Autonomous Robots 10, 135–147, 2001

Internet Control Architecture for Internet-Based Personal Robot

KUK-HYUN HAN, SINN KIM, YONG-JAE KIM AND JONG-HWAN KIM
Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Kusong-dong, Yusong-gu, Taejon-shi, 305-701, Republic of Korea
khhan@vivaldi.kaist.ac.kr
skim@vivaldi.kaist.ac.kr
yjkim@vivaldi.kaist.ac.kr
johkim@vivaldi.kaist.ac.kr

Abstract. This paper proposes a novel direct internet control architecture for internet-based personal robot, which is insensitive to the inherent internet time delay. The personal robot can be controlled using a simulator provided at a local site. Since the internet time delay is affected by the number of nodes and the internet loads, it is variable and unpredictable so that a large internet delay makes some control inputs distorted. The proposed control architecture guarantees that the personal robot can avoid obstacles and reduce the path error and the time difference between a virtual robot at the local site and a real robot at the remote site. This architecture is extended for an uncertain environment. Simulations and experimental results in the real internet environment demonstrate the effectiveness and applicability of the proposed internet control architecture.

Keywords: personal robot, internet control, telerobotics, remote control, internet interface

1. Introduction

The internet-based personal robot considered in this paper is a kind of service robot which can be used for a person's convenience in a house/office environment. It has a wireless LAN system for the internet remote control. Internet users can control it easily using a Web browser or a TCP/IP application program. The internet-based personal robot system may become popular in the near future.

Recently many researches on internet robotics have progressed actively because of the merits of the internet which enables users to access any system on the network cheaply. The robot arm control system (Taylor and Dalton, 1997) through a Web browser was designed, and TeleGarden system (Sutter and Wiegley, 1995) and Mars Pathfinder (Volpe et al., 1996) were developed. The sensor-based mobile robot system (Chen and Luo, 1997) which can be controlled using a Web browser and the internet-based supervisory architecture (Brady and Tarn, 1998) were also reported. The concept of a personal tele-embodiment (Paulos and Canny, 1998) and an intelligent telerobot (Aude et al., 1999) were introduced recently. Most of them have the supervisory control scheme which enables users to issue high level commands. The internet time delay is variable and unpredictable so that the design of a direct control scheme which enables users to control the motion of the robot continuously may be not easy. The direct control scheme (Oboe and Fiorini, 1998) on the internet was proposed, but the modeling of the internet time delay is somewhat unreasonable.

This paper proposes a novel internet control architecture for the internet-based personal robot, which guarantees that the personal robot can avoid obstacles and reduce the path error and the time difference between a virtual robot at the local site and a real robot at the remote site. An internet user can control the real robot using a simulator provided at the local site, and he/she can know a real environment of the remote site since the simulator has a virtual environment. The path error and the time difference of the internet-based personal

robot system are caused by the unpredictable internet time delay and the difference between the real environment of the remote site and the virtual environment of the local site. It is not easy to model the internet time delay so a control architecture which is insensitive to the time delay is needed. Main components of the proposed internet control architecture are a command filter to recover the information loss of control commands, and a path generator and a path-following controller to reduce the time difference between the real robot and the virtual robot. The difference between the real robot and the virtual robot model of the simulator can be overcome by a posture estimator. The problem caused by the difference between the two environments can be solved by applying a virtual environment supervisor to the control architecture.

This paper is organized as follows. In Section 2, the developed internet-based personal robot system, the modeling of a mobile robot, and the characteristics of the internet time delay are described. In Section 3, the step by step design of the proposed internet control architecture which is insensitive to the internet time delay is described. The extended internet control architecture is proposed for an uncertain environment. Section 4 presents the simulation results of the proposed internet control architecture. Real experiments with a micro-robot are provided to show the effectiveness and applicability of the proposed internet control architecture. Concluding remarks follow in Section 5.

2. Internet-based Personal Robot Systems (IPR systems)

2.1. System Description

The internet-based personal robot (IPR), a kind of service robot, can be used for a person's convenient life in a house/office or any indoor environment. It has a personal computer (PC) as a main part, and it can obtain information about environmental changes by using vision cameras, sonar sensors, etc. Actuators enable the robot to move and to carry out physical works. It has a wireless LAN system for the internet remote control. A user can control the IPR using a simulator provided at a local site. It has the intelligence to gather the data from the sensors and to process them to decide its action.

The overall system consists of computers at the local sites, internet, wireless LAN system and the IPR. Users can access the IPR located at the remote site via the internet using a computer at the local site. The wireless LAN system connects the IPR to the internet.

At the local site to control the IPR, a remote control architecture of the IPR system should be designed considering the inherent internet time delay. The basic remote control architecture of the IPR system is described in the following. A user controls a virtual robot in a simulator provided at the local site, and the virtual robot uses a virtual environment for obstacle avoidance. The command signals given by the user are sent to the IPR at the remote site via the internet. The IPR moves like the virtual robot, and avoids obstacles using sensor information. The posture of the virtual robot in the simulator can be updated by the feedback of the IPR posture information through the internet.

It should be noted that an internet interface between a local site and a remote site is needed in the IPR system. Two interfacing methods can be considered. The first one utilizes a Web server and a Web browser for the internet interfacing. Control commands are sent to the IPR via CGI (Common Gateway Interface). The simulator provided at the local site can be implemented by Java. The other one consists of a server program and a client program implemented by C or C++ language. Users utilize the client program to control the IPR. The server program receives control commands from the client program, and controls the IPR directly as per the command.

The developed IPR has a square body of which size is 45 cm × 52 cm × 75 cm as shown in Fig. 1. The weight is about 75 kg. It is a 4-wheeled drive with two fixed wheels and two auxiliary off-centered orientable wheels. It consists of a personal computer (Pentium II 333 Mhz), a wireless LAN (Samsung MagicWave, 2 Mbps), a head with two vision color cameras,

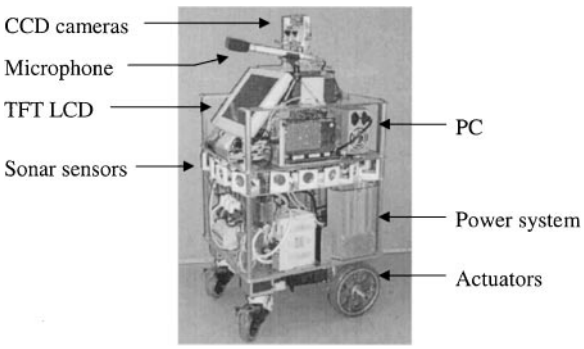

Figure 1. The IPR hardware.

a 12.1 inch TFT monitor, a speaker, a microphone, sonar sensors (16 pairs), a 12 V 100 Ah battery (5 hr 80 Ah), and two AC servo motors (LG Industrial Systems, 200 W). Two cameras mounted on top of can rotate around a vertical and a horizontal axis under the command of the three DC motors. The IPR is connected to the internet through the wireless LAN, and it works as a server. The user can connect to the IPR using a Web browser or a TCP/IP application anywhere and also the user can order motion commands using buttons in the Web browser or the TCP/IP application.

2.2. *Modeling of a Mobile Robot*

The robot model is a prerequisite for the implementation of the simulator. Two wheeled mobile robots with non-slipping and pure rolling are considered. The velocity vector $\mathbf{u} = [v \ \ \omega]^T$ consists of the translational velocity of the center of robot and the rotational velocity with respect to the center of robot. The velocity vector $\mathbf{u}$ and a posture vector $\mathbf{P_c} = [x_c \ \ y_c \ \ \theta_c]^T$ are associated with the robot kinematics as follows:

$$\dot{\mathbf{P}}_c = \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\theta_c} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{J}(\theta)\,\mathbf{u} \quad (1)$$

$$\mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \quad (2)$$

where v_R is the right wheel velocity, v_L is the left wheel velocity, and L is the distance between the two wheels. To derive the robot's position and angle, (1) should satisfy the following nonholonomic constraint:

$$\mathbf{G}\,\dot{\mathbf{P}}_c = [\sin\theta_c \quad -\cos\theta_c \quad 0] \begin{bmatrix} \dot{x_c} \\ \dot{y_c} \\ \dot{\theta_c} \end{bmatrix} = \dot{x}_c \sin\theta_c - \dot{y}_c \cos\theta_c = 0 \quad (3)$$

where $\mathbf{G}$ is a normal vector with respect to the side of wheel. This means that the instant heading direction is the same as the angle of the front side of the robot.

Control command $\mathbf{u}$ is fed to the robot at every sampling time T where $t = kT$, $k = 0, 1, 2, \ldots$. The trajectory of the robot in a sampling time T is shown in Fig. 2, where XY is a global coordinate and $X'Y'$ is a local coordinate rotated by θ_c at time $t = (k-1)T$, and the origin is the posture of robot at time $t = (k-1)T$. The posture at time $t = kT$ can be obtained as follows:

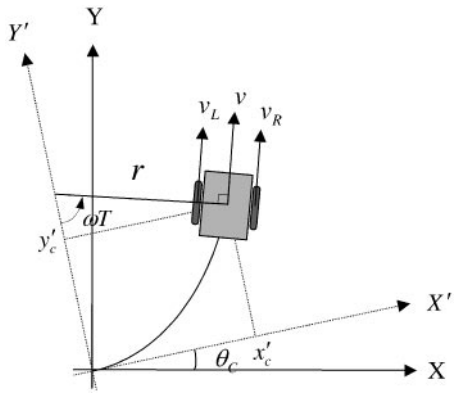

Figure 2. Trajectory of a mobile robot.

$$\begin{aligned} &\mathbf{P}_c(kT) \\ &\quad = \mathbf{P}_c((k-1)T) \\ &\qquad + \begin{bmatrix} \cos\theta_c((k-1)T) & -\sin\theta_c((k-1)T) & 0 \\ \sin\theta_c((k-1)T) & \cos\theta_c((k-1)T) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &\qquad \times \begin{bmatrix} r(kT)\sin\omega(kT)T \\ r(kT)(1-\cos\omega(kT)T) \\ \omega(kT)T \end{bmatrix} \\ &\quad = \mathbf{P}_c((k-1)T) + \mathbf{T}(\theta_c((k-1)T)) \\ &\qquad \times \begin{bmatrix} r(kT)\sin\omega(kT)T \\ r(kT)(1-\cos\omega(kT)T) \\ \omega(kT)T \end{bmatrix} \end{aligned} \quad (4)$$

where $\theta_c((k-1)T)$ is the value of $\theta_c(t)$ at $t = (k-1)T$, $r(kT)$, $v(kT)$ and $\omega(kT)$ are the values of $r(t)$, $v(t)$, and $\omega(t)$ at $t = kT$, respectively, and $r(kT)$ is the radius of rotation, that is, $\frac{v(kT)}{\omega(kT)}$.

If the angular velocity $\omega(kT)$ is 0, the trajectory can be obtained from:

$$\begin{aligned} \mathbf{P}_c(kT) = \mathbf{P}_c((k-1)T) \\ + \mathbf{T}(\theta_c((k-1)T)) \begin{bmatrix} v(kT)T \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (5)$$

For notational simplicity the trajectory equation is rewritten as

$$\mathbf{P}_c(k+1) = \mathbf{P}_c(k) + \mathbf{T}(\theta_c(k))\mathbf{H}(v(k), \omega(k)) \quad (6)$$

where

$$\mathbf{T}(\theta_c(k)) = \begin{bmatrix} \cos\theta_c(k) & -\sin\theta_c(k) & 0 \\ \sin\theta_c(k) & \cos\theta_c(k) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}(v(k), \omega(k)) = \begin{cases} [T\,v(k)\ 0\ 0]^T, & \text{if } \omega(k) = 0 \\ \begin{bmatrix} r(k)\ \sin\omega(k)T \\ r(k)(1-\cos\omega(k)T) \\ \omega(k)T \end{bmatrix}, & \text{otherwise.} \end{cases}$$

2.3. Internet Time Delay

There are many internet nodes between a local site and a remote site. Assume that the physical distance between the two sites is D. In order to examine the distribution of internet time delay, the local site and the remote site were set in the same computer and a reflector was provided at a place of distance $D/2$. The reflector returns the received data immediately. Internet time delay was measured in two cases. One was for a connection in the same domain, and the other was for a connection in the same country. The parameters for the measurements of the internet time delay were as follows: in the same domain, $D = 300$ m, number of nodes $= 11$, bandwidth$(b_0, b_n) = 10$ Mbps, and bandwidth$(b_{\frac{n}{2}-1}, b_{\frac{n}{2}}) = 10$ Mbps, and in the same country, $D = 300$ Km, number of nodes $= 29$, bandwidth$(b_0, b_n) = 10$Mbps, and bandwidth$(b_{\frac{n}{2}-1}, b_{\frac{n}{2}}) =$ 56 Kbps, where n is the number of nodes. Tables 1 and 2 show the regional and weekly variations of the delay measured every one minute for 24 hours each. According to the measurements, the internet time delay increases with distance, but the delay depends also on the number of nodes traversed. Also the delay strongly depends on the internet load so that it cannot be modeled for prediction.

The internet time delay is characterized by the processing speed of nodes, the load of nodes, the connection bandwidth, the amount of data, the transmission speed, etc. Especially the dominating factors are the processing speed and the load of nodes. The internet time delay $T_d(k)$ can be described as follows:

$$\begin{aligned} T_d(k) &= \sum_{i=0}^{n} \left[\frac{l_i}{C} + t_i^R + t_i^L(k) + \frac{M}{b_i} \right] \\ &= \sum_{i=0}^{n} \left(\frac{l_i}{C} + t_i^R + \frac{M}{b_i} \right) + \sum_{i=0}^{n} t_i^L(k) \\ &= d_N + d_L(k) \end{aligned} \tag{7}$$

where l_i is the ith length of link, C the speed of light, t_i^R the routing speed of the ith node, $t_i^L(k)$ the delay caused by the ith node's load, M the amount of data, and b_i the bandwidth of the ith link. d_N is a term which is independent of time, and $d_L(k)$ is a time-dependent term. Because of the term $d_L(k)$ it is impossible to predict the internet time delay at every instant.

Since the internet time delay is affected by the number of nodes and the internet loads, it is variable and unpredictable. Also, a large internet time delay disturbs some control inputs. Figure 3 shows the influence of the internet time delay on the control information. The received data at the remote site was distorted severely, and the information of the sine function $y(t) = 5\sin(0.2\pi t) + 5$, which was used as a test

Table 1. Regional variations of the internet time delay.

	Mean	Standard dev.	Min. value	Max. value
Same domain	20 m sec	51.7	6 m sec	0.89 sec
Same country	4194 m sec	2.25×10^4	39 m sec	343 sec

Table 2. Weekly variations of the internet time delay.

	Mean	Standard dev.	Min. value	Max. value
Mon.	531 m sec	1.42×10^3	48 m sec	15 sec
Tue.	6880 m sec	2.89×10^4	45 m sec	343 sec
Wed.	1480 m sec	7.77×10^3	44 m sec	154 sec
Thu.	1030 m sec	4.16×10^3	45 m sec	68 sec
Fri.	779 m sec	2.59×10^3	49 m sec	35 sec

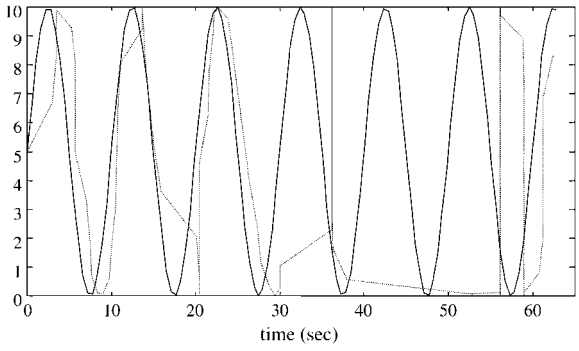

Figure 3. Influence of the internet time delay.

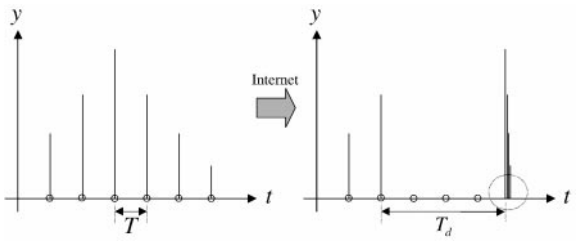

Figure 4. Information loss of command signals.

function, was almost lost. Figure 4 shows the information loss of command signals when the time delay is T_d, where a command is given to the IPR via internet every sampling time T. N control commands input to the remote site at the same time after the delay T_d so that their informations is lost, where $N = integer(\frac{T_d}{T})$. Thus, a novel internet control architecture is needed to control the IPR, which is insensitive to the internet time delay.

3. Internet Control Architecture Design

A user can control the IPR at the remote site through the internet using a simulator provided at the local site. The user regards the status of the virtual IPR at the local site as that of the real IPR at the remote site. Since the user cannot recognize the environment of the remote site, he/she expects that the real IPR moves as the virtual IPR does. However, because of the delay we have to compensate for the path error and the time difference between the real IPR and the virtual IPR, which increase as time goes on. Moreover, it is not possible to recognize the current status of the real robot, so the possibility of colliding with the obstacles is very high. If the collision happens, the robot can be damaged and persons near the robot can be hurt. Thus, the IPR should have the function of obstacle avoidance (Borenstein and Koren, 1991; Fujimori et al., 1997).

In this section, a novel internet control architecture is designed step by step to minimize the effect of the internet time delay. The proposed architecture is completed by three design steps, and its effectiveness is verified through the simulations and experiments.

3.1. Internet Control Architecture

The internet control architecture-I consists of a user interface, simulator, virtual environment and posture estimator, which can be devised from the basic concept as shown in Fig. 5. In the figure, $\mathbf{u}_r(i)$ is the ith control command $[v_r(i)\ \omega_r(i)]^T$ from a user, $\mathbf{u}_r^d(i)$ the ith control command passed through the internet, $\mathbf{P}_c(i)$ the ith robot posture, $\mathbf{P}_c^d(i)$ the ith robot posture passed through the internet, $\hat{\mathbf{P}}_c(i)$ the ith estimated posture, and $\mathbf{P}_c^s(i)$ the ith posture of the virtual robot. In order to correct the posture error between the virtual robot and the real robot, the real robot generates feedback signals such as posture information of the real robot, to the simulator. The architecture-I can be considered as a basic structure.

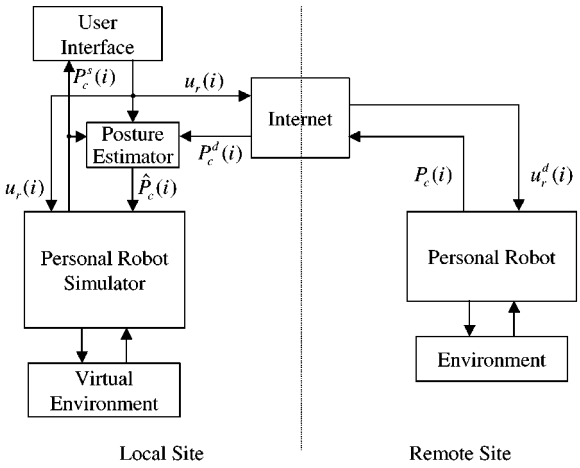

Figure 5. Internet control architecture-I (Step 1).

User Interface which can be implemented by Java, C++, etc., enables a user to control a remote IPR. *Posture estimator* estimates the current posture of the virtual IPR based on the feedback information of the real IPR. *Personal robot simulator* is the same as the virtual mobile robot at the local site. *Virtual environment* has the information of the real environment so that it enables the virtual robot to avoid obstacles. *Personal robot* is the same as the real mobile robot at the remote site. *Environment* is a circumstance where a real IPR is working.

The *posture estimator* corrects the error between the real robot and the virtual robot. The error can be increased by the difference between the two robots, the internet time delay, the difference between the two environments, etc. The *posture estimator* can be described in the following. To estimate the IPR posture, (6) and the following observer can be used:

$$\hat{\mathbf{P}}_c(i+1) = \hat{\mathbf{P}}_c(i) + \mathbf{T}(\theta_c^d(i))\mathbf{H}(v_r(i), \omega_r(i)) + \mathbf{K}_e(\mathbf{P}_c^d(i) - \hat{\mathbf{P}}_c(i)) \quad (8)$$

The current posture of the IPR can be estimated as follows:
If $i = j + 1$:

$$\hat{\mathbf{P}}_c(i) = (\mathbf{I} - \mathbf{K}_e)\hat{\mathbf{P}}_c(j) + \mathbf{T}(\theta_c^d(j))\mathbf{H}(v_r(j), \omega_r(j)) + \mathbf{K}_e\mathbf{P}_c^d(j) \tag{9}$$

If $i > j + 1$:

$$\hat{\mathbf{P}}_c(i) = (\mathbf{I} - \mathbf{K}_e)\hat{\mathbf{P}}_c(j) + \mathbf{T}(\theta_c^d(j))\mathbf{H}(v_r(j), \omega_r(j)) + \mathbf{K}_e\mathbf{P}_c^d(j) + \sum_{k=j+1}^{i-1} \mathbf{T}(\hat{\theta}_c(k))\mathbf{H}(v_r(k), \omega_r(k)) \tag{10}$$

The internet control architecture-I has a weak point that the information loss of control commands increases when the internet time delay occurs as shown in Fig. 4. The *posture estimator* can recover the information loss eventually, but the time required for the recovery becomes too long. The architecture which can get rid of the cause of the information loss is needed. Figure 6 shows the internet control architecture-II, where a *command filter* is introduced. The *command filter* can recover the information loss of control commands caused by the internet time delay. It means that the *command filter* reduces the path error between the real robot and the virtual robot. The function of the *command filter* is shown in Fig. 7. Command signals received at the same time after the internet time delay T_d are regenerated with the sampling time T in the *command filter*. The *command filter* consists of two

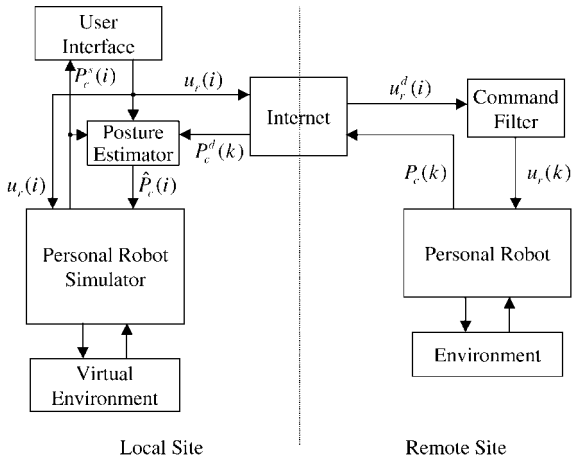

Figure 6. Internet control architecture-II (Step 2).

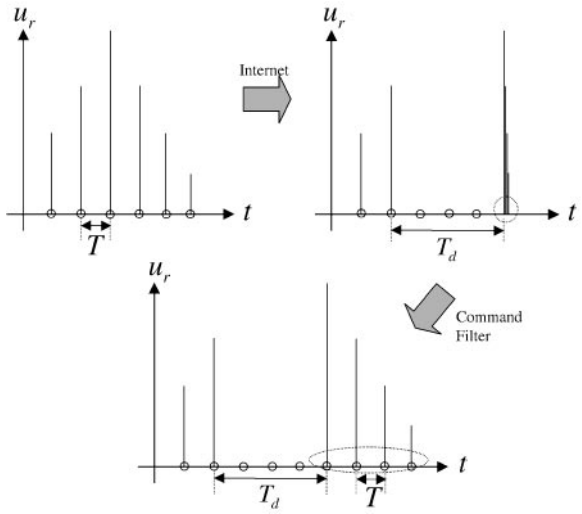

Figure 7. Function of *command filter*.

modules such as a *command queue* and a *command generator*. The *command filter* and the two modules can be defined by DEVS (Discrete Event Systems Specifications) formalism as in (11), (12) and (13), respectively. The *command filter* receives a control command, and stores it in the *command queue*. The *command generator* pulls out the command from the *command queue*, and outputs it each sampling time T. The input and output can be rewritten as $U_{in} = u_r^d(i)$, $U_{out} = u_r(k)$.

$$\begin{aligned} Filter &= < X, Y, M, EIC, EOC, IC, Select > \\ X &= \{U_{in}\} \\ Y &= \{U_{out}\} \\ M &= \{Queue, Generator\} \\ EIC &= \{(Filter.U_{in}, Queue.U_{in})\} \\ EOC &= \{(Generator.U_{out},\ Filter.U_{out})\} \\ IC &= \{(Queue.U_{out},\ Generator.U_{in}), \\ &\quad (Generator.U_{out},\ Queue.U_{in})\} \\ Select &: Select(\{Queue, Generator\}) = Queue \end{aligned} \tag{11}$$

$$\begin{aligned} Queue &= < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta > \\ X &= \{U_{in}, done\} \\ Y &= \{U_{out}\} \\ S &= \{(n, status) \mid n \in \{0, 1, 2, \cdots\}, \\ &\quad status \in \{Busy, Free\}\} \\ \delta_{ext} &: \delta_{ext}((n, \_), U_{in}) = (n + 1, \_) \\ &\quad \delta_{ext}((\_, Busy), done) = (\_, Free) \\ \delta_{int} &: \delta_{int}((n \neq 0, Free)) = (n - 1, Busy) \\ \lambda &: \lambda((n \neq 0, Free)) = U_{out} \\ ta &: ta((n \neq 0, Free)) = 0,\ otherwise\ ta = \infty \end{aligned} \tag{12}$$

$$
\begin{aligned}
Generator &= < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta > \\
X &= \{U_{in}\} \\
Y &= \{U_{out}\} \\
S &= \{status | status \in \{Busy, Free\}\} \\
\delta_{ext} &: \delta_{ext}(Free, U_{in}) = Busy \\
\delta_{int} &: \delta_{int}(Busy) = Free \\
\lambda &: \lambda(Busy) = U_{out} \\
ta &: ta(Busy) = T \ (sampling\ time), \\
& \quad ta(Free) = \infty
\end{aligned}
\tag{13}
$$

The internet control architecture-II can recover the information loss of control commands though the internet time delay exists, but it still has the serious problem that the time difference between the real robot and the virtual robot increases, as the internet time delay T_d is accumulated in the *command filter*. In order to solve this problem, the internet control architecture-III is finally designed.

The internet control architecture-III guarantees that the path error and the time difference between the real IPR at the remote site and the virtual IPR at the local site can be reduced. The proposed architecture includes *path generator* and *path-following controller*. The *path generator* restores the moving path of the virtual robot. The *path-following controller* guarantees that the real robot follows the generated path. The time difference between the real robot and the virtual robot can be reduced by the *path generator* and the *path-following controller*. As the control input of the real robot is separated from the control command passed through the internet by the two modules, the *command generator* in the *command filter* can be modified by replacing sampling time T of the time advance function *ta* with the processing time T_p which is shorter than T. The processing time is the computing interval for generating a path segment for one control command. The function of the modified *command filter* is shown in Fig. 8. The modified time advance function reduces the time

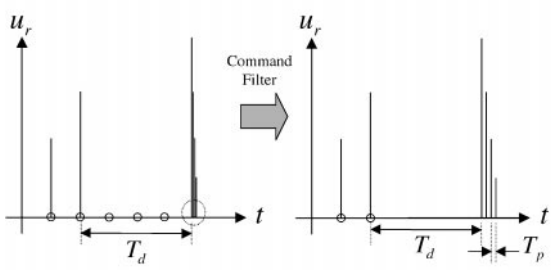

Figure 8. Function of the modified *command filter*.

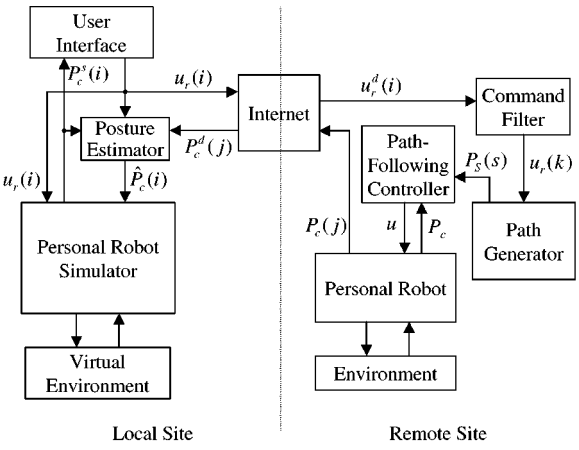

Figure 9. Internet control architecture-III (Step 3).

difference by Δt:

$$\Delta t = NT - NT_p = N(T - T_p)$$

where N is the number of commands received at the same time after time delay.

Figure 9 shows the internet control architecture-III, where $\mathbf{P}_S(s)$ is the moving path of the virtual robot, $\mathbf{u}$ is the control input of the *path-following controller*, $\mathbf{P}_c$ is the current posture of the real robot, and $\mathbf{P}_c(j)$ is the jth robot posture to be fed back to the simulator.

In this paper, the *path-following controller* is implemented with the uni-vector field navigation method (Kim et al., 1998; Kim et al., 1998). The uni-vector field makes the mobile robot converge to a desired path. The uni-vector field is generated from the following equation:

$$\theta_N(l) = \theta_d - sgn(l)\tan^{-1}(g|l|^{1/c}),\ g > 0,\ c > 1 \tag{14}$$

where l is the shortest distance between the robot and the desired path, the sign of l is positive at the left side of the moving direction, and vice versa, $\theta_N(l)$ is the angle of uni-vector field at the robot position and θ_d is the angle of the desired path.

The control law which guarantees convergence to the vector field is obtained as

$$
\begin{aligned}
\omega &= K_\omega sgn(\theta_e)\sqrt{|\theta_e|} + \left(\cos\theta_c \frac{\partial \theta_N}{\partial x} + \sin\theta_c \frac{\partial \theta_N}{\partial y}\right) v \\
v &= K_v \Delta s
\end{aligned}
\tag{15}
$$

where θ_e is the difference between the angle of the vector θ_N and the angle of the robot θ_c, that is, $\theta_e = \theta_N - \theta_c$, and Δs is the distance left.

3.2. *Extended Internet Control Architecture for an Uncertain Environment*

The internet control architecture which is insensitive to internet time delay has been designed under the assumption that the *virtual environment* is equivalent to the *environment*. But the real environment changes frequently. If the assumption is not satisfied, the architecture cannot guarantee stability of the IPR system. The extended internet control architecture is designed to overcome this problem considering the real environment. The *virtual environment supervisor* added in the new architecture makes the decision whether or not the sensor information is about obstacles, distinguishes new obstacles from the environment already known, and adds the new information to the *virtual environment*. As the obstacle informations updated newly have a high probability of change, it can be changed again in a short time period. It can be overcome by assigning a lifetime to the cells of the obstacle in the *virtual environment*. The *virtual environment* consists of many cells which are of the same size. The value of cell C_{mn} should be zero or one, where C_{mn} is the (mth, nth) cell of the *virtual environment*. In the simulation, $m = 125$, $n = 61$ were used. The value "1" means that there is an obstacle in the cell. If a cell is formed at time t_0, the value of the cell C_{mn} can be obtained from

$$C_{mn}(t - t_0) = sgn(max(L_{mn}(t - t_0), L_{TH}) - L_{TH}) \tag{16}$$

where L_{mn} is the lifetime of the cell C_{mn}, L_{TH} is a threshold lifetime, and $sgn()$ is a sign function with $sgn(0) = 0$. The cell's lifetime can be designed as

$$\begin{aligned} L_{mn}(t - t_0) &= e^{-\frac{t-t_0}{\gamma C^S_{mn}}} \\ C^S_{mn} &= \sum_{p=m-1}^{m+1} \sum_{q=n-1}^{n+1} C_{pq} - C_{mn}. \end{aligned} \tag{17}$$

Figure 10 shows the extended internet control architecture where $\mathbf{d}(j)$ is the sensors information at time step

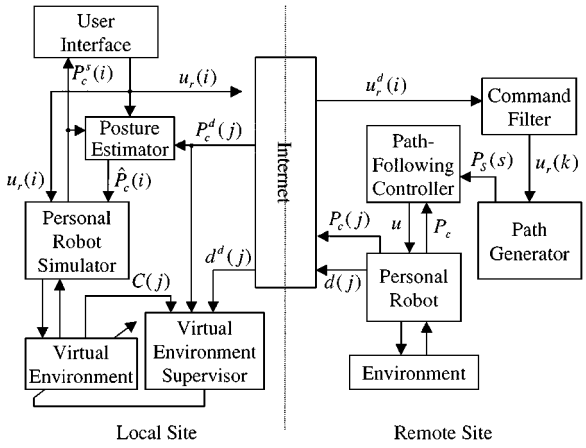

Figure 10. Extended internet control architecture.

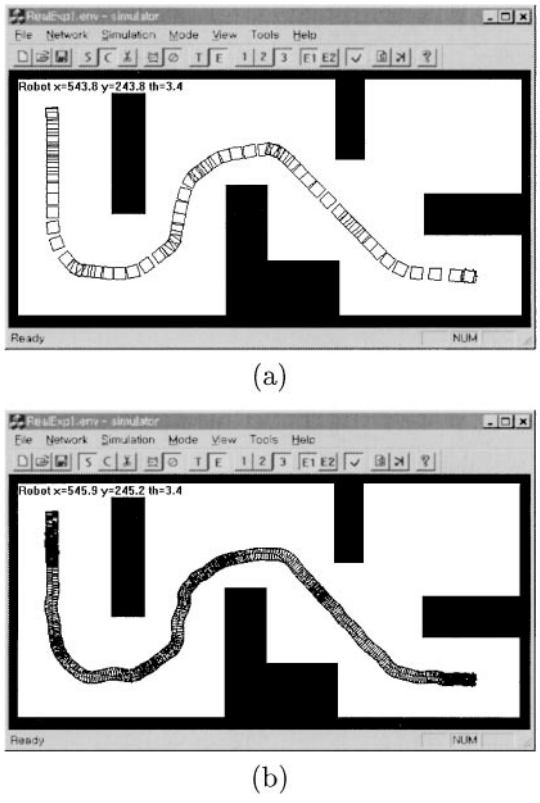

Figure 11. Simulation results by the internet control architecture-III. (a) Virtual robot path at the local site. (b) Real robot path at the remote site.

j, that is $[d_0(j)\ d_1(j) \cdots d_{N_s}(j)]^T$, and $\mathbf{C}(j)$ includes the information of mn cells of the *virtual environment* at time step j, that is, $\mathbf{C}(j)$ is a matrix consisting of the elements C_{mn}.

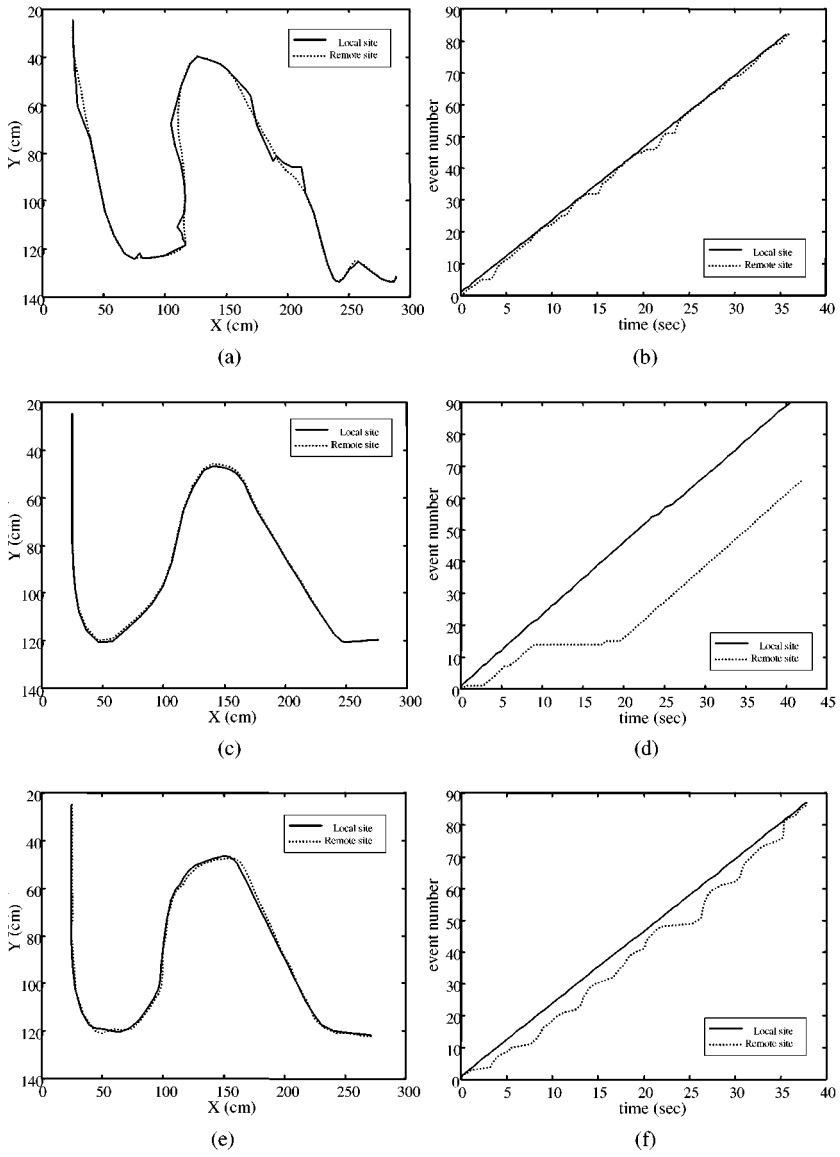

Figure 12. Simulation results. (a) Path error (internet control architecture-I). (b) Time difference (internet control architecture-I). (c) Path error (internet control architecture-II). (d) Time difference (internet control architecture-II). (e) Path error (internet control architecture-III). (f) Time difference (internet control architecture-III).

4. Simulations and Experiments

4.1. Simulation Results

Simulations were performed in the real internet environment. In the simulation, the proposed internet control architecture was implemented as a TCP/IP application version. The physical distance between the local site and the remote site was about 300 km in Korea. The virtual robot path controlled by a user at the local site and the real robot path controlled by the internet control architecture-III at the remote site are shown in Fig. 11. Figure 12 shows the path error and the time difference between the two robots.

It should be noted that the architecture-I had cumulative errors. Because of this, it might be inconvenient for the user to control the robot in the simulation environment by the architecture-I. By the architecture-II

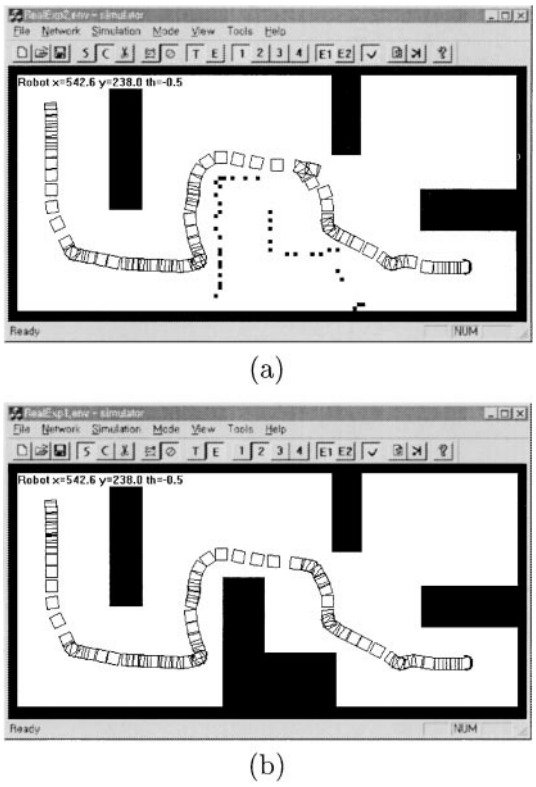

Figure 13. Simulation results on changing environment. (a) Updated virtual environment. (b) Real robot path at the remote site.

the path error could be reduced, but the time difference increased continuously. However, by the architecture-III, the time difference as well as the path error was very small, although the internet time delay was quite variable.

Virtual environment at the local site is different from *Environment* at the remote site as shown in Fig. 13. As time goes on, from the sensor information the *virtual environment* generates obstacles at the center like the *environment* at the remote site. The results demonstrate that the extended internet control architecture can overcome the problem caused by changing environment.

4.2. Experimental Results

Experiment was performed with a small mobile robot whose size is 7.5 cm × 7.5 cm × 7.5 cm with an overhead CCD camera for global positioning. The robot has two driving wheels and two passive centered orientable wheels, and its weight is about 430 g. The CPU of the robot is an AT89C52 microprocessor running at 24 MHz, and DC motors with LM629 motion controllers are mounted on the driving wheels. The LM629 chip is used as a velocity controller which receives only a reference velocity as the input signal and implicitly produces torque according to the reference velocity via the L298 driver chip.

Figure 14 shows the perspective view of the experiment setting with the vision camera, the micro-robot,

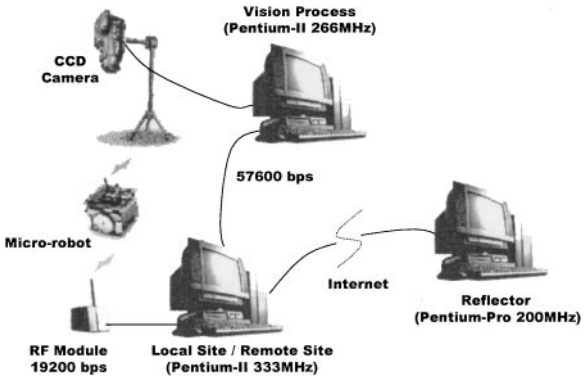

Figure 14. Overview of the experiment settings.

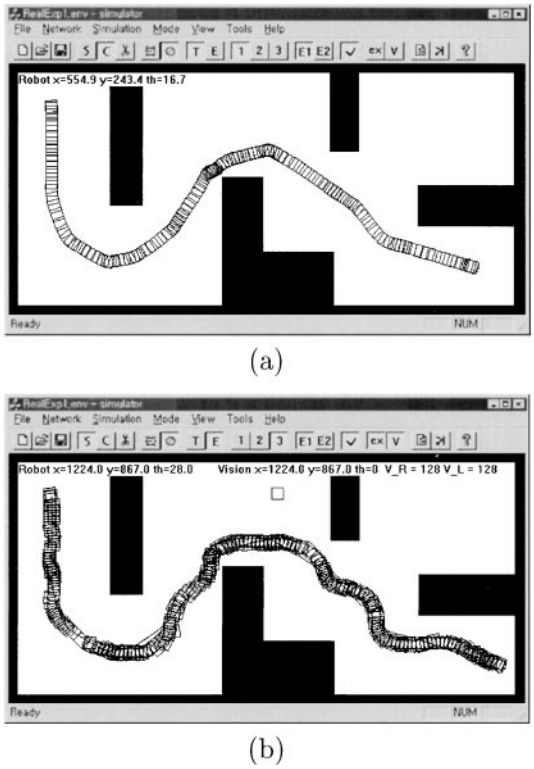

Figure 15. Experimental results by the internet control architecture-III. (a) Virtual robot path at the local site. (b) Real robot path at the remote site.

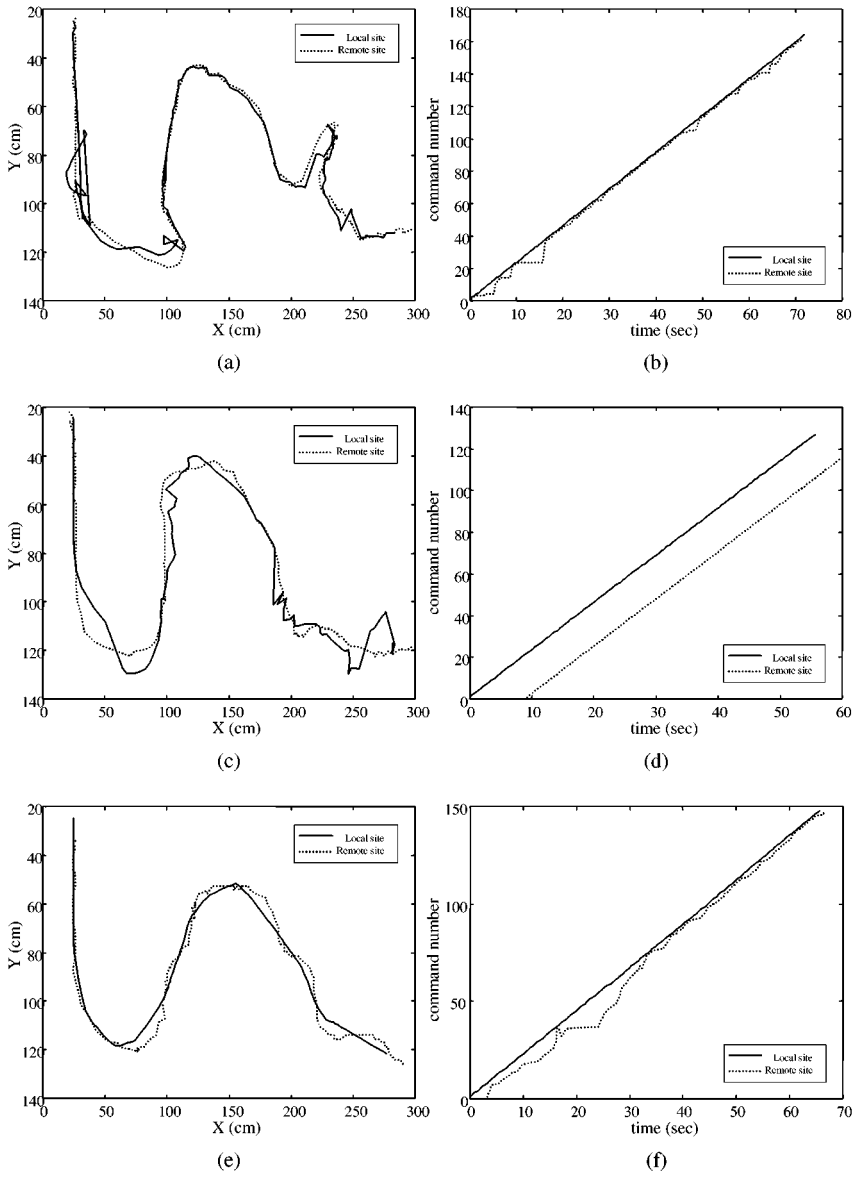

Figure 16. Experimental results. (a) Path error (internet control architecture-I). (b) Time difference (internet control architecture-I). (c) Path error (internet control architecture-II). (d) Time difference (internet control architecture-II). (e) Path error (internet control architecture-III). (f) Time difference (internet control architecture-III).

and three computers for vision processing, local/remote site implementation, and the reflector, respectively.

In this experiment, the proposed internet control architecture was implemented as a TCP/IP application version, and the physical distance between the local/remote site and the reflector was about 150 km, which was the same condition as that of computer simulations. The virtual robot path and the real robot path of the internet control architecture-III are shown in Fig. 15. Figure 16 shows the path error and the time difference between the two robots. In the experimental results of the internet control architecture-I, the path error was caused by the information loss of control commands. The information loss of control commands made the real robot path different from the virtual robot path, and the feedback of the real robot posture made the movement of the virtual robot discontinuous. This made it difficult for the user to control the virtual robot

at the local site. In the results obtained by the internet control architecture-II, the path error was quite large, even though it was small in the simulation. This was caused by the difference between the two models of the virtual robot and the real robot. For instance, assumptions of non-slipping and pure rolling made the two models different. Since the error caused by the difference of models accumulated in a sampling time with an internet time delay, the movement of the virtual robot became discontinuous after the feedback of the real robot posture. In the results by the internet control architecture-III, the path error and the time difference between the two robots was quite small. The experimental results demonstrated the effectiveness and the applicability of the proposed internet control architecture-III as the simulation results did.

5. Conclusions

A novel internet control architecture for the internet-based personal robot (IPR) was proposed. The extended architecture considering an uncertain environment was designed, and the IPR was developed. The proposed architecture was insensitive to the internet time delay and guaranteed that the path error and the time difference between a real IPR and a virtual IPR could be reduced. The path error could be reduced by the *command filter*, and the time difference could be decreased by the *path generator* and the *path-following controller*. The problem caused by changing environment could be overcome with the *virtual environment supervisor*. Simulations and experimental results in a real internet environment demonstrated the effectiveness and the applicability of the proposed internet control architecture.

As a further work, the proposed internet control architecture will be implemented into the developed personal robot for practical application. For this purpose, a global positioning system for the personal robot is under development. However, it should be mentioned that the basic functions for tele-presence and voice communication with the developed personal robot have been implemented.

Acknowledgments

The authors would like to thank the support given by MIC (Ministry of Information and Communication), Korea, to the development of the internet-based personal robot.

References

Aude, E.P.L., Caneiro, G.H.M.B., Serdeira, H., Silveira, J.T.C., Martins, M.F., and Lopes, E.P. 1999. CONTROLAB MUFA: A multi-level fusion architecture for intelligent navigation of a telerobot. In *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 465–472.

Borenstein, J. and Koren, Y. 1991. The vector field histogram — fast obstacle avoidance for mobile robots. *IEEE J. Robot. Automat.*, 7(3):278–288.

Brady, K. and Tarn, T.J. 1998. Internet-based remote teleoperation. In *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 65–70.

Chen, T.M. and Luo, R.C. 1997. Remote supervisory control of an autonomous mobile robot via World Wide Web. In *Proc. IEEE Int. Symposium on Industrial Electronics*, vol. 1, pp. ss60–ss64.

Fujimori, A., Nikiforuk, P.N., and Gupta, M.M. 1997. Adaptive navigation of mobile robots with obstacle avoidance. *IEEE Trans. Robot. Automat.*, 13(4):596–602.

Kim, Y.-J., Kim, D.-H., and Kim, J.-H. 1998. Evolutionary programming-based vector field method for fast mobile robot navigation. In *Proc. Second Asia-Pacific Conf. on Simulated Evolution And Learning*, vol. 1.

Kim, J.-H., Kim, K.-C., Kim, D.-H., Kim, Y.-J., and Vadakkepat, P. 1998. Path planning and role selection mechanism for soccer robots. In *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 3216–3221.

Oboe, R. and Fiorini, P. 1998. A design and control environment for internet-based telerobotics. *Int. Journal of Robotics Research*, 17(4):433–449.

Paulos, E. and Canny, J. 1998. Designing personal tele-embodyment. In *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 3173–3178.

Sutter, C. and Wiegley, J. 1995. Desktop teleoperation via the World Wide Web. In *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 654–659.

Taylor, K. and Dalton, B. 1997. Issues in internet telerobotics. In *Int. Conf. on Field and Service Robotics*.

Volpe, R., Balaram, J., Ohm, T., and Ivlev, R. 1996. The Rocky 7 Mars Rover Prototype. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1558–1564.

Kuk-Hyun Han received his B.S. and M.S. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1997 and 1999, respectively. He is currently working toward the Ph.D. degree in Electrical Engineering at this institute. His research interests are in the areas of internet-based personal robot systems, evolutionary computation and intelligent control. He is a member of IEEE. Mr. Han is the recipient of the Presidential Award at the 38th Nationwide Science Exhibition in 1992, the Certificate of Appreciation at the 1996 Micro-Robot World Cup Soccer Tournament (MiroSot'96), and the 2nd Award at the S-MiroSot'97.

Sinn Kim received his B.S. and M.S. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology, Taejon, Korea, in 1994 and 1996, respectively. He is currently working toward the Ph.D. degree in Electrical Engineering at this institute. His research interests are in the areas of localization of a mobile vehicle, robust observer and intelligent control. Mr. Kim is the recipient of the Best Post Award at the 1995 IEEE International Conference on Evolutionary Computation, held at Nagoya, Japan.

Yong-Jae Kim received his B.S. and M.S. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology, Taejon, Korea, in 1996 and 1998, respectively. He is currently working toward the Ph.D. degree in Electrical Engineering at this institute. His research interests include motion planning of mobile systems, and machine intelligence.

Jong-Hwan Kim received his B.S., M.S., and Ph.D. degrees in Electronics Engineering from Seoul National University, Korea, in 1981, 1983, and 1987, respectively. Since 1988, he has been with the Department of Electrical Engineering at the Korea Advanced Institute of Science and Technology, where he is currently a Professor. He was a Visiting Scholar at Purdue University from September 1992 to August 1993. His research interests are in the areas of Evolutionary Multi-agent Robotic Systems. He is the Associate Editor of the IEEE Transactions on Evolutionary Computation, and of the International Journal of Intelligent and Fuzzy Systems. He is one of the co-founders of Asia-Pacific Conference on Simulated Evolution and Learning. He is the General Chair of the Congress on Evolutionary Computation'2001. His name is included in the Barons 500 Leaders for the New Century as the Founder of FIRA (Federation of International Robot-soccer Association) and of IROC for Robot Olympiad. He is now serving FIRA and IROC as President. He was the Guest Editor of the special issue on MiroSot'96 of the Journal of Robotics and Autonomous Systems and on Soccer Robotics of the Journal of Intelligent Automation and Soft Computing. Dr. Kim is the recipient of the 1988 Choongang Young Investigator Award from Choongang Memorial Foundation, the LG YonAm Foundation Research Fellowship in 1992, the Korean Presidential Award in 1997, and the SeoAm Foundation Research Fellowship in 1999.