Predictive Simulation of Autonomous Robots for Reliable Visualization over the Internet

Dirk Schulz Wolfram Burgard Armin B. Cremers
Department of Computer Science III, University of Bonn, 53117 Bonn, Germany

Abstract

Visual feedback is an important precondition for successful tele-operation of instruct-able mobile robots. Especially connections with varying and limited bandwidth, such as the Internet, prohibit the continuous transmission of video signals. In this paper we propose a predictive simulation technique which is designed to permit the reliable visualization of the robot's actions in tele-operation systems communicating with the user over the Internet. It differs from previous approaches, in that it includes a global path planner, a reactive collision avoidance in addition to a odometry and sensor simulation to predict the actions of the robot. The advantage of this approach is that the simulation of the robot's actions improves the accuracy of the visualization especially when large transmission delays of several seconds occur. We present experiments carried out with a real robot in a structured office environment illustrating the improvements in the visualization.

1. Introduction

Due to the increased costs of laboratory equipment such as mobile robots and due to the increased specialization of research on the other hand, there is a growing need for cooperation between research groups. The Internet can be regarded as one of the most important media for cooperation over large distances, today. Tele-robotics over the Internet can become an important means for the collaboration of research groups, for example during the development and the testing of control systems or for the demonstration of new systems.

Unfortunately the Internet does only provide a restricted bandwidth and arbitrary large transmission delays can occur, so that video streams cannot be used for the visualization of the robot's actions. Obviously, accurate visualizations are important for the observation of experiments and during demonstrations. Furthermore, with the development of web-operated robots, for example tele-operated museum tour-guide robots, there is a growing need for tele-operation interfaces which can be used even by untrained people. In this context, accurate visualizations of the robot's actions becomes more and more important.

In this paper we describe a technique which provides the basis for smooth and reliable real-time 3D visualizations of the movements of an autonomous mobile robot. It uses a predictive simulation scheme (PSS) to ensure that the visualization remains synchronized with the true state of the world even during larger transmission delays of several seconds. This simulation scheme includes the robot's collision avoidance and path planning components in addition to the odometry and sensor simulation of the robot. We compare the PSS to a variant that uses the last motion vector obtained from the robot to predict its future behavior. In extensive experiments we show that PSS provides significantly better results when transmission delays of more than three seconds occur.

The remainder of this paper is organized as follows. After discussing related work in the next section, Section 3 introduces the components of PSS in detail and finally Section 4 gives experimental results illustrating the strength of the approach.

2. Related Work

A variety of web based tele-operation interfaces for autonomous robots has been developed over the last few years. Three of the earlier systems are the Mercury Project, the "Telerobot on the Web" and the Tele-Garden [6, 7, 15]. These systems allow people to perform simple tasks with a robot arm via the web. They provide still images from a camera mounted on the robot arm after a requested movement task has been completed, no feedback is given while the robot arm is moving.

XAVIER [14] is a web-operated autonomous mobile robot. It can be adviced by web users to move to an office and to tell a joke. The web interface relies on client-pull and server-push techniques to provide images taken by the robot and a map indicating the robot's current position.

The autonomous mobile robots RHINO and Minerva, which were deployed as autonomous museum tour-guides in the "Deutsches Museum Bonn" in 1997 [3] respectively in the Smithsonian Museum of American History in 1998 [16], additionally offer Java applets for instant updates of information. For example, the interfaces provide applets displaying information about what the robot is currently doing and offering a smooth 2D animation of the robot's trajectory in a map of the environment.

Hirukawa et al. [8] describe web-operation interfaces, where web users can perform manipulation tasks using a 3D graphics simulation contained in the web browser. These interfaces follow the tele-programming approach. Tasks are first tested on a simulator and the tested sequence of actions can afterwards be transmitted to the real robot.

Simulation based delay compensation techniques and virtual environment interfaces are in use for space and undersea tele-robotics for several years now [4, 12, 18, 1]. Here comparable large transmission delays occur, but in contrast to Internet based systems, these systems are mostly operated via communication links with a guaranteed bandwidth and known transmission delays. Our simulation scheme is conceptually similar to the one used during the ROTEX experiment on board the space-shuttle COLUMBIA in 1993 [9]. During this experiment a 6 DOF robot arm was controlled from ground employing telesensor programming. This robot carried out a sequence of actions autonomously relying on it's sensors but the autonomy did not include planning, which was carried out by a human operator from ground. Autonomous robots, however, make their own decisions, for example, which path to take. Our mobile robot RHINO uses a reactive collision avoidance and a path planning component to make such decisions. Therefore, we integrate these components into the predictive simulation and this way achieve a reliable visualization even in case of large transmission delays of several seconds.

3. The Predictive Simulation Scheme PSS

The task of PSS is to achieve a better visualization accuracy over the Internet when long transmission delays and high packet loss rates occur. The current implementation of PSS is integrated into the RHINO system. RHINO is an RWI B21 robot equipped with 2 laser range finders, a ring of 24 ultrasonic sensors as well as with several tactile and IR sensors (see Figure 1). RHINO is a research platform used for the development of different service robot applications such as office delivery [13, 2] or giving tours to visitors of the "Deutsches Museum Bonn", Germany [3]. The RHINO system is a distributed software package assembled from over 25 modules. The PSS uses a simulator of the robot and it replicates the RHINO system's collision avoidance module and the path planning module to predict the robot's actions.

Figure 1. The RWI B21 robot RHINO.

3.1. Extrapolation of robot motion

A B21 robot is controlled by setting the translational and rotational accelerations $(\dot{v}, \dot{\omega})$ of it's synchro-drive at discrete points in time [5]. During a time interval $[t_0, t_n]$ only a finite number of acceleration commands can be issued. Let us assume, that acceleration commands are given at times $\{t_0, \ldots, t_{n-1}\}$ and that they remain constant during the intervals $[t_i, t_i + 1[$ for the duration $\Delta_i = t_{i+1} - t_i$. Under these assumptions, the dynamic of the synchro-drive is expressed by the functions

$$\begin{aligned}
x(t_n) &= x(t_0) + \sum_{i=0}^{n-1} \int_0^{\Delta_i} \mathrm{tv}(i,t) \cdot \cos(\mathrm{hd}(i,t))\, dt \\
y(t_n) &= y(t_0) - \sum_{i=0}^{n-1} \int_0^{\Delta_i} \mathrm{tv}(i,t) \cdot \sin(\mathrm{hd}(i,t))\, dt \\
\theta(t_n) &= \mathrm{hd}(n-1, t_n - t_{n-1}).
\end{aligned}$$

Here the functions *tv* and *hd* denote the translational velocity and the heading of the robot at some point in time x,

$$\begin{aligned}
\mathrm{tv}(k,x) &= (v(t_k) + \dot{v}_k \cdot x) \\
\mathrm{hd}(k,x) &= \theta(t_k) + \omega(t_k) \cdot x + \frac{1}{2}\dot{\omega}_k \cdot x^2
\end{aligned}$$

However, if we extrapolate the robot's motion, we have to compute the position many times per second. Therefore, we approximate the motion assuming that the velocities of the robot remain constant from one simulation step to the next. Thus the integral above can be simplified to

$$F_x^i = \int_0^{\Delta_i} v_i \cdot \cos(\theta(t_i) + \omega_i \cdot t)\, dt$$

for the x-coordinate. F_y^i is defined analogously. Here v_i and w_i denote the constant velocities during the time interval $[t_i, t_{i+1}[$. Solving the integral yields

$$F_x^i = \begin{cases} \frac{v_i}{w_i}(\sin(\theta(t_i) + \omega_i \cdot \Delta_i) - \sin(\theta(t_i))) & \omega_i \neq 0 \\ v_i \cdot \cos(\theta(t_i)) \cdot \Delta_i & \omega_i = 0 \end{cases}$$

for the x-coordinate and

$$F_y^i = \begin{cases} -\frac{v_i}{w_i}(\cos(\theta(t_i) + \omega_i \cdot \Delta_i) - \cos(\theta(t_i))) & \omega_i \neq 0 \\ v_i \cdot \sin(\theta(t_i)) \cdot \Delta_i & \omega_i = 0 \end{cases}$$

for the y-coordinate. Thus, the extrapolation inductively computes the robot's position given the position at time t_0 according to the following equations

$$\begin{aligned} x(t_{i+1}) &= x(t_i) + F_x^i \\ y(t_{i+1}) &= y(t_i) + F_y^i \\ \theta(t_{i+1}) &= \theta(t_i) + \omega_i \cdot \Delta_i \\ v_{i+1} &= v_i + \dot{v}_i \cdot \Delta_i \\ \omega_{i+1} &= \omega_i + \dot{\omega}_i \cdot \Delta_i \end{aligned}$$

Notice that the robot will move on a straight line if $\omega_i = 0$ and on a circular arc if $\omega_i \neq 0$.

3.2. Integration of the DWA Collision Avoidance

RHINO's collision avoidance protects the robot from colliding with obstacles. It is replicated in the PSS for two reasons, (1) the collision avoidance module is the component of the RHINO system, which generates the acceleration commands for the robot, and (2) it prevents the simulated robot from driving into obstacles, which guarantees that the simulation remains at least consistent with the laws of physics in case of transmission delays.

The collision avoidance is based on the dynamic window algorithm (DWA) [5] which uses several hard and soft constraints to control the velocities of the robot based on sensor input and a "target location" prescribed by the path planner. In the current implementation DWA decides on new velocities 4 times per second. The decision process is carried out in two phases, in the first phase the space of velocities is restricted to velocities satisfying the hard constraints, e. g. velocities which are admissible with respect to the robot's inertia and torque limits – this subspace forms the dynamic window –, and velocities which are save with respect to the obstacles detected by the proximity sensors.

In the second phase the reduced search space is searched for the best velocity vector (v_i, ω_i). This is accomplished by maximizing the objective function

$$G(v, \omega) = \sigma(\alpha \cdot \text{heading}(v, \omega) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{vel}(v, \omega))$$

which computes the weighted sum of the three soft constraints,

(a) *heading*, measuring the progress towards the goal location. It is maximal if the robot moves straight towards the target.

(b) *dist*, denoting the distance of the closest obstacle on the curvature (v, ω), and

(c) *vel*, the forward velocity of the robot, supporting fast movements.

The function σ smoothes the weighted sum of the three components and results in more side-clearance from obstacles.

3.3. Integration of Path Planning

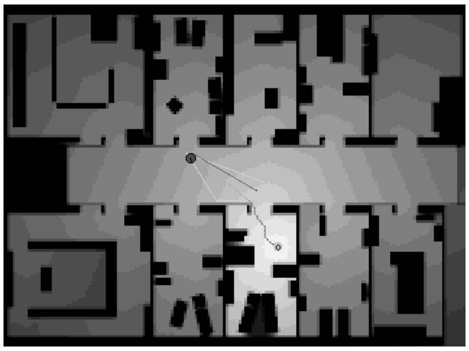


Figure 2. The path planner computes the optimal path to goal position (marked by 0) and the next intermediate goal position which is sent to the collision avoidance.

The task of RHINO's path planner [17] is to determine shortest paths from one point in the environment to another (see Figure 2). It generates a sequence of intermediate "target locations" thereby taking visibility considerations and increasing the side-clearance into account. Target locations are then passed on to the collision avoidance in advance.

The path planner is based on value iteration, a popular dynamic programming algorithm. It uses a grid map representation of the environment for planning, which is computed from the 3D model through a simple projection. Value iteration iteratively computes values $V_{x,y}$ for each grid cell $\langle x, y \rangle$. Initially, the grid cell containing the goal position is set to 0, and all others are set to ∞. During iteration the values of all unoccupied grid cells are set to the value of their best unoccupied neighbor plus the costs of moving there. After convergence, each value $V_{x,y}$ corresponds to the distance between $\langle x, y \rangle$ and the goal position and steepest decent in the value function leads to the shortest path to the goal. Figure 2 shows an example value function after convergence. Higher costs are represented by darker grey levels, the goal position is at the white spot. The minimal cost path to the goal is marked as a grey line.

After convergence, the value iteration algorithm has determined the optimal moving direction towards the goal position for every grid cell, independent from the robot's position. In PSS this gets important after prediction errors have occurred; the resynchronization will force the simulated robot to "jump" to the correct position. In such situations, the path planner is able to compute a new intermediate target point for the collision avoidance immediately. Consequently, the simulation can quickly be synchronized if new information is obtained from the robot.

3.4. The B21 Simulator

The core part of the PSS is the B21 robot simulator. The simulator we developed provides a simulation of the robot's odometry and of its proximity sensors. The odometry simulation employs the iterative scheme described in section 3.1 .

The simulation of the proximity sensors is achieved using ray tracing to calculate the distance between the sensor and the next obstacle in the sensing direction. Ray tracing is performed within the same 3D world model of the environment, which is also used for the visualization. The simulation takes sensor specific measuring errors into account. For example, the simulation includes the possibility of specular reflections of ultrasound beams. According to experiments carried out with RHINO the probability of such reflections increases with the angle α between the surface normal and the acoustic axis of the beam and also depends on the distance d to the obstacle. In case of a reflection the sensor does not perceive an echo and thus erroneously indicates free space. The probability $P(r \mid \alpha, d)$ of a specular reflection is defined as

$$P(r \mid \alpha, d) = \begin{cases} 0 & \alpha < \theta(d) \\ (\alpha - \theta(d))/\beta & \theta(d) \leq \alpha < \theta(d) + \beta \\ 1 & \alpha \geq \theta(d) + \beta \end{cases}$$

Here β denotes the angle range in which the failure probability increases linearly from 0 to certainty and $\theta(d)$ is a linearly decreasing function. The values of β and $\theta(d)$ depend on the material the obstacle is made of. Examples for a typical generated scan in an office environment is shown in Figure 3.

For the more reliable laser-range finder we use the distance to the next obstacle in the sensing direction directly and add an appropriate Gaussian noise to model the small measuring errors.

The 3D model of the environment uses a boundary representation of the objects. Realistic scene descriptions for 3D visualizations usually contain a large number of objects,

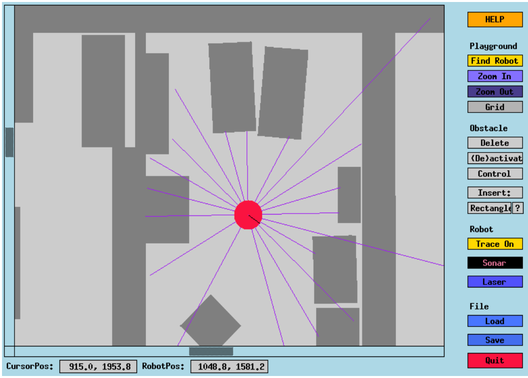


Figure 3. The RHINO simulator; The 2D visualization displays the length of the simulated sonar measurements.

such that for the distance calculations a large number of line-surface intersections have to be performed. In order to achieve the sensor simulation in real-time, the number of necessary line-surface intersections has to be reduced. We use a spatial indexing technique based on the rectangular tiling of the scene [11] for this purpose.

3.5. Synchronization

The current version of PSS has been integrated into the RHINO software. This prototype is a client-server system. The server provides the data for the synchronization of the visualization with the robot. As a module of the RHINO system it collects all the information required and transmits it to the clients. In addition, the server has to maintain the robot's position in the environment. Due to dead reckoning errors of the robot, it can not rely on the robot's odometry information directly for this purpose. It employs the RHINO systems localization facility, instead, to correct the odometry information before synchronization.

One synchronization message contains the current position of the robot, the velocities, the accelerations, the next goal position of the planner and a time stamp. This information is sufficient to resynchronize the PSS. The time stamp is used by the clients to compensate for the transmission time of a synchronization message. For this reason it is important, that the clients and the server have synchronized clocks. We rely on the "Network Time Protocol" [10] for this purpose. The synchronization is state-less, i. e. it only requires the latest synchronization message, a fact which is important because the system uses a fast but unreliable UDP socket connection for the Internet communication.

4. Experimental Results

To demonstrate the prediction accuracy of the PSS we performed a typical office delivery experiment and compared the quality of the visualization of PSS with a visualization obtained by using the extrapolation scheme described in section 3.1 alone. Here, RHINO started in the corridor of our department and visited five different offices in the order which is depicted in Figure 4. The length of the trajectory is 80.8 m and the robot needed 320 seconds to complete this task. The numbers in the offices correspond to the target-points sent to the path planning module.

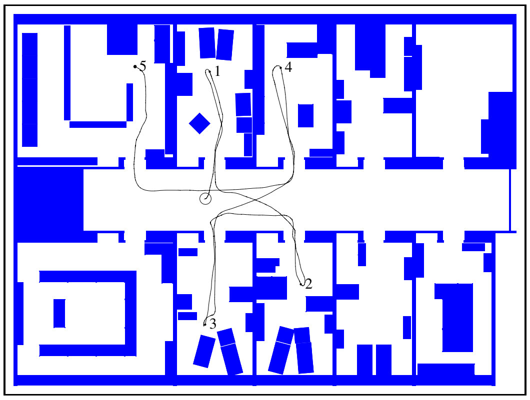


Figure 4. Trajectory of the robot during the office delivery task.

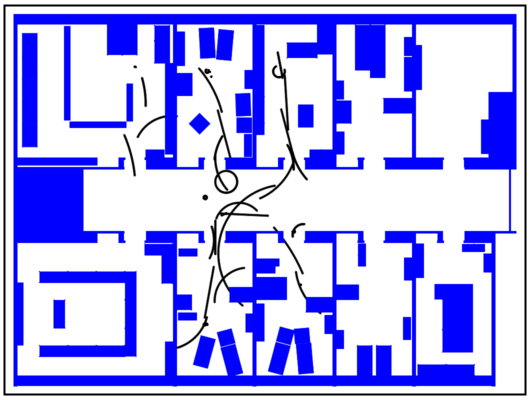


Figure 5. Trajectory estimated by extrapolation.

In the experiment, we manually decreased the packet transmission rate to approximately 10 packets per minute. We used a constant transmission interval, so that the time delay between subsequent packets was approximately 6 seconds. Since packets sometimes get lost, the effective time-delay between subsequent packets in several cases was $n \times 6$ seconds. Figures 5 and 6 show typical trajectories which are obtained if the robot's trajectory is extrapolated based on the most recent packet received (Figure 5) or if the behavior of the robot is simulated using PSS (Figure 6). The trajectories already demonstrate that PSS provides a better estimation of the robot's position than the extrapolation approach.

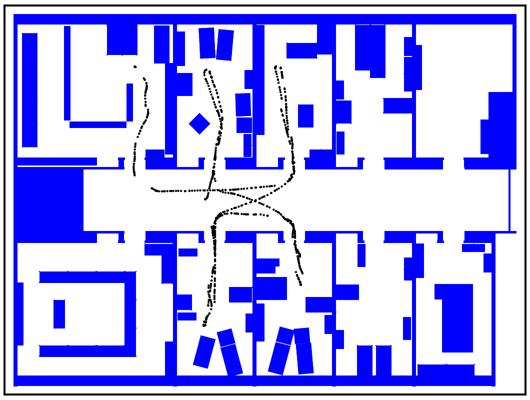


Figure 6. Trajectory predicted by PSS.

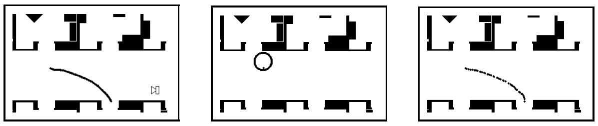
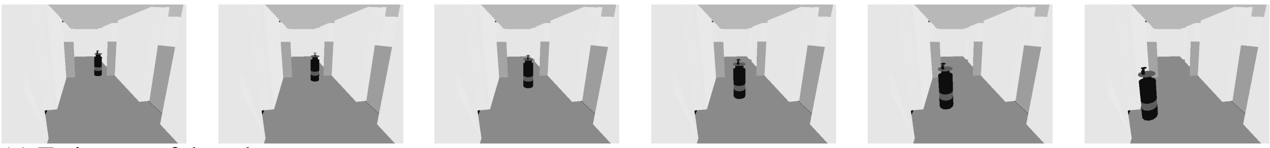


Figure 7. Trajectory of the robot taken after leaving room 1 (left), circular trajectory obtained by extrapolation (center), and trajectory predicted by PSS (right).

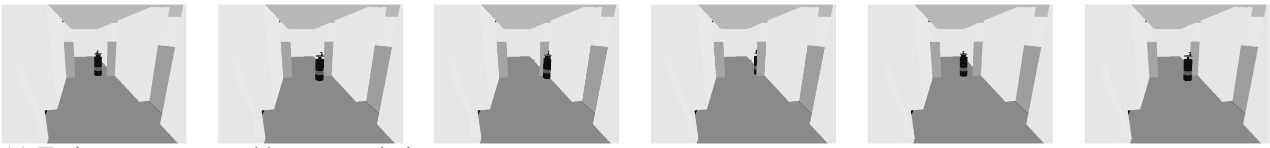
To illustrate the improvements in the visualization we now consider one subset of these trajectories corresponding to the packet, received by the client after the robot leaves room 1.

Figure 7 shows fractions of all three trajectories after the robot left room 1. In this particular case, the next packet was lost so that the system had to predict the robot's trajectory for a period of 12.5 seconds. Since the extrapolation does not change the velocities of the robot during a transmission gap, the robot moves on a circular trajectory. In contrast, PSS uses the path planner and the collision avoidance modules to compute intermediate velocities, which results in a trajectory which is close to that of the real robot.

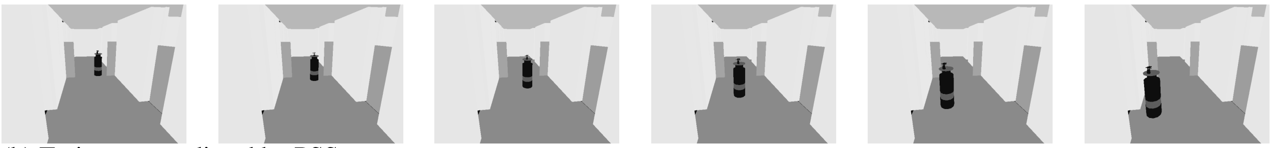
The corresponding sequences of computed images for the 3D visualization are shown in Figure 8. The time delay between subsequent images is 2 seconds. The first row shows the images obtained using the correct position of the robot, the second row the images computed by an extrapolation, and the third row the images obtained using PSS.



(a) Trajectory of the robot.



(c) Trajectory computed by extrapolation.



(b) Trajectory predicted by PSS.

Figure 8. A sequence of computed images showing RHINO moving through the corridor of our department. The time-difference between consecutive images is approximately 2 seconds. The camera position is illustrated in the left image of Figure 7.

Obviously PSS significantly improves the quality of the visualization.

We repeated this experiment several times for the whole trajectory using recorded data and varying the points in time when packets were transmitted. Figure 9 shows the average

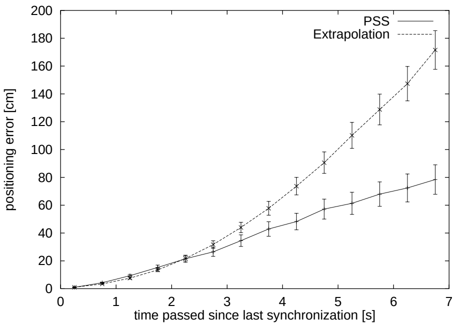


Figure 9. Average displacement depending on transmission gap.

positioning error – the distance between the predicted position and the position of the real robot – as a function of the time passed since the last packet was received. This shows, that PSS is able to significantly reduce the average positioning error compared to the extrapolation technique after transmission gaps of at least 2.5 seconds. Since these differences depend on the speed of the robot, which was $34.5\,\mathrm{cm/s}$ on average in this experiment, we additionally computed the average displacement depending on the distance traveled after the latest synchronization packet. The

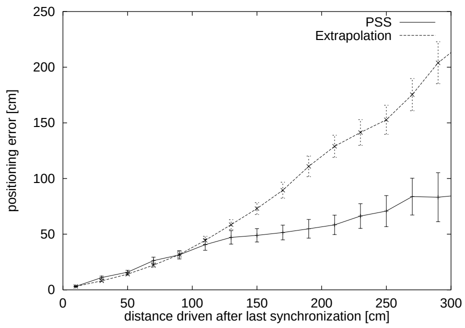


Figure 10. Average displacement depending on the distance traveled after the last synchronization.

resulting plots are shown in Figure 10. Thus, at least after 1 m of travel, PSS provides a significantly better prediction than extrapolation. In both figures, the error bars indicate the 95% confidence interval of the average mean.

5. Summary and Conclusions

In this paper we presented PSS, a predictive simulation system for the accurate visualization of tele-operated autonomous robots. In addition to odometry and sensor system simulators this system includes a global path planner and a reactive collision avoidance system to predict the robot's trajectory in the case of transmission gaps.

The PSS has been implemented using the RHINO system, the distributed control software for our mobile robot RHINO. The task of the PSS is to predict the robot's behavior in order to achieve accurate visualizations while watching the robot operate over the Internet. In this paper we demonstrated that PSS provides accurate predictions even in situations in which no packet has been received for a period of over 12 seconds. In experiments with an office delivery task, the predictions of PSS were significantly better than those obtained by extrapolating the robot's trajectory based on the velocities given with the last synchronization message, provided that the robot traveled at least 1 m during the transmission gap.

Despite these encouraging results there are several warrants for future research. The most important topic concerns the extension to dynamic environments and the visualization of manipulation tasks. This requires a reliable model update mechanism based on the sensor interpretation capabilities provided by the robot system.

References

[1] B. Bon and S. Homayoun. Real-time model-based obstacle detection for the NASA ranger telerobot. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, April 1997.

[2] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The mobile robot Rhino. *AI Magazine*, 16(2):31–38, Summer 1995.

[3] W. Burgard, A. B. Cremers, D. Fox, G. Lakemeyer, D. Hähnel, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998.

[4] L. Conway, R. A. Volz, and M. W. Walker. Teleautonomous systems: Projecting and coordinating intelligent action at a distance. In *IEEE Transactions on Robotics and Automation*, volume 6, April 1990.

[5] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, Mar. 1997.

[6] K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley. Desktop tele-operation via the world wide web. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1995.

[7] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, J. Wiegley, and E. Berger. The telegarden. In *Proc. of ACM SIGGRAPH*, 1995.

[8] H. Hirukawa, T. Matsui, and H. Onda. Prototypes of teleoperation systems via a standard protocol with a standard human interface. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[9] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl. Sensor-based space robotics–ROTEX and its telerobotic features. In *IEEE Transactions on Robotics and Automation*, volume 9, October 1993.

[10] D. L. Mills. Network time protocol (version 3) specification, implementation and analysis. In *Request for Comments (RFC) 1305, Internet Engineering Task Force (IETF)*, March 1992.

[11] H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley Publishing Company, 1990.

[12] T. B. Sheridan. Space teleoperation through time delay: Review and prognosis. In *IEEE Transactions on Robotics and Automation*, volume 9, October 1993.

[13] R. Simmons. The 1994 AAAI robot competition and exhibition. *AI Magazine*, 16(2):19–29, Summer 1995.

[14] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proceedings of the First International Conference on Autonomous Agents*, Marina del Rey, CA, February 1997.

[15] K. Taylor and J. Trevelyan. A telerobot on the world wide web. In *Proceedings of the 1995 National Conference of the Australian Robot Association*, July 1995.

[16] S. Thrun, M. Bennewitz, W. Burgard, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second-generation museum tour-guide robot. unpublished.

[17] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1997.

[18] Y. Tsumaki and M. Uchiyama. A model-based space teleoperation system with robustness against modeling errors. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, April 1997.