Gain Scheduling Middleware for Networked Mobile Robot Control

Yodyium Tipsuwan
yyt@ku.ac.th

Mo-Yuen Chow
chow@eos.ncsu.edu

Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911 USA

***Abstract*—This paper proposes a novel control methodology for remote mobile robot control over a network via middleware. The controller output is adapted via middleware with respect to current network traffic conditions. The middleware can be implemented in a modular structure. Thus, a controller upgrade or modification for other types of network protocols or different control objectives can be achieved easily. A case study on a mobile robot path-tracking with IP network delays is described. The effectiveness of the proposed approach is verified by experimental results.**

***Index Terms*—Internet, networks, adaptive control, distributed control, real time system, mobile robots, telerobotics.**

I. Introduction

RESEARCH in remote mobile robot control and teleoperation was initiated with the concern of safety and convenience in hazardous environments such as spaces and nuclear reactor plants. This research area has recently gained much attention due to the rapid advancements in data and communication network technologies and several benefits for control systems [1]. However, when a mobile robot is controlled over a network, its performance can be degraded by network-induced delays, and the mobile robot system can even become unstable. Several techniques have been developed to handle network delay effects, and some promising results have been reported. These control methodologies are based on different techniques such as buffering [2], nonlinear and perturbation theory [3], and optimal gain scheduling [4]. Some techniques are developed for specific robotic applications. These techniques include robust gain scheduling [5], wave variables [6], and event-based control [7]. Nevertheless, most of these techniques have been developed for a specific network characteristic or a specific protocol. Porting a controller developed by one of these techniques to a different type of network is usually not an easy or convenient task. Some of these techniques are event-driven rather than time-driven so that network delays do not affect the robot stability. Thus, these techniques may not be able to satisfy a time-based optimal requirement such as minimal time control.

This paper proposes a novel control methodology for remote mobile robot control over a network via middleware. A mobile robot path-tracking control with IP network delays is used to illustrate the proposed methodology. The methodology is developed based on an optimal gain scheduling technique to adapt the controller output signals without modifying or interrupting internal controller operations [8]. In the proposed structure, the part to compute control algorithm and the part to handle network connections are separated. Thus, the overall system can be portable to different types of network protocols. The control objective can also be modularly changed to serve different optimal control requirements such as minimal time control.

II. System Description of a Case Study

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 [9], and is described as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2} & \frac{\rho}{2} \\ \frac{\rho}{W} & -\frac{\rho}{W} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}, \quad (1)$$

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

$$\dot{x} = v\cos\theta, \quad (3)$$

$$\dot{y} = v\sin\theta, \quad (4)$$

$$\dot{\theta} = \omega, \quad (5)$$

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 [8]:

B. Path-tracking algorithm

A generalization of the quadratic curve approach proposed in [9] 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 $\left(x_p(s), y_p(s)\right)$, where s is the distance traveled on the path. This algorithm is suitable for real-time usage because of its simple computation with minimal amount of information compared to other approaches and is outlined as:

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

$$\min_s \sqrt{\left(x_p(s) - x(i)\right)^2 + \left(y_p(s) - y(i)\right)^2}, \qquad (6)$$

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. An example of a robot path shown in Fig. 6 is the combination of:

- *Segment 1*: Straight line:

$$x_p(s) = 0,\ y_p(s) = s, \text{ if } s_{e,0} \le s \le s_{e,1}, s_{e,0} = 0, s_{e,1} = 1, \qquad (7)$$

- *Segment 2*: Arc with a radius of 1:

$$x_p(s) = 1 - \cos(s-1),\ y_p(s) = 1 + \sin(s-1),$$
$$\text{if } s_{e,1} < s \le s_{e,2},\ \ s_{e,2} = 1 + \pi, \qquad (8)$$

- *Segment 3*: Arc with a radius of 0.2:

$$x_p(s) = 2.2 - 0.2\cos 5(s - 1 - \pi),$$
$$y_p(s) = 1 - 0.2\sin 5(s - 1 - \pi),$$
$$\text{if } s_{e,2} < s \le s_{e,3},\ \ s_{e,3} = 1 + 1.2\pi, \qquad (9)$$

where $\theta_p(s)$ is the tangent angle at $\left(x_p(s), y_p(s)\right)$, j is the index of the j-th segment of the path, $\kappa_j = d\theta_p(s)/ds$ is the curvature of the j-th segment, $s_{e,j}$ is the endpoint of the j-th segment. The reference $\mathbf{x}_r(i) = \begin{bmatrix} x_r(i) & y_r(i) & \theta_r(i) \end{bmatrix}^{\mathrm{T}}$ is computed from $x_r(i) = x_p\left(s(i)\right)$, $y_r(i) = y_p\left(s(i)\right)$, $\theta_r(i) = \theta_p\left(s(i)\right)$, where $s(i)$ is the reference distance traveled and is determined by the procedures in Fig. 1.

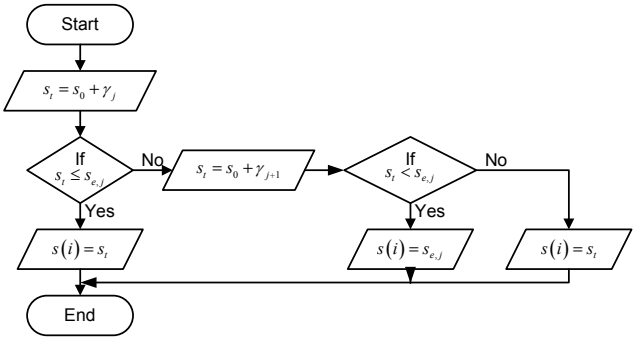

Fig. 1. Procedures for determining the reference distance traveled $s(i)$.

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

$$s(i) = s_t = s_0 + \gamma_j, \gamma_j = \frac{s_{\max}}{1 + \beta\kappa_j}, \qquad (10)$$

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 $\left(x_p(s_0), y_p(s_0)\right)$. 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 j-th segment of the path to track. The reference point will be closer to $\left(x_p(s_0), y_p(s_0)\right)$ if κ_j is high. However, if $s(i) = s_t > s_{e,j}$, $s(i)$ will not be on the j-th 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:

$$s(i) = s_t = s_0 + \gamma_{j+1}. \qquad (11)$$

When $s(i) = s_t < s_{e,i}$, $s(i)$ may be less than $s(i-1)$, which causes the robot to move backtrack if $\left(x_p\left(s(i)\right), y_p\left(s(i)\right)\right)$ is used as the reference position. Therefore, the better choice of $s(i)$ in this case should be $s_{e,i}$ 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:

$$\mathbf{e}(i) = \begin{bmatrix} e_x & e_y & e_\theta \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\mathbf{x}_r(i) - \mathbf{x}(i)\right). \qquad (12)$$

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

$$y_M = A(i)x_M^2, \text{ where } A(i) = \mathrm{sgn}(e_x)\frac{e_y}{e_x^2}. \qquad (13)$$

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:

$$v_r(i) = \mathrm{sgn}(e_x)\sqrt{\dot{x}_M^2\left(1 + 4A^2(i)x_M^2\right)}, \qquad (14)$$

$$\omega_r(i) = \frac{2A(i)\dot{x}_M^3}{v_r^2(i)}. \qquad (15)$$

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

$$x_M = K(i)(t - t_i), \qquad (16)$$

where

$$K(i) = \mathrm{sgn}(e_x)\frac{\alpha}{1 + |A(i)|}, \qquad (17)$$

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:

$$v_r^2(i) = K^2(i)\left(1 + 4A^2(i)K^2(i)(t - t_i)^2\right) \simeq K^2(i). \quad (18)$$

Thus, (14) and (15) can be approximated by:

$$\hat{v}_r(i) \simeq K(i), \quad (19)$$

$$\hat{\omega}_r(i) \simeq 2A(i)K(i). \quad (20)$$

The reference speeds of both wheels are calculated by:

$$\omega_{R,r}(i) = \frac{\hat{v}_r(i)}{\rho} + \frac{W\hat{\omega}_r(i)}{2\rho}, \quad (21)$$

$$\omega_{L,r}(i) = \frac{\hat{v}_r(i)}{\rho} - \frac{W\hat{\omega}_r(i)}{2\rho}. \quad (22)$$

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

III. Path-Tracking Control over a Network

To control a robot to track a predefined path over a network, the path-tracking controller can compute and send the reference speed $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ across the network at every iteration *i* to the robot as shown in Fig. 2.

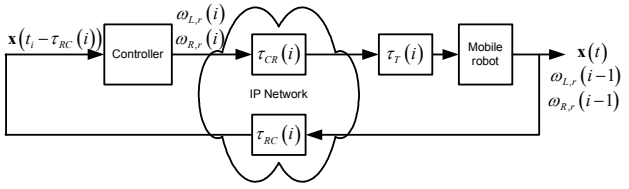

Fig. 2. Data flow of networked mobile robot.

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 relatively insignificant and the computation could be assumed to finish at $t = t_i$. 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 $\tau_T(i)$.

The algorithm described in the previous section is not suitable for direct networked path-tracking control because:

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 (18) may be no longer valid. Thus, the robot may not follow a desired quadratic trajectory.

IV. Gain Scheduler Middleware

The middleware in this paper is defined as the *Gain Scheduler Middleware (GSM)*. We assume that the GSM handles all network connections between the controller and the remote system to be controlled over a network. These include typical network operations such as sending and receiving packets, and other general middleware operations such as negotiation and resource reservation [10]. The basic components of the GSM shown in Fig. 3 are:

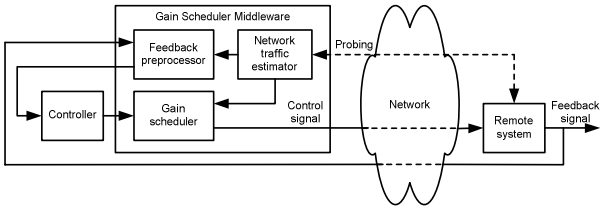

Fig. 3. Structure of gain scheduler middleware (GSM).

A. 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. From this assumption, we can approximate the robot movement during $\left[t_i - \tau_{RC}(i), t_i + \tau_{CR}(i) + \tau_T(i)\right]$ using:

$$\begin{aligned}\Delta_\tau \mathbf{x}(i) &= \mathbf{x}(t_i + \tau_{CR}(i) + \tau_T(i)) - \mathbf{x}(t_i - \tau_{RC}(i)), \\ &= \left[\Delta_\tau x(i) \quad \Delta_\tau y(i) \quad \Delta_\tau \theta(i)\right]^{\mathrm{T}}, \end{aligned} \quad (23)$$

where

$$\Delta_\tau \theta(i) \simeq \hat{\omega}_r(i-1)\tau(i), \tau(i) = \tau_{CR}(i) + \tau_T(i) + \tau_{RC}(i), \quad (24)$$

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

$$\Delta_\tau x(i) \simeq \left(\hat{v}_r(i-1)/\hat{\omega}_r(i-1)\right) \cdot \left[\sin\theta\left(t_i + \tau_{CR}(i) + \tau_T(i)\right) - \sin\theta\left(t_i - \tau_{RC}(i)\right)\right], \quad (25)$$

$$\Delta_\tau y(i) \simeq \left(\hat{v}_r(i-1)/\hat{\omega}_r(i-1)\right) \cdot \left[\cos\theta\left(t_i - \tau_{RC}(i)\right) - \cos\theta\left(t_i + \tau_{CR}(i) + \tau_T(i)\right)\right], \quad (26)$$

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

$$\Delta_\tau x(i) \simeq \hat{v}_r(i-1)\tau(i)\cos\theta\left(t_i - \tau_{RC}(i)\right), \quad (27)$$

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

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

$$\hat{\mathbf{x}}\left(t_i + \tau_{CR}(i) + \tau_T(i)\right) = \mathbf{x}\left(t_i - \tau_{RC}(i)\right) + \Delta_\tau \hat{\mathbf{x}}(i). \quad (29)$$

where $\Delta_\tau \hat{\mathbf{x}}(i)$ is the approximation of $\mathbf{\Delta}_\tau \mathbf{x}(i)$ computed from (23)-(28).

B. 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, 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 define the following cost function:

$$\min \hat{J}_1(i+1) = \left\|\mathbf{\Delta}_\tau \hat{\mathbf{x}}(i+1)\right\|_2, \quad (30)$$

$$= \left\|\hat{\mathbf{x}}\left(t_{i+1} + \tau_{CR}(i+1) + \tau_T(i+1)\right) - \hat{\mathbf{x}}\left(t_{i+1} - \tau_{RC}(i+1)\right)\right\|_2,$$

$$\min \hat{J}_2(i+1) = -\left|K(i)\right|, \quad (31)$$

where $\|\bullet\|_2$ is the Euclidean norm, $\hat{\mathbf{x}}\left(t_{i+1} - \tau_{RC}(i+1)\right) \approx \hat{\mathbf{x}}\left(t_i + \tau_{CR}(i) + \tau_T(i)\right)$, and $\hat{\mathbf{x}}\left(t_{i+1} + \tau_{CR}(i+1) + \tau_T(i+1)\right)$ is the predicted position, which can be determined by using $\hat{\mathbf{x}}\left(t_{i+1} - \tau_{RC}(i+1)\right)$ 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. This cost function implies 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 linear functions of $K(i)$. Minimizing $\hat{J}_2(i+1)$ is equivalent to maximizing $\left|K(i)\right|$. Depending on the actual robot performance requirement (e.g., maximal efficiency control), other cost functions could be also used.

From (24)-(28), $\hat{J}_1(i+1)$ could be also expressed as:

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

$$\hat{J}_1(i+1) = 0, \quad (32)$$

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

$$\hat{J}_1(i+1) = \left|\hat{\omega}_r(i)\hat{\tau}(i+1)\right|, \quad (33)$$

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

$$\hat{J}_1(i+1) = \sqrt{\frac{1-\cos\hat{\omega}_r(i)\hat{\tau}(i+1)}{2A^2(i)} + \left(\hat{\omega}_r(i)\hat{\tau}(i+1)\right)^2}, \quad (34)$$

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

$$\hat{J}_1(i+1) = \left|\hat{v}_r(i)\hat{\tau}(i+1)\right|. \quad (35)$$

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 $\left|K(i)\right|$ 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 (20), these updates are equivalent to adjusting the gain $K(i)$ in (19) and (20). The optimal values of $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$ in (33) and (35) can be determined by solving $\hat{\omega}_r(i)$ and $\hat{v}_r(i)$, respectively, whereas (34) requires a numerical method to solve for $\hat{\omega}_r(i)$ to find the optimal $\omega_{L,r}(i)$ and $\omega_{R,r}(i)$. Since $\hat{\omega}_r(i) \simeq 2A(i)K(i)$, (34) could be viewed as a function of $A(i)$, $K(i)$ and $\hat{\tau}(i+1)$. Because $A(i)$ and $K(i)$ 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. 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 $K(i)$ 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. 4 shows the surfaces of $\hat{J}_1(i+1)$ with respect to $A(i)$, $K(i)$, and $\hat{\tau}(i+1)$ with $\varepsilon = 0.2$. The surface of the optimal $K(i)$ in this case is shown in Fig. 5.

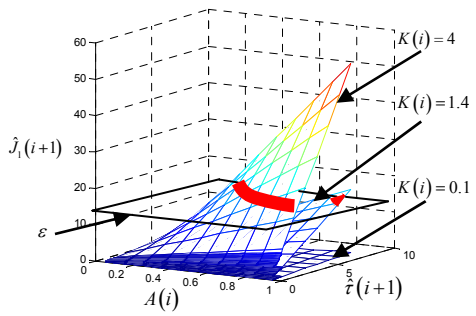

Fig. 4 Cost surfaces of $\hat{J}_1(i+1)$ with respect to $A(i)$, $K(i)$, and $\hat{\tau}(i+1)$ with $\varepsilon = 0.2$.

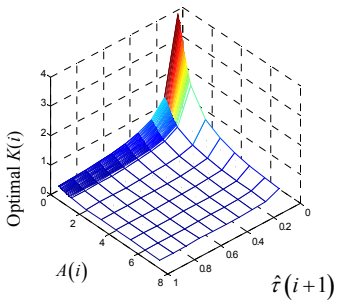

Fig. 5. Optimal $K(i)$ surface with respect to $A(i)$ and $\hat{\tau}(i+1)$ with $\varepsilon = 0.2$.

As shown in Fig. 4, 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. As shown in Fig. 5, 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 the quadratic curve is small and the delay is short. 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, 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 $\hat{J}_1(i+1) \le \varepsilon$.

C. Network traffic estimator

In this paper, we illustrate the GSM concept using delays from an actual IP network and use the network traffic estimator to estimate the delay τ. This delay is estimated from the roundtrip time (RTT) delay measurements between the controller and the mobile robot on an IP network.

Several papers have proposed to approximate the RTT delay on IP networks by a generalized exponential distribution [8, 11]:

$$P[\tau] = \begin{cases} \frac{1}{\phi} e^{-(\tau-\eta)/\phi}, & \tau \ge \eta, \\ 0, & \tau < \eta, \end{cases} \qquad (36)$$

where the expected value of the RTT delay $E[\tau] = \phi + \eta$, and variance $\sigma^2 = \phi^2$. If η is known, ϕ can be easily approximated from η, and an experimental value of $E[\tau]$ or the mean μ by $\phi = E[\tau] - \eta$.

An important concern is what should be a good representative value of RTT delays to be used as τ. The feedback processor requires a delay value that is closed to the actual 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 defined as $\text{Med}(\tau)$ can be a good representative value [8]. In this case, a majority of RTT delays should not be much different from $\text{Med}(\tau)$, and 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 $\hat{J}_1(i+1) \le \varepsilon$. 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 $\text{Med}(\tau)$. 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 $\text{Med}(\tau)$. However, in actual real-time traffic measurements with a limited number of probing packets [8], we may have $\text{Med}(\tau) > \mu$. To handle this case, we propose to use the larger value between $\text{Med}(\tau)$ and μ:

$$\hat{\tau} = \max\{\text{Med}(\tau), \mu\}, \qquad (37)$$

Both $\text{Med}(\tau)$ and μ in a specific time interval can be computed by sending probing packets as mentioned earlier.

V. Experimental Results

A. Testing environment and parameters

To verify the effectiveness of the GSM concept, the RTT network delays of UDP (User Datagram Protocol) packets between ADAC lab at North Carolina State University and KU (Kasetsart University), Thailand, are measured for 24 hours (00:00-24:00) [8]. The reason to use UDP for networked robot control is to avoid additional delays from retransmission. The use of UDP is a common practice for real-time networked control applications. These data are used in the experimental setup 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 two and is utilized as τ_{RC} and τ_{CR} .The path of the robot used for actual experimental verification is the same path described in section II. The controller and robot parameters used for the proposed GSM verification are: $s_{\max}$=0.5, α =0.25, β =0.5, ε =0.2, W=0.48, ρ =0.07, T_P =0.01 s, T_C =0.1 s, and T=0.002 s.

B. Results

An experimental mobile robot platform is built to verify

the effectiveness of the proposed GSM. To focus specially on the effects of network delays, we create an experimental simulation scenario of IP network delays by delaying data transfers between a computer and a 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 V.A. The reasons of using the collected IP delay data rather than using the real IP network is that the experiment is ensured to be repeatable for various future investigations.

The network traffic estimator is set to compute the mean and median of RTT delays for every 10 probing packet roundtrips. The initial position of the robot is arbitrarily set to (-0.01,-0.01). The robot will stop if:

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

Fig. 6 shows the experimental results of the IP-based robot path-tracking. In addition, Fig. 7 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.

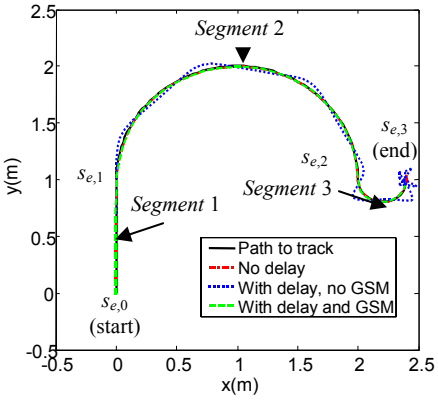

Fig. 6. Robot tracks from experiments; Dashed-dotted line: The robot is controlled without IP network delay; Dotted line: The robot is controlled with IP network delays from ADAC to KU and no GSM; Dashed line: The robot is controlled with IP network delays from ADAC to KU using the GSM.

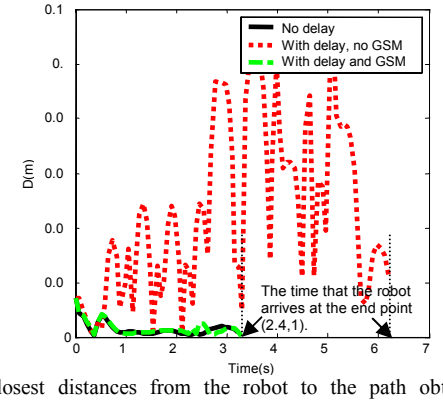

Fig. 7. Closest distances from the robot to the path obtained from experiments; Solid line: The robot is controlled without IP network delay; Dotted line: The robot is controlled with IP network delays from ADAC to KU and no GSM; Dashed line: The robot is controlled with IP network delays from ADAC to KU using the GSM.

As shown in Fig. 6 and Fig. 7, 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. Also, the robot spends a longer time to reach the final destination as shown in Fig. 7. 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 and closer to the robot without delay as shown in Fig. 6 and Fig. 8.

VI. Conclusion

The GSM approach has shown significant improvement on the robot path-tracking performance with the existence of IP network delays. The GSM illustrated in this paper may be quite specific. Nevertheless, the concept of the GSM and external gain scheduling could be extended to be applied on different applications, path-tracking algorithms, or networks by reformulating an external gain scheduling scheme based on these concerns. Because the GSM can be implemented separately from the controller, modification of the GSM using other types of prediction algorithms or a different optimal objective for different applications could be achieved easily.

Acknowledgment

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

References

[1] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control Engineering Practice*, vol. 11, no. 10, pp. 1099-1111, 2003.

[2] R. Luck and A. Ray, "An observer-based compensator for distributed delays," *Automatica*, vol. 26, no. 5, pp. 903-908, 1990.

[3] G. C. Walsh, H. Ye, and L. G. Bushnell, "Stability analysis of networked control systems," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 3, pp. 438-446, 2002.

[4] M.-Y. Chow and Y. Tipsuwan, "Gain adaptation of networked DC motor controllers based on QoS variations," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, 2003.

[5] A. Sano, H. Fujimoto, and M. Tanaka, "Gain-scheduled compensation for time delay of bilateral teleoperation systems," presented at IEEE ICRA'98, Leuven, Belgium, 1998.

[6] 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.

[7] K. Brady and T.-J. Tarn, "Internet-based teleoperation," presented at IEEE ICRA'01, Seoul, South Korea, 2001.

[8] Y. Tipsuwan and M.-Y. Chow, "On the gain scheduling for networked PI controller over IP Network," presented at IEEE/ASME AIM'03, Kobe, Japan, 2003.

[9] K. Yoshizawa, H. Hashimoto, M. Wada, and S. M. Mori, "Path tracking control of mobile robots using a quadratic curve," presented at IEEE Intelligent Vehicles Symposium'96, Tokyo, Japan, 1996.

[10] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, "QoS negotiation in real-time systems and its application to automated flight control," *IEEE Transactions on Computers*, vol. 49, no. 11, pp. 1170-1183, 2000.

[11] J. W. Park and J. M. Lee, "Transmission modeling and simulation for Internet-based control," presented at IEEE IECON '01, Denver, CO, 2001.