry (b_1)	0.0865 m
Distance between the center of turn and center of gravity (b_2) (negative value here means CT is behind CG)	-0.09 m
Radius of the wheel (r)	0.036 m
Mass of each wheel and rotor (m_w)	0.025 kg
Moment of inertia of each wheel about the wheel diameter (I_m)	3.375e-06 $kg \cdot m^2$
Moment of inertia of the wheel about the wheel axis (I_w)	1.620e-05 $kg \cdot m^2$
Mass of the chassis (m_c)	1.5625 kg
Moment of inertia of the chassis around the center of turn (I_c)	0.02434 $kg \cdot m^2$

Table 2. Motor parameters.

Parameters	**values**
Electric resistance (R_{motor})	7.4 ohms
Electric inductance (L_{motor})	2e-3 H
Damper (b_{motor})	2e-7 $N \cdot m \cdot s$
Moment of inertia of the rotor (J_{motor})	1.5e-7 $kg \cdot m^2$
Gear ratio (N_{motor})	333.6375
Torque constant (K_t) which equals to generator constant (K_e)	2.2179e-3 $N \cdot m/A$

Table 3. Quadratic curve path-tracking parameters.

Parameters	values
Maximum reference speed (α)	0.45 m/s
Maximum look ahead distance (d_{max})	0.2473 m
Minimum look ahead distance (d_{min})	0.0771 m
Look ahead distance adjustment factor (β)	0.7509

As we operate iSpace at NCSU in local area network, the communication delay is small compared to the delay caused by the image acquisition and image processing delay. The sum of image acquisition and image processing delay represents the RTT delay in the control loop as shown in the histogram in Fig. 9.

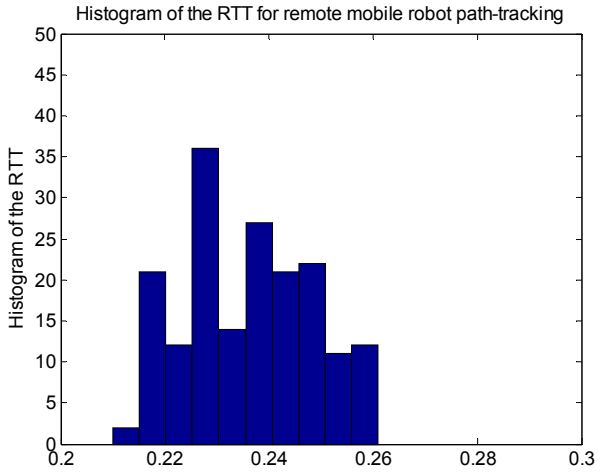

Fig. 9. Histogram of the random delay between ADAC lab and Hashimoto lab.

From Fig. 9, the RTT delay in iSpace at NCSU is random with the mean value of 0.2369 sec. The RTT has the minimum value at 0.2100 sec and the maximum value at 0.2810 sec with the standard deviation value of 0.0127 sec. We will use this set of random delay in the simulation to generate the results.

B. Generated gain table

As represented by $K_{new}(\kappa_{UGV}, \hat{\tau}^{k_{i+1}})$, the gain table is a function of UGV trajectory curvature κ_{UGV} and estimated value of the next RTT delay $\hat{\tau}^{k_{i+1}}$. The values within the gain table represent the adjusted speed of the UGV. Given a value of ε, the tolerance of tracking error, one can generate gain table by using the bisection algorithm as discussed in the appendix. The generated gain table of $K_{new}(\kappa_{UGV}, \hat{\tau}^{k_{i+1}})$ with $\varepsilon = 0.12$ is shown by surface graph in Fig. 10.

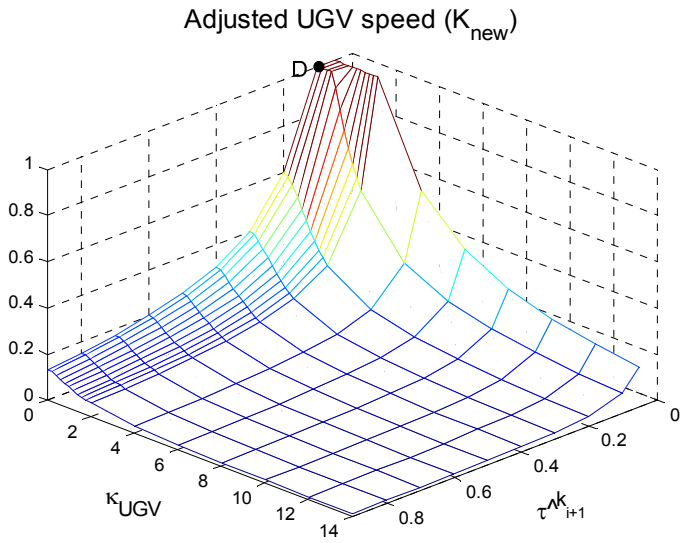

Fig. 10. 2-D gain for GS module.

From Fig. 10, we can observe that K_{new} is large when the radius of UGV trajectory is large (small κ_{UGV}) and $\hat{\tau}^{k_{i+1}}$ is small and vice versa. Therefore, the UGV path-tracking using GSM will use high speed when it tracks a straight line with small RTT. In the case that RTT is very small, K_{new} can be very large and beyond the physical speed of the actual UGV. Therefore, we set an upper bound for K_{new} at 1 m/s, which is fastest speed of the UGV we are

using in the simulation.

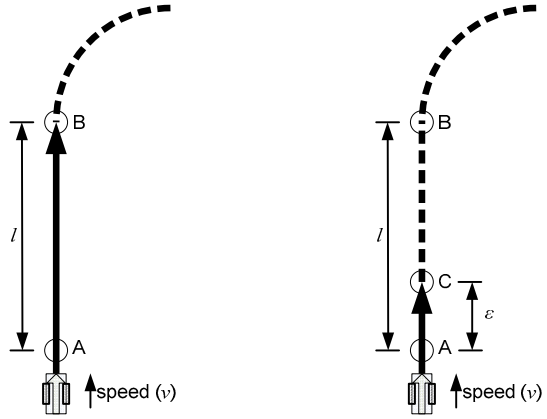

Fig. 11. Scenario when the UGV tracks a straight path length l (left) and the distance ε that the UGV travels because of the adjusted speed (right).

The implication of the gain table in Fig. 10 for the behavior of the UGV is as follows. Consider Fig. 11 as the UGV needs to move along a tracking path composed of a straight line length l and a circular path. When the UGV is at point A, it receives a control signal to move forward. Because of the network delay, the UGV receives an updated control signal only every τ sec. Ideally, the adjusted speed should make the UGV move as fast as possible from point A upto point B (where it receives an updated control signal to make a turn at the right time). The adjusted speed should be equal to $v = l/\tau$ because the UGV will move for the distance l within τ sec. GSM is a module inserted in remote UGV path-tracking system between the path-tracking controller and the network. In this thesis, the cost function used to develop the GSM table only receives the information of the control signal and the UGV state at the current time that passes through the module. The cost function does not consider external information such as the path curvature. Therefore, the GS module does not have the information of the length of the path l and cannot schedule the optimized speed $v = l/\tau$ for

the UGV in this segment. Instead, GS module schedules the gain so that the UGV will move forward for a "safety distance" ε meters before receiving an updated control signal as shown in Fig. 11 (right). This setting allows the UGV to get the control signal soon enough so that it can correctly make a turn when it is near point B. As a result, the values of the UGV speed in the gain table when $\kappa_{UGV} = 0$ is $K_{new} = v = \varepsilon/\tau$. As we set the upperbound of the speed equal to 1 m/s to reflect physical limitation, the graph of the speed gain is clipped at $\tau = \varepsilon/v = 0.12/1$ sec as shown by point D in Fig. 10 and Fig. 12.

We also generate the gain table when we fix either the value of κ_{UGV} or the value of $\hat{\tau}^{k_{i+1}}$ to compare the performance of using 2-D and 1-D gain table. Our 1-D gain table can be represented by either $K_{new}(\hat{\tau}^{k_{i+1}})$ or $K_{new}(\kappa_{UGV})$ as shown in Fig. 12. Later on in this section, we will compare the performance of remote UGV path tracking while using 2-D gain table and 1-D gain table to see the characteristics of GSM according to each independent variable in gain table.

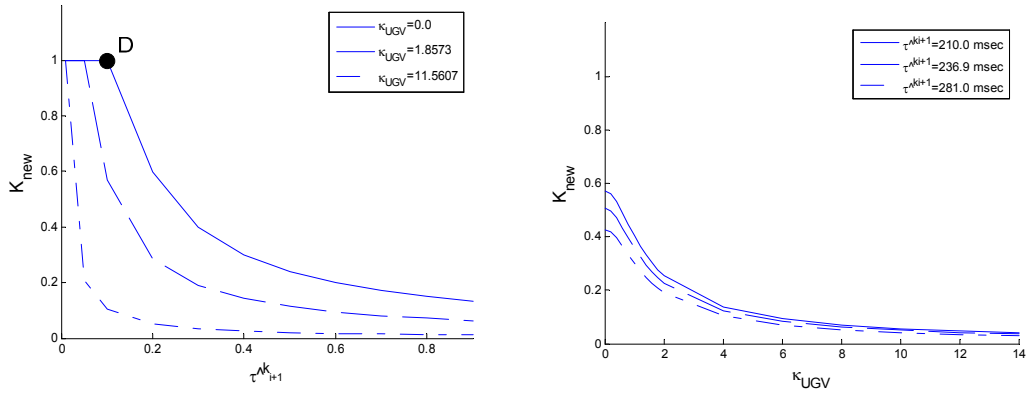

Fig. 12. 1-D gain table for GS module, $K_{new}(\hat{\tau}^{k_{i+1}})$ on the left and $K_{new}(\kappa_{UGV})$ on the right.

The left figure of Fig. 12 shows the values of the speed gain $K_{new}(\hat{\tau}^{k_{i+1}})$ in three gain

tables generated by using different fixed values of κ_{UGV}. We select three values of κ_{UGV} so that it will cover different range of tracking path curvatures. The minimum value of κ_{UGV} is 0.0 since it is corresponding to a straight path. The maximum value of κ_{UGV} is 11.5607 = 1/ 0.0865 since the distance between the wheel and the UGV's center of turn is 0.0865 m and the practical path should not have the radius of turn smaller than that. The other value of κ_{UGV} =1.8573 is selected from the average path curvature value of the tracking path used in this chapter. As we can expect, the highest values of K_{new} is corresponding to the lowest value of κ_{UGV} at 0.0. The lowest values of K_{new} is corresponding to the highest value of κ_{UGV} at 11.5607.

For the right figure of Fig. 12, we show the speed gain $K_{new}(\kappa_{UGV})$ in the gain table generated by using fixed value of $\hat{\tau}^{k_{i+1}}$. We select three values of $\hat{\tau}^{k_{i+1}}$, the minimum, the mean, and the maximum, to cover different range of the RTT in iSpace at NCSU whose histogram is shown in Fig. 9. The minimum value of RTT is 210.0 msec, the mean value of RTT is.236.9 msec, and the maximum value of RTT is 281.0 msec. We observe that the smallest RTT is corresponding to the largest gain. Vice versa, the largest RTT is corresponding to the smallest gain.

C. Simulation results and discussion

In this section, we present the simulation result of the remote UGV path-tracking for several different setups. We can separate the results into three set so that they can be easily compared. The setups of all cases are shown in Table 4. By considering these simulation results, we can gain a better understanding on how GS operates and how the gain table

should be prepared for using with GSM.

For each case, we use the same tracking path as shown in Fig. 13. The tracking path can be separated into 5 different segments according to different path curvatures. For comparison, the performance between different segments of the path can be distinguished by dashed lines as shown in the simulation results.

Table 4. Setups of the remote UGV path-tracking to be simulated.

Set A	**Set B**	**Set C**
No delay, No GSM	With delay & GSM, 1-D gain table at κ_{UGV} =0.0	With delay & GSM, 1-D gain table at $\hat{\tau}^{k_{i+1}}$ =210.0 msec
With delay, No GSM	With delay & GSM, 1-D gain table at κ_{UGV} = 1.8573	With delay & GSM, 1-D gain table at $\hat{\tau}^{k_{i+1}}$ =236.9 msec
With delay & GSM	With delay & GSM, 1-D gain table at κ_{UGV} = 11.5607	With delay & GSM, 1-D gain table at $\hat{\tau}^{k_{i+1}}$ = 281.0 msec

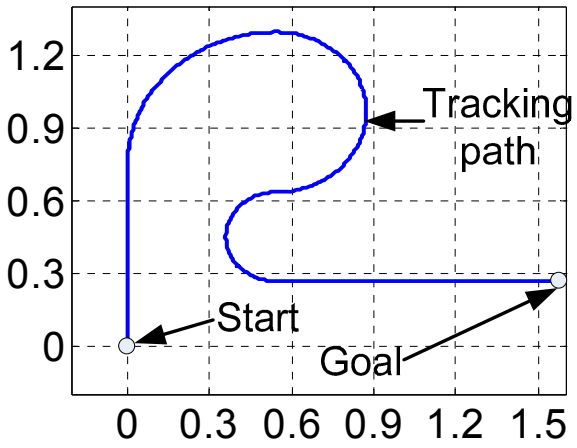

Fig. 13. Tracking path used in the simulation.

The path tracking performance is indicated by the UGV accumulated tracking error and the tracking speed. The quantitative value of the accumulated tracking error can be calculated by J_1 as shown in Eqn. (7 where $D(\cdot)$ is the shortest distance between the robot position $\left(x(t), y(t)\right)$ and the tracking path as depicted in Fig. 14. The tracking speed is calculated by J_2

in Eqn. (8). The overall cost for evaluation J can be calculated by weighting between two cost functions as shown in Eqn. (9).

$$J_1 = \int_{t_0}^{t_f} D(x(t), y(t))\,dt = \int_{t_0}^{t_f} \min_{s} \sqrt{\left(x(t)-x_p(s)\right)^2 + \left(y(t)-y_p(s)\right)^2}\,dt \tag{7}$$

$$J_2 = t_f - t_0 \tag{8}$$

$$J = w_1 J_1 + w_2 J_2 \tag{9}$$

where t_0 is the starting time, t_f is the time when the UGV reaches the destination. $w_1 = 8.8652$ and $w_2 = 0.0272$ as we normalized the costs so that J_1 and J_2 have an equal weight to constitute the cost J, when the mobile robot tracking the path without network delay, to be equal to 1.

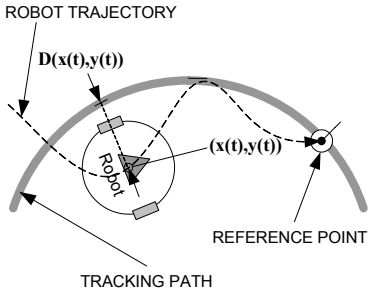

Fig. 14. The mobile robot while tracking the predefined path.

The simulation result for each set is illustrated as follows:

(1) Simulation result for set A

From the result in Fig. 15, the first column illustrates the UGV trajectory while tracking the path. The second column shows the path-tracking error. The third column shows the

accumulated path-tracking error and the forth column shows the speed of the UGV. On the first row, the result shows the case when there is no delay (i.e. the delay from image acquisition and image processing is omitted) and no GSM is used. The UGV driven by QC can closely track the path. QC also adjusts the speed of the UGV according to the path curvature. The UGV is slowed down while tracking the high curvature path (e.g. path segment #4) and uses higher speed while tracking a straight path (path segment #1 & #5). On the second row, the result shows the case when there is delay time but no GSM. The UGV path-tracking is degraded by time delay which makes the tracking error larger and the tracking period longer. As the UGV oscillates around the path, the UGV speed changes up and down significantly. This is because QC algorithm implies that the path curvature keeps changing as the quadratic curve between the UGV and the reference point on the path has a changing coefficient. On the third row, the result shows the case when there is delay time and GSM is applied. Obviously, the result shows a substantially less path-tracking error than not using GSM. In this particular case, the speed of the UGV is quite similar to the no delay case. It shows that FP can alleviate the delay problem and the help from GS is not required. This may happen because RTT is relatively small and GS schedules the UGV speed gain to the values higher than the speed gain from QC algorithm. Therefore, the speed from QC algorithm (which is lower) is used to drive the UGV because GS will not schedule the speed gain to a higher value than the original speed gain as it might be over the physical limit of the UGV.

For a quantitative comparison, the path-tracking costs are calculated as shown in Table 5. The calculated costs is consistent with the tracking result in Fig. 15 as the overall cost *J*

largely increases when there is delay in the system. After GSM is introduced into the system, the cost J is then decreased significantly toward the level of the case when there is no delay.

Table 5. Path tracking cost for the simulation result set A.

Set A	**J1**	**J2**	**J**
No delay, No GSM	0.0564	18.3654	1.00
With delay, No GSM	0.6022	28.7812	6.12
With delay & GSM	0.0746	21.1533	1.24

For comparison, the case that the RTT is larger is shown in Fig. 16 and quantitatively shown in Table 6 as a constant delay is added to the iSpace delay by 0.35 sec. The result shows that GS schedules the gain by generally making the speed slower according to path curvature so that the UGV path-tracking can still keep small path-tracking error. It also shows how GS module adjusts the UGV speed according to the delay time so that we do not need to worry if the characteristics of the network delay is changed.

Table 6. Path tracking cost for the simulation result set A with additional network delay.

Set A	**J1**	**J2**	**J**
No delay, No GSM	0.0564	18.3654	1.00
With delay, No GSM	10.4125	141.7000	96.16
With delay & GSM	0.2038	46.6640	3.08

(2) Simulation result for set B

As we use 1-D gain table $K_{new}(\hat{\tau}^{k_{i+1}})$ for the simulation set B, we expect to see a compromised path-tracking performance compared with the result from 2-D gain table because the speed gain will not be scheduled by changing the UGV trajectory curvature. Because the speed gain is changed only according to the RTT and the random RTT has a relatively small variance compared with the change of path curvature, the change of the UGV speed while tracking the path is expected to be small. However, the result of set B in Fig. 17

shows different profiles of UGV speed. First, consider the third case where the gain table is generated for the highest curvature in the tracking path at κ_{UGV}=11.5607. We can observe steady speed profile on the forth column as we expected. As the gain table is tuned for a smaller value of κ_{UGV}, the speed profile should be steady but higher in magnitude. However, the speed seems to change according to the curvature as shown in the first and second case. This happens because the UGV will use the smaller speed gain between the speed gain from GS and the original speed from QC. Therefore, if the looked up speed gain from GS is too large, the original speed will be used.

Considering the quantitatively calculated cost in Table 7, the cost *J* in the second and third case is greater than the GSM with 2-D gain table in Table 5. However, when the 1-D gain table with fixed κ_{UGV}=0.0 is used, the cost is slightly lower than using 2-D gain. This reflects one nature of GS module, that GS uses the gain optimizing the cost for only one small step at a time. GS does not optimize for the whole path-tracking since GS does not have the knowledge of the whole path beforehand. Therefore, the path-tracking cost while using 2-D gain table may be small but may not reflect the minimum cost.

Table 7. Path tracking cost for the simulation result set B.

Set B	**J1**	**J2**	**J**
With delay & GSM, 1-D gain table at κ_{UGV}=0.0	0.0772	20.1441	1.23
With delay & GSM, 1-D gain table at κ_{UGV} = 1.8573	0.0687	22.0942	1.21
With delay & GSM, 1-D gain table at κ_{UGV} = 11.5607	0.3760	94.2455	5.90

(3) Simulation result for set C

Simulation set C use 1-D gain table $K_{new}(\kappa_{UGV})$ with different fixed values of $\hat{\tau}^{k_{i+1}}$. We expect to see a compromised path-tracking performance compared with the result from 2-D gain table because the speed gain will not be scheduled by the changing RTT. Fig. 18 shows the result that conforms to our expectation as the tracking error for all three cases are generally larger than using the 2-D gain table. This can also be observed by the calculated cost in Table 8 as the cost *J* for all cases are larger than 2-D GS. For the first case, when using fixed RTT at the smallest value at 210.0 msec, the UGV uses faster speed compared with the others and the UGV reaches the destination in a shortest time. The last case with fixed RTT at the largest value at 281.0 msec, the UGV uses slowest speed and has the smallest tracking error. For all three cases, the costs *J* only have a small difference from each other. Since the cost for all three cases are only slightly different from 2-D gain table, we may consider using 1-D gain table with fixed RTT instead of 2-D gain table for easier implementation given that the characteristics of the random RTT is known and the variance of the RTT is small.

Table 8. Path tracking cost for the simulation result set C.

Set C	**J1**	**J2**	**J**
With delay & GSM, 1-D gain table at $\hat{\tau}^{k_{i+1}} = 210.0$ msec	0.0933	21.1110	1.40
With delay & GSM, 1-D gain table at $\hat{\tau}^{k_{i+1}} = 236.9$ msec	0.0940	22.1754	1.44
With delay & GSM, 1-D gain table at $\hat{\tau}^{k_{i+1}} = 281.0$ msec	0.0804	23.9761	1.36

VI. CONCLUSION

In this chapter, we introduce a new concept of application called Intelligent Space or iSpace as it can provide intelligent services to human beings. A preliminary project, iSpace at NCSU, is used as a platform to study the problem of time-delay in iSpace. The delay problem is demonstrated through a simulation of remote Unmanned Ground Vehicle (UGV) path-tracking in iSpace. The problem is alleviated by a technique called Gain Scheduler Middleware (GSM) that can apply to a nonlinear system like a UGV or mobile robot. Three set of simulation results are demonstrated to compare seven implementations of gain tables in GSM such as two dimensional and one dimensional gain table to show the advantage, limitation, and practical consideration of gain scheduler module within GSM. The result shows that two dimension gain table is more adaptive to the changing system parameters such as path curvature and delay magnitude. In the case that characteristics of system parameter is known and the parameter has only a small variation, we can consider using one dimensional gain table due to its simplicity of implementation.

REFERENCES

[1] H. Hashimoto, "Intelligent space - How to make spaces intelligent by using DIND?," Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia, 2002.

[2] R. Vanijjirattikhan, M.-Y. Chow, P. Szemes, and H. Hashimoto, "Mobile Agent Gain Scheduler Control in Inter-Continental Intelligent Space," Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA05), Barcelona, Spain, 2005.

[3] Y. Tipsuwan and M.-Y. Chow, "Gain Scheduling Middleware for Networked Mobile Robot Control," Proceedings of American Control Conference 2004 (ACC2004), Boston, MA, 2004.

[4] W.-L. Lueng, R. Vanijjirattikhan, Z. Li, L. Xe, T. Richards, B. Ayhan, and M.-Y. Chow, "Intelligent space with time sensitive applications," Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Monterey, California USA, 2005.

[5] Y. Tipsuwan and M.-Y. Chow, "Gain scheduler middleware: A methodology to enable existing controllers for networked control and teleoperation: PART II: teleoperations," *IEEE Transactions on Industrial Electronics*, vol. 51, 2004.

[6] Y. Tipsuwan and M.-Y. Chow, "Control Methodologies in Networked Control Systems," *Control Engineering Practice*, vol. 11, pp. 1099-1111, 2003.

[7] Y. Tipsuwan, "Gain scheduling for networked control system," North Carolina State University, 2003.

[8] S. Mizik, P. Baranyi, P. Korondi, and M. Sugiyama, "Virtual Training of Vector Function based Guiding Styles," *Automatic Control and Computer Science*, vol. 46(60), pp. 81-86, 2001.

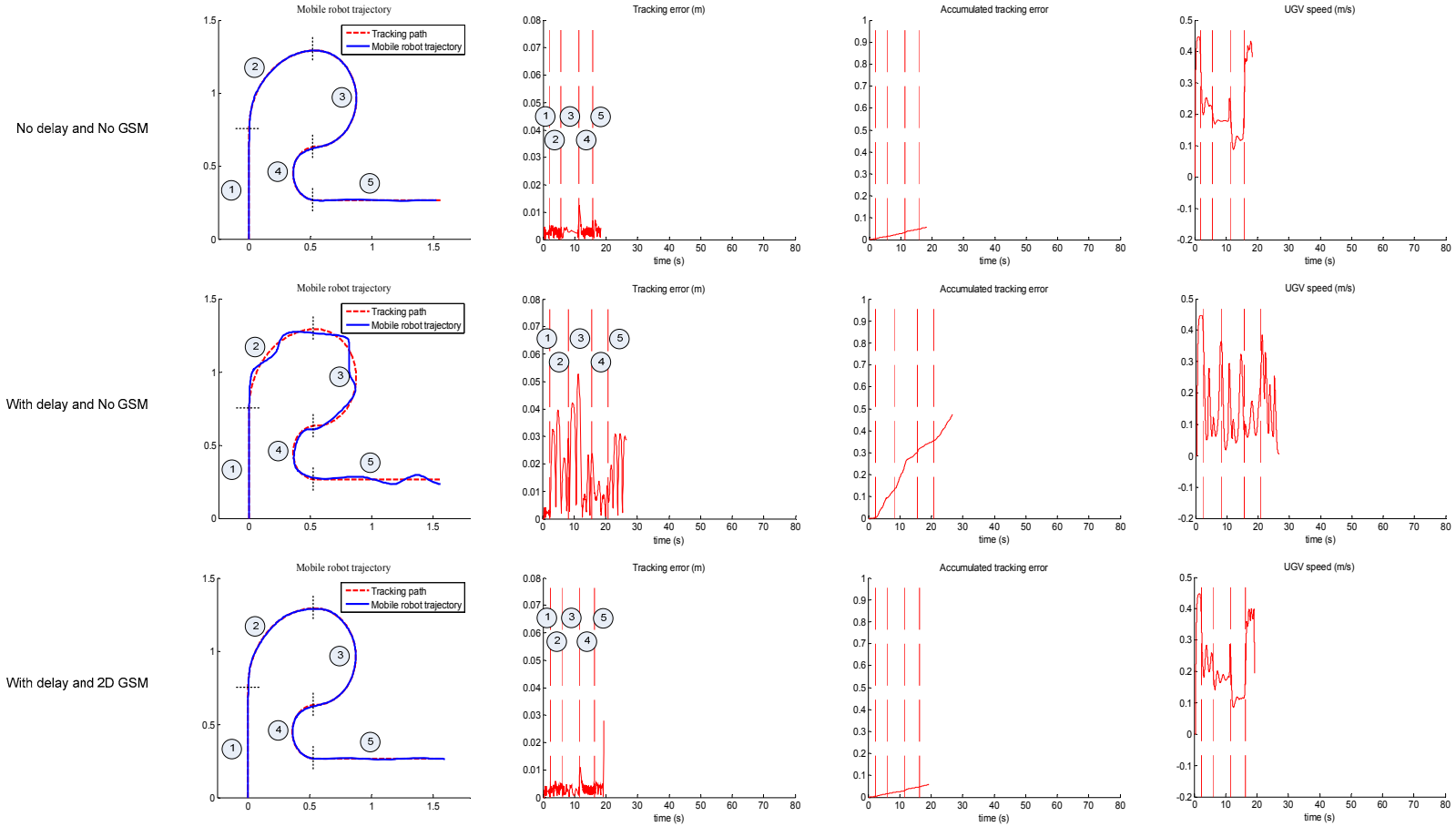

Fig. 15. Remote UGV path-tracking simulation result for set A when there is no delay, when there is delay but without GSM, and when there is delay and GSM.

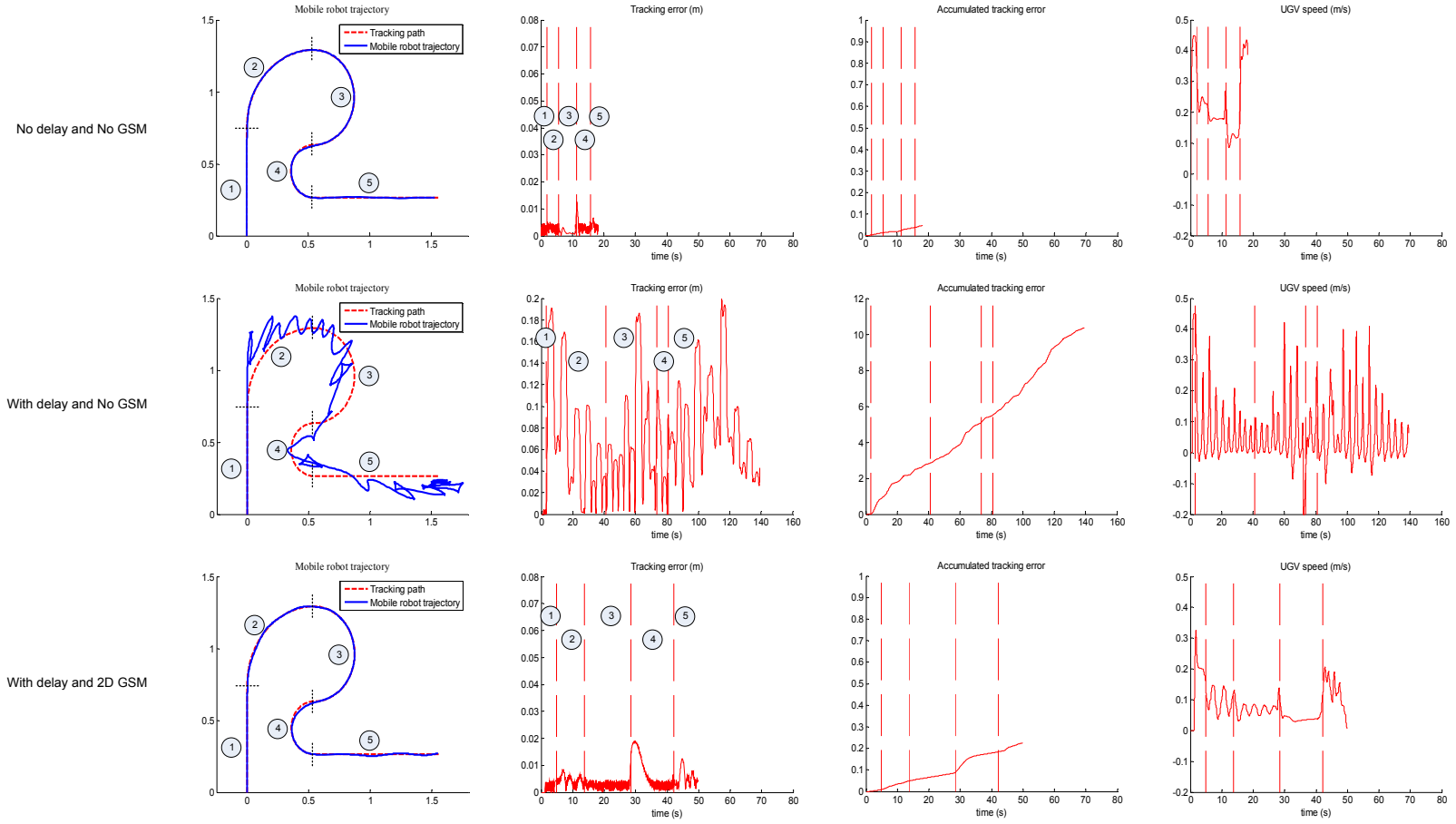

Fig. 16. Remote UGV path-tracking simulation result for set A with 0.35 sec additional delay.

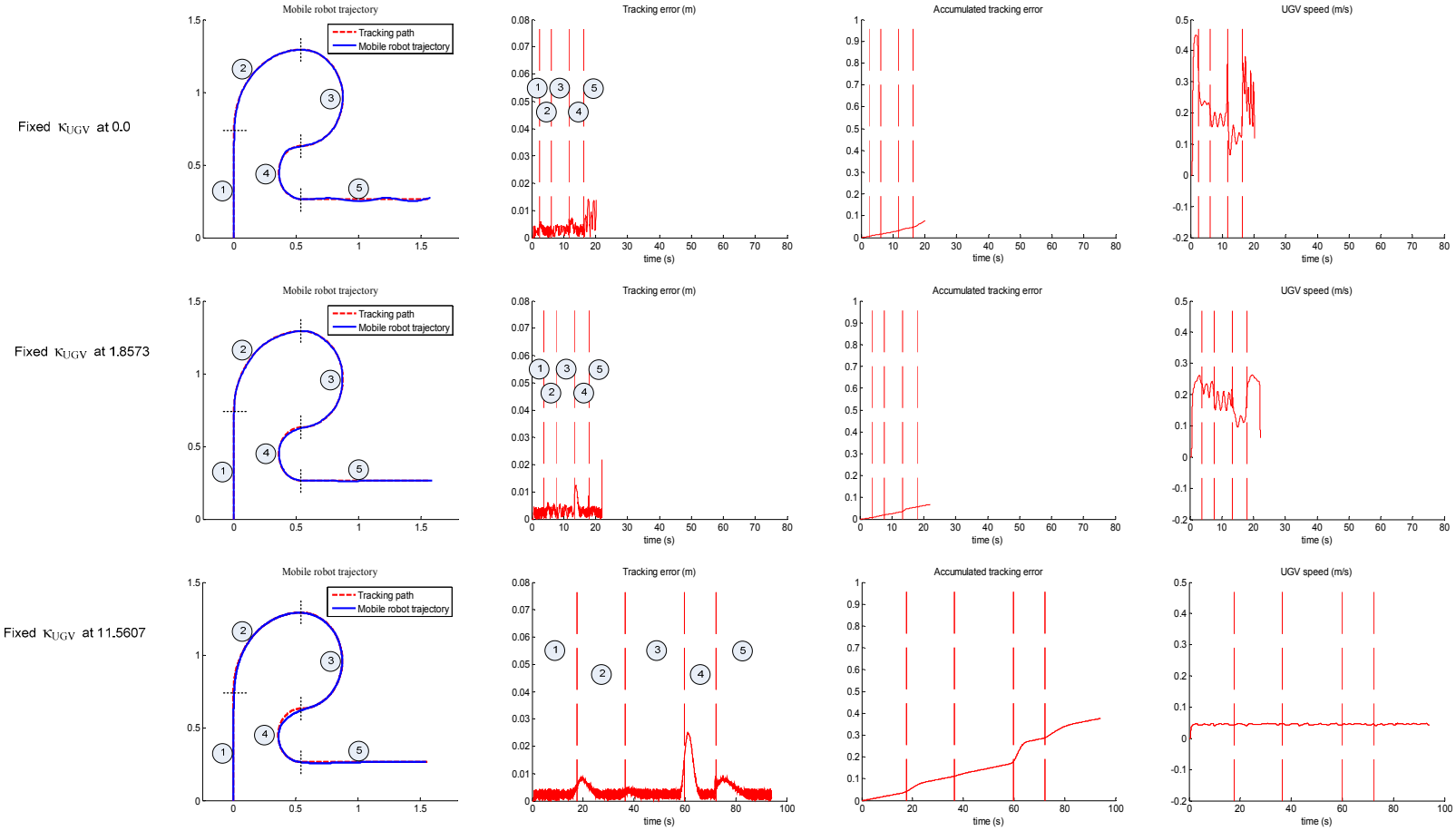

Fig. 17. Remote UGV path-tracking simulation result for set B when using GSM with 1-D generated gain table having κ_{UGV} fixed at 0.0, 1.8573, and 11.5607 respectively.

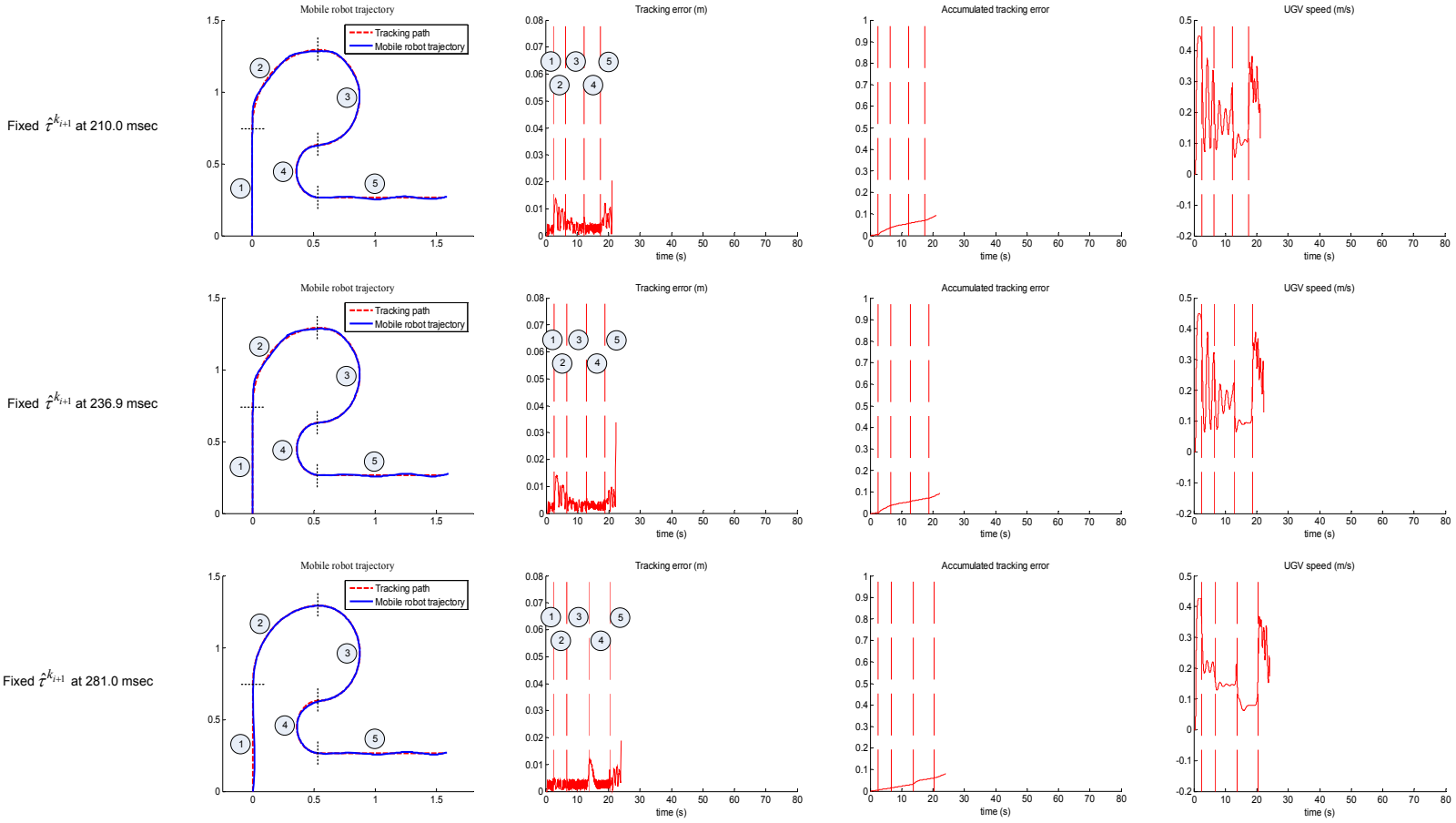

Fig. 18. Remote UGV path-tracking simulation result for set C when using GSM with 1-D generated gain table having $\hat{\tau}^{k_{i+1}}$ fixed at 210.0 msec, 236.9 msec, and 281.0 msec respectively.

CHAPTER VI

CONCLUSION

The research presented in this dissertation proposes the methods for time delay problem alleviation in remote mobile robot path-tracking system in which the path-tracking controller communicates with the mobile robot through a communication network. The effect of delay time in the system can range from system performance degradation to system instability. The uniqueness of the proposed methods is their non-requirement of the controller redesign and their applicability to a range of different mobile robot characteristics and operating environments.

In chapter III, we propose an analytical sensitivity analysis of the Network-based control mobile robot path-tracking system according to operating environment. The analytical results are obtained from a mathematical formulation of a path-tracking scenario. The sensitivity data can be used to determine how the path-tracking cost is sensitive to each system parameter and which parameter must be carefully looked at. The sensitivity data also provides us a guideline on how to choose an operation condition for a networked control system (NCS) problem given a magnitude of the change of a system parameter. In chapter IV, we introduce an intermediate parameter, *UGV response time*, used to relate the demonstrated remote mobile robot path-tracking performance to other UGVs in general. Chapter IV also investigates the effects of using a Feedback Preprocessor (FP) based on the concept of the Smith predictor in NBC path tracking problem. FP estimates the UGV state in

the future to be used by the path-tracking controller in order to compensate the effect of time delay induced by the network. FP augments an existing controller, designed to operate in delay-free condition, to externally compensate the time delay effect for Network-Based Control applications without redesigning the controller. The approach and results from chapter IV can be used to estimate the performance of any similar UGV operating in a NBC environment via UGV response time. In chapter V, the delay problem is alleviated by a technique called Gain Scheduler Middleware (GSM) that can apply to a nonlinear system like a UGV or mobile robot. Several set of simulation results are demonstrated to compare several implementations of gain tables in GSM such as two dimensional and one dimensional gain table to show the advantage, limitation, and practical considerations of gain scheduler module within GSM. The result shows that two dimensional gain table is more adaptive to the changing system parameters such as path curvature and delay magnitude. In the case that characteristics of system parameter is known such as the random distribution of the network delay, we can consider using one dimensional gain table for the simplicity of implementation.

As the path-tracking controller should be finely tuned before operating either in no delay and Network-based control case, we also propose a path-tracking parameter tuning method for geometrical path-tracking algorithm in chapter II. We can use this method to find a qualified set of parameters so that the robot can track the path with the path deviation error less than a certain bound. The detail of quadratic curve path-tracking algorithm used to control the mobile robot is demonstrated in appendix B for reference. We also demonstrate the mathematical model of the mobile robot used in implementing the simulation program

and GSM in appendix B and C respectively so that the result can be reduplicated.

For future research, we can consider enhancing the Gain Scheduler Middleware module by using the whole tracking trajectory for gain tuning to optimize the overall path-tracking cost instead of optimizing the cost for one step at a time. This research direction may improve the NBC path-tracking performance as the controller gain can be scheduled based on the costs of several path-tracking steps ahead instead of the costs from only one step ahead. Moreover, we can consider using sensitivity data in gain table generation so that the effect of noise in remote mobile robot path-tracking is carefully taken into account.

APPENDICES

APPENDIX A

MODEL OF UNMANNED GROUND VEHICLE (UGV)

The UGV is composed of a chassis, two driving wheels, a free running support wheel, two motors, and two proportional-integral (PI) speed controllers. The reference UGV speed and reference turn rate (how fast the UGV turns) from the path tracking controller will be used to calculate reference wheel speeds for PI speed controllers. Based on reference wheel speeds, PI speed controllers will provide electricity to drive the motors which generate torques to drive the wheels of the UGV. The simulator in this paper uses the UGV dynamics model to describe the behavior of UGV according to the driving torques. The UGV dynamics is described by the mathematical model of the non-holonomic two driving wheeled mobile robot (illustrated in Fig. 1) from [1].

As shown in Fig. 1, the center of gravity (CG) of the UGV does not need to be the same point as the center of turn (CT). The wheels with radius *r* and the chassis have their own mass and moment of inertia.

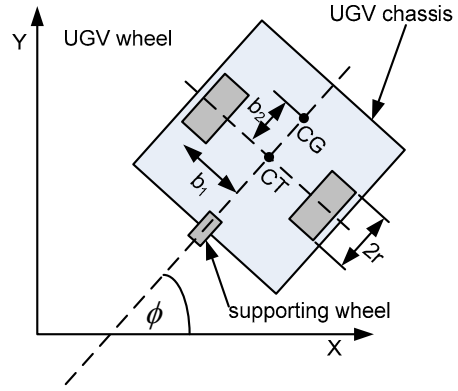

Fig. 1. Unmanned Ground Vehicle (UGV).

Combining the UGV dynamics model, the dc motor model, and the PI controller gives the UGV model that describes the behaviour of the UGV according to the reference speed and turn rate. The state of the UGV composed of the state of UGV kinetics, DC motors, and PI controllers is described by $\mathbf{x} = \left[x\, y\, \theta_r\, \theta_l\, \dot{\theta}_r\, \dot{\theta}_l\, i_{a_r}\, i_{a_l}\, z_r\, z_l \right]^T$ where (x, y) is the UGV coordinate, θ and $\dot{\theta}$ are the angular displacement and angular velocity of the wheel, i_a is the state of the motor electrical dynamics, z is the state of the integrator in PI speed controller. *r* and *l* denote the right and the left wheel, respectively.

The model of UGV can be separated into components as shown by the block diagram in Fig. 2. The input of the model is $\mathbf{u} = \left[v_{ref} \quad \omega_{ref} \right]^T$ while the output of the model is $\mathbf{y} = \left[x \quad y \quad \phi \right]^T$ where v_{ref} is the reference speed of the UGV (in m/s), ω_{ref} is the reference turn rate (in rad/s), (x, y) is the position coordinate of the robot, and ϕ is the heading angle of the robot.

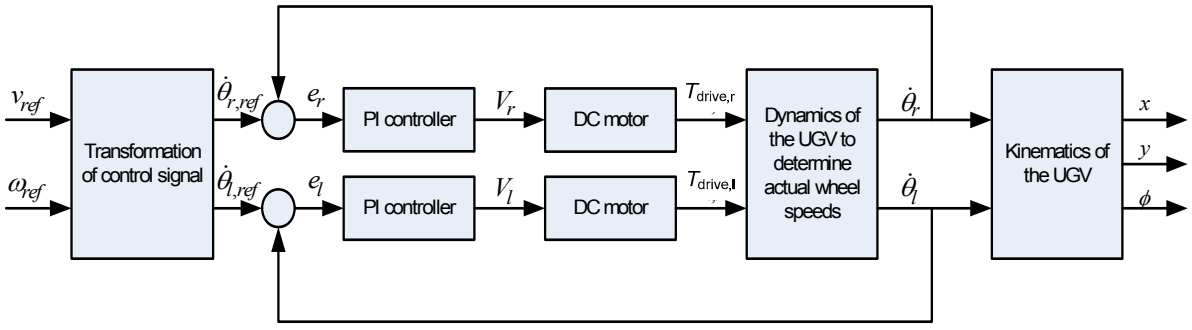

Fig. 2. Block diagram of the Unmanned Ground Vehicle (UGV).

From Fig. 2, v_{ref} and ω_{ref} is transformed to the reference speeds for the right and the left

wheel of the UGV ($\dot{\theta}_{r,ref}$ and $\dot{\theta}_{l,ref}$). Then, subtracted by the actual speed of the wheels ($\dot{\theta}_r$ and $\dot{\theta}_l$), the error signals are fed to the PI controller to generate the driving voltages (V_r and V_l) for the DC motors. After that, the DC motors provide the driving torques ($T_{drive,r}$ and $T_{drive,l}$) to the wheels of the UGV which make the UGV run with the actual wheel speeds. Finally, the position of the UGV can be calculated from the actual wheel speeds by using UGV kinematics.

Each component in Fig. 2 can be described as follows:

(1) Transformation of control signal

Based on the v_{ref} and ω_{ref}, the reference speed for each wheel of the UGV $\dot{\theta}_{r,ref}$ and $\dot{\theta}_{l,ref}$ needed to be determined in order to control the wheel speed to meet the required reference speed and reference turn rate. To do that, we need to know the dimension of the UGV such as the wheel radius, r, and the distance between the wheels, which equals to $2b_1$ as shown in Fig. 1. The calculation of $\dot{\theta}_{r,ref}$ and $\dot{\theta}_{l,ref}$ is illustrated in Eqn. (1)

$$\begin{bmatrix} \dot{\theta}_{r,ref} \\ \dot{\theta}_{l,ref} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2b_1} & -\frac{r}{2b_1} \end{bmatrix}^{-1} \begin{bmatrix} v_{ref} \\ \omega_{ref} \end{bmatrix} \tag{1}$$

(2) PI controller

PI controller generates the driving voltage, V_r and V_l, from the error between the reference speed and the actual speed, e_r and e_l. The calculation of the driving voltage for the right wheel of the UGV (which is the same as the left wheel) is illustrated in Eqn. (2)-(4) where z_r is the state of the integrator used in the PI controller, K_p and K_i is the proportional

and integral gain respectively.

$$e_r = \dot{\theta}_{r,ref} - \dot{\theta}_r \tag{2}$$

$$\dot{z}_r = e_r \tag{3}$$

$$\begin{aligned} V_r(t) &= K_P e_r(t) + K_I \int_0^t e_r(\xi) d\xi \\ V_r(t) &= K_P e_r(t) + K_I z_r \end{aligned} \tag{4}$$

(3) Motor with gear train

The motor coupled with a gear train can provide a larger driving torque albeit at the cost of the speed. An important parameter describing how the motor speed is decreased by the gear train is gear ratio N_a as shown in Eqn. (5) where $\dot{\theta}_m$ is the speed of the motor shaft, and $\dot{\theta}$ is the speed of the gear shaft connected to the wheel. N_a also relates the load torque of the motor T_L with the load torque that drives the wheel of the UGV T_{drive} as shown in Eqn. (6). Note that the notations illustrated in this section are for the right wheel of the UGV which can also be used to describe the left wheel.

$$\dot{\theta}_{m,r} = N_a \times \dot{\theta}_r \tag{5}$$

$$T_{drive,r} = N_a \times T_{L,r} \tag{6}$$

The DC motor model with gear train can be simplified and described by Eqn. (7)-(8). The electrical dynamics of the motor is illustrated in Eqn. (7) where L_{motor} and R_{motor} are electrical inductance and electrical resistance respectively, K_e is the generator constant, i_{a_r} is the armature current. The mechanical dynamics of the motor is illustrated in Eqn. (8) where J_{motor} is the moment of inertia of the rotor and B_{motor} is the damper constant. The relationship

between electrical dynamics and mechanical dynamics is described in Eqn. (9) where K_t is the torque constant.

$$L_{motor}\frac{di_{a_r}}{dt}+R_{motor}i_{a_r}=V_r-K_e\dot{\theta}_{m,r} \tag{7}$$

$$J_{motor}\ddot{\theta}_{m,r}+B_{motor}\dot{\theta}_{m,r}+T_{L,r}=T_{m,r} \tag{8}$$

$$T_{m,r}=K_t i_{a_r} \tag{9}$$

Note that the effect of gear train such as the moment of inertia of the gear and damper can be embedded in the aforementioned motor parameters. The detailed model of the motor with gear train can be found in [2].

(4) UGV dynamics for determining the actual wheel speeds

The actual wheel speed of the UGV exerted by the motor driving torque is governed by the UGV dynamics. The calculation of the wheel speeds needs to consider the dimension of the UGV, the mass and moment of inertia of UGV's chassis, wheels, and motor rotors. This part of the UGV dynamics is determined by the Lagrange's equation of motion as illustrated in detail in [3]. For convenience, we repeat the result derived in [3] here as shown in Eqn. (10)-(13).

$$S^T MS\begin{bmatrix}\ddot{\theta}_r\\ \ddot{\theta}_l\end{bmatrix}+S^T M\dot{S}\begin{bmatrix}\dot{\theta}_r\\ \dot{\theta}_l\end{bmatrix}+S^T E=\begin{bmatrix}T_{drive,r}\\ T_{drive,l}\end{bmatrix} \tag{10}$$

$$S=\begin{bmatrix}cb_1\cos\phi & cb_1\cos\phi\\ cb_1\sin\phi & cb_1\sin\phi\\ 1 & 0\\ 0 & 1\end{bmatrix};\ c=2r/b_1 \tag{11}$$

$$M = \begin{bmatrix} m & 0 & -m_c c b_2 \sin\phi & m_c c b_2 \sin\phi \\ 0 & m & m_c c b_2 \cos\phi & -m_c c b_2 \cos\phi \\ -m_c c b_2 \sin\phi & m_c c b_2 \cos\phi & I_c^2 + I_w & -I_c^2 \\ m_c c b_2 \sin\phi & -m_c c b_2 \cos\phi & -I_c^2 & I_c^2 + I_w \end{bmatrix} \tag{12}$$

$$E = \begin{bmatrix} -m_c b_2 \dot{\phi}^2 \cos\phi \\ -m_c b_2 \dot{\phi}^2 \sin\phi \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

where

m_c is the mass of the UGV chassis without the driving wheels and the rotors of the DC motors;

m_w is the mass of each driving wheel plus the rotor of its motor;

I_c is the moment of inertia of the chassis without the driving wheels and the rotors of the motors about a vertical axis through the center of turn;

I_w is the moment of inertia of each wheel about the wheel axis;

I_m is the moment of inertia of each wheel and the motor rotor about the wheel diameter.

We can use Eqn. (10) as a model to simulate the wheel speed of a UGV given UGV parameters and driving torques at both driving wheels.

(5) UGV kinematics

We can calculate the position of the UGV (x,y) and the heading angle ϕ from the actual speed of the UGV by using geometry as shown in Eqn. (14) and (15). Integrating both side of Eqn. (15) and assuming that all the initial values are zero, we can find the ϕ from the displacement of the right and left wheel as shown in Eqn. (16).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} r\cos\phi & r\cos\phi \\ r\sin\phi & r\sin\phi \end{bmatrix} \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix} \tag{14}$$

$$\dot{\phi} = \frac{2r}{b_1}\left(\dot{\theta}_r - \dot{\theta}_l\right) \tag{15}$$

$$\phi = \frac{2r}{b_1}\left(\theta_r - \theta_l\right) \tag{16}$$

Therefore, we can find the output of the overall UGV model by Eqn. (17).

$$\begin{aligned} y &= \begin{bmatrix} x & y & \phi \end{bmatrix}^T = \mathbf{C}\mathbf{x} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2r/b_1 & 2r/b_1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x\, y\, \theta_r\, \theta_l\, \dot{\theta}_r\, \dot{\theta}_l\, i_{a_r}\, i_{a_l}\, z_r\, z_l \end{bmatrix}^T \end{aligned} \tag{17}$$

REFERENCES

[1] X. Yun and Y. Yamamoto, "Internal dynamics of a wheeled mobile robot," Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93), 1993.

[2] J. J. D'Azzo and C. H. Houpis, *Linear control system analysis and design : conventional and modern*, 4th ed. New York: McGraw-Hill, 1995.

[3] Y. Yamamoto, "Control and Coordination of Locomotion and Manipulation of a Wheeled Mobile Manipulator," University of Pennsylvania, 1994.

APPENDIX B

QUADRATIC CURVE PATH-TRACKING CONTROLLER

The path tracking controller ($\mathbf{g}(\cdot)$) generates the reference speed and reference turn rate, $\mathbf{u} = \left[v_{ref}, \omega_{ref} \right]^T$, for the UGV to track the path. Among the published path tracking algorithms, *pure pursuit* [1] is a well-known and a very efficient algorithm. Based on a simple geometric consideration, pure pursuit tracks the path by repeatedly fitting a circular arc trajectory for the vehicle from its current position to a reference point on the path. As the vehicle moves forward, the reference point also moves forward along the path. This algorithm will generate the reference speed for both wheels of the vehicle to track the fitted arcs and will eventually drive the vehicle to track the path. However, the pure pursuit method does not specify "*how*" to choose the reference point on the path. Therefore, we use an alternative algorithm called *quadratic curve* algorithm [2]. This algorithm is slightly modified from the pure pursuit algorithm, in which it also provides a means to choose an appropriate reference point **r** such that the UGV can effectively track a predefined path. The conceptual diagram of the quadratic curve algorithm is depicted in Fig. 1.

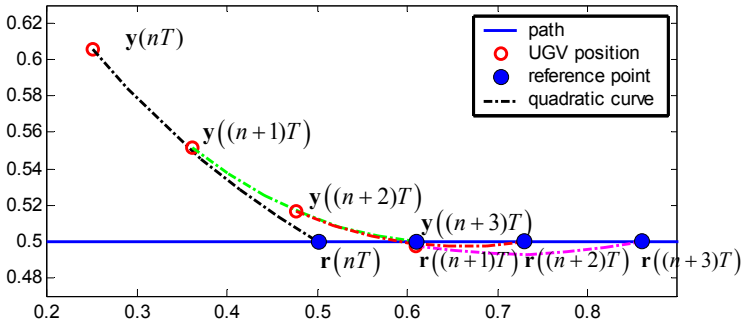

Fig. 1. Conceptual diagram for quadratic curve algorithm.

The UGV position is represented by a set of coordinate and a heading angle, $\mathbf{y} = [x\ y\ \phi]^T$. As shown in Fig. 1, the algorithm will drive the UGV position to track a parabolic curve. The curve is repeatedly drawn from the position of the UGV to the tracking path. The algorithm will drive the vehicle along the curve and eventually track the desired path [2]. The algorithm can be described in steps as follows:

(1) Find reference point

The path tracking controller is embedded with a navigation module for providing an updated reference point $\mathbf{r} = [x_{ref}\ y_{ref}]^T$ for the UGV to follow. The path description is assumed known to the navigation module and is described by

$$\mathbf{P}(d) \quad = [x_p(d) \quad y_p(d)]^T = \mathbf{x}_p(d) \tag{1}$$

where $x_p(d)$ and $y_p(d)$ are the mathematical functions parameterized by d to describe the path. For example, a 90 degree turn path can be described by

$$x_p(d) \quad = \begin{cases} 0; & d \le 0.5 \\ d - 0.5; & 0.5 < d \le 1.5 \end{cases} \tag{2}$$

$$y_p(d) \quad = \begin{cases} d; & d \le 0.5 \\ 0.5; & 0.5 < d \le 1.5 \end{cases} \tag{3}$$

Given $\mathbf{x}_p(d)$, the navigation module acquires a new reference point for the UGV by looking for a point on the path that keeps a certain distance d_0 away from the UGV. d_0 can be determined from the quadratic curve algorithm,

$$d_0 = d_{\max} / (1 + \beta |A|) \tag{4}$$

where $d_{\max}$ is the maximum look ahead distance, β is a constant, and A is the quadratic curve coefficient from the previous calculation of control signal described in Eqn. (7). The appropriate values of $d_{\max}$ and β was discussed in chapter II.

The new reference point is acquired by solving the following equation:

$$\sqrt{(x - x_p(d))^2 + (y - y_p(d))^2} = d_0 . \tag{5}$$

where d is the variable to be found and the reference point is $\mathbf{r} = [x_{ref} \; y_{ref}]^T = [x_p(d) \; y_p(d)]^T$.

Because the reference point should only move forward along the path to guide the UGV to the destination, we enforce a constraint that our reference point will not move backward along the path. Therefore, in the case that the UGV wanders away from the path or move backward, we will use the previous acquired reference point. Moreover, in addition to the original algorithm, we enforce a constraint to limit the minimum value of d_0 so that d_0 is always greater than $d_{\min}$ where $d_{\min}$ is a constant discussed in chapter II. This can eliminate the situation that d_0 is too small causing the UGV to highly oscillate around the path.

(2) Find error between the reference point and the UGV

Define the UGV coordinate during $t \in [kT, (k+1)T), i = 1, 2, 3, \ldots$ as the coordinate that has the origin at the center of the UGV at time $t = kT$ and has the X axis point to the same direction of the UGV. For each point of time, the UGV has the error between the reference point and the current position on the UGV coordinate calculated by:

$$\begin{bmatrix} e_x & e_y \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_{ref} - x \\ y_{ref} - y \end{bmatrix}. \tag{6}$$

(3) Find quadratic curve coefficient

The error between the reference point and current UGV position will be used to find a quadratic curve line that links between current UGV position and reference point. From the parabola curve equation, we have $e_y = A \times e_x^2$ where A is the parabolic curve coefficient. We can find A while considering the negative value of e_x as follows:

$$A = \mathrm{sgn}(e_x) \frac{e_y}{e_x^2}. \tag{7}$$

(4) Find control signal

We need to find the control signal $\mathbf{u} = \begin{bmatrix} v_{ref} & \omega_{ref} \end{bmatrix}^T$ that will drive the UGV along the quadratic curve. We can find v_{ref} by considering a small change of the distance the UGV travelled along the quadratic path on the UGV coordinate by:

$$ds = \sqrt{(dx_{UGV})^2 + (dy_{UGV})^2} = \sqrt{(dx_{UGV})^2 + 4A^2 x_{UGV}^2 (dx_{UGV})^2}. \tag{8}$$

$$v_{ref} = \frac{ds}{dt} = \sqrt{\dot{x}_{UGV}^2 \left(1 + 4A^2 x_{UGV}^2\right)}\,. \tag{9}$$

where $\left(x_{UGV}, y_{UGV}, \phi_{UGV}\right)$ represents the position of the UGV on the UGV coordinate.

Note that Eqn. (9) always gives a positive value regardless of the position of the reference point. In order to consider the reference point behind the UGV, we assign the value of v_{ref} as

$$v_{ref} = sgn(e_x)\sqrt{\dot{x}_{UGV}^2 (1 + 4A^2 x_{UGV}^2)}\,. \tag{10}$$

Next, we need to find ω_{ref}. We can find it by considering the definition of ϕ_{UGV} and the derivative of ϕ_{UGV} according to time t as follows:

$$\tan\phi_{UGV} = \frac{dy_{UGV}}{dx_{UGV}}\,. \tag{11}$$

$$\omega_{ref} = \frac{d\phi}{dt} = \frac{d\phi_{UGV}}{dt} = \frac{d\tan^{-1}\left(dy_{UGV}/dx_{UGV}\right)}{dt}\,. \tag{12}$$

$$\omega_{ref} = \frac{2A\dot{x}_{UGV}^3}{\dot{x}_{UGV}^2\left(1 + 4A^2 x_{UGV}^2\right)} = \frac{2A\dot{x}_{UGV}^3}{v_{ref}^2}\,. \tag{13}$$

Eqn. (10) and (13) can be simplified by assuming that the control signal will be used only for a small period of time. Let x_{UGV} at $t \in \left[kT, (k+1)T\right)$ be given by:

$$x_{UGV} = K \times \left(t - kT\right). \tag{14}$$

where $K = \mathrm{sgn}\left(e_x\right)\dfrac{\alpha}{1 + |A|}$, α is a constant representing the maximum reference speed.

Substituted x_{UGV} in Eqn. (10) and (13) and assumed that x_{UGV} is small as $(t-kT)$ is small, the control signal driving the UGV along the quadratic curve can be estimated as:

$$v_{ref} \cong K , \tag{15}$$

$$\omega_{ref} \cong 2AK . \tag{16}$$

Note that the UGV's PI controller will control the speed of the UGV so that the UGV's speed and turn rate will be close to v_{ref} and ω_{ref} as much as possible. It can be illustrated by geometry that the UGV's circular trajectory has the radius equal to R_{UGV} where $R_{UGV} = v_{ref} / \omega_{ref} = 1/\kappa_{UGV}$, and κ_{UGV} is defined as the curvature of the UGV trajectory.

REFERENCES

[1] A. Ollero and G. Heredia, "Stability analysis of mobile robot path tracking," Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'95), Pittsburgh, PA, 1995.

[2] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path tracking control of mobile robots using a quadratic curve," Proceedings of the 1996 IEEE Intelligent Vehicles Symposium, 1996.

APPENDIX C

GSM FOR REMOTE UGV PATH TRACKING

Gain Scheduler Middleware (GSM) is a gain scheduling method for alleviating the delay time effect on a NBC system. GSM externally adjusts the controller gain according to the operating condition and also preprocesses the feedback data from the plant before the controller uses them to calculate control signal. As shown in Fig. 1, there are three modules within GSM:

- Feedback preprocessor (FP)
- Network traffic estimator (NTE)
- Gain scheduler (GS)

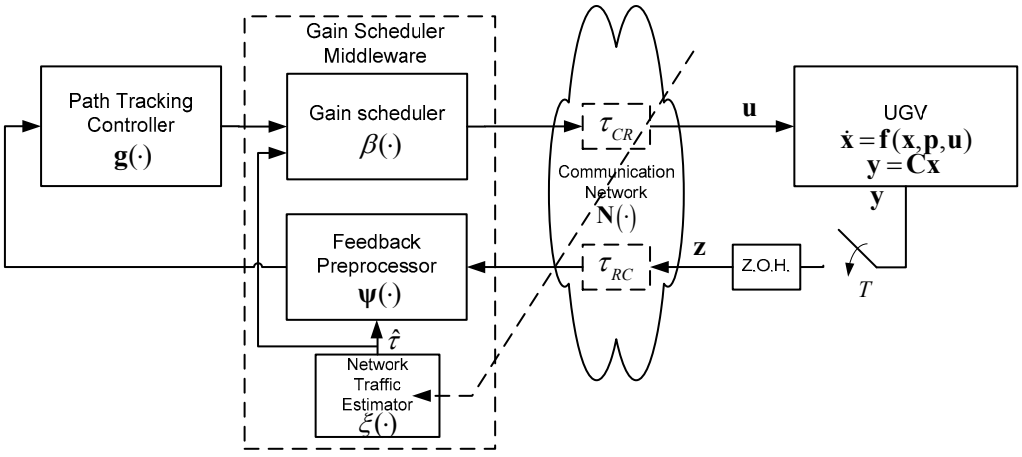

Fig. 1. Gain Scheduler Middleware for unmanned ground vehicle path-tracking.

In general, FP is employed to preprocess feedback data before it arrives at the controller to decrease the effect of communication network. Depended on the application, FP can be used to perform various related operations such as noise filtering and plant state estimation.

To preprocess data according to the communication network, NTE is useful for estimating the network condition that will be experienced by the next data packet. The estimated network condition can be useful for both the FP and GS module. After the controller acquires the preprocessed feedback data and generates control signal, GS module scales the control signal so that the effect is the same as adjusting the controller gain for performance optimization according to predefined costs.

This section focuses on the implementation of these three modules specifically for the remote UGV path-tracking based on the symbols described in chapter IV as follows:

A. Feedback Preprocessor (FP)

Feedback preprocessor (FP) alleviate time delay effect on the remote UGV path-tracking system by estimating the UGV's position when the UGV receives the control signal in the future. The estimated position of the UGV will be used for calculating control signal instead of the actual position in the past. When the UGV receives the control signal, the position of the UGV would be closer to the estimated position than the actual position in the past because of the UGV movement. Therefore, by using the estimated position provided by FP, the path tracking performance would be less affected by time delay. We can compare the situation between not using FP and using FP as shown in Eqn. (1) and Eqn. (2) respectively.

$$\text{Not using FP, } \mathbf{u}(t^+) = \mathbf{g}\left(\mathbf{y}(t-\tau^{k_i})\right),\ t \in \left\{k_i T + \tau^{k_i}, i = 1,2,3,...\right\} \tag{1}$$

$$\text{Using FP, } \mathbf{u}(t^+) = \mathbf{g}\left(\hat{\mathbf{y}}(t)\right),\ t \in \left\{k_i T + \tau^{k_i}, i = 1,2,3,...\right\} \tag{2}$$

Eqn. (1) shows that the control signal the UGV gets at time $k_i T + \tau^{k_i}$ is calculated based on the position of the UGV in the past at time $k_i T$. Distinct from Eqn. (1), Eqn. (2) shows

that the control signal is calculated based on the position the UGV is estimated to be at time *t*.

FP can be described by a function as shown in Eqn. (3). Note that FP requires the position of the UGV in the past $\mathbf{y}(t-\tau^{k_i})$, the control signal in the past $\mathbf{u}(t-\tau^{k_i})$, and the estimated RTT $\hat{\tau}^{k_i}$.

$$\hat{\mathbf{y}}(t) = \boldsymbol{\psi}\left(\mathbf{y}(t-\tau^{k_i}), \mathbf{u}(t-\tau^{k_i}), \hat{\tau}^{k_i}\right) \tag{3}$$

As described in detail in chapter IV, FP can be implemented by using Eqn. (4).

$$\begin{aligned} \hat{\phi}(t) &= \phi(t-\tau^{k_i}) + \omega_{ref}(t-\tau^{k_i}) \times \hat{\tau}^{k_i} \\ \hat{x}(t) &= x(t-\tau^{k_i}) + \frac{v_{ref}(t-\tau^{k_i})}{\omega_{ref}(t-\tau^{k_i})}\left(\sin\left(\phi(t-\tau^{k_i}) + \omega(t-\tau^{k_i}) \times \hat{\tau}^{k_i}\right) - \sin\phi(t-\tau^{k_i})\right) \\ \hat{y}(t) &= y(t-\tau^{k_i}) + \frac{v_{ref}(t-\tau^{k_i})}{\omega_{ref}(t-\tau^{k_i})}\left(\cos\phi(t-\tau^{k_i}) - \cos\left(\phi(t-\tau^{k_i}) + \omega(t-\tau^{k_i}) \times \hat{\tau}^{k_i}\right)\right) \end{aligned} \tag{4}$$

where $\hat{\mathbf{y}}(t) = \begin{bmatrix} \hat{x} & \hat{y} & \hat{\phi} \end{bmatrix}^T$. Note that $v_{ref}(t-\tau^{k_i})$ and $\omega_{ref}(t-\tau^{k_i})$ is used to estimate the speed and turn rate of the UGV at time $t-\tau^{k_i}$. In the case that the information of the actual speed and turn rate of the UGV is available and the UGV has a slow response to the reference speed and turn rate, we should replace $v_{ref}(t-\tau^{k_i})$ and $\omega_{ref}(t-\tau^{k_i})$ by using the actual speed and turn rate instead.

B. Network traffic estimator (NTE)

For the application of GSM in remote UGV path-tracking, Network traffic estimator is a module used to predict the round-trip-time (RTT) delay for the next data communication loop. NTE has an important effect on the performance of GSM in remote UGV path tracking because it provides an estimated RTT that will be used in Feedback preprocessor to predict

the position of the UGV in the future and will be used in Gain scheduler to adjust the gain of the controller to match current communication network condition.

NTE uses the history of the RTT in order to perform the estimation. The history of RTT can be collected from the difference in the time at which the path-tracking controller received consecutive sampled data from the UGV. NTE can be described as a function shown in Eqn. (5).

$$\hat{\tau}^{k_i} = \xi\left(\tau^{k_{i-1}}, \tau^{k_{i-2}}, ..., \tau^{k_{i-N_{NTE}}}\right) \tag{5}$$

NTE for remote UGV path-tracking estimates RTT by statistics such as mean and median of the RTT history. The implementation of FP can be described as

$$\hat{\tau}^{k_i} = \max(\eta, \mu) \tag{6}$$

where μ and η is the mean and median of the delay history respectively. We consider the mean value here, because, statistically, it is a good estimation of a random value as it reflects the least expected difference between the estimated value and the actual value. The median value is also considered here because, in some case when the distribution of the random delay is not normal but has a long tail either on the left or on the right, the median value can represents the majority of the random data better than the mean value. Then, we select the maximum value between the mean and median because we consider the worst case if we use the estimated delay in Gain scheduler module. The larger estimated delay would make GS lookup for the gain that is corresponding to slower UGV and has a larger safety margin for path-tracking error.

We calculate mean and median values of the RTT based on a number of the latest delay

data. We may keep this latest delay data in a FIFO queue, whose size denoted by N_{NTE}, by inserting new data in and fetching oldest data out. Therefore, the data remained in the queue is $\tau^{k_{i-1}}, \tau^{k_{i-2}}, ..., \tau^{k_{i-N_{NTE}}}$. Then, μ and η can be calculated from the data in the queue as shown in Eqn. (7)-(8).

$$\mu = \sum_{j=i-1}^{i-N_{NTE}} \tau^{k_j} \Big/ N_{NTE} \tag{7}$$

$$\eta = \begin{cases} \tilde{\tau}^i \;\; ; \;\; i = \dfrac{N_{NTE}+1}{2} \text{ if } \mathrm{N_{NTE}} \text{ is an odd number} \\ \dfrac{\tilde{\tau}^i + \tilde{\tau}^{i+1}}{2} \;\; ; \;\; i = \dfrac{N_{NTE}}{2} \text{ if } \mathrm{N_{NTE}} \text{ is an even number} \end{cases} \tag{8}$$

where $\left\{\tilde{\tau}^1, \tilde{\tau}^2, ..., \tilde{\tau}^{N_{NTE}}\right\}$ is the ascending sorted sequence of the data in the queue.

C. Gain scheduler

Gain scheduler for remote UGV path tracking provides a gain adjusted control signal that optimizes path-tracking cost based on current communication network and path tracking conditions. GS probes the communication network condition by using estimated RTT from FP. With limited information fed to GS, it probes the path tracking condition by using the control signal from the controller. The application of GS can be described by Eqn. (9)

$$\mathbf{u}(t^+) = \beta\left(\mathbf{g}\left(\hat{\mathbf{y}}(t)\right), \hat{\tau}^{k_{i+1}}\right) \times \mathbf{g}\left(\hat{\mathbf{y}}(t)\right) \;\; , t \in \left\{k_i T + \tau^{k_i}, i = 1, 2, 3, ...\right\} \tag{9}$$

where $\mathbf{g}\left(\hat{\mathbf{y}}(k_i T + \tau^{k_i})\right)$ is the control signal calculated by the path-tracking controller. $\mathbf{g}\left(\hat{\mathbf{y}}(k_i T + \tau^{k_i})\right) = \left[v_{ref}(k_i T + \tau^{k_i}) \quad \omega_{ref}(k_i T + \tau^{k_i})\right]^T = \left[K \quad K/R_{UGV}\right]^T$ where K is the speed gain of the UGV, and R_{UGV} is the UGV's radius of turn. After GS is applied, $\mathbf{u}(k_i T + \tau^{k_i})$ is

equal to $\left[K_{new} \quad K_{new}/R_{UGV}\right]^T$ where K_{new} is the new speed gain.

The objective of GS here is to find K_{new} that optimizes the path-tracking costs. There are two quantities here that we should be concerned about for the performance of the remote UGV path tracking, the path tracking error and the UGV tracking speed. Therefore, we need to define cost functions subjected to an optimization that would reflect the lowest path-tracking error and the highest tracking speed. Since the control signal from GS module will be applied only for one period until the UGV receive the next control signal, we define two cost function, J_{A} and J_{B}, to be evaluated for each period between two consecutive times the UGV received the control signal from the controller as illustrated in Eqn. (10) and (11).

$$J_A(i+1) = \sqrt{\Delta x^2(i+1) + \Delta y^2(i+1) + \Delta \phi^2(i+1)} \tag{10}$$

where $\left[\Delta x(i+1)\ \Delta y(i+1)\ \Delta\phi(i+1)\right]^T = \mathbf{y}(k_{i+1}T + \tau^{k_{i+1}}) - \mathbf{y}(k_iT + \tau^{k_i})$.

$$J_B(i+1) = -\left|\frac{s(k_{i+1}T + \tau^{k_{i+1}}) - s(k_iT + \tau^{k_i})}{(k_{i+1}T + \tau^{k_{i+1}}) - (k_iT + \tau^{k_i})}\right|. \tag{11}$$

From Eqn. (10), J_{A} is defined by the second norm of the difference between two consecutive states of the UGV. The larger J_{A} reflects the larger path tracking error as the UGV running for too far before receiving an updated control signal. From Eqn. (11), J_{B} is defined by the negative value of the average UGV speed magnitude where $s(\cdot)$ represents the distance the robot at the specified time travelled along its trajectory from the starting point.

For a typical optimization problem, we need to minimize the overall cost aggregated from J_{A} and J_{B}, for example the overall cost can be defined by weighting between J_{A} and J_{B} such as $J = w_A J_A + w_B J_B$. However, this remote UGV path tracking problem is related to a

nonlinear system whose analytical solution of the optimization problem is difficult to find. Moreover, we need to find the solution in realtime in order to schedule the speed gain for each control signal sent to the UGV. Therefore, we use the optimization constraints that minimize only J_B according to a predefined condition on J_A as shown in Eqn. (12) and (13).

$$J_A(i+1) \le \varepsilon \tag{12}$$

$$\min J_B(i+1) \tag{13}$$

The proposed optimization constraints limit J_A within an ε bound so that the UGV's path-tracking error caused by excessive RTT is limited to a certain bound. The constraints also minimize J_B so that the UGV will run as fast as possible which contributes to better performance.

Now, we need to find K_{new} , which contributes to the values of J_A and J_B that are corresponding to the optimization constraints. Note that the values of J_A(i+1) and J_B(i+1) belong to the future time period which is not here yet. Therefore, we need to be able to calculate predicted values of these costs based on the value of K_{new} and select K_{new} that makes the optimization constraints valid. The predicted values of the cost can be calculated based on the estimated position of the UGV which can be computed by using the same function as FP. Predicted J_A denoted by $\hat{J}_A$ can be calculated as shown in Eqn. (14)-(15) as we assume that T is small compare with τ^{k_i} , then $k_i T + \tau^{k_i} \cong k_{i+1} T$.

$$\hat{J}_A(i+1) = \sqrt{\Delta\hat{x}^2(i+1) + \Delta\hat{y}^2(i+1) + \Delta\hat{\phi}^2(i+1)} \tag{14}$$

where $\left[\Delta\hat{x}(i+1)\ \Delta\hat{y}(i+1)\ \Delta\hat{\phi}(i+1)\right]^T = \hat{\mathbf{y}}(k_{i+1}T + \tau^{k_{i+1}}) - \hat{\mathbf{y}}(k_i T + \tau^{k_i})$.

$$\begin{aligned}\hat{\mathbf{y}}(k_{i+1}T+\tau^{k_{i+1}})-\hat{\mathbf{y}}(k_iT+\tau^{k_i}) &\cong \hat{\mathbf{y}}(k_{i+1}T+\tau^{k_{i+1}})-\mathbf{y}(k_{i+1}T)\\ &= \boldsymbol{\psi}\left(\mathbf{y}(k_{i+1}T),\mathbf{u}(k_{i+1}T),\hat{\tau}^{k_{i+1}}\right)-\mathbf{y}(k_{i+1}T)\end{aligned} \tag{15}$$

$\hat{J}_A$ can be determined and be classified into two cases when $\kappa_{UGV}\neq 0\left(R_{UGV}\neq\infty\right)$ as in Eqn. (16) and when $\kappa_{UGV}=0\left(R_{UGV}=\infty\right)$ as in Eqn. (17).

When $\kappa_{UGV}\neq 0\left(R_{UGV}\neq\infty\right)$,

$$\begin{aligned}\hat{J}_A(i+1) &= \sqrt{\frac{v_{ref}^2(k_{i+1}T)}{\omega_{ref}^2(k_{i+1}T)}\left\{2-2\left(\cos\left(\omega_{ref}(k_{i+1}T)\times\hat{\tau}^{k_{i+1}}\right)\right)\right\}+\omega_{ref}^2(k_{i+1}T)\times\left(\hat{\tau}^{k_{i+1}}\right)^2}\\ &= \sqrt{\left\{2-2\left(\cos\left(K_{new}\cdot\kappa_{UGV}\cdot\hat{\tau}^{k_{i+1}}\right)\right)\right\}\Big/\kappa_{UGV}^2+\left(K_{new}\cdot\kappa_{UGV}\cdot\hat{\tau}^{k_{i+1}}\right)^2}\ .\end{aligned} \tag{16}$$

When $\kappa_{UGV}=0\left(R_{UGV}=\infty\right)$,

$$\begin{aligned}\hat{J}_A(i+1) &= \left|v_{ref}(k_{i+1}T)\times\hat{\tau}^{k_{i+1}}\right|\\ &= \left|K_{new}\times\hat{\tau}^{k_{i+1}}\right|.\end{aligned} \tag{17}$$

For the predicted value of J_B, denoted by $\hat{J}_B$, we can use the reference speed or K_{new} to predict the avarage speed of the UGV as $\hat{J}_B$ can be calculated by Eqn. (18)

$$\hat{J}_B(i+1) = -\left|K_{new}\right| \tag{18}$$

Now, we can calculate $\hat{J}_A$ and $\hat{J}_B$ based on the value of K_{new}, κ_{UGV}, and $\hat{\tau}^{k_{i+1}}$ where $\hat{\tau}^{k_{i+1}}\cong\hat{\tau}^{k_{i+1}}$. Therefore, for each pair of κ_{UGV}, and $\hat{\tau}^{k_i}$, we can find one value of K_{new} that complies with the constraints in Eqn. (12) and (13). K_{new} can be written as a function as $K_{new}(\kappa_{UGV},\hat{\tau}^{k_{i+1}})$. The purpose of GS module is to find $K_{new}(\kappa_{UGV},\hat{\tau}^{k_{i+1}})$ that generates $\hat{J}_A$ and

$\hat{J}_B$ according to the constraints. Since we need to find K_{new} in real-time, it is appropriate to implement $K_{new}(\kappa_{UGV}, \hat{\tau}^{k_{i+1}})$ by using lookup table in which we can do offline search for optimized values to fill.

To do offline search for K_{new}, we define a range of possible K_{new} and use bisection algorithm as shown in Fig. 2 to find a qualified K_{new} for each pair of κ_{UGV}, and $\hat{\tau}^{k_i}$. The algorithm separates the possible range into two equal ranges and checks for the range that contains the qualified value of K_{new}. The eligible range is then subdivided into smaller and smaller ranges until the size of the range is very small and the difference of K_{new} and the boundary of the range is negligable. We can set the number of iterations we need, *N,* to be used in the algorithm.

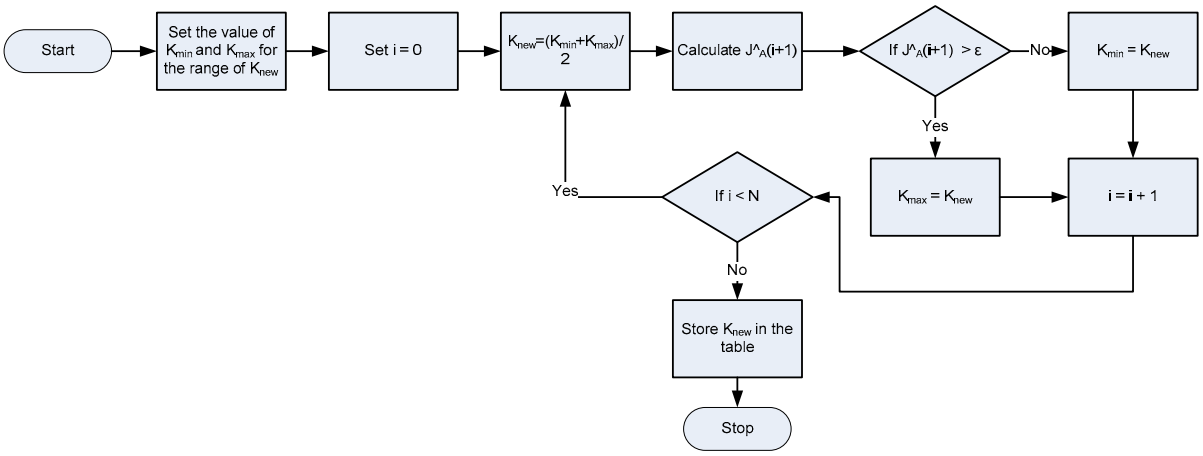

Fig. 2. Bisection algorithm for each operating condition in GS gain table generation.

Note that we can use this algorithm because $\hat{J}_B$ is a decreasing function of K_{new} and $\hat{J}_A$ is a nondecreasing function of K_{new}. (We know that $\hat{J}_A$ is nondecreasing because the derivative of $\hat{J}_A$ according to K_{new} is always greater than or equal to zero.) This allow us to

decide if a range of K_{new} does not contain a qualified value by checking only the value of K_{new} at the boundary of the range.