2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Automatic Regulation of the Information Flow in the Control Loops of a Web Teleoperated Robot*

J.-A. Fernández-Madrigal, C. Galindo, E. Cruz-Martín, A. Cruz-Martín, and J. González
System Engineering and Automation Department
University of Málaga, (Spain)
e-mails: {jafma,cipriano,anacm,jgonzalez}@ctima.uma.es; elenacm@isa.uma.es

Abstract— **The use of the World Wide Web for robot teleoperation is growing in the last years due mainly to the pervasiveness of internet and web browsers, although web interfaces usually use ethernet networks that exhibit time unpredictability. Most recent research in the area has been focused on improving time predictability of the network under delays, jitter, and no guaranteed bandwidth. However, we believe that: i) not only the network, but every component in the interfaced system exhibit time unpredictability; and ii) improving time predictability is not the only solution: adapting the interfaced system to unpredictable conditions is also a possibility. In this paper we consider a web interfaced robot as a set of control loops and describe and implement an hysteresis controller for regulating the flow of information through the loops as a method to satisfy the system time requirements under some unpredictable and varying conditions. For demonstrating the goodness of our algorithm, we a) compare it with a near-optimal one automatically generated through reinforcement learning, and b) show an implementation of the algorithm for the direct teleoperation of a service mobile robot, obtaining a better behavior than the same system without flow regulation.**

Keywords- Web Interfaces, Mobile Robots, Teleoperation.

I. INTRODUCTION

Networked robots that are teleoperated through the World Wide Web are increasing their popularity, mainly due to the pervasiveness of internet and web browsers. Their applications include telecare robotics ([1],[2]), museum assistants ([3],[4]), education ([5],[6]), etc.

Several topics have been addressed recently in the research literature on web teleoperation of robots: network performance ([7],[8]), virtual reality for replacing real information when it is not available at appropriate time rates ([9],[10]), internet multimedia systems [11], traditional teleoperation systems ([12],[13]), communication protocols ([14], [15]), or soft computing controllers [16].

In particular, we believe that trying to improve the predictability of the network lacks the generality needed for solving the problem. The whole interfaced system (the client-side interface, the robot, and the network) should be taken under consideration as a whole, since undesirable time effects may appear in several parts of it. For example, the time consumption of the web interface may be of a magnitude comparable to network delays, or even much greater if the network uses high-speed technology. Thus, the design of systems that adapt to unpredictable or varying time conditions seems a reasonable approach.

In a previous work [17] we have presented a probabilistic model of the control loops of a web interfaced robot that allows the user interface to select automatically, among a set of control loops with different time requirements, the one that is most likely to satisfy the timing constraints. We have called this "coarse adaptation" of the web interface, and it has demonstrated its suitability when any component in the loops changes drastically its performance. In the same work we have also considered the possibility of deactivating certain graphical components to improve the time consumption of the system before changing to a different control loop. That has been called "medium adaptation". The drawback of both actions is that they lead to abrupt changes in the modality of control of the user over the robot.

In this paper we focus in a third type of regulation: "fine adaptation", which allows us to deal with changes in time performance that are not important enough to deactivate parts of the interface or to deactivate the loop. Fine adaptation is aimed to affect as little as possible to the user control of the system and thus it should be used more frequently than coarse or medium adaptation; its goal is to automatically regulate the amount of sensory information gathered by the robot and flowing through the system to be displayed on the web interface[1]. We do this through a simple hysteresis controller [18] that reduces that information if the time requirement for the current loop is not likely to be met, and increases it when the probability of satisfying that requirement rises again. The whole probabilistic approach (coarse + medium + fine) is aimed to build web interfaces for robot teleoperation more adaptable than conventional ones, constituting a framework that is also compatible with any network-improving approach.

The suitability of the hysteresis controller that we present in this paper has been stated through two methods. We have firstly compared it to an automatically generated algorithm that is supposed to yield the best results with respect to a mathematically defined goodness measure. We have included in that goodness measure both the probability of satisfying the time requirements of the control loop and the density of sensory information shown to the user. The framewo

[1] Sensory data is by far the most bandwidth demanding among the ones that flow through a control loop; usually, actuation signals are only constituted of a few bytes.

rk for constructing the near-optimal algorithm has been reinforcement learning, and in particular Q-learning [19], since this methodology yields policies (=algorithms) that tend to the optimal ones. We have found that the Q-based algorithm performs similarly to our controller, thus providing a satisfactory justification for the use of the latter. Secondly, we have implemented our algorithm in a real web-interfaced robot and have found that the behavior of the system is better when the automatic flow regulation is activated.

The paper is organized as follows: section II describes a general web interfaced robotic system and the probabilistic models used for its time consuming parts. Section III explains our algorithm information flow regulation and how it has been compared to a near-optimal algorithm generated through Q-learning. Section IV shows the results of evaluating the proposed controller in a real web-interfaced robot. The paper ends with some conclusions and future work.

II. Control Loop Modeling in a Web-Interfaced Robotic System

We have modelled a web interfaced robotic system as shown in fig. 1. All the control loops that exist in the system are assumed to fit into the following scheme: user's actuation (on a given actuation widget[2]) generates some service requests that are transmitted through the network to the suitable modules of the software architecture of the robot; once the requests are completed, their return data plus the readings from the sensors associated to the sensory widgets (obtained from other services of the robot), are sent back to the display. For portability reasons, the client-side application is assumed to be a Java Applet [20], while the robot software architecture is assumed to be implemented upon the CORBA middleware [21].

In the described model there are several time-consuming classes of components (please refer to fig. 1):

1) *Processing Components.* These components (Translation, CORBA processing, Service Processing, and Display Processing) involve the processing of some data to yield another. The time consumption of these operations depends basically on: the size of the data, the computational complexity of the processing algorithms, and the CPU scheduling provided by the operating system (multitasking assumed). The first two sources of time complexity are well approximated by polynomial functions, since these algorithms are O(n^k), with k typically being 1 or 2. This has been modelled by uniform probability distributions in order to cope with slight variations due to conditional statements in the code or imprecisions in the measurement of time. In non-real-time environments, sporadic high time consumptions can appear due to the third source. We have modelled this by adding exponential probability distributions when needed.

2) *User Reaction Components.* Human reaction-time depends on several factors, as varied as: amount of information interpretable by the user, spatial arrangement of that information [22], rate of change in sensory data, etc. The majority of models for human reaction time in the literature are based con ex-Gaussians [23], which are the convolution of a Gaussian and an exponential distribution. For the sake of simplicity, in our work we model human reaction-time as a Gaussian probability distribution.

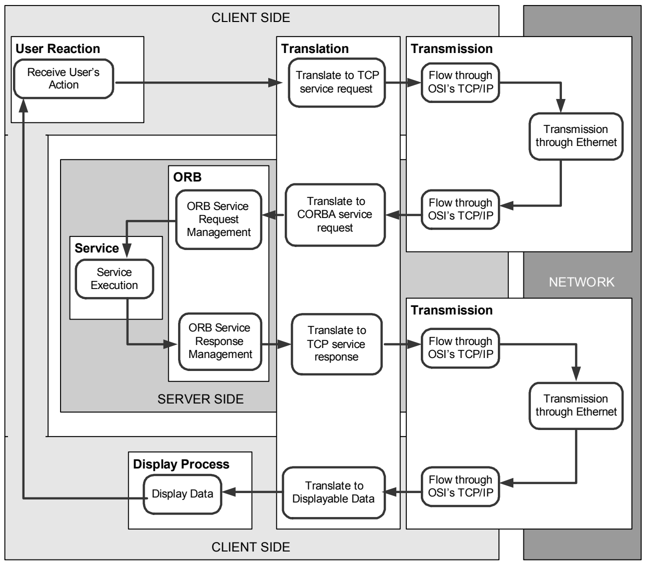

CLIENT SIDE
User Reaction
Receive User's Action
Translation
Translate to TCP service request
Transmission
Flow through OSI's TCP/IP
Transmission through Ethernet
Flow through OSI's TCP/IP
Translate to CORBA service request
ORB
ORB Service Request Management
Service
Service Execution
ORB Service Response Management
Translate to TCP service response
NETWORK
Transmission
Flow through OSI's TCP/IP
Transmission through Ethernet
Flow through OSI's TCP/IP
SERVER SIDE
Display Process
Display Data
Translate to Displayable Data
CLIENT SIDE

Fig. 1. Proposed general scheme for a control loop in a web interfaced robotic system. All the time-consumption steps are indicated, assuming a CORBA middleware for the robot software architecture.

3) *Network Transmission Components.* This class includes components of the physical network, queuing buffers, and the OSI protocol processes. In the literature it has been stated that the arrival time of ethernet communications tends to a Poisson process as long as the network is fast enough, thus the interarrival time can be modelled as an exponential distribution [24]. Experimentally, it has been shown that in cases where the network is slower it is more appropriate a beta distribution [25].

III. Flow Regulation of Sensory Information

The regulation action considered in this paper is based on reducing/increasing the amount of sensory information transmitted to the web interface (for example, the number of pixels of a camera image or the number of samples of a laser range scanner). It is intuitive that degrading the amount of sensory information provided to the user should degrade the overall performance of the system gracefully (that is, the capability of controlling the system by the user should be maintained), although the control should be better when more data is available. Fig. 2 illustrates this intuition through the results of some experiences we have conducted. In such experiments, people control remotely the movement of a simulated mobile robot for following a circular corridor along its middle line. The only sensor available is a laser scanner that provides range measurements. The shape of the corridor has been chosen to force the user to actuate continuously. The figures show how the average deviation from the desired path increases when less sensory information is available, and then the user reaction time is

[2] A widget is a component in a graphical interface (buttons, panels, etc).

smaller since the user must react faster to unexpected situations (when more sensory data is available, the user spends more time planning better actions). In summary, our experiments show that an acceptable control is still possible when sensory data is reduced.

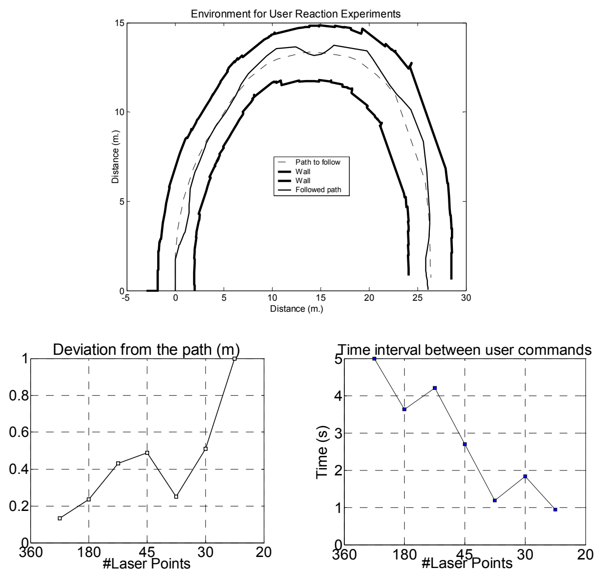


Fig. 2. Up) Simulated environment where the user drives remotely a mobile robot receiving only data from a laser scanner. Bottom-Left) Average deviation from the desired path under different densities of sensory information. Bottom-Right) Time between user commands (speed/orientation changes) for the same densities. The web interface used is the same as in experiments in section IV. The simulated robot runs in a remote computer. Communications are via ethernet twisted-pair.

Next we describe a hysteresis controller that allows the interfaced system to regulate automatically the amount of sensory information that is shown to the user (and therefore, the information flows through the system) in order to adapt to varying and unpredictable time conditions. Also, we present a method for constructing near-optimal regulation algorithms with respect to a given goodness measure, and this is used to justify the suitability of the former.

In both the hysteresis controller and the near-optimal algorithm, we assume that the client-side interface of the system displays a finite set of sensory widgets associated to the current control loop, let say $W=\{w_i\}$. Each widget w_i has a finite set of possible density states $d(w_i)=\{d_{ij}\}$, with each density state d_{ij} indicating a given amount of data $a(d_{ij})$ that the widget shows to the user when it is in that state.

A. Hysteresis Controller Algorithm

A pseudocode for the hysteresis controller appears in fig. 3 (experimental results of its implementation are given in section IV). The algorithm reduces gradually the amount of information associated to the sensory widgets until: a) the probability of the loop to satisfy its time requirements falls under a given "critical threshold" (then medium or coarse adaptations must be done), or b) that probability rises over a given "safety threshold" (then the loop is satisfying its time requirements comfortably). When b) occurs, the sensory widgets that did not show all the information that they could, recover their densities gradually. When the probability lies between both thresholds, the sensory widgets reduce their densities in an orderly fashion. The critical and safety thresholds and the loop time requirement must be specified by the user or the programmer.

```
O <- list of sensory widgets ordered by decreasing density
O' <- equal to O, but in increasing order of density
I <- 1
Do
  P <- probability of satisfying time requirement of the loop
  If (P < critical threshold)
    Do Medium and Coarse adaptations.
  Else if (P < safety threshold)
    If (widget O(I) density can be decreased)
     Set current density state of O(I) to its next lower state.
    Else
      If (I < number of widgets)
          I <- I+1 /* next widget */
      Endif
    Endif
  Else /* P >= safety threshold */
    If (widget O'(I) can increase its density)
      Set density state of O'(I) to the next higher state.
    Else
      If (I < number of widgets)
          I <- I+1 /* next widget */
      Endif
    Endif
  Endif
Enddo
```

Fig. 3. Pseudocode of the hysteresis controller that regulates the amount of flow information in a control loop of the web interfaced system.

B. Near-Optimal Algorithm

We propose now a method for constructing near-optimal algorithms for sensory flow regulation automatically. This method is too slow to use it at run-time, but our goal is rather to calculate, by comparison, the optimality of the algorithm presented in the previous section.

We have chosen reinforcement learning (RL) [19] as the framework for the near-optimal algorithm. RL can be used for learning the optimal policy (= sequence of actions) to perform by an agent in a complex scenario. At any moment of the RL process, the agent (the web interfaced system in our case) is in a state *s* (set of widget densities and current probability of satisfying the control loop time requirement) and decides to execute some action *a* (changing the density of some widget), turning its state into state *s'* and getting a reinforcement signal or reward *r* for its decision (which sets the optimality measure of its behaviour). These experience tuples (*s*,*a*,*s'*,*r*) are used for finding a policy π that maximizes the long-term reward. It is straightforward to interpret the policy as an algorithm, as we will do here.

There are several methods for solving RL problems. We have selected Q-learning since it does not need the probabilistic model of the agent's environment. In spite of that, Q-learning yields policies that tend to the optimal ones. It uses the following value function to resume the learning procedure, which can be recursively computed:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \quad \textbf{(1)}$$

where α is the *learning rate* (it must decrease slowly [3]), γ is the *discount factor* (which represents the importance of future rewards), and *Q(s',a')* refers to the Q-values of the

[3] We have chosen $\alpha(i)=1/i^c$, where *i* is the current iteration index of the Q-learning algorithm and *c* is a constant that we have calculated for α to equal 0.05 in the last iteration. This function is demonstrated to produce a good convergence rate [27].

next state *s'* for any action *a'*. The final *Q* is the best policy under the agent experience, and therefore, the best known algorithm to follow when the agent is confronted with the same scenario again. The *Q* function is a matrix of *(number of states x number of actions)*. For a given state, the action for which *Q* is maximal in that state is the best decision according to the policy. Therefore the obtained *Q* can be used as an algorithm by running the procedure shown in fig. 4.

```
Q <- near-optimal values produced by Q-learning
Do
  If (P < critical threshold)
    Do Medium or Coarse adaptations.
  Else
    S <- current state of the system
    A <- A' for which Q(S,A') is maximum
        (if several maxima, at random)
    Do A (change one of the widgets densities)
  Endif
Enddo
```

Fig. 4. Pseudocode that interprets a learned *Q* as an algorithm for regulation of information flow.

C. *Optimality of the Intuitive Algorithm*

Now we compare the hysteresys controller to the Q-learning algorithm. The goal is to provide a scientific justification for the optimality of the former.

We have based the experiments on the real interface for controlling an assistant robot that is described in section IV. The network is a mixture of twisted-pair 100 Mbps segments and a wireless 802.11g segment. However, for carrying out a sufficiently large number of learning steps and rich comparisons (which would not be possible using the real application), we have employed the probabilistic models described in section II for simulating the whole interfaced system. We have gathered time measurements of the different components of the real interfaced system and entered them into those models (see table 1), thus obtaining very realistic simulations.

TABLE 1

PROBABILISTIC MODELING OF SOME OF THE COMPONENTS OF A WEB-INTERFACED ROBOTIC SYSTEM, CONSIDERING PROCESSING OF 1 BYTE OF DATA

Component	**Model**	**Parameters**
User reaction	Gaussian	μ=100, σ=50
Network transmission	Exponential	λ=268
ORB processing	Uniform+Exp.	a=-0.7, b=0.7, λ=0.5

The interfaced system has a control loop which allows the teleoperator to drive the robot by sending direct motion commands (speed/direction). The loop has one sensory widget that displays the readings of the robot laser, which has a range of 180º, and a low-resolution image captured by a camera mounted on the top of the robot.

We have set the laser widget densities to four possible values: widget off, and displaying 90, 180, and 360 range points. We have used three densities for the camera widget: camera OFF, camera in black & white, and camera in colour. Fig. 5 shows the cumulative probability distribution function of time consumption in the control loop with the camera widget set permanently to each of its densities. We have included a time overload in the system (for simulating unpredictable delays) modelled by a Gaussian probability distribution function with μ=600 ms and σ=200 ms. Notice that the effect of the camera is evident. In fact, changes in the laser widget density are not shown in the figure since they are very close to the camera widget main curves.

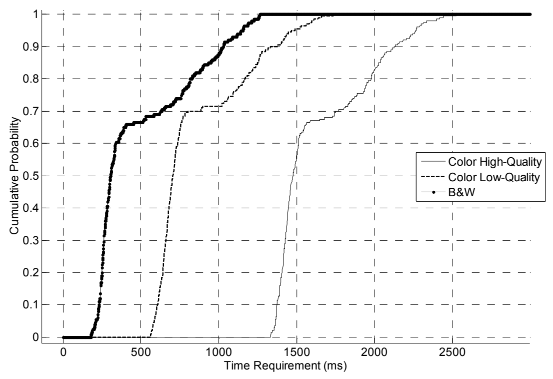


Fig. 5. Cumulative probability distribution functions of the time consumption of the control loop for different camera densities. The horizontal axis indicates a time requirement for the control loop, while the vertical axis gives, for that requirement, the probability of closing the loop in an equal or shorter time.

For this setting, we have discretized the state space of the system into 48 states that are the combination of 4 probability ranges of satisfying the control loop time requirement, of 4 laser densities, and of 3 camera densities. We have established 7 possible actions (to set one of the widgets to one of its densities). If we define *p(s)* as the integer discretization of the probability of closing the loop under the time requirement (1-> 0-50%, 2-> 51-75%, 3-> 76-85%, 4-> 86-100%), *d(s)* as the density of the laser widget (1, 2, 3, or 4), and *c(s)* as the camera widget density (1, 2, or 3), the reward obtained from a given selection of widget densities and probability to satisfy loop requirements can be calculated as[4]:

$$r(s)=\begin{cases} 0 \text{ if } p(s)=1 \vee \left(c(s)=1 \wedge d(s)=1\right) \\ 10\left(0.6\frac{p(s)-1}{3}+0.1\frac{d(s)-1}{3}+0.3\frac{c(s)-1}{2}\right) \textit{otherwise} \end{cases}$$

For learning the Q matrix we have executed repeatedly equation (1) with $\alpha(1)$=1 and γ=0.9, during 10000 iterations with a desired time requirement for the loop of 1450 ms. Table 2 shows the portion of the learned Q that contains the most visited states (those that have been sufficiently explored during learning).

We have then compared both algorithms (the Q-based one vs. the hysteresis controller), measuring their respective cumulative rewards over time (= their optimalities). For that, we have launched both for 100 iterations, each one being the closing of the control loop for 40 times (which supposes about 10 to 40 seconds of real time execution; summing a total comparison time of about 1000-4000 seconds). Fig. 6-Left shows the total reward collected. The average

[4] Generally, in Q-learning the reward is a function of both the current state and the selected action, which is useful in the case that carrying out actions has some cost. In this paper, the reward only depends on the current state since we do not distinguish among the costs of carrying out different actions (we consider them all to be null).

optimalities are 301.7 (Q) and 280.6 (hyst.), with standard deviations of 17.7 and 21.2 respectively, which makes them indistinguishable, showing that our intuitive approach is close to the optimal. Fig. 6-Right shows a similar result for the same web-interfaced system but with only one widget, the laser, and a time requirement of 250 ms. We have obtained an average reward for Q of 12.2 with a standard deviation of 6.4, and of 5.5 with a standard deviation of 4.3 for the hysteresis controller.

TABLE 2

LEARNED Q (ONLY SUFFICIENTLY EXPLORED STATES)

State	Best Action according to Q
6 (prob<51%, laser 90, cam. Color)	Camera B&W
9 (prob<51%, laser 180, cam. Color)	Camera B&W
12 (prob<51%, laser 360, cam. Color)	Camera OFF
15 (prob 51-75%, laser OFF, cam. Color)	Laser 90
18 (prob 51-75%, laser 90, cam. Color)	Camera B&W
21 (prob 51-75%, laser 180, cam. Color)	Camera OFF
35 (prob 76-85%, laser 360, cam. Color)	Laser 360
37 (prob>85%, laser OFF, cam. OFF)	Laser 90
38 (prob>85%, laser OFF, cam. B&W)	Laser 360
40 (prob>85%, laser 90, cam. OFF)	Laser 180
41 (prob>85%, laser 90, cam. B&W)	Laser 360
43 (prob>85%, laser 180, cam. OFF)	Camera B&W
44 (prob>85%, laser 180, cam. B&W)	Laser 90
46 (prob>85%, laser 360, cam. OFF)	Camera B&W
47 (prob>85%, laser 360, cam. B&W)	Laser 360

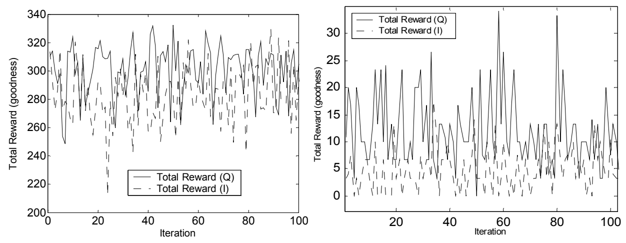
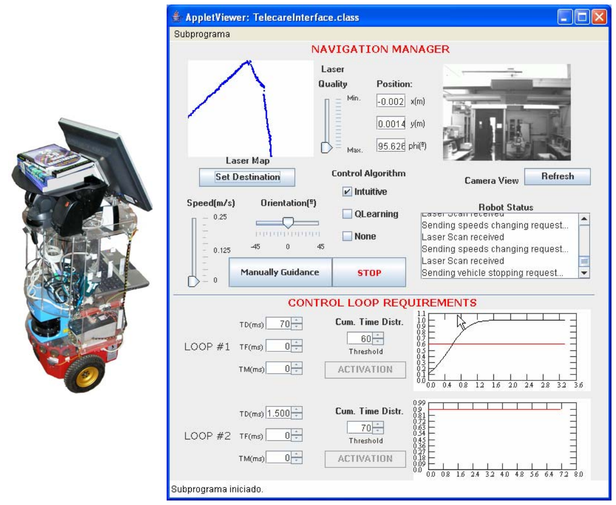


Fig. 6. Total rewards of both the Q-learned algorithm (solid line) and the hysteresis controller (dotted line) for left) a web-interfaced system composed of two sensory widgets, and right) the same system with only one sensory widget.

IV. REAL IMPLEMENTATION

Once we have shown the near-optimality of our controller for regulating the information flow of the control loops, we have implemented it in the real interface of fig. 7 and evaluated it under the control loop with one sensory widget (the laser). The real robot is a service robot called SANCHO intended for pick-and-delivery, museum guiding, or fair hosting.

The experiment has consisted in driving the robot remotely from a given location to another through simple speed/direction commands. The user reaction time has not been considered in this experiment (only the time from the user action to the displaying of sensory data), since people do not always do control (for example, they do not act when the robot is already in the right direction and speed). A user waiting for the next control action would increase the loop time, which would be considered by the system as a problem for satisfying the requirement, which is not necessary.



Position: -0.002 x(m), 0.0014 y(m), 95.626 phi(º)

Control Algorithm: [✓ Intuitive] QLearning None

Robot Status:
```
Sending speeds changing request...
Laser Scan received
Sending speeds changing request...
Laser Scan received
Sending vehicle stopping request...
```

LOOP #1: TD(ms) 70, TF(ms) 0, TM(ms) 0, Threshold 60

LOOP #2: TD(ms) 1.500, TF(ms) 0, TM(ms) 0, Threshold 70

Subprograma iniciado.

Fig. 7. On the left, the SANCHO robot we have used for implementing a real web-controlled system. On the right, the client-side web interface. The robot is based on a Pioneer 3DX mobile platform enhanced with an on-board computer, wi-fi connectivity, and several sensors (laser, sonar, camera, infrared).

We have measured the final cumulative probability distribution functions under two situations: i) when the hysteresis controller is activated, and ii) without using regulation flow algorithms (that is, setting the laser density to a fixed number of sample points), obtaining the results shown in fig. 8-Up. This figure is the result of 75 passes of the control loop. Also, we have logged the densities of the laser widget that the controller has set during the experiment (fig. 8-Bottom). As shown in the figure, the controller has a good probability of satisfying the time requirement for the control loop (75 ms) similar to that of setting the density of the widget to a small number points. However, this good time satisfaction has been achieved with laser densities that are, most of the time, very high (360 points). Notice that the laser widget is disabled during short periods of time when the time conditions were hard. If these periods exceed some predetermined duration, the other regulation actions (coarse and medium) take place as explained in [17].

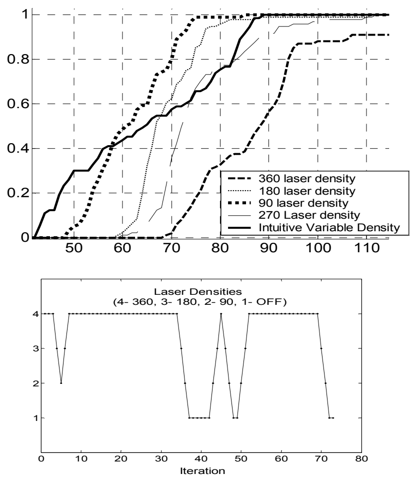


Fig. 8. Up) Cumulative probability distributions for the real experiments under different sensory information flows. Bottom) Densities of the sensory widget during the use of the hysteresis controller. Each density value is logged after one pass of the control loop.

In [26] you can find a multimedia file that shows the overall regulated-flow teleoperation of the robot and a coarse adaptation action due to critical threshold underpassing, in which a second control loop consisting in navigating reactively to a manually-set goal location is activated.

V. CONCLUSIONS AND FUTURE WORK

In this paper we propose a new way of approaching the typical unpredictability and time variation problems of web networked robots that is of greater generality than improving predictability and time efficiency of network transmissions (the most common line of research in this field in literature).

In particular, we have focused on designing an algorithm that automatically regulates the optimal amount of information that flows through the system. The algorithm is a simple hysteresis controller, and we have developed a framework for generating automatically near-optimal algorithms for the same regulation task, through Q-learning, in order to evaluate its optimality. We have developed several experiments based on a real interfaced robot that show that our algorithm is as optimal as the near-optimal one generated by Q-learning. Also, a real implementation showing the suitability of our approach has been described.

In the future, we plan to implement more web applications for assistant and service robots, in order to test our approach under different conditions. Also, the use of the Q-learned algorithm in real-time will be explored.

REFERENCES

[1] Jia S., Hada Y., Ye G., and Takase K. Distributed Telecare Robotic Systems Using CORBA as a Communication Architecture. IEEE Intl. Conf. on Robotics & Automation, 2002, pp. 1659 – 1664.

[2] Camarinha-Matos L.M., and Afsarmanesh H. Tele-Care and Collaborative Virtual Communities in Elderly Care. Int. Workshop on Tele-Care and Collaborative Virtual Communities in Elderly Care, TELECARE 2004, pp, 1-12. Porto, Portugal, April 2004.

[3] Thrun S., Beetz M., Bennewitz M., Burgard W., Cremers A., Dellaert F., Fox D., Hahnel D., Rosenberg C., Roy N., Schulte J., and Schulz D. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. Journal of Robotics Research, 19(11), pp. 972-999, 2000.

[4] Goldberg S. and Bekey A. Online Robots and the Remote Museum Experience. In Beyond Webcams: An Introduction to Online Robots. MIT Press, 2002, pp. 295 - 305.

[5] Lixiang Y., Pui T., Quan Z., and Huosheng H. A Web-based Telerobotic System for Research and Education at Essex, 2001 IEEWASME International Conference on Advanced Intelligent Mechatronics, 8-12 July 2001 Como, Italy, pp.284-289.

[6] Safaric R., Sinjur S., Zalik B., and Parkin R.M. Control of Robot Arm with Virtual Environment Via the Internet. Proc. of the IEEE, 91(3), March 2003 pp. 422 – 429.

[7] Liu P.X., Meng M.Q-H., Gu J., Yang S.X., and Hu C. Control and Data Transmission for Internet Robotics, IEEE Intl. Conf. on Robotics & Automation, Taipei, Taiwan., 2003, pp. 1659-1664.

[8] Oboe R. Web-interfaced, force-reflecting teleoperation systems. IEEE Trans. On Indusltrial Electronics, 48 (6), 2001, p.p. 1257-1265.

[9] Belousov I.R., Chellali R., and Clapworthy G.J. Virtual reality tools for Internet Robotics.ICRA 2001, May 21-26, Seoul, Korea, p.p. 1878-1883.

[10] Jiacheng T. and Clapworthy G.J. Virtual environments for Internet-based robots. I. Modeling a dynamic environment. Proc. of the IEEE. Vol.91, Issue 3, March 2003 pp. 383 - 388.

[11] Wang X. and Schulzrinne H. Comparison of Adaptive Internet Multimedia Applications. IEICE Transactions on Communications, Vol. E82-B, No. 6, June 1999.

[12] Andreu D., Fraisse P., Roqueta V. and Zapata R. Internet enhanced teleoperation: toward a remote supervised delay regulator. 2003 IEEE International Conference on Industrial Technology, Vol. 2, pp. 663-688.

[13] Imer O. C., Yüksel S., and Basar T. Optimal control of LTI systems over unrealiable communication links. Automatica 42 (2006), 1429-1439.

[14] Liu X. P., Meng M. Q.-H., andYang S. X. Data Communications for Internet Robots. Autonomous Robots 15, 213-223, 2003.

[15] Fiorini P. and Oboe R. Internet-Based Telerobotics: Problems and Approaches. International Conference on Advanced Robotics (ICAR'97), pp. 765-770, Monterey, CA, USA, July.

[16] Lin W. K. W., Wong A. K. Y., and Dillon T. S. Application of Soft Computing Techniques to Adaptive User Buffer Overflow Control on the Internet. IEEE Transactions on Systems, Man and Cybernetics-Part C: Applications and Reviews, Vol. 36, No. 3, May 2006.

[17] Fernández-Madrigal J.A., Cruz-Martín E., Cruz-Martín A., González J., and Galindo C. Adaptable Web Interfaces for Networked Robots. IROS'2005, Edmonton (Canada), 2-6 August 2005.

[18] C.H. Phillips and R.D. Harbor, Feedback Control Systems, Prentice Hall, 2000.

[19] Kaelbling L.P., Littman M.L., and Moore A.W. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4 (1996) 237–277.

[20] Schildt H. JAVA 2:The Complete Reference, McGraw-Hill, 2002.

[21] Henning M. and Vinovski S. Advanced CORBA Programming with C++, Addison-Wesley, 1999.

[22] Fitts P. M. The information capacity of the human motor system in controlling the amplitude of movement, Journal of Experimental Psychology vol. 47, pp. 381-391, 1954.

[23] Zaitsev A. V. and Skorik Y. A. Mathematical Description of Sensorimotor Reaction Time Distribution Human Physiology, Vol. 28, No. 4, 2002, pp. 494-496.

[24] Cao J., Cleveland W.S., Lin D., and Sun D.X. Internet Traffic Tends Toward Poisson and Independent as the Load Increases, in NonLinear Estimation and Classification, Holmes et al. eds., Springer, New York, 2002.

[25] Kobayashi H. Modeling And Analysis. An Introduction to System Performance Evaluation Methodology, Addison-Wesley, 1978.

[26] http://www.isa.uma.es/personal/jafma/experiments.htm

[27] Even-Dar E. and Mansour Y. Learning Rates for Q-learning, Journal of Machine Learning Research, vol 5, pp. 1-25, 2003.