TELEOPERATION OF COLLABORATIVE MOBILE ROBOTS WITH FORCE FEEDBACK OVER INTERNET

Ivan Petrović
Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, Zagreb, Croatia
ivan.petrovic@fer.hr

Josip Babić
Končar - Institute of Electrical Engineering, Baštijanova bb, Zagreb, Croatia
babic.josip@gmail.com

Marko Budišić
Department of Mechanical Engineering, University of California, Santa Barbara, USA
marko.budisic@ieee.org

Keywords: teleoperation, mobile robots, force feedback

Abstract: A teleoperation system has been developed that enables two human operators to safely control two collaborative mobile robots in unknown and dynamic environments from any two PCs connected to the Internet by installing developed client program on them and by using simple force feedback joysticks. On the graphical user interfaces, the operators receive images forwarded by the cameras mounted on the robots, and on the the joysticks they feel forces forwarded by developed obstacle prevention algorithm based on the dynamic window approach. The amount and direction of the forces they feel on their hands depend on the distance and direction to the robot's closest obstacle, which can also be the collaborating robot. To overcome the instability caused by the unknown and varying time delay an event-based teleoperation method is employed to synchronize actions of each robot with commands from its operator. Through experimental investigation it is confirmed that developed teleoperation system enables the operators to successfully accomplish collaborative tasks in complex environments.

1 INTRODUCTION

Teleoperation is often employed in controlling mobile robots navigating in unknown and unstructured environments. This is largely because teleoperation makes use of the sophisticated cognitive capabilities of the human operator (Sheridan, 1992), (Murphy and Rogers, 1996). Conventional teleoperated mobile robots rely on visual contact with the operator, either directly or through video transmissions. Guiding such a robot is a formidable task, often complicated by the limited view from the camera. Under such conditions, a human teleoperator must exercise extreme care, especially in obstacle-cluttered environments. In order to increase the system performance and to reduce the operator stress and the task errors, force feedback from the robot to the human operator is usually employed, see e.g. (Sheridan, 1992), (Lee et al., 2002).

With the rapid development of information technology, the Internet has evolved from a simple data-sharing media to an amazing information world where people can enjoy various services, teleoperation beeing one of them. The use of Internet for teleoperation tasks has become one of the hottest topics in robotics and automation, see e.g. (Munir, 2001)–(Lo et al., 2004). On the other hand, the Internet also entails a number of limitations and difficulties, such as restricted bandwidth, arbitrarily large transmission delays, delay jitter, and packet lost or error, all of which influence the performance of Internet-based telerobotics systems. A number of approaches have been proposed to ensure stability of the force feedback loop closed over Internet, were the majority of them are based on passivity theory (Munir, 2001), (Niemeyer, 1996), (Niemeyer and Slotine, 2004) or on the event based action synchronization using a non-time refernce (Wang and Liu, 2005), (K. Brady, 2002), (Luo and Su, 2003). The later approach is used in this paper because of its simplicity and effectiveness.

There are many complicated and sophisticated tasks that cannot be performed efficiently by a single robot or operator, but require the cooperation of number of them. The cooperation of multiple robots and/or operators is particularly beneficial in cases

when robots must operate in unknown and dynamic environments. Teleoperation of multiple robots by multiple operators over Internet has been extensively studied for couple of years. A good survey can be found in (Lo et al., 2004).

In this paper we present a teleoperation system that consists of two mobile robots, where one of them serves as *Scout* and the other one as *Manipulator*. Scout explores the remote site and with its sensory information assists the operator controlling the Manipulator, which executes tasks of direct interaction with working environment by using the robot arm mounted on it. In order to guarantee safe robots navigation in dynamic environments and/or at high-speeds, it is desirable to provide a sensor-based collision avoidance scheme on-board the robots. By having this competence, the robots can react without delay on changes in its surrounding. Usually used methods for obstacle prevention are based on virtual forces creation between the robot and the closest obstacle, see e.g. (Borenstein and Koren, 1990) and (Lee et al., 2006), where the force is inversely proportional to the distance between them. In this paper, we propose a new obstacle avoidance algorithm based on the dynamic window (DW) approach presented in (Fox et al., 1997). Main advantage of our algorithm is that it takes robot dynamic constraints directly into account, which is particularly beneficial for safe navigation at high-speeds as the safety margin depends not only on distances between the robot and the nearby obstacles, but also on the velocity of robot motion. The algorithm is implemented on both robots and each robot considers the other one as the moving obstacle.

The paper is structured as follows. In Section 2, we present overview of developed mobile robot teleoperation system. Force feedback loops implementations are described in section 3. Section 4 describes experimental results. We conclude with a summary and a discussion of future work.

2 OVERVIEW OF THE SYSTEM

The teleoperation system considered in this paper is schematically illustrated in Fig. 1. It consists of two mobile robots (Scout and Manipulator) operating in a remote environment and two operator stations with PCs and force feedback joysticks. While the operators' PCs (clients) are directly connected to the Internet, robots' on-board PCs (servers) are connected to Internet via wireless LAN. Each operator controls a single robot and receives visual and other data from both robots.

Mobile robots that we use are *PIONEER 2DX* (Scout) and *PIONEER 3DX* (Manipulator) manufactured by *ActivMedia Robotics*. Scout is equipped with an on-board PC, a ring with sixteen sonar sensors, laser distance sensor and a *Sony EVI-D30* PTZ camera. Manipulator carries an external laptop computer, a ring with sixteen sonars and a *Cannon VC-C50i* PTZ camera. Additionally, a *Pioneer arm* with five degrees of freedom and a gripper is mounted on the Manipulator. Sonars on each robot are used for obstacle detection.

Any personal computer with an adequate Internet connection, a force feedback joystick and developed client application can serve as an operator station. The client application has graphical user interface (GUI) that enables the operator to preset the operation mode (*driving*, *manipulation*, *observation*) and to supervise both mobile robot actions.

Logitech WingMan Force 3D Joystick used in our experiments has two axes on which it can read inputs and generate force: x (stick up-down) and y (stick left-right) and two additional axes that can only read inputs: z (stick twist) and throttle. GUI provides additional operator input, e.g. when switching between operating modes. Both joystick references and GUI input are collectively referred to as *commands*. When client application is actively connected with the mobile robot and *driving* operating mode is chosen on the GUI, joystick x and y axes are used to input desired translational and rotational velocities, respectively. Force feedback is applied on the same two axis, defying commands that would lead robot toward detected obstacles. If *manipulation* operating mode is chosen two joystick buttons are used to chose one of arm's 6 joints, third button sends the arm in its home position and y axis is used to input chosen joint velocity and to display reflected force. In *observation* operating mode joystick x axis is used to tilt the camera, y axis to pan it, throttle is used for adjusting zoom, and one joystick button sends camera to its home position.

Communication between server applications running on the mobile robots' on-board PCs and client applications running on the operators' PCs is initialized and terminated by client applications. In special cases, e.g. when robot is turned off or malfunctioning or in case of communication failure, a server application can refuse or terminate the connection. Communication is implemented using three independent communication modules: control module, image transfer module and info module. Modules are executed within separate threads of applications and communicate using separate communication sockets. Modular structure decreases individual socket load and enables each module to transfer specific type of

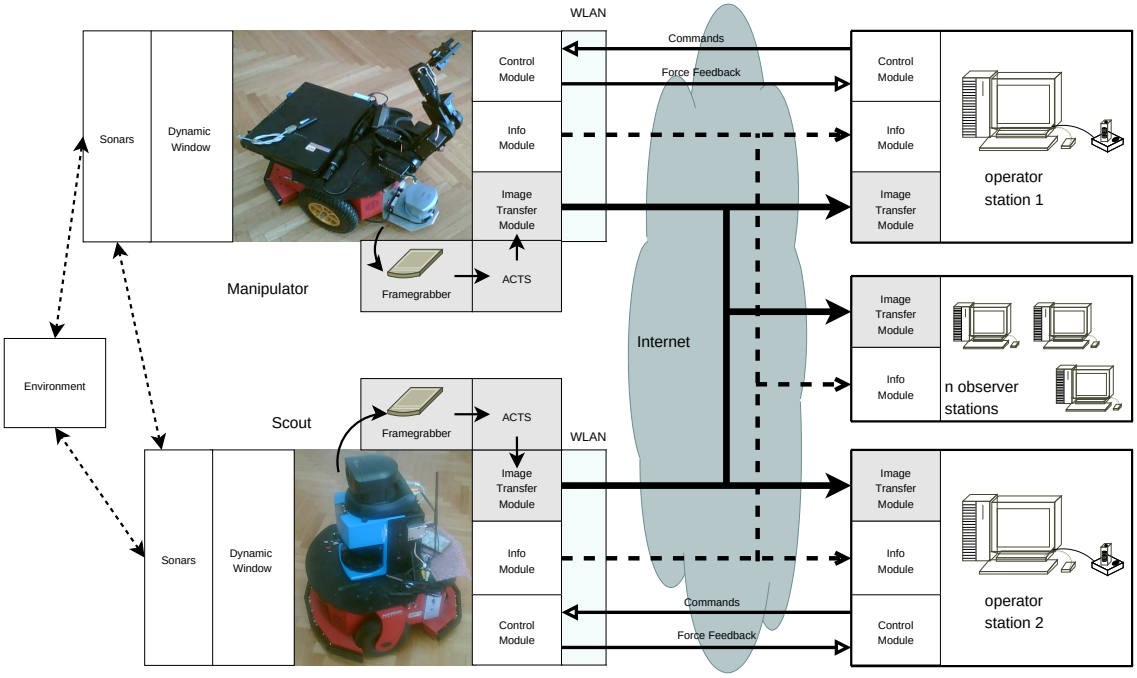

Figure 1: Schematic overview of the teleoperation system

data without employing complicated packet scheduling schemes.

Control module is used to transmit commands from the joystick to the robot and force feedback signal from the robot to the joystick. Depending on the operation mode, chosen on the client application GUI, these command can be robot's translational and angular velocities (driving mode), angular velocity of individual joints of the robot arm (manipulation mode) or they could be commands sent to the camera: pan and tilt angular velocity and zoom value (observation mode). If the operator controlls mobile robot's movement or one of robot's arm joints, control module delivers reflected force from the robot to the joystick. In case the operator controlls the camera reflected force is zero and it is only used for action synchronization.

Image transfer module transmits the frames of visual feedback signal from robots' cameras to operators via GUI of the client application. It delivers video signal image at a time. Cameras are connected to the PCs via frame grabber cards and images are fetched using *ActivMedia Color Tracking System* (ACTS) application. ACTS is an application which, in combination with a color camera and frame grabber hardware in a PC, enables custom applications to actively track up to 320 colored objects at a full $30[fps]$ image acquisition rate (ActivMedia, 2003). ACTS serves images to applications connected to it through TCP/IP. This functionality was not used and communication module was developed so that frequency and order in which clients receive images can be directly controlled. Server side communication module periodically requests an image form the ACTS and sends it to the connected clients.

Info module transmits time noncritical information, and is primarily used to synchronize local clocks and transmit information about robot velocities and arm positions to the clients.

3 FORCE FEEDBACK LOOPS

Force feedback from the remote environment gives important information to the human operator. Two force feedback loops have been implemented. One, which is implemented on both mobile robots, forwards the force to the corresponding operator in case of possible collision with the nearby obstacles or with the collaborating robot. Anther one, which is implemented only on Manipulator, forwards the force to its operator's hand when he controls the robot arm.

3.1 Event Based Action Synchronization

Main drawback of closing a control loop over the Internet is the existence of stochastic and unbounded

communication delay that can affect system performance and even make it unstable. These problems are usually addressed by ensuring passiveness of the control loop, see e.g. (Munir, 2001), (Niemeyer, 1996) and (Niemeyer and Slotine, 2004) or by using a non-time reference for action synchronization, as presented in (Wang and Liu, 2005), (K. Brady, 2002) and (Luo and Su, 2003). The later approach is used here, because there is no need for additional signal processing and consequently the control system is computationally much simpler. The stability of the control system with event-based action synchronization is ensured if a nondecreasing function of time is used as the action reference (Xi and Tarn, 2000). The number of completed control cycles, which is obviously a monotone increasing function of time, was chosen for this reference. Each control cycle (Fig. 2) is a sequence of the following actions:

1. Server application fetches the most recent force feedback from the buffer and sends it to the client application.
2. Client application receives force feedback.
3. Received force is applied to the joystick.
4. New commands are read from the joystick.
5. The commands are sent to the server application.
6. Server application receives new commands and refreshes the buffer. Depending on the operation mode, DW algorithm or arm controller periodically fetches the command from the buffer and refreshes the force feedback on the buffer after its execution. Commands are refreshed once for every force feedback signal sent to the client application.

Proper order of arrival of information packets is crucial to stability of control system, even more than their timely delivery. For this reason, UDP protocol is used for sending operator commands and force feedback. UDP, unlike TCP, does not resend lost packets and is therefore more suitable for real time applications. Additionally, UDP packets tend to be smaller in size than corresponding TCP packets which yields a smaller load on the communication channel. However, unreliable delivery of control packages could break the control cycle and longer communication delays may destabilize the system. To avoid this, server application monitors control cycle duration and if it exceeds a prescribed maximal value (e.g. 1 second), server application initiates a new control cycle by sending a fresh force feedback packet. All packets that arrive outside their control cycles are simply ignored.

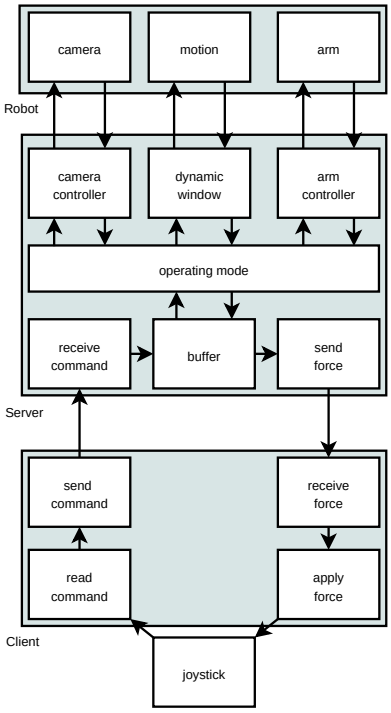

Figure 2: Event-Based Action Synchronization between client and server control modules

Described event-triggered control ensures that the force applied to the joystick corresponds to the command sent in the previous control cycle and that the buffer state is refreshed with the command that corresponds to the force feedback sent in the current control cycle. Therefore, action synchronization is achieved using the control cycle number as a time independent reference and the system is stable.

Action synchronization between Manipulator and its operator station is executed independently from action synchronization between Scout and its operator station. This arrangement is referred to as decentralized event-based control (Lo et al., 2004) and it assures that control cycle duration of one robot-operator station pair is not affected by communication delays of the other pair.

3.2 Collision Prevention

For safe robots navigation and cooperation in dynamic environments it is necessary to provide a sensor-based collision prevention scheme on-board each mobile robot. Here, we applied the DW algorithm, which is a velocity space based local reactive avoidance technique (Fox et al., 1997). Unlike directional reactive collision avoidance approaches (e.g. potential filed, vector field histograms), the DW algo-

rithm takes robot's kinematic and dynamic constrains directly into account by performing a search in space of translational and rotational velocities. DW produces trajectories that consist of circular and straight line arcs.

The DW algorithm can be integrated with a global path planing algorithm, e.g. FD* algorithm as in (Seder et al., 2005), for executing autonomous tasks in partially unknown environments. While global path planing algorithm calculates optimal path to a specific goal, the DW algorithm takes into account unknown and changing characteristics of the environment based on the local sensory information. We used DW algorithm in a teleoperation system, without global path planing algorithm, just to ensure safe motion of the mobile robot and to help the operator to better perceive obstacles. Therefore some modifications to the original algorithm had to be made.

Operator issues translational and rotational velocities' references. DW algorithm evaluates the given command while taking into account local sonar readings and kinematic and dynamic robot constrains. Commands that are not safe are not executed and force feedback is generated in order to warn the operator.

Proposed DW-based collision prevention algorithm consists of the following steps:

1. Desired velocities (v_d, ω_d) are fetched from the buffer.
2. They are constrained by maximal and minimal achievable velocities:
$$v_d \in [v_{min}, v_{max}], \quad \omega_d \in [\omega_{min}, \omega_{max}]. \tag{1}$$
3. Resulting velocities are additionally constrained to values from the set of velocities $V_{nc} = \{v_{nc}, \omega_{nc}\}$ achievable in one robot control cycle.
$$v_d \in v_{nc} = [v_c - T\dot{v}_m, v_c + T\dot{v}_m], \quad \omega_d \in \omega_{nc} = [\omega_c - T\dot{\omega}_m, \omega_c + T\dot{\omega}_m], \tag{2}$$
where v_c and ω_c are current velocities, $\dot{v}_m$ and $\dot{\omega}_m$ are maximal accelerations/decelerations and T is robot control cycle duration.
4. Minimal stopping path is calculated. Stopping time and applied translational deceleration must be established first. If condition
$$|\frac{v_d}{\dot{v}_m}| > |\frac{\omega_d}{\dot{\omega}_m}| \tag{3}$$
is satisfied, maximal translational deceleration a_s is applied during stopping time t_s:
$$t_s = |\frac{v_d}{\dot{v}_m}|, \quad a_s = \dot{v}_m. \tag{4}$$

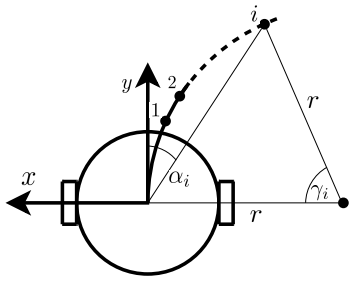

Figure 3: Trajectory of a robot is described as a circular arc

If (3) is not satisfied it takes more time to stop the rotation of the robot than its translation. Then translational deceleration smaller than maximal is applied:
$$t_s = |\frac{\omega_d}{\dot{\omega}_m}|, \quad a_s = |\frac{v_d}{t_s}|. \tag{5}$$
Minimal stopping path s_s is then:
$$s_s = v_d(t_s + t_{stm}) - \frac{a_s t_s^2}{2}, \tag{6}$$
where t_{stm} is additional safety time margin.

5. For the given velocities (v_d, ω_d) coordinates of N_p points on circular or straight line arc (Fig. 3) are calculated using following equations:
$$\gamma_i = 2\alpha_i = 2TN_c\frac{i}{N_p}, \quad x_i = x_c + sgn(v_d)rsin(\gamma_i), \quad y_i = y_c + sgn(v_d)sgn(\omega_d)rcos(\gamma_i), \tag{7}$$
where (x_c, y_c) is the center of the arc, $r = \frac{v_d}{\omega_d}$ its radius, $i = 1, \ldots, N_p$ is the index specifying the point on the trajectory, from 1 to N_p and N_c is the number of cycles for which the algorithm is executed.
6. Minimal allowed distance to obstacle ρ_{min} is:
$$\rho_{min} = r_r + s_{c1} + +(s_{c2} - s_{c1})\frac{v_d}{v_{max}} + s_{c\omega}\frac{\omega}{\omega_{max}}, \tag{8}$$
where r_r is robot radius, s_{c1} safety clearance at low translational velocities, s_{c2} safety clearance at high translational velocities and $s_{c\omega}$ rotational safety clearance. Rotational safety clearance is set to $s_{c\omega} = 0$ because it is considered safe to rotate the robot in place. This clearance can be set to a higher value if rotation in place is not considered safe, e.g. when rotating the robot with extended

robot arm. At low translational velocities (i.e. velocities close to zero) s_{c2} contributes little or not at all to ρ_{min} ($\rho_{min} = r_r + s_{c1}, v_d = 0, s_{c\omega} = 0$). At high translational velocities (i.e. velocities close to maximum value) s_{c1} has little or no influence on ρ_{min} value ($\rho_{min} = r_r + s_{c2}, v_d = v_{max}, s_{c\omega} = 0$).

7. For every point on the trajectory (v_d, ω_d), starting with the one closest to the robot, distance to the closest obstacle $\rho_i(v_d, \omega_d)$ is calculated and if the condition:
$$\rho_i(v_d, \omega_d) \geq \rho_{min}(v_d, \omega_d) \quad (9)$$
is true the calculation is executed for the next point on the trajectory. If (9) is satisfied for all N_p points on the trajectory then it is considered clear, force feedback is zero and the algorithm is finished.
8. If the condition (9) is not satisfied, path to the i-th point on the trajectory is calculated:
$$s_p = v_d \frac{i}{N_p} T N_c \quad (10)$$
9. If the path to the i-th point s_p is smaller than stopping path s_s
$$s_p < s_s, \quad (11)$$
i.e. it is possible to stop the robot before it comes closer than ρ_{min} to the obstacle, trajectory is considered safe, force feedback is calculated and the algorithm is finished. Force feedback is calculated as follows:
$$F_{amp} = \beta \sqrt{x_o^2 + y_o^2},$$
$$F_{ang} = atan2(y_o, x_o), \quad (12)$$
where F_{amp} and F_{ang} is force feedback amplitude (scaled to $[0, 1]$ with scaling factor β) and direction, respectively, (x_o, y_o) is the closest obstacle position in mobile robots coordinates and *atan2* is arctangent function.
10. If the condition (11) is not met, the investigated trajectory leads to collision, i.e. leads robot closer than ρ_{min} to the obstacle. To prevent collision, desired velocities magnitudes are decreased:
$$v_{d(i+1)} = v_{di} - v_{d1}/N_s,$$
$$\omega_{d(i+1)} = \omega_{di} - \omega_{d1}/N_s, \quad (13)$$
where $v_{d(i+1)}$ and $\omega_{d(i+1)}$ are velocities for the next iteration of the algorithm, v_{di} and ω_{di} are velocities of the current iteration, v_{d1} and ω_{d1} are original commanded velocities that entered the first iteration of the algorithm and N_s is the number of steps in which velocities are decremented. If the new values are different than zero, algorithm is repeated from step 4) until safe trajectory is found.

3.3 Force Feedback from the Manipulator's Arm

During robot arm movement force is reflected when the joint being controlled rotates to an angle that is less than 10° away from its maximum value:
$$F_{arm} = F_{max}(1 - \frac{\xi_{max} - \xi_i}{10}),$$
$$|\xi_{max} - \xi_i| < 10^\circ, \quad (14)$$
where F_{arm} is the reflected force amplitude corresponding to the current joint angle ξ_i, F_{max} maximal reflected force amplitude, and ξ_{max} maximal joint angle. Feedback force informs the Manipulator's operator that the controlled joint approaches its maximal angle.

4 EXPERIMENTS

In order to validate the developed system a number of experiments have been carried out during which different signals have been recorded. Results of four illustrative experiments are given here.

In the first experiment the operator drove the robot around an obstacle. The commanded velocities on client side (CS), commanded velocities on server side (SS), measured velocities, applied force feedback direction and amplitude were recorded. During the first phase of the experiment (from 0 s to 3 s, Fig. 4) robot moves forward and the obstacle is outside the DW sensitivity region. Measured velocities match the commanded ones and no force is reflected. When the operator starts turning the robot, the obstacle enters DW sensitivity region and force is reflected at -80° informing the operator that the obstacle is on the right from the robot (from 3 s to 6 s), and at the same time DW limits the robot angular velocity, while translational velocity matches the commanded one. At approximately $t = 6s$, the operator stops turning the robot, and the reflected force falls down, DW algorithm limits for a short time first the robot translational velocity and then angular velocity. Then the robot moves away from the obstacle and both velocities match the commanded ones, but the reflected force again slowly increases, which is caused by the next obstacle entering DW sensitivity region.

In the second experiment two operators, located at different places, attempted to drive the Scout and Manipulator robots into a head-on collision. This is the most difficult case for the modified DW algorithm to handle as both robots see the other as a moving obstacle. Velocities of both robots drop rapidly when the

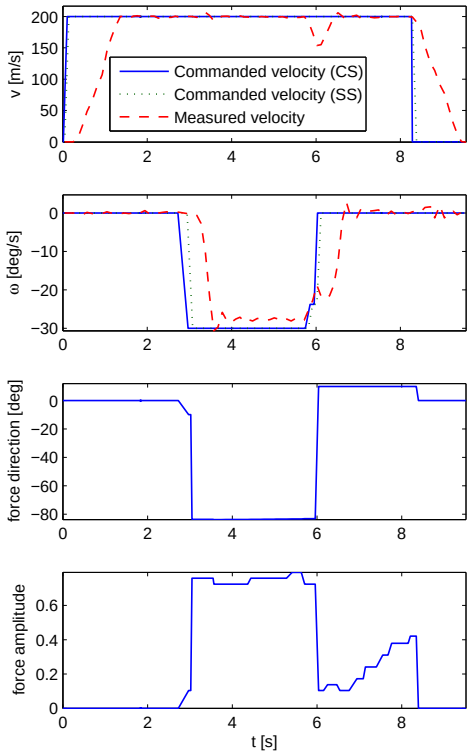

Figure 4: One robot experiment: driving around a corner

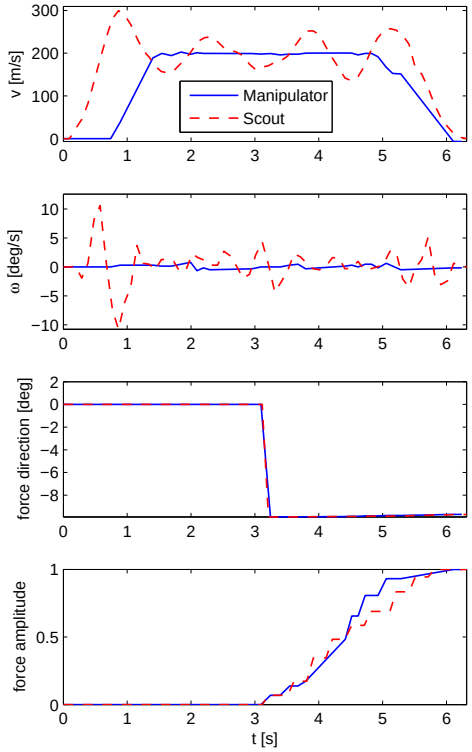

Figure 5: Velocities and force feedback recorded during two robot collision experiment

distance between them approaches the DW sensitivity range (at approximately $t = 3s$, Fig. 5). At the same time reflected forces sent to both operators rise. When force feedback signals reach their maximal values velocities of both robots drop to zero. Robots stop and the collision is avoided. Fluctuation of the Scouts's rotational and translational velocities is due to difficulties that its low-level velocity control system had while following reference values. In spite of this modified DW algorithm successfully prevented robots from colliding. Force feedback angle should be 0° during this experiment as the obstacle is directly in front of the robot. However, this angle varies between 0° and 10°. No stochastic processing was implemented and sonars were treated as rays whose orientation depends on sonar's position on the robot. Such an error is tolerated due to the fact that the force feedback is used only to provide the operator with a general notion about the position and distance to the closest obstacle.

In the last two experiments the task was to pick up a small cardboard box of the floor, place it on the Manipulator's back and put the arm at the home position. Only one operator controlling Manipulator accomplished it after few attempts and with some unnecessary adjustments (Fig. 6). Adjustments were needed as the camera is mounted close to the ground (for better object grasping) not covering the entire workspace of the arm. Another problem is the lack of information about the third dimension which complicates the adjustments of joint angles even when the arm is visible to the operator. The same task can be accomplished much easier and faster (approximately 60 seconds in contrast to approximately 110 seconds) if Scout and Manipulator cooperate (Fig. 7). Scout's assistance gives the Manipulator's operator better view of the distant site enabling him to see the arm during the whole procedure and giving him a feel of the third dimension.

5 CONCLUSION

A teleoperation system has been developed that enables two human operators to safely control two mobile robots in unknown and dynamic environments over the Internet. Each operator receives images displayed on the graphical user interface, which are forwarded by the cameras mounted on the robots, and force feedback on the joystick, which is reflected from

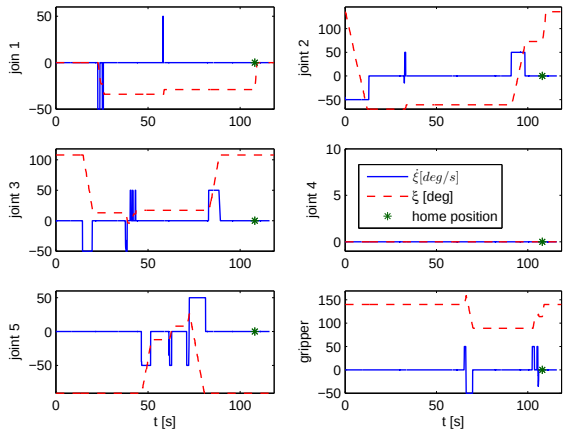

Figure 6: Motion of the robot arm joint angles during the one experiment

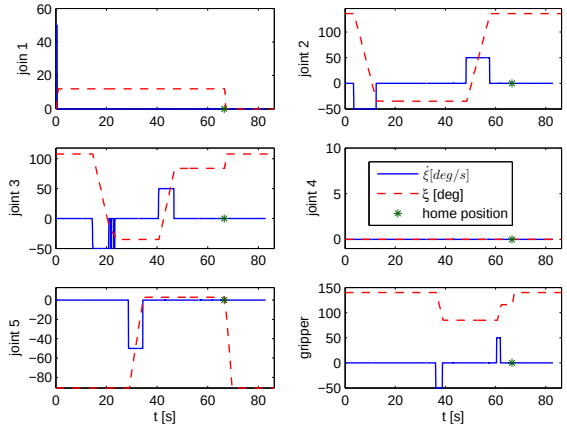

Figure 7: Motion of the robot arm joint angles during the two robots experiment

the robot controlled by him. To overcome the instability caused by the unknown and varying time delay, event-based teleoperation system is employed to synchronize actions of each robot with command from its operator. Through experimental investigation it is confirmed that developed teleoperation enables the operator to successfully accomplish teleoperation tasks in complex environments.

Developed teleoperation system could be easily adjusted to different robot and sensor types to allow application to different tasks. A possible application could be in missions of finding and rescuing victims from collapsed buildings in cases of natural disasters, fires or terrorist attacks. For example, Scout could be a small flying robot with various sensors that could easily maneuver the site searching for the victims while Manipulator could be stronger robot able to clear its way into the wreckage and carry them out of danger zone.

REFERENCES

ActivMedia (2003). *ACTS ActivMedia Robotics Color Tracking System — User Manual*. ActivMedia Robotics LLC, Amherst.

Borenstein, J. and Koren, Y. (1990). Teleautonomous guidance for mobile robots. In *IEEE Transactions on Systems, Man and Cybernetics*.

Fox, D., Burgard, W., and Thurm, S. (1997). The dynamic window approach to collision avoidance. In *IEEE Robotics & Automation Magazine*.

K. Brady, T. J. T. (2002). *An Introduction to Online Robots*. The MIT Press, London.

Lee, D., Martinez-Palafox, O., and Spong, M. W. (2006). Bilateral teleoperation of a wheeled mobile robot over delayed communication network. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*.

Lee, S., Sukhatme, G., Kim, G., and Park, C. (2002). Haptic control of a mobile robot: A user study. In *IEEE/RSJ International Conference on Intelligent Robots and Systems EPFL*.

Lo, W., Liu, Y., Elhajj, I. H., Xi, N., Wang, Y., and Fukada, T. (2004). Cooperative teleoperation of a multirobot system with force reflection via internet. In *TIEEE/ASME Transactions on Mechatronics*.

Luo, R. C. and Su, K. L. (2003). Networked intelligent robots through the internet: issues and opportunities. In *IEEE Proc.*

Munir, S. (2001). *Internet-Based Teleoperation*. PhD thesis, Georgia Instituteof Technology, Atlanta.

Murphy, R. and Rogers, E. (1996). Cooperative assistance for remote robot supervision. In *Presence: Teleoperators and Virtual Environments*.

Niemeyer, G. (1996). *Using Wave Variables in Time Delayed Force Reflecting Teleoperation*. PhD thesis, MIT, Cambridge, MA.

Niemeyer, G. and Slotine, J.-J. E. (2004). Telemanipulation with time delays. In *The International Journal of Robotics Research*.

Seder, M., Maček, K., and Petrović, I. (2005). An integrated approach to real-time mobile robot control in partially known indoor environments. In *IECON*.

Sheridan, T. (1992). *TTelerobotics, Automation, and Human Supervisory Control*. MIT Press, Cambridge, MA.

Wang, M. and Liu, J. N. K. (2005). Interactive control for internet-based mobile robot teleoperation. In *Robotics and Autonomous Systems*.

Xi, N. and Tarn, T. J. (2000). Stability analysis of non-time referenced internet-based telerobotic systems. In *Robotics and Autonomous Systems*.