Gain Scheduler Middleware: A Methodology to Enable Existing Controllers for Networked Control and Teleoperation—Part II: Teleoperation

Yodyium Tipsuwan, *Member, IEEE,* and Mo-Yuen Chow, *Senior Member, IEEE*

***Abstract*—This paper is the second of two companion papers. The foundation for the external gain scheduling approach to enable an existing controller via middleware for networked control with a case study on a proportional–integral (PI) controller for dc motor speed control over IP networks was given in Part I. Part II extends the concepts and methods of the middleware called gain scheduler middleware (GSM) in Part I to enable an existing controller for mobile robot path-tracking teleoperation. By identifying network traffic conditions in real-time, the GSM will predict the future tracking performance. If the predicted tracking performance tends to be degraded over a certain tolerance due to network delays, the GSM will modify the path-tracking controller output with respect to the current traffic conditions. The path-tracking controller output is modified so that the robot will move with the fastest possible speed, while the tracking performance is maintained in a certain tolerance. Simulation and experimental results on a mobile robot path-tracking platform show that the GSM approach can significantly maintain the robot path-tracking performance with the existence of IP network delays.**

***Index Terms*—Adaptive control, control systems, dc motors, distributed control, Internet, mobile robots, networks, real-time system, telerobotics.**

I. Introduction

PART I of this paper [1] introduced the fundamental concept of external gain scheduling and the structure of gain scheduler middleware (GSM) to enable an existing controller for networked control and teleoperation with a case study on a proportional–integral (PI) controller for dc motor speed control over IP network delays [1]. The PI controller gains are adapted externally at the control output via the GSM with respect to the current network traffic conditions. Since an existing PI controller can be utilized for networked control without interrupting the internal controller operation, the GSM approach can save much cost and time for practical controller upgrades. In Part II, another case study to apply GSM on a more complicated teleoperation application, which is mobile robot path-tracking control over an IP network, is presented. The similar GSM concept and structure will be applied to enable an existing robot path-tracking controller for teleoperation control. By identifying network traffic conditions in real time, the GSM will predict the future tracking performance. If the predicted tracking performance tends to be degraded over a certain tolerance due to network delays, the GSM will modify the path-tracking controller output with respect to the current traffic conditions. The path-tracking controller output is modified so that the robot will move with the fastest possible speed, while the tracking performance is maintained in a certain tolerance.

Manuscript received September 10, 2003; revised May 31, 2004. Abstract published on the Internet September 10, 2004. This work was supported in part by the Royal Thai Government.

Y. Tipsuwan is with the Department of Computer Engineering, Kasetsart University, Bangkok 10900, Thailand (e-mail: yyt@ku.ac.th).

M.-Y. Chow is with the Advanced Diagnosis And Control (ADAC) Laboratory, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27606 USA (e-mail: chow@eos.ncsu.edu).

Digital Object Identifier 10.1109/TIE.2004.837865

II. Case Study: GSM for Remote Mobile Robot Path Tracking

A mobile robot path-tracking problem is used to illustrate the GSM concept and its effectiveness for mobile robot teleoperation. The robot model and path-tracking algorithm are described as follows.

A. Mobile Robot Model

The robot used to illustrate the proposed approach is a differential drive mobile robot with two driving wheels and two caster wheels [2] as shown in Fig. 1.

The mobile robot is described as

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2} & \frac{\rho}{2} \\ \frac{\rho}{W} & -\frac{\rho}{W} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = \begin{bmatrix} \omega_{R,r} + \varepsilon_R \\ \omega_{L,r} + \varepsilon_L \end{bmatrix} \tag{2}$$

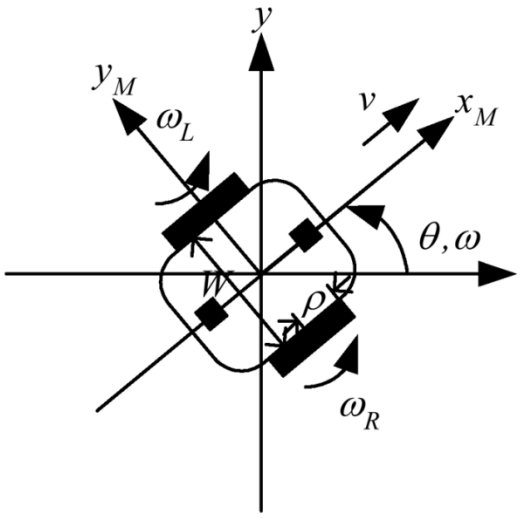
$$\dot{x} = v\cos\theta \tag{3}$$

$$\dot{y} = v\sin\theta \tag{4}$$

$$\dot{\theta} = \omega \tag{5}$$

where (x, y) is the position in the inertial coordinate, (x_M, y_M) is the position in the robot coordinate, θ is the azimuth angle of the robot, v is the linear velocity of the robot, W is the distance between the two wheels, ρ is the radius of the wheels, ω is the angular velocity of the robot, ω_L and ω_R are the angular velocities of the left and right wheels, $\omega_{L,r}$ and $\omega_{R,r}$ are the reference angular velocities for wheel speed controllers at the left and right wheels, and ε_L and ε_R are the differences between the reference velocities and the actual velocities of the left and right wheels, respectively. The speed of each wheel is controlled by a PI controller

$$u(t) = K_P e(t) + K_I \int_0^t e(\xi)d\xi \tag{6}$$



(a)

(b)

Fig. 1. Differential drive mobile robot. (a) Robot drawing diagram. (b) Actual mobile robot platform.

where K_P is the proportional gain, K_I is the integral gain, $u(t)$ is the input voltage to a dc motor, and $e(t)$ is the error, which can be either ε_L or ε_R.

B. Path-Tracking Algorithm

A generalization of the quadratic curve approach proposed in [2] is used as the path-tracking algorithm in our illustration. The main concept of this path-tracking algorithm is to move the robot along a quadratic curve to a reference point on a desired path. A point on the path is described in the inertial coordinate as $(x_p(s), y_p(s))$, where s is the distance traveled on the path. By assuming that the orientation of the mobile robot moves close to the desired value in the motion along the reference path, this algorithm controls only the position of the robot regardless to its orientation. This algorithm is suitable for real-time usage because of its simple computation with minimal amount of information compared to other approaches. The algorithm is outlined as follows.

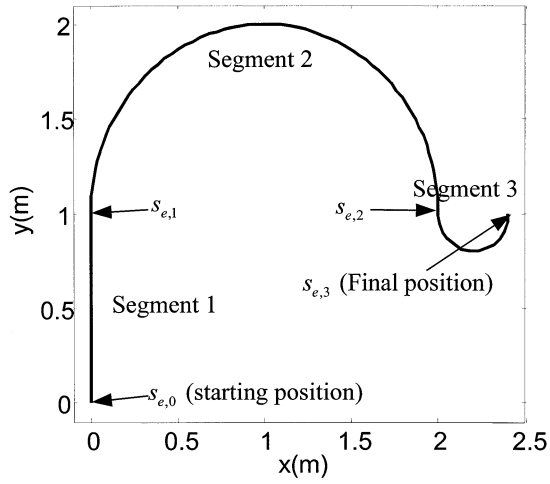


Fig. 2. Example of robot path.

1) Based on the current robot position $\mathbf{x}(i) = [x(i) \quad y(i) \quad \theta(i)]^{\mathrm{T}}$, where $i \in \mathbb{N}^+$ is the iteration number, optimize

$$\min_s \sqrt{(x_p(s) - x(i))^2 + (y_p(s) - y(i))^2} \tag{7}$$

to find $s = s_0$ that gives the closest distance between the robot and the path. Depending on the forms of $x_p(s)$ and $y_p(s)$, this optimization could be performed in real-time by using a closed-form solution, or a lookup table and a numerical technique such as linear interpolation. The iteration number i can be thought of as the sampling time index of the path-tracking controller if $t_{i+1} - t_i$ is constant.

2) Compute the reference point for the robot to track. Without loss of generality, in this paper, the path is constructed by a combination of lines and curves. Each line or curve has a constant curvature.Fig. 2 shows an example of a robot path, which is the combination of

• Segment 1: Straight line:

$$x_p(s) = 0, y_p(s) = s, \quad \text{if } s_{e,0} \le s \le s_{e,1}, s_{e,0} = 0, s_{e,1} = 1 \tag{8}$$

• Segment 2: Arc with a radius of 1:

$$\begin{aligned} x_p(s) &= 1 - \cos(s-1), \\ y_p(s) &= 1 + \sin(s-1), \quad \text{if } s_{e,1} < s \le s_{e,2}, s_{e,2} = 1 + \pi \end{aligned} \tag{9}$$

• Segment 3: Arc with a radius of 0.2:

$$\begin{aligned} x_p(s) &= 2.2 - 0.2\cos 5(s - 1 - \pi) \\ y_p(s) &= 1 - 0.2\sin 5(s - 1 - \pi) \\ &\text{if } s_{e,2} < s \le s_{e,3}, s_{e,3} = 1 + 1.2\pi \end{aligned} \tag{10}$$

where $\theta_p(s)$ is the tangent angle at $(x_p(s), y_p(s))$, j is the index of the jth segment of the path, $\kappa_j = d\theta_p(s)/ds$

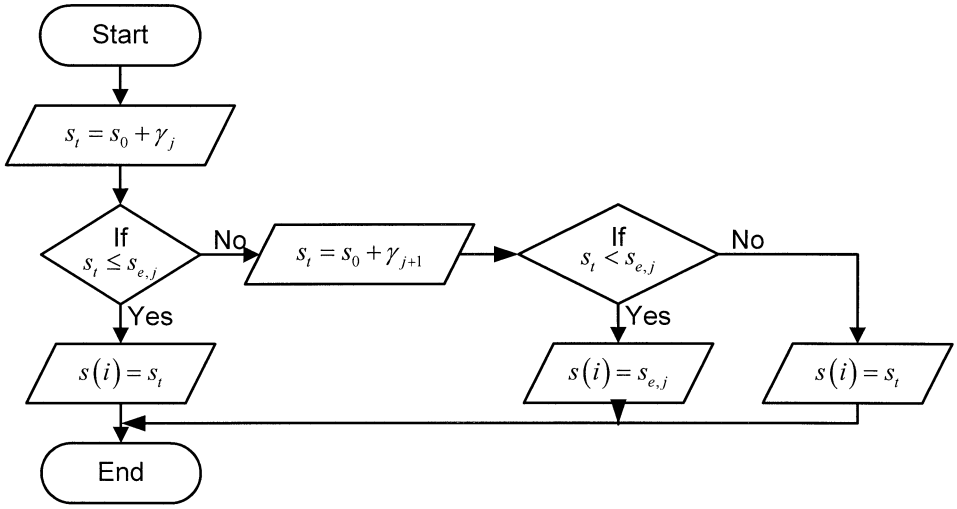


Fig. 3. Procedures for determining the reference distance traveled $s(i)$.

is the curvature of the j-th segment, $s_{e,j}$ is the endpoint of the j-th segment. The reference position $\mathbf{x}_r(i) = [x_r(i) \quad y_r(i) \quad \theta_r(i)]^{\mathrm{T}}$ is computed from $x_r(i) = x_p(s(i))$, $y_r(i) = y_p(s(i))$, $\theta_r(i) = \theta_p(s(i))$, where $s(i)$ is the reference distance traveled and is determined by the procedures shown in Fig. 3.

The value of $s(i)$ is initially determined from s_0 by

$$s(i) = s_t = s_0 + \gamma_j, \gamma_j = \frac{s_{\max}}{1 + \beta\kappa_j} \tag{11}$$

where s_t is a temporary variable, γ_j is the projecting distance, $s_{\max} \le s_{e,j} - s_{e,j+1}, \forall j$, is the maximal projecting distance, and $\beta \in \mathbb{R}^+$ is a positive constant. The projecting distance indicates how far the reference distance traveled should be projected ahead from $(x_p(s_0), y_p(s_0))$. The values of the constants $s_{\max}$ and β depend on the robot path, the robot configuration, and the designer's preference, whereas the curvature κ_j depends only on the jth segment of the path to track. The reference point will be closer to $(x_p(s_0), y_p(s_0))$ if κ_j is high. However, if $s(i) = s_t > s_{e,j}$, $s(i)$ will not be on the jth segment. In this case, the controller needs to evaluate if the robot should track the path based on segment j or segment $j+1$. For evaluation, s_t is recomputed by

$$s(i) = s_t = s_0 + \gamma_{j+1}. \tag{12}$$

When $s(i) = s_t < s_{e,j}$, $s(i)$ may be less than $s(i-1)$, which causes the robot to move backtrack if $(x_p(s(i)), y_p(s(i)))$ is used as the reference position. Therefore, the better choice of $s(i)$ in this case should be $s_{e,j}$ in order to guarantee that the robot will not move backtrack.

3) Compute the error $\mathbf{x}_r(i) - \mathbf{x}(i)$, and transform this error to the error in the robot coordinate as

$$\mathbf{e}(i) = [e_x \quad e_y \quad e_\theta]^{\mathrm{T}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{x}_r(i) - \mathbf{x}(i)). \tag{13}$$

4) Find a quadratic curve that links between $\mathbf{x}(i)$ and $\mathbf{x}_r(i)$ from

$$y_M = A(i) x_M^2, \text{ where } A(i) = \mathrm{sgn}(e_x) \frac{e_y}{e_x^2}. \tag{14}$$

The robot will move forward if $\mathbf{x}_r(i)$ is in front of the robot ($e_x > 0$). On the other hand, the robot will move backward if $\mathbf{x}_r(i)$ is behind of the robot ($e_x < 0$).

5) Compute the reference linear and angular velocities of the robot along the quadratic curve. The original equations of the velocities are

$$v_r(i) = \mathrm{sgn}(e_x) \sqrt{\dot{x}_M^2 (1 + 4A^2(i) x_M^2)} \tag{15}$$

$$\omega_r(i) = \frac{2A(i)\dot{x}_M^3}{v_r^2(i)}. \tag{16}$$

Let x_M at $t_i \le t < t_{i+1}$ be given by

$$x_M = K(i)(t - t_i) \tag{17}$$

where

$$K(i) = \mathrm{sgn}(e_x) \frac{\alpha}{1 + |A(i)|} \tag{18}$$

and α is a positive constant used as a speed factor. The robot will move fast if α is set to a high value, and vice versa. In order to control the robot to move fast so it can arrive at a destination with the minimal time requirement, a large gain $K(i)$ is usually required. If $t - t_i$ is very small, $v_r(i)$ can be approximated during $t_i \le t < t_{i+1}$ by

$$v_r^2(i) = K^2(i)\left(1 + 4A^2(i)K^2(i)(t - t_i)^2\right) \simeq K^2(i). \tag{19}$$

Thus,(15) and (16) can be approximated by

$$\hat{v}_r(i) \simeq K(i) \tag{20}$$

$$\hat{\omega}_r(i) \simeq 2A(i)K(i). \tag{21}$$

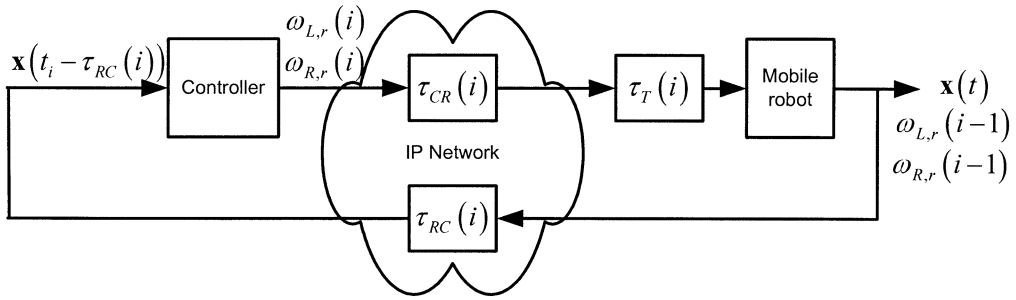


Fig. 4. Data flow of networked mobile robot.

The reference speeds of both wheels are then calculated by

$$\omega_{R,r}(i) = \frac{\hat{v}_r(i)}{\rho} + \frac{W\hat{\omega}_r(i)}{2\rho} \tag{22}$$

$$\omega_{L,r}(i) = \frac{\hat{v}_r(i)}{\rho} - \frac{W\hat{\omega}_r(i)}{2\rho}. \tag{23}$$

6) Repeat all steps by going back to 1) and set $i = i + 1$.

C. Path-Tracking Control Over an IP Network

In order to control a mobile robot to track a predefined path over an IP network, the path-tracking controller computes and sends the reference speed $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ in a packet across the network at every iteration i to the robot as shown in Fig. 4.

The path-tracking computation at iteration i starts when the controller receives the feedback data in a packet from the mobile robot at time $t = t_i$. Compared with the network delays, the computation time at the controller is usually and relatively insignificant. Thus, the computation could be assumed to finish at $t = t_i$ as well. The basic arrival feedback data in this case are the reference speeds $\omega_{L,r}(i-1)$ and $\omega_{R,r}(i-1)$, and the robot position $\mathbf{x}(t_i - \tau_{RC}(i))$, where $\tau_{RC}(i)$ is the network delay from the robot to the controller at i. The controller then sends $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ to the robot once the computation is finished. Likewise, $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are also delayed by the network. The network delay to send these reference speeds to the mobile robot is defined as $\tau_{CR}(i)$. The robot then periodically monitors and updates the reference speeds by the newly arrival data of $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ at every sampling time period T. The waiting time to update the reference speeds is defined as $\tau_T(i)$.

Network Delay Effect on Path-Tracking Algorithm: To modify the controller output with respect to network conditions characterized by $\mathbf{q}$, the effects of network delays on the mobile robot have to be analyzed. There are two concerns in the use of the original path-tracking algorithm due to $\tau_{CR}(i)$ and $\tau_{RC}(i)$.

1) Due to $\tau_{RC}(i)$, the controller does not have the current robot position $\mathbf{x}(t_i)$, but $\mathbf{x}(t_i - \tau_{RC}(i))$.
2) The reference speeds $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are computed at $t = t_i$, but will be applied at $t = t_i + \tau_{CR}(i) + \tau_T(i)$.

If the controller directly uses $\mathbf{x}(t_i - \tau_{RC}(i))$ as $\mathbf{x}(i)$ to compute $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, and if $\mathbf{x}(t_i - \tau_{RC}(i))$ and $\mathbf{x}(t_i)$ are very different, then the result may be far away from what it actually should be. In addition, even if the controller uses $\mathbf{x}(t_i)$ to compute $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, the robot might have already moved to another position at $t = t_i + \tau_{CR}(i) + \tau_T(i)$ when $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are applied. Thus, the robot response can be undesirable. The delay of $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ is crucial if the robot moves far away from a desired position. In addition, with long network delays, $t - t_i$ may be large, and the approximation in (19) may be no longer valid. Thus, the robot may not follow a desired quadratic trajectory.

Feedback Preprocessor: In this paper, we use feedback preprocessor to predict the future position of the mobile robot at $t = t_i + \tau_{CR}(i) + \tau_T(i)$. The predicted position is then forwarded to the path-tracking controller. A future position at $t = t_i + \tau_{CR}(i) + \tau_T(i)$, defined as $\mathbf{x}(t_i + \tau_{CR}(i) + \tau_T(i))$, is predicted from $\mathbf{x}(t_i - \tau_{RC}(i))$. This predicted position, defined as $\hat{\mathbf{x}}(t_i + \tau_{CR}(i) + \tau_T(i))$, is then used instead of $\mathbf{x}(i)$ for the path-tracking controller. Thus, $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ should be properly applied when the packet containing the reference speeds reaches the robot $t = t_i + \tau_{CR}(i) + \tau_T(i)$.

Before the mobile robot receives the reference speeds $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, both left and right wheels have been controlled by using $\omega_{L,r}(i-1)$ and $\omega_{R,r}(i-1)$ as the reference speeds. Thus, the robot can be assumed to move with constant linear and angular velocities if the wheel speed controllers of both wheels work perfectly such that $\varepsilon_L \to 0$, $\varepsilon_R \to 0$ quickly, and the weight of the robot is light. From this assumption, (3)–(5), (20), and (21), we can approximate the robot movement during $[t_i - \tau_{RC}(i), t_i + \tau_{CR}(i) + \tau_T(i)]$ using

$$\begin{aligned}\boldsymbol{\Delta}_\tau \mathbf{x}(i) &= \mathbf{x}(t_i + \tau_{CR}(i) + \tau_T(i)) - \mathbf{x}(t_i - \tau_{RC}(i)) \\ &= [\,\Delta_\tau x(i) \quad \Delta_\tau y(i) \quad \Delta_\tau \theta(i)\,]^{\mathrm{T}}\end{aligned} \tag{24}$$

where

$$\Delta_\tau \theta(i) \simeq \hat{\omega}_r(i-1)\,\tau(i), \tau(i) = \tau_{CR}(i) + \tau_T(i) + \tau_{RC}(i). \tag{25}$$

1) If $\hat{\omega}_r(i-1) \neq 0$

$$\begin{aligned}\Delta_\tau x(i) \simeq &\frac{\hat{v}_r(i-1)}{\hat{\omega}_r(i-1)} \\ &\times [\sin\theta(t_i + \tau_{CR}(i) + \tau_T(i)) - \sin\theta(t_i - \tau_{RC}(i))]\end{aligned} \tag{26}$$

$$\begin{aligned}\Delta_\tau y(i) \simeq &\frac{\hat{v}_r(i-1)}{\hat{\omega}_r(i-1)} \\ &\times [\cos\theta(t_i - \tau_{RC}(i)) - \cos\theta(t_i + \tau_{CR}(i) + \tau_T(i))].\end{aligned} \tag{27}$$

2) If $\hat{\omega}_r(i-1)=0$

$$\Delta_\tau x(i) \simeq \hat{v}_r(i-1)\,\tau(i)\cos\theta(t_i-\tau_{RC}(i)) \quad (28)$$
$$\Delta_\tau y(i) \simeq \hat{v}_r(i-1)\,\tau(i)\sin\theta(t_i-\tau_{RC}(i)). \quad (29)$$

The delay variable $\tau(i)$ is estimated by the network traffic estimator. The predicted position $\hat{\mathbf{x}}(t_i+\tau_{CR}(i)+\tau_T(i))$ is then computed from

$$\hat{\mathbf{x}}(t_i+\tau_{CR}(i)+\tau_T(i)) = \mathbf{x}(t_i-\tau_{RC}(i)) + \Delta_\tau\hat{\mathbf{x}}(i) \quad (30)$$

where $\Delta_\tau\hat{\mathbf{x}}(i)$ is the approximation of $\boldsymbol{\Delta}_\tau\mathbf{x}(i)$ computed from (24)–(29).

Gain Scheduler: To avoid the robot deviating far from a desired position, the gain scheduler is used to first evaluate the predictive movement of the robot. If the robot tends to move too fast and could be farther from the desired position because of network delays, the gain scheduler will update $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ to compensate the network delay $\tau(i)$ before sending the reference speed signals out. To evaluate the robot movement with respect to $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ ahead of time, we could utilize (24)–(30) to define the following cost function:

$$\begin{aligned}\text{Min}\,\hat{J}_1(i+1) &= \|\boldsymbol{\Delta}_\tau\hat{\mathbf{x}}(i+1)\|_2, \\ &= \|\hat{\mathbf{x}}(t_{i+1}+\tau_{CR}(i+1)+\tau_T(i+1)) \\ &\quad -\hat{\mathbf{x}}(t_{i+1}-\tau_{RC}(i+1))\|_2 \quad (31)\\ \text{Min}\,\hat{J}_2(i+1) &= -|K(i)| \quad (32)\end{aligned}$$

where $\|\bullet\|_2$ is the Euclidean norm, $\hat{\mathbf{x}}(t_{i+1}-\tau_{RC}(i+1)) \approx \hat{\mathbf{x}}(t_i+\tau_{CR}(i)+\tau_T(i))$, and $\hat{\mathbf{x}}(t_{i+1}+\tau_{CR}(i+1)+\tau_T(i+1))$ is the predicted position, which can be determined by using $\hat{\mathbf{x}}(t_{i+1}-\tau_{RC}(i+1))$ and a predicted delay $\hat{\tau}(i+1) \approx \tau_{RC}(i+1)+\tau_{CR}(i+1)+\tau_T(i+1)$. Likewise, assume that $\hat{\tau}(i+1)$ is estimated by the network traffic estimator. The cost function $\hat{J}_1(i+1)$ represents the amount of robot movement with respect to the predicted network delay after the robot receives the reference speed signals $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$. A large value of $\hat{J}_1(i+1)$ implies that the robot could significantly be affected by network delays. On the other hand, $\hat{J}_2(i+1)$ is linearly proportional to the speed of the robot since both $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are a linear function of $K(i)$. Minimizing $\hat{J}_2(i+1)$ is equivalent to maximizing $|K(i)|$. Depending on the actual robot performance requirement (e.g., maximal efficiency control), other cost functions could also be used.

From (25)–(29), $\hat{J}_1(i+1)$ could be also expressed as follows.

1) If $\hat{v}_r(i)=0$ and $\hat{\omega}_r(i)=0$

$$\hat{J}_1(i+1)=0. \quad (33)$$

2) If $\hat{v}_r(i)=0$ and $\hat{\omega}_r(i)\neq 0$

$$\hat{J}_1(i+1)=|\hat{\omega}_r(i)\,\hat{\tau}(i+1)|. \quad (34)$$

3) If $\hat{v}_r(i)\neq 0$ and $\hat{\omega}_r(i)\neq 0$

$$\hat{J}_1(i+1)=\sqrt{\frac{1-\cos\hat{\omega}_r(i)\,\hat{\tau}(i+1)}{2A^2(i)}+(\hat{\omega}_r(i)\,\hat{\tau}(i+1))^2}. \quad (35)$$

4) If $\hat{v}_r(i)\neq 0$ and $\hat{\omega}_r(i)=0$

$$\hat{J}_1(i+1)=|\hat{v}_r(i)\,\hat{\tau}(i+1)|. \quad (36)$$

In a vigorous approach, to find the optimal $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, a weighted cost function based on $\hat{J}_1(i+1)$ and $\hat{J}_2(i+1)$ can be formed and an optimal control strategy can be applied to minimize $\hat{J}_1(i+1)$ and $\hat{J}_2(i+1)$. However, this approach may not be suitable for the path-tracking algorithm used in real time because the algorithm is highly nonlinear with uncertain delays and disturbances. A heuristic approach can provide a feasible solution by maximizing $|K(i)|$ while maintaining $\hat{J}_1(i+1)\leq\varepsilon$, where ε is defined as the tracking performance degradation tolerance. In this case, $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ are modified based on their original values so that the robot will move as fast as possible by minimizing $\hat{J}_2(i+1)$ while $\hat{J}_1(i+1)$ is maintained at an acceptable small value. This approach does not minimize $\hat{J}_1(i+1)$ as in the vigorous approach, but can provide feasible $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$, which are optimal under the condition $\hat{J}_1(i+1)\leq\varepsilon$. In practice, gain scheduler will optimally modify $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ when $\hat{J}_1(i+1)>\varepsilon$ so that the robot will move as fast as possible based on $\hat{\tau}(i+1)$.

Because $A(i)$ is fixed by the path-tracking algorithm as the requirement of the robot trajectory in (21), the speed modification is equivalent to adjusting the gain $K(i)$ in (20) and (21). The optimal values of $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ in (34) and (36) can be determined by solving $\hat{\omega}_r(i)$ and $\hat{v}_r(i)$, respectively, whereas (35) requires a numerical method to solve (35) for $\hat{\omega}_r(i)$ to find the optimal $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$. Since $\hat{\omega}_r(i)\simeq 2A(i)K(i)$, (35) could be viewed as a function of $A(i)$, $K(i)$ and $\hat{\tau}(i+1)$. Because $A(i)$ and $\hat{\tau}(i+1)$ are given, $\hat{J}_1(i+1)$ will be determined by $K(i)$. The optimal values of $K(i)$ with respect to $A(i)$ and $\hat{\tau}(i+1)$ subject to $\hat{J}_1(i+1)\leq\varepsilon$ can be found by computing $\hat{J}_1(i+1)$ from various combinations of $A(i)$, $K(i)$ and $\hat{\tau}(i+1)$ in actual ranges of operating conditions. Then, by fixing $A(i)$ and $\hat{\tau}(i+1)$, we can search for the optimal $K(i)$ with an iterative approach that gives $\hat{J}_1(i+1)\leq\varepsilon$. These optimal $K(i)$ values are then stored in a lookup table and will be utilized by gain scheduler to compute the optimal $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$. For example, Fig. 5 shows the cost surfaces of $\hat{J}_1(i+1)$ with respect to $A(i)$, $K(i)$, and $\hat{\tau}(i+1)$ with $\varepsilon=0.2$.

As shown in Fig. 5, ε can be thought of as a plane cutting through multiple surfaces of cost $\hat{J}_1(i+1)$ with different values of $K(i)$. The optimal $K(i)$ with respect to $A(i)$ and $\hat{\tau}(i+1)$ chosen to modify $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ in this case is the largest $K(i)$ that has to be under or at the ε plane.

Fig. 6 shows the surface of the optimal $K(i)$ with respect to $A(i)$ and $\hat{\tau}(i+1)$ with $\varepsilon=0.2$. As indicated in Fig. 6(a)–(c), if $A(i)$ and $\hat{\tau}(i+1)$ are low, the optimal $K(i)$ is large. This implies that the robot can move very fast if the curvature of

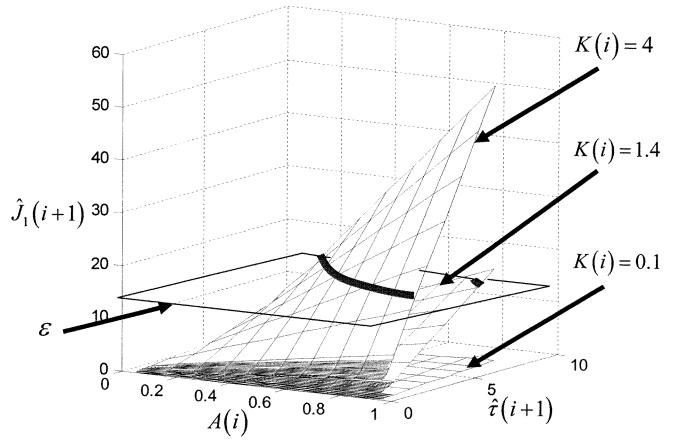


Fig. 5. Cost surfaces of $\hat{J}_1\,(i+1)$ with respect to $A\,(i)$, $K\,(i)$, and $\hat{\tau}\,(i+1)$ with $\varepsilon = 0.2$.

the quadratic curve is small and the network delay is short. According to Fig. 6(b), a larger $A\,(i)$ enforces the optimal $K\,(i)$ to be small because the GSM has to reduce the robot speed in order to follow the quadratic guideline with the higher curvature closely. Likewise, as shown in Fig. 6(c), with a longer delay $\hat{\tau}\,(i+1)$, the GSM has to apply a small optimal $K\,(i)$ to reduce the robot speed so that the robot will not deviate far from the guideline and will still satisfy $\hat{J}_1\,(i+1) \leq \varepsilon$.

Network Traffic Estimator: Network variables representing network traffic conditions are typically subjective to the algorithm used in the feedback preprocessor and in the gain scheduler. Characterization of network conditions to network variables for use in both parts also depends on typical behaviors and characteristics of the network used. In this paper, we illustrate the GSM concept using delays from an actual IP network. As mentioned in earlier sections, the required network variable is the delay τ, which is estimated from the round-trip time (RTT) delay measurements between the controller and the mobile robot on an IP network.

An important concern is what should be a good representative value of RTT delays to be used as the network variable τ. The feedback processor requires a delay value that is closed to the actually delay as much as possible. If RTT delays on an actual IP network are assumed to have the distribution similarly to the generalized exponential distribution, the median of RTT delays η can be a good representative value [3]. In this case, a majority of RTT delays should not be much different from η, and η could be used in the feedback preprocessor. On the other hand, the gain scheduler requires the value of the delay τ so the networked mobile robot does not violate $\hat{J}\,(i+1) \leq \varepsilon$. We can relax this value by using a slightly larger τ for some purposes such as reducing the effects from robot modeling errors or delay prediction errors in case that an actual RTT delay is larger than η. By assuming the network traffic distribution is the generalized exponential distribution, we proposed to use the mean of RTT delays μ in this case, which is ideally larger than η. However, in actual real-time traffic measurements with a limited number of probing packets, we may have $\eta > \mu$. To handle this case, we propose to use the larger value between η and μ

$$\hat{\tau} = \max\{\eta, \mu\}. \tag{37}$$

Both η and μ in a specific time interval can be computed by sending probing packets as mentioned in [1, Sec. II-B]. Unlike the PI controller in [1], the probing packet transmission in this case is separated from the transmission of the reference speeds.

III. Simulation and Experimental Results

A. Testing Environment and Parameters Used for the Simulation and Experiment of the Mobile Robot Path-Tracking Control Over a Network

1) IP Network Delay: To verify the effectiveness of the GSM concept, the RTT network delays of User Datagram Protocol (UDP)packets between the Advanced Diagnosis And Control (ADAC) Laboratory at North Carolina State University, Raleigh, and Kasetsart University (KU), Bangkok, Thailand (www.ku.ac.th at), are measured for 24 h (00:00–24:00). The reason to use UDP for networked control is to avoid additional delays from retransmission. This data is used in the simulation and experimental setups of the networked mobile robot path-tracking control with the assumption that there is no packet loss. Each value of these RTT delays is divided by 2 and is utilized as τ_{RC} and τ_{CR}.

2) Path for the Robot to Track: The path of the robot used for simulation and actual experimental verification is the same path described in Section II.

3) Controller and Robot Parameters: The controller and robot parameters used for the proposed GSM verification are listed in Table I.

B. Simulation Results

The networked mobile robot path-tracking simulation program is setup in a Matlab/Simulink environment to investigate the effectiveness of the GSM. Three scenarios are compared in the simulation.

1) The robot is controlled without IP network delay.
2) The robot is controlled with IP network delays from ADAC to KU. The GSM is not applied.
3) The robot is controlled with IP network delays from ADAC to KU. The GSM is applied.

The network traffic estimator is set to compute η and μ with the number of packet roundtrips for evaluation $N = 10$. The initial position of the robot is arbitrarily set to $(-0.01, -0.01)$. The robot will stop if

$$\sqrt{\left(x_p\left(s_{e,3}\right) - x\left(i\right)\right)^2 + \left(y_p\left(s_{e,3}\right) - y\left(i\right)\right)^2} \leq 0.05. \tag{38}$$

Fig. 7 shows the results from simulations.

As shown in Fig. 7, the original path-tracking controller performs superbly when there is no IP network delay. The robot track is basically overlaps with the path to be tracked. With IP network delays, the robot without GSM fails to track the path closely because the position feedback and the reference speeds are delayed by IP network delays. On the contrary, the robot with the GSM can track the path much better. The predicted position applied for path-tracking computation and the gain scheduling scheme have compensated the network delay effects on the mobile robot system.

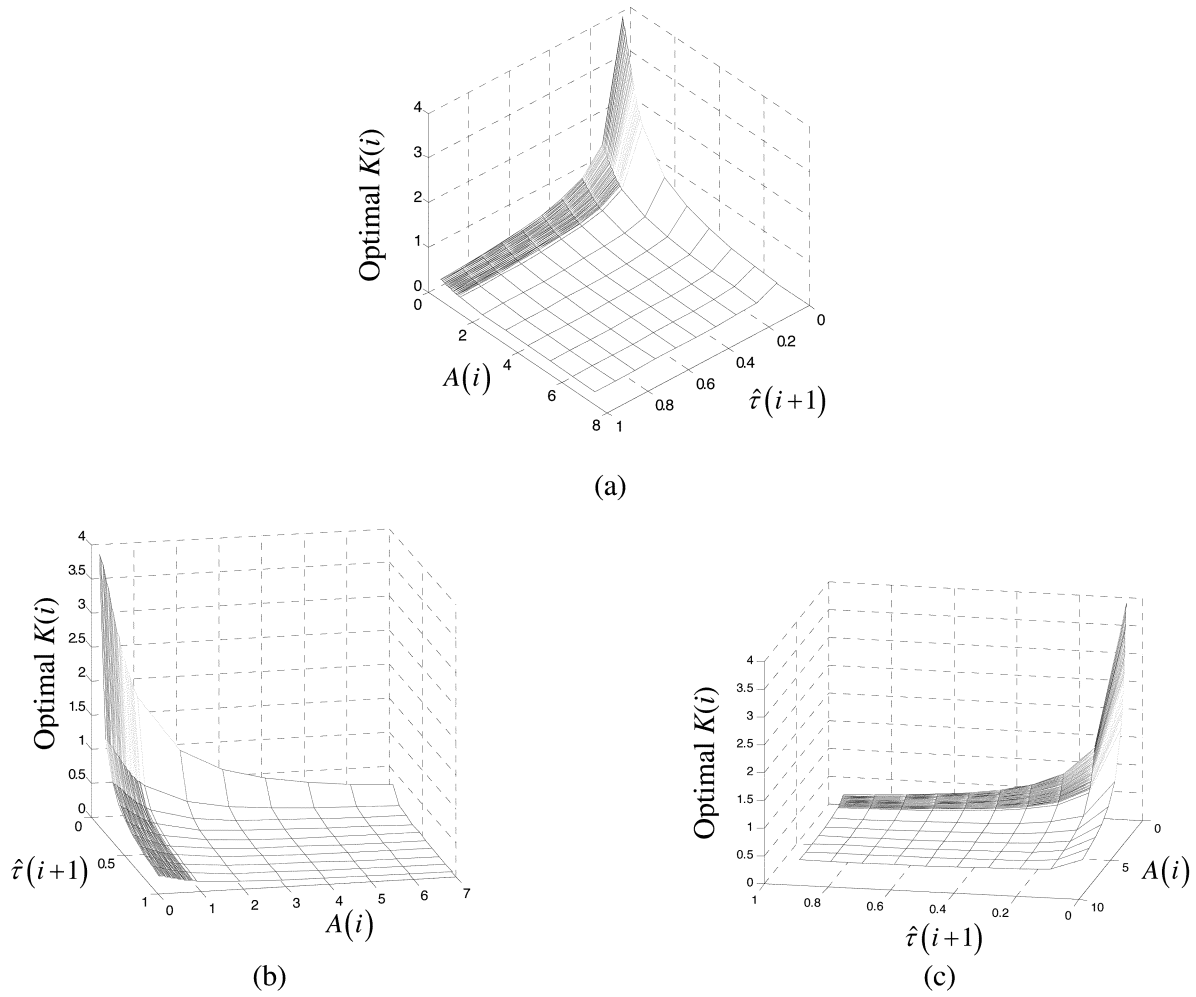


Fig. 6. Optimal $K\,(i)$ surface with respect to $A\,(i)$ and $\hat{\tau}\,(i+1)$ with $\varepsilon = 0.2$. (a) Front view. (b) Side view of $A\,(i)$. (c) Side view of $\hat{\tau}\,(i+1)$.

TABLE I
CONTROLLER AND ROBOT PARAMETERS

Parameter	Description	Value
$s_{\max}$	Maximal projecting distance	0.5
α	Speed factor	0.25
β	Projecting factor	0.5
ε	Cost tolerance	0.2
W	Distance between two wheels	0.4826 m
ρ	Radius of the wheels	0.073 m
T_P	Sampling time of probing packet transmission	0.01 s
T_C	Sampling time of path-tracking controller	0.1 s
T	Sampling time for robot to update reference speeds	0.002 ms

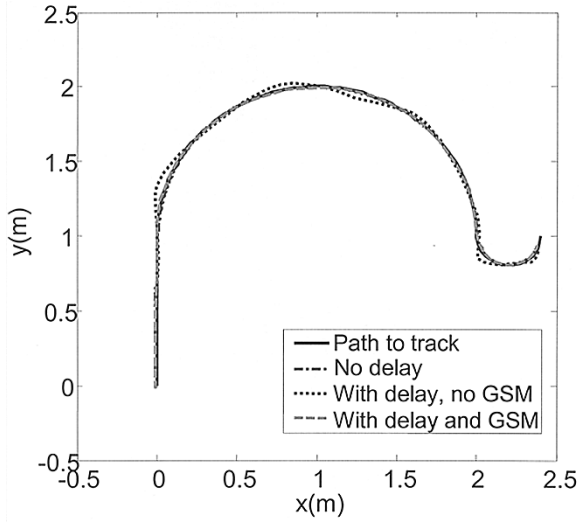


Fig. 7. Comparison of robot tracks from simulations. (a) Dashed–dotted line: robot is controlled without IP network delay. (b) Dotted line: robot is controlled with IP network delays from ADAC to KU. GSM is not applied. (c) Dashed line: robot is controlled with IP network delays from ADAC to KU. GSM is applied.

C. Experimental Results

An experimental mobile robot platform is built to verify the effectiveness of the proposed GSM using the same delay scenario as in the simulations. The block diagram of the robot setup is illustrated in Fig. 8. The mobile robot platform is composed of a Celeron 1.5-GHz notebook computer, a C515C Siemens/Infineon microcontroller board, two LMD18200 motor driver chips, and two dc motors with two 500 pulses/rev optical encoders. The dc motor models used are Maxon A-max 236668. The microcontroller board controls the speeds of the two dc motors by PI control algorithm. It reads the motor rotations from the

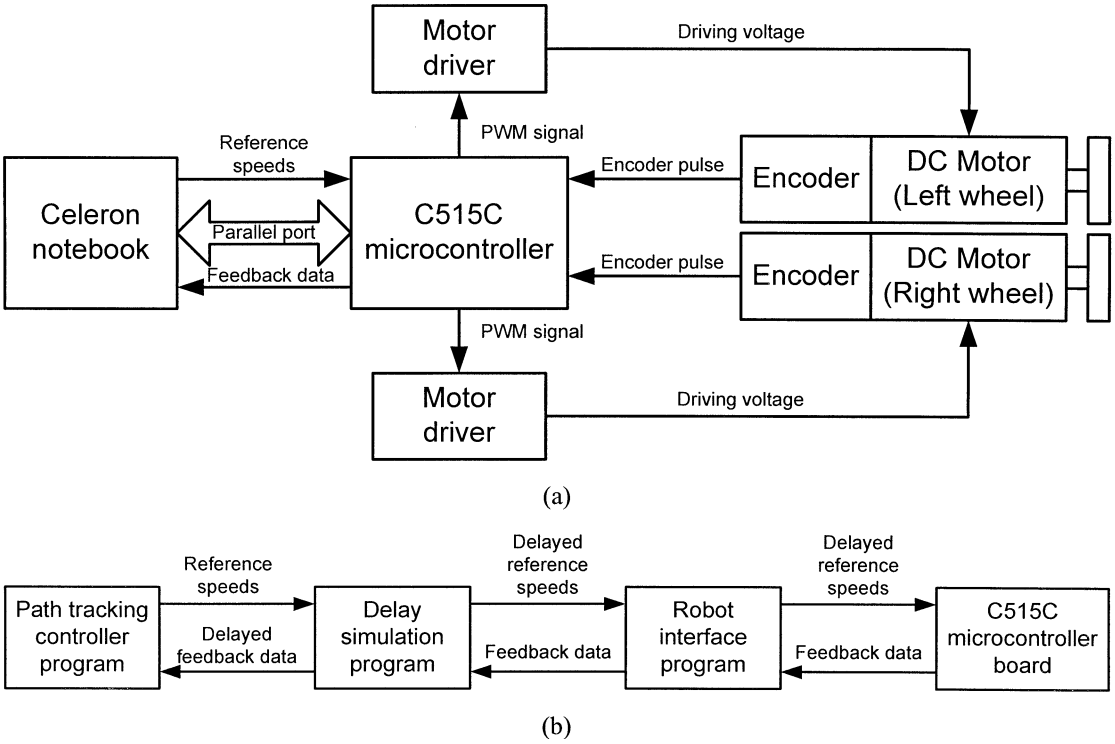


Fig. 8. Mobile robot teleoperation experimental setup. (a) Hardware schematic diagram. (b) Software schematic diagram.

encoders and converts them to the feedback data. The feedback data is then sent to the notebook via a parallel port. The notebook uses this feedback data to perform path tracking by computing the reference speeds of both dc motors, and sending them back to the microcontroller board via the parallel port as well. The microcontroller board then applies these references to perform PI control computation, and send control signals in the forms of TTL-level PWM signals to the motor drivers. The motor drivers will amplify the pulsewidth-modulation (PWM) signals to 0–12-V signals for driving the dc motors.

To focus specially on the effects of network delays, we create an experimental simulation scenario of IP network delays by delaying data transfers between the notebook computer and the microcontroller board using real-time software and a hardware timer. The delay applied in this program is the actual measured IP network delays in Section III-A. The reasons for using the collected IP delay data rather than using the real IP network are as follows.

1) The experimental results can be compared with the simulation results using the same delay data.
2) The experiment is ensured to be repeatable for various future investigations.

This scenario is implemented by using three real-time programs running on RTLinux 3.2:

1) Delay Simulation Program: The delay simulation program delays data transfers between the path-tracking controller program and the robot interface program by using two linked-list structures. The delay can range from 0 s to the maximal delay measurement with the resolution about 0.1 ms.

2) Robot Interface Program: The robot interface program handles parallel port communication between the notebook computer and the microcontroller board. The interface program receives the reference speeds from the path-tracking controller program via the delay simulation program. Likewise, it forward the feedback data from the microcontroller board through the delay simulation program to the path-tracking controller program.

3) Path-Tracking Controller Program: This program contains two parts: path-tracking implementation and the GSM. The GSM is responsible for data communication with the robot interface program through the delay simulation program, for position prediction, and for the gain scheduling.

Fig. 9 shows the experimental results of the IP-based robot path tracking. In addition, Fig. 10 shows the distance from the robot to the path. This distance indicates how close the robot is to the path when the robot is tracking the path.

As shown in Fig. 9, without IP network delay, the original path-tracking controller performs superbly. When there are IP network delays, the robot without the GSM cannot track the path closely because the position feedback and the reference speeds are delayed. The GSM can improve the path-tracking performance by using predicted position and gain scheduling to compensate the delay effects so the robot can track the path more effectively as shown in Fig. 9. Fig. 10 also shows that the robot without the GSM deviates from the path relatively more than the other two cases, and spends a longer time to reach the final destination. On the other hand, the robot with the GSM can track the path much closer to the robot without delay.

IV. Conclusion

This paper has extended the concepts and methods of the GSM described in [1] to enable an existing controller for

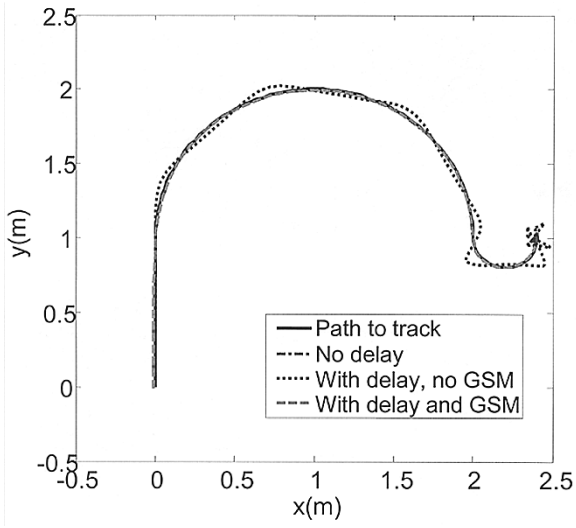


Fig. 9. Comparison of robot tracks from experiments. (a) Dashed–dotted line: robot is controlled without IP network delay. (b) Dotted line: robot is controlled with the existence of IP network delays from ADAC to KU. GSM is not applied. (c) Dashed line: robot is controlled with the existence of IP network delays from ADAC to KU. GSM is applied.

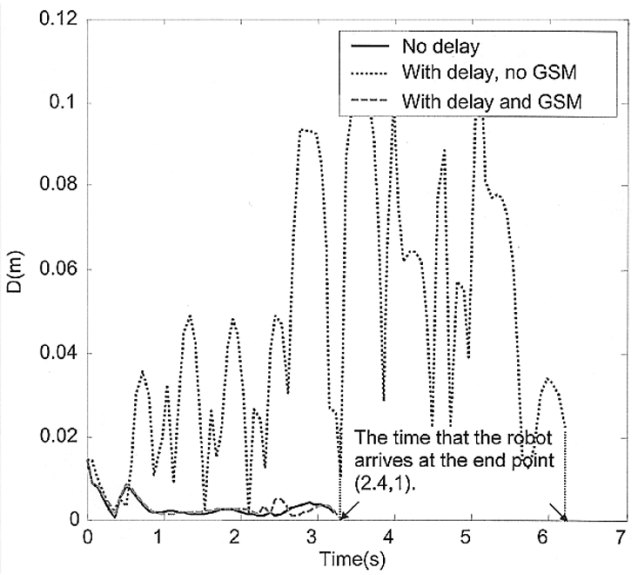


Fig. 10. Closest distances from the robot to the path obtained from experiments. (a) Solid line: robot is controlled without IP network delay; (b) Dotted line: robot is controlled with the existence of IP network delays from ADAC to KU. GSM is not applied; (c) Dashed line: robot is controlled with the existence of IP network delays from ADAC to KU. GSM is applied.

teleoperation. The GSM concept for teleoperation is illustrated on mobile robot path-tracking control with the existence of IP network delays. In this case study, the GSM adjusts the path-tracking controller output with respect to the current network traffic conditions without interrupting the internal design or structure of the existing controller. Thus, the existing path-tracking controller can be still utilized to save cost and time instead of installing the whole new teleoperation system. The GSM can also be implemented along with other middleware features such as in [4]–[9] to assist teleoperation in other ways. The GSM illustrated in this paper may be quite specific to the path-tracking algorithm and the IP network environment. Nevertheless, the concept of GSM and external gain scheduling could be extended to be applied on different teleoperation applications, different path-tracking algorithms, different network environments, or different performance measures by reformulating an external gain scheduling control scheme based on these concerns. In addition, the modular structure of the GSM allows convenient algorithm updates in each part. For example, another prediction algorithm can be updated in the feedback preprocessor separately, or the network traffic condition can be identified by using different models to be used on other types of networks without modifying the other parts.

ACKNOWLEDGMENT

The authors would like to acknowledge Maxon Precision Motor, Inc. for the motor donation.

REFERENCES

[1] Y. Tipsuwan and M.-Y. Chow, "Gain scheduler middleware: A methodology to enable existing controllers for networked control and teleoperation—Part I: Networked control," *IEEE Trans. Ind. Electron.*, vol. 51, pp. 1218–1227, Dec. 2004.

[2] K. Yoshizawa, H. Hashimoto, M. Wada, and S. M. Mori, "Path tracking control of mobile robots using a quadratic curve," in *Proc. IEEE Intelligent Vehicles Symp.*, Tokyo, Japan, 1996, pp. 58–63.

[3] Y. Tipsuwan and M.-Y. Chow, "On the gain scheduling for networked PI controller over IP network," in *Proc. IEEE/ASME AIM'03*, Kobe, Japan, 2003, pp. 640–645.

[4] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro-middleware for mobile robot applications," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 493–497, Aug. 2002.

[5] D. Brugali and M. E. Fayad, "Distributed computing in robotics and automation," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 409–420, Aug. 2002.

[6] D. C. Schmidt, "Middleware techniques and optimizations for real-time, embedded systems," in *Proc. Int. Symp. System Synthesis*, San Jose, CA, 1999, pp. 12–16.

[7] B. Li and K. Nahrstedt, "A control-based middleware framework for quality-of-service adaptations," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1632–1650, Sept. 1999.

[8] S. Song, J. Huang, P. Kappler, R. Freimark, and T. Kozlik, "Fault-tolerant Ethernet middleware for IP-based process control networks," in *Proc. IEEE Conf. Local Computer Networks*, Tampa, FL, 2000, pp. 116–125.

[9] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS negotiation in real-time systems and its application to automated flight control," *IEEE Trans. Comput.*, vol. 49, pp. 1170–1183, Nov. 2000.

Yodyium Tipsuwan (S'99–M'04) was born in Chiangmai, Thailand. He received the B.Eng. degree (with honors) in computer engineering from Kasetsart University, Bangkok, Thailand, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from North Carolina State University, Raleigh, in 1999 and 2003, respectively.

Following receipt of the Ph.D. degree, he joined the Department of Computer Engineering, Kasetsart University, as an Instructor. His main interests are distributed networked control systems, mobile robotics, and computational intelligence. His current research topic is networked control over IP networks.

Dr. Tipsuwan was awarded the Royal Thai Scholarship for his graduate studies.

Mo-Yuen Chow (S'81–M'82–SM'93) received the B.S. degree in electrical and computer engineering from the University of Wisconsin, Madison, in 1982, and the M.Eng. and Ph.D. degrees from Cornell University, Ithaca, NY, in 1983 and 1987, respectively.

Upon completion of the Ph.D. degree, he joined the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, as an Assistant Professor. He became an Associate Professor in 1993 and a Professor in 1999. He was a Senior Research Scientist with the U.S. Army TACOM TARDEC Division during the summer of 2003. He spent his sabbatical leave in 1995 as a Visiting Scientist in the ABB Automated Distribution Division. He has also been a consultant to Duke Power Company, Otis Elevator Company, Taiwan Power Company, J. W. Harley Company and a faculty intern at Duke Power Company. His core technology is diagnosis and control, artificial neural networks, and fuzzy logic. Since 1987, he has been applying his core technology to areas including motor systems, power distribution systems, network-based distributed control systems, and unmanned vehicles. He has served as a Principal Investigator in several projects supported by the National Science Foundation, Center for Advanced Computing and Communication, Nortel Company, Electric Power Research Institute, Duke Power Company, ABB Company, Electric Power Research Center, NASA, and the U.S. Army. He established the Advanced Diagnosis and Control (ADAC) Laboratory at North Carolina State University. He has authored one book, several book chapters, and over 100 journal and conference articles related to his research work.

Prof. Chow is an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS and an AdCom Member of the IEEE Industrial Electronics Society (IES). He served as the IES Vice President of Member Activities during 2000–2001.