Robust Visualization for Web-based Control of Mobile Robots

Dirk Schulz
Department of Computer Science
University of Bonn
D-53117 Bonn

Wolfram Burgard
Department of Computer Science
University of Freiburg
D-79110 Freiburg

10th July 2001

Abstract

The World Wide Web provides a unique opportunity to connect robots to the Internet. Easy to use Web interfaces enable people all over the world to control mobile robots and to monitor their operation from the distance. Additionally, connecting a robot to the Internet offers distributed research groups the opportunity to perform joint experiments. In this context visualizing the robot's actions plays a key role. In this paper we present different techniques designed to provide robust visualizations of Web-controlled mobile robots in dynamic environments. These techniques address two major drawbacks of the Internet, the limited bandwidth and the unreliability of connections resulting in packet loss. Our technique uses a 3D-model of the environment and visualizes the robot´s actions using this model. It employs predictive simulation to bridge transmission gaps and applies probabilistic techniques to estimate the state of dynamic objects which influence the behavior of the robot. This way accurate visualizations of the actions of a mobile robot and of relevant aspects of its environment can be achieved.

1 Introduction

The Internet provides a unique opportunity to tele-operate and monitor mobile robots. Web-controlled mobile robots can give people all over the world the ability to become tele-present at distant places. Additionally, Internet-based tele-experimentation systems for mobile robots give distributed research groups located at distant places the ability to carry out joint experiments. This way, they can share expensive robot platforms and furthermore save travel expenses. Finally, the Internet can be used for on-line demonstrations with mobile robots, for example during the presentation of research results at conferences.

All these applications of Web interfaces for mobile robots require accurate visualization techniques. This includes a high level of detail as well as high update rates. Unfortunately, the Internet does only provide a restricted bandwidth and arbitrarily large transmission gaps can occur. Because of the limited bandwidth video streams with appropriate resolutions do not achieve the frame rate required for a smooth visualization of the robot's actions. Installing static monitoring cameras, a method frequently used on the Internet, is also not a feasible approach; they only provide a reduced field of view and important details may also be occluded.

In this paper we present a system which provides accurate and smooth real-time 3D visualizations of the movements of an autonomous mobile robot over the Internet. The visualization component uses a 3D model of the robot and its environment to display the current state of the robot. Given this model, only the current state of the robot needs to be transfered over the Internet in order to animate the robot's actions. As this is only a small amount of information, these updates can be carried out at high frequencies. However, on the Internet several of the state updates may get lost. To bridge the resulting transmission gaps, the visualization component of our system also performs predictive simulations. This way, it is able to accurately visualize the state of the robot even during transmission gaps of several seconds.

The system is designed for mobile robots operating in dynamic environments. In this situation, the position and orientation of several objects in the environment may have a direct influence on the behavior of the robot. Typical objects whose states are relevant for the robot's actions are doors, tables, wastebaskets, etc. To accurately deal with such dynamic objects, their current states have to be visualized. Our system therefore includes a probabilistic technique to continuously estimate the states of these objects based on the data obtained with the robot's sensors. It determines the current object states (position and/or orientation) and transfers them to all sites connected over the Internet, which instantly update the visualization accordingly.

This paper presents its architecture of the system. It describes the predictive simulation technique as well as the state estimation method. After discussing related work in the following section, we describe the key ideas of the visualization system in Section 3. In Section 4 we present several experiments illustrating the advantages and the robustness of our approach.

2 Related Work

A variety of Web-based tele-operation interfaces for 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 [8, 9, 23]. These systems allow people to perform simple tasks with a robot arm via the Web. Since these arms operate in prepared workspaces without any unforeseen obstacles, all movement commands issued by a Web user can be carried out in a deterministic manner. Additionally, it suffices to provide still images from a camera mounted on the robot arm after a requested movement task has been completed. Our system, in contrast, is designed to visualize the

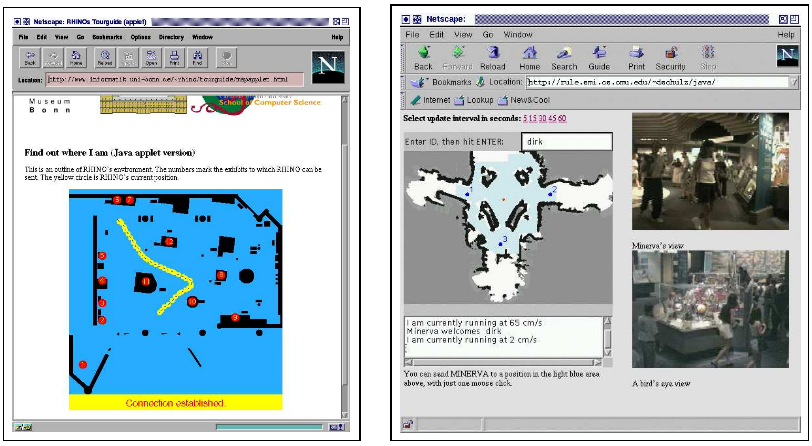


Figure 1: Java-based control interfaces of the robots Rhino (left) and Minerva (right)

actions of an autonomous mobile robot operating in a dynamic environment. According to that, the the execution of the same task depends on the current situation in the robot's environment.

Xavier [22] is a Web-operated autonomous mobile robot. It can be advised 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 visual feedback of the robot's movements; this includes images taken by the robot as well as a map indicating the robot's current position. However, Xavier's interface does not include any techniques for bridging transmission gaps or to reflect changes of the environment.

The autonomous mobile robots Rhino and Minerva, which were deployed as interactive museum tour-guide robots in the "Deutsches Museum Bonn" in 1997 and in the Smithsonian Museum of American History in 1998, could also be operated over the Web [2, 24]. In addition to image streams, recorded with on-board and off-board cameras, their interfaces offered Java applets for instant updates of information. For example, the interfaces displayed information about the current actions of the robot and included applets providing smooth 2D animations of the robot's trajectory in a map of the environment [20] (see Figure 1). Although both robots used the communication interface described in this paper, the interfaces did not use the predictive simulation to compensate transmission gaps. The map employed by the visualization was completely static and dynamic aspects were not handled at all.

3D graphics visualizations for Internet-based robot control have already been suggested by Hirukawa et al. [10]. Their interface allows Web users to carry out manipulation tasks with a mobile robot, by controlling a 3D graphics simulation of the robot contained in the Web browser. In contrast to our system, these interfaces do not provide on-line visualizations of the robot's actions. Instead, they follow the tele-programming approach. Tasks are first tested off-line in the 3D simulation environment and are afterwards transmitted via the Internet for execution by the real robot.

Simulation-based delay compensation techniques and virtual environment interfaces are frequently used for space and undersea robotics [3, 26, 21, 1]. These methods are designed to deal with large transmission delays but, in contrast to Internet based systems, rely on communication links with a guaranteed bandwidth and known transmission gaps. A visualization method, which is often used in tele-manipulation systems are predictive displays [13]. Methods of this type combine low-bandwidth video transmissions with graphics visualizations.

The simulation scheme we use is conceptually similar to the one used during the ROTEX experiment on board the space-shuttle COLUMBIA in 1993 [11]. During this experiment a 6 DOF robot arm was controlled from ground employing tele-sensor programming. A human operator planned a sequence of "elemental operations" which was afterwards carried out by the robot. Elemental operations are short-term reactive control tasks, like for example grasping operations, which the robot can carry out autonomously using its sensors. Truly autonomous robots, however, include methods to plan even complex actions and to react to unforeseen circumstances. For example, most environments are not completely accessible by a mobile robot. The robot must therefore have the ability to determine the shortest path to a given target point. In addition, it must be able to re-plan dynamically whenever unexpected obstacles block its path.

The predictive simulation system described in this paper is designed to provide accurate and detailed visualizations of the actions carried out by a mobile robot over the Internet. The main advantage of this system is that it includes techniques to deal with unreliable connections without a guaranteed bandwidth such as the Internet. In the case of transmission gaps, our system uses a robot simulator to simulate the robot's actions. Additionally, it uses path planning components to accurately forecast the robot's actions even if transmission gaps of more than ten seconds occur.

In general, it is not sufficient to regard the environment as static and to display the robot within a static model of the environment. In practice, the environment of a mobile robot contains several non-static objects which influence the behavior of the robot. Among them are doors, chairs, and tables, which often change their position and can prevent the robot from taking a previously planned trajectory. To acquire and maintain a model of the environment is a major research area in mobile robotics. The most frequently used types of maps are metric and topological maps. Topological models, as used in [14, 17], describe the environment at a coarse resolution. Because of the lack of necessary details, these types of maps are only of limited use for visualizations. Metric maps, on the other hand, describe the environment at a finer level of detail. A popular approach are discrete occupancy grids proposed in [5, 16]. Each cell of such a grid contains the probability that the corresponding space in the environment is occupied. The major advantage of occupancy maps lies in the existence of techniques for their acquisition and maintenance based on sensor information. However, most of the occupancy grid techniques are intended for static environments only. Since all cells are considered independently, they cannot appropriately represent dynamic objects such as doors, tables or chairs. In this paper we use a 3D model of the environment containing all relevant objects of the environment. We apply a probabilistic approach to estimate the state of dynamic objects. Changes of these states are also updated in the 3D model so that the visualization can quickly be adapted to the current state of the environment.

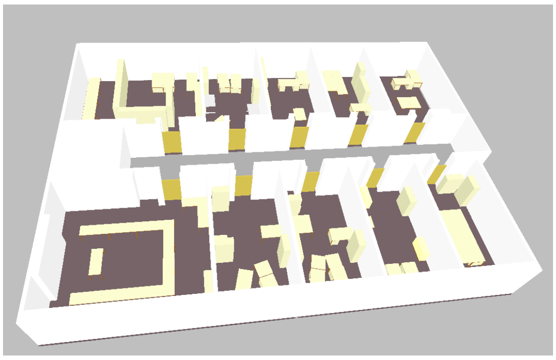


Figure 2: Three-dimensional boundary representation of the robot's environment

3 The Visualization System

To display the robot and its actions in the environment we use a 3D visualization system. This visualization system requires a 3D model of the robot's environment represented by a hierarchical scene graph, in which objects are described in the boundary representation. Figure 2 shows the 3D model of our department in which the experiments described below were carried out. Given a corresponding description of the robot it can provide a visualization of the robot at arbitrary locations in the environment.

Our system has a client-server architecture. Initially, each client receives the 3D model. Afterwards, the server sends updates of the current state of the robot in regular intervals to all its clients. Unfortunately, the bandwidth of the Internet is limited and the delivery of packets is unreliable so that larger transmission gaps can occur especially on overseas connections. To cope with these limitations, our system provides a predictive simulation in order to predict the robot's actions even when larger transmission gaps of several seconds occur.

As pointed out above, the environment of the robot is not static. Generally, there are several dynamic objects in the real world which influence the robot's behavior. To correctly reflect the state of these objects, the remote clients also receive the states of these objects from the server, in order to update the world model according to changes. To provide accurate states of objects, the server is connected to a state estimation component, which continuously determines the state of objects using the robot's laser range-finders.

3.1 State Estimation of Dynamic Objects

To estimate the state of dynamic objects, a mobile robot has to use its sensors. The information provided by the sensors, however, is inherently noisy so that state estimates relying on this information are always to some degree uncertain. Estimating the state of objects based on sensor information provided by a mobile robot, furthermore depends on the robot's position within the environment. This position also has to be determined by a state estimation process and therefore is also uncertain. To correctly deal with these uncertainties we use probabilistic state estimation techniques.

The correct probabilistic approach to continuously maintain a dynamic world model obviously is to maintain the joint distribution over all states of the dynamic objects within the environment. Unfortunately, this approach is not tractable since the size of the joint state space grows exponentially in the number of objects. Therefore, we consider a marginal distribution and estimate the states of the dynamic objects independently.

Since the measurements o of the robot are inherently uncertain, we are interested in the most likely state $\tilde{s}$ of each object, i.e. the state which maximizes the probability $p(s \mid o)$

$$\tilde{s} = \operatorname*{argmax}_{s} p(s \mid o). \tag{1}$$

Here $p(s \mid o)$ is the probability that the currently considered object is in state s given the observation o.

To compute $p(s \mid o)$ we have to take into account that the position l of the robot from which the observation o is obtained is generally uncertain. We therefore assume that the robot maintains a belief $p(l)$ over all possible locations. To take the current belief of the robot into account, we integrate over all possible locations of the robot:

$$p(s \mid o) = \int_{l} p(s \mid o, l) \cdot p(l)\, dl. \tag{2}$$

Given Bayes rule using l as background knowledge we obtain

$$p(s \mid o, l) = \frac{p(o \mid s, l) \cdot p(s \mid l)}{p(o \mid l)}, \tag{3}$$

where the denominator $p(o \mid l)$ can be be rewritten as

$$p(o \mid l) = \int_{s} p(o \mid s, l) \cdot p(s)\, ds. \tag{4}$$

Inserting Equation (3) into Equation (2) leads to

$$p(s \mid o) = \int_{l} \frac{p(o \mid s, l) \cdot p(s \mid l)}{p(o \mid l)} \cdot p(l)\, dl. \tag{5}$$

Under the strong assumption that the state s of the object and the position l of the robot are independent, we simplify this to

$$p(s \mid o) = \int_{l} \frac{p(o \mid s, l) \cdot p(s) \cdot p(l)}{p(o \mid l)}\, dl. \tag{6}$$

Thus, all we need to know are the quantities $p(s)$ which is the prior distribution of the state s of the currently considered object, the current belief $p(l)$ of the position of the robot as well as the term $p(o \mid s, l)$.

To estimate $p(l)$, i.e. the probability that l is the robot's current position, we apply Monte Carlo localization [4, 6] which is an efficient variant of Markov localization [7]. In several robot systems applied in real-world scenarios, Markov localization has been proven to be robust and to provide accurate position estimates. The key idea of Markov localization is to maintain a probability distribution over all possible positions of the robot. Monte Carlo localization is a variant, which uses a set of samples to represent this density. Our current system uses this sample set as input for the state estimation process. Since the samples provided by Monte Carlo localization are randomly drawn from the probability density about the robot's current position, Equation 6 can be reduced to

$$p(s \mid o) \quad = \quad \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \frac{p(o \mid s, \lambda) \cdot p(s)}{p(o \mid \lambda)} \tag{7}$$

where Λ is the set of samples representing the robot's current belief about its position.

The term $p(o \mid s, l)$ belongs to the crucial aspects of the state estimation process since it describes the probability of making a certain observation o given the state s of the object and the location l of the robot. The usual way to compute this probability is a template matching process which compares the current measurement to the expected measurement given the position of the robot and the state of the object. Our system uses a laser range-finder for this purpose. A scan of a laser range-sensor generally consists of a set of beams representing the distance to the next obstacles at a fine angular resolution (see, for example, the left image of Figure 4). Under the assumption that the individual beams are independent given the position of the robot, we can compute $p(o \mid s, l)$ in the following way:

$$p(o \mid s, l) \quad = \quad \prod_{i=1,\ldots,n} p(d_i \mid \hat{d}_i(s, l)). \tag{8}$$

Here n is the number of beams, d_i is the length of beam i and $\hat{d}_i(s, l)$ is the expected distance for beam i given the state s of the object and the location l of the robot. In our current implementation we use a Gaussian distribution centered around the expected distance $\hat{d}_i(s, l)$ to determine the probability $p(d_i \mid \hat{d}_i(s, l))$.

3.2 Predictive Simulation

As already pointed out above, one of the major drawbacks of the Internet in the context of tele-operated mobile robots is the missing guarantee of a certain transmission rate and the loss of packets. Thus, it is generally impossible to obtain updates of the current position of the robot with a guaranteed frequency. To cope with this problem, visualization techniques have to extrapolate the actions of the robot between two subsequent packets in order to provide smooth animations of the robot's actions. In this context, a common technique is to extrapolate the robot's actions using the current velocities. This, in fact is the same as performing dead reckoning based on the latest robot position and velocities received. Our approach, in contrast, uses a predictive simulation technique, which also respects the real robot's path planning component to achieve better long term predictions. As a result, our system provides accurate predictions for more than ten seconds.

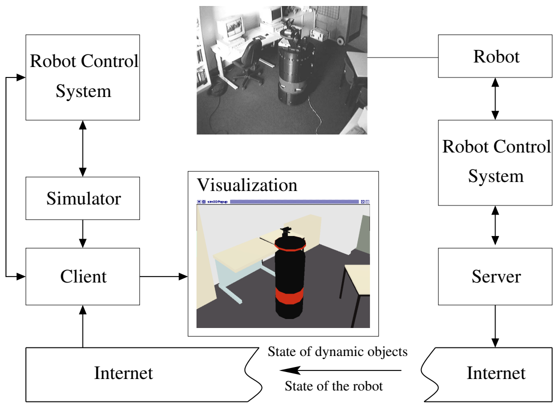


Figure 3: Architecture of the predictive simulation and visualization system.

Figure 3 shows the client-server architecture of the predictive simulation system. The server, which is located at the robot's site, sends synchronization messages via the Internet to its clients. Each synchronization message contains the current state of the robot consisting of the current position, velocities and target point of the robot. It furthermore includes the current state of the dynamic objects as well as the current time.

Instead of the standard TCP protocol utilized by most applications, our system uses the unreliable UDP protocol to transfer the state information from the server to its clients. Since the Internet is subject to packet loss, the use of a reliable protocol like TCP can increase transmission delays, as the re-transmission of lost messages prevents the arrival of new information. In several experiments carried out with typical long distance connections, we frequently observed packet loss rates of up to 30 percent whereas the transmission time of a single packet rarely exceeded 0.3 seconds. To deal with longer transmissions each client warns the user if a packet needed more than 0.4 seconds for the delivery. The synchronization of the clocks of the clients and the server is obtained by the "Network Time Protocol" [15].

After receiving a state message, the client updates the position of the robot and the state of the dynamic objects. Afterwards it starts a simulation to predict the robot's actions until the next packet is received. The current implementation of our system [19] includes a simulator to simulate the robot's actions and its proximity sensors. It is also connected to the path planning components [25] used by the real robot.

The simulation component provides methods for carrying out ray casting within the world model which builds the basis for the proximity sensor simulation, required to simulate the behavior of the path planning components. Since realistic scene descriptions generally contain a large number of objects, a huge amount of line-polygon intersections has to be calculated to simulate the proximity sensors. To achieve this

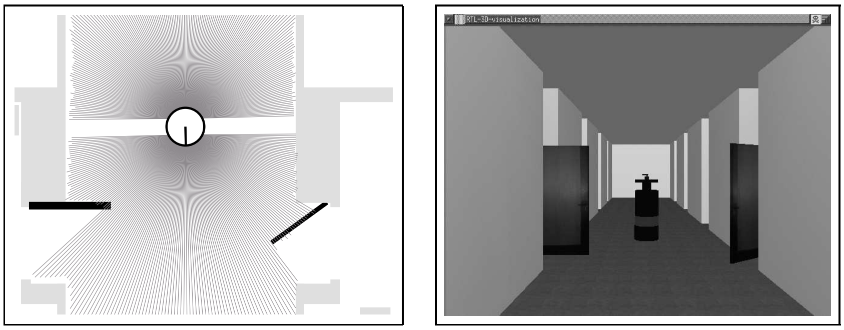


Figure 4: Rhino detecting the state of two doors with its laser range-finder (left image). The estimated states of the doors are depicted in black. The resulting 3D-visualization is shown in the right image.

in real-time, the simulation system applies a spatial indexing technique based on the rectangular tiling of the scene [18].

4 Experimental Results

The state estimation technique and the predictive simulation method have been implemented and extensively tested and evaluated using our mobile robot Rhino, an RWI B21 robot. Rhino is equipped with two laser range-finders covering almost 360 degrees of the robot's surrounding. These sensors are used for the evaluation of the state estimation technique described above. The visualization system is based on OpenGL so that the actions of the robot can be visualized with at least 20 Hz frame rate using 3D hardware acceleration on a standard low-cost PC.

4.1 State Estimation for Dynamic Objects

To evaluate the state estimation technique and its advantages for a robust visualization of the robot's activities we performed several experiments in our office environment. In these experiments, the task of the robot was to estimate the state of the doors while the robot was traveling along the 26m long corridor of our department building.

To efficiently carry out the state estimation process, our current implementation uses a discrete set of possible states. For doors we typically use 10 states ranging from 0 to 90 degrees. To obtain more accurate estimates of the state of objects we integrate over a small set of different measurements. Figure 4 shows a typical situation in which the robot measures two doors. The resulting states of the doors are depicted in black. Figure 4 also shows the corresponding 3D visualization.

Figure 5 illustrates the trajectory of the robot as well as the states of the doors which were kept unchanged during the experiment. Table 4.1 shows the results of the state estimation process for each door. As can be seen from these data our technique

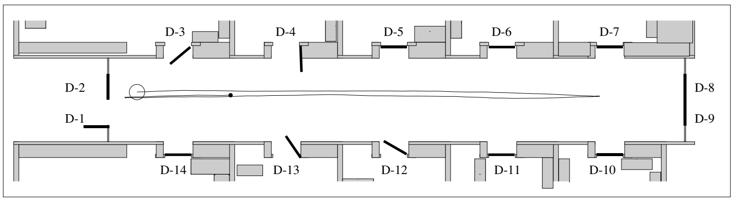


Figure 5: Trajectory of the robot and state of the doors in the corridor of the Department of Computer Science, Bonn.

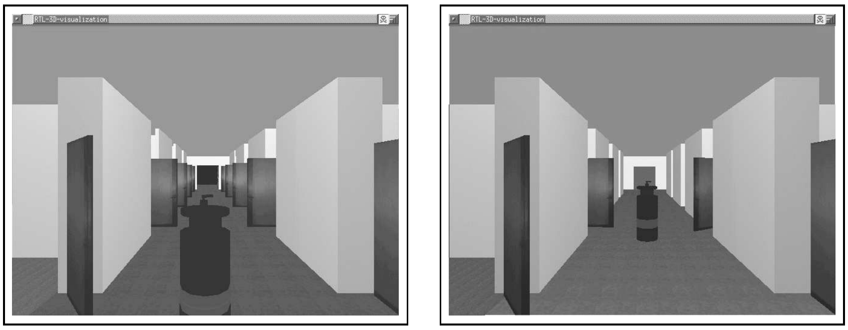


Figure 6: Visualization of the corridor at the beginning (left image) and at the end of the trajectory (right image).

provides highly accurate results. In most of the cases the standard deviation is below 5 degrees.

Door	D-1	D-2	D-3	D-4	D-5	D-6	D-7	D-8	D-9	D-10	D-11	D-12	D-13	D-14
avg. state	180	270	220	264	182	180	182	270	90	176	178	155	124	173
std. dev.	0.0	0.0	5.3	6.4	5.0	0.0	4.3	0.0	0.0	4.6	3.9	5.0	7.1	6.5
measurements	21	25	43	23	17	17	33	27	26	37	13	15	19	41
exact values	180	270	218	270	180	180	180	270	90	180	180	150	122	180

Table 1: Estimates of the state of the doors depicted in Figure 5

4.2 Predictive Simulation

To demonstrate the prediction accuracy of the predictive simulation system, we performed several navigation experiments with our mobile robot Rhino. The goal of these experiments is to demonstrate the advantages of the predictive simulation system over a visualization technique based on dead reckoning given the latest state information received from the server.

In the first experiment, Rhino started in the corridor of our department and moved

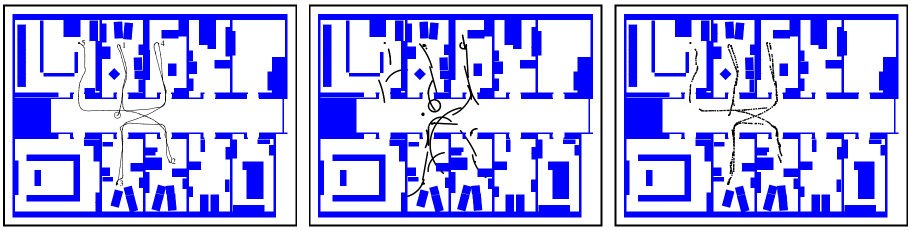


Figure 7: Trajectory of the robot (left image), trajectory computed by dead reckoning (center image), and trajectory computed by the predictive simulation technique (right image).

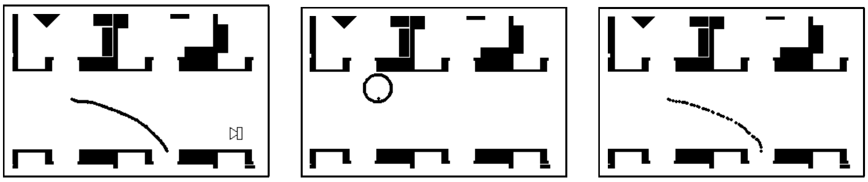
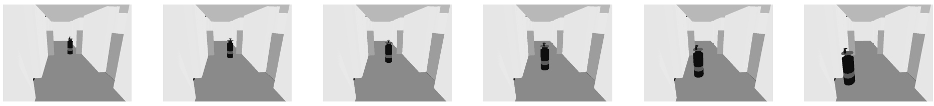


Figure 8: Trajectory of the robot taken after leaving room 1 (left), circular trajectory obtained by extrapolation (center), and trajectory extrapolated by the predictive simulation (right).

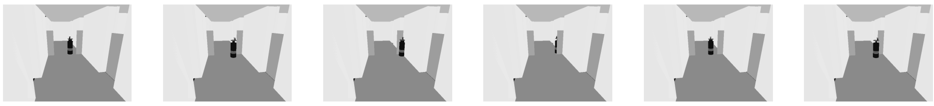
into five different offices in the order indicated by the numbers in the left image of Figure 7. The length of the trajectory is 80.8 m and the robot needed 320 seconds to complete this task.

To evaluate the capabilities of the predictive simulation system, we manually decreased the packet transmission rate to approximately 10 packets per minute. We used a constant transmission interval, so that the time gap between subsequent packets was approximately 6 seconds. Since packets sometimes get lost, the effective time-gap between subsequent packets in several cases was $6n$ seconds. The center plot of Figure 7 shows the trajectory obtained if the robot's movements are extrapolated using dead reckoning. Obviously the resulting trajectory differs significantly from the true trajectory of the robot. It furthermore contains several long gaps of more than $1m$ illustrating the limited quality of the standard dead reckoning approach. The predictive simulation system, however, provides a trajectory which is much closer to the robot's actual trajectory (see right image of Figure 7).

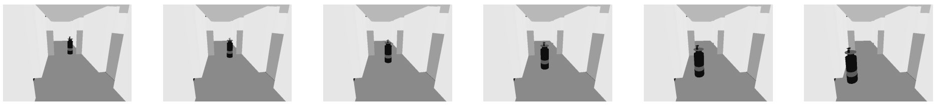
To illustrate the improvements in the visualization we now focus on a small fraction of the trajectory of the robot after it left room 1. Figure 8 shows the corresponding trajectory of the robot (left image), the trajectory obtained by extrapolation (center image), and the trajectory computed by the predictive visualization system (right image). In this particular case one packet was lost so that the system had to predict the robot's trajectory for a period of 12.5 seconds. Since the dead reckoning does not change the velocities of the robot during a transmission gap, the robot moves on a circular trajectory. In contrast, the predictive simulation system uses the path planning components



(a) Trajectory of the robot.



(c) Trajectory obtained using dead reckoning.



(b) Trajectory computed by the predictive simulation system.

Figure 9: 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 8.

to forecast the actions of the robot. Therefore, the resulting trajectory is much closer to that of the real robot.

The corresponding sequences of images of the 3D visualization are shown in Figure 9. The time delay between consecutive frames is 2 seconds. The first row shows the images obtained using the correct position of the robot. The second row contains the images computed by the dead reckoning extrapolation, and the third row includes the images obtained using the predictive visualization system. Obviously, the predictive visualization system significantly improves the quality of the visualization.

To obtain a quantitative assessment of the prediction quality, we repeated the whole experiment several times using the data shown in the left image of Figure 7. In each individual run we randomly selected the packets which were not transmitted. The left plot in Figure 10 shows the average positioning error – the distance between the predicted position and the true position of the robot – as a function of transmission gap. Here, the predictive visualization system is able to significantly reduce the average positioning error compared to dead reckoning after transmission gaps of at least 2.5 seconds. Since these differences depend on the speed of the robot, which was $34.5\,cm/s$ on average during this experiment, we additionally computed the average displacement depending on the distance traveled after the latest synchronization packet. The resulting plot is shown in the right half of Figure 10. Obviously, after at least $1m$ of travel, the predictive visualization system provides significantly better position estimates. In both figures, the error bars indicate the 95% confidence interval of the average mean.

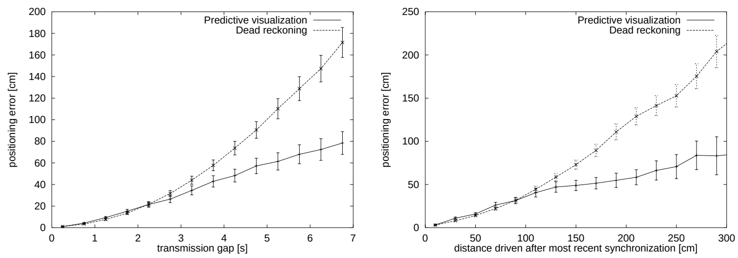


Figure 10: Average displacement depending on the length of the transmission gap (left image) and depending on the distance traveled after the last synchronization (right image).

5 Conclusions

In this paper we presented an approach for the robust 3D visualization of the actions of a mobile robot over the Internet. Instead of image streams our technique uses a 3D model of the environment and visualizes the actions of the robot and the current state of the environment by 3D rendering. According to that, only the current state of the robot and of dynamic objects within the environment have to be transferred over the Internet which results in a serious data reduction. Our approach applies a probabilistic state estimation technique to continuously determine the current states of dynamic objects. It also includes a predictive simulation component to bridge transmission gaps of several seconds, which frequently occur when the Internet is used as the communication medium.

Our approach has been implemented and tested extensively in a typical office environment. The results show that the predictive simulation provides significantly more reliable 3D visualizations than the standard dead reckoning approach which forecasts the actions of the robot based on the latest state information. We furthermore demonstrated that our state estimation procedure can robustly and efficiently estimate the state of dynamic objects in the environment of the robot. As a result, we obtain accurate visualizations of the robot and its environment even if larger transmission gaps of 10 seconds occur.

There are several aspects for future research. For example, using 3D visualizations has the advantage that a user can watch the robot from arbitrary perspectives by selecting different view-points for the virtual camera. However, choosing appropriate view-points is not at all an easy task, especially for non-expert users. The system therefore requires techniques for selecting view-points automatically and appropriate user interfaces. One approach towards this direction, is described in [12].

Since the robots operate in populated environments, we additionally would like to include techniques for detecting and tracking people in the robot's surrounding. In first experiments we found that the template matching approach presented here is not adequate for this task, especially if people are walking through the robot's environment. Moreover, since people often show up in groups they cannot be tracked independently. In this context, techniques known from multiple hypothesis tracking appear to be a more promising approach.

6 Acknowledgments

This project has been supported by the Northrhine-Westphalia Ministry for Schools, Education, Science and Research (MSWWF-NRW, Germany), under contract number IVA3-10703596. The authors furthermore would like to thank Dieter Fox, Michael Beetz, and Sebastian Thrun as well as the members of the research lab for autonomous mobile systems at the University of Bonn for fruitfull discussions.

References

[1] Bruce Bon and Seraji Homayoun. Real-time model-based obstacle detection for the NASA ranger telerobot. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, April 1997.

[2] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000. To appear.

[3] 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.

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1999.

[5] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Transactions on Robotics and Automation*, 3(3):249–265, 1987.

[6] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1999.

[7] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

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

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

[10] H. Hirukawa, T. Matsui, and H. Onda. Prototypes of teleoperation systems via a standard protocol with a standard human interface. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1997.

[11] 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.

[12] A. Hopp, D. Schulz, W. Burgard, A.B. Cremers, and D. Fellner. Virtual reality visualization of distributed tele-experiments. In *Proc. of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 1998.

[13] W.S. Kim and A.K. Bejczy. Demonstration of a high-fidelity predictive/preview display technique for telerobotic servicing in space. In *IEEE Transactions on Robotics and Automation*, volume 9, October 1993.

[14] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proc. of the Twelfth National Conference on Artificial Intelligence*, pages 979–984, 1994.

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

[16] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, Summer 1988.

[17] D. Pierce and B. Kuipers. Learning to explore and build maps. In *Proc. of the Twelfth National Conference on Artificial Intelligence*. AAAI Press, 1994.

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

[19] D. Schulz, W. Burgard, and A.B. Cremers. Robust visualization of navigation experiments with mobile robots over the internet. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1999.

[20] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A.B. Cremers. Web interfaces for mobile robots in public places. *IEEE Robotics & Automation Magazine*, 2000. To appear.

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

[22] 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.

[23] K. Taylor and J. Trevelyan. A telerobot on the World Wide Web. In *Proceedings of the 1995 National Conference of the Australian Robot Association*, 1995.

[24] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1999.

[25] 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.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1997.

[26] Y. Tsumaki and M. Uchiyama. A model-based space teleoperation system with robustness against modeling errors. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1997.