Towards virtual control of mobile manipulators

Arnaud Lelevé, Philippe Fraisse, André Crosnier, Pierre Dauchez, François Pierrot

TOWARDS VIRTUAL CONTROL OF MOBILE MANIPULATORS

A.LELEVE, P.FRAISSE, A.CROSNIER, P.DAUCHEZ, F.PIERROT
LIRMM UMR 5506 CNRS / Université Montpellier II
161 rue Ada 34392 Montpellier Cedex 5 - France -

ABSTRACT

This paper introduces the realisation of a setup platform for the study of teleoperation through *Internet*. The platform is a vehicle including a PUMA manipulator. The communication uses an Ethernet radio network, a PC laptop located on the vehicle and a Silicon Graphics Indigo for the base man-machine interface. This platform will permit us to study control laws to cope with variable delays.

KEYWORDS : Teleoperation, Internet.

INTRODUCTION

The teleoperation of any remote system raises at one and the same time some difficulties about technics, logistic and cost. Indeed, the long distance command of a remote system requires the use of different media of communication. (computer networks, satellites, relay broadcasting stations, ...) that may introduce variable delays in the command loop that may be then instable.

A first approach [1] consists in designing two overlapped loops. The first one, restricted to the vehicle, locally controls it to fit the reference values sent by the base and it resolves critical situations as obstacle avoidance for instance. The second loop, which dynamics are slower than the first one, includes the first loop and the base. This one will be the victim of probable delays. In this way, the dialogue between the base and the vehicle consists in sending (only when the operator modifies the configuration of the remote system) desired state values.

However, the operator usually has a vision of the operation site limited in terms of quality by the bandwidth of the transmission channel. He may be quickly disturbed by the presence of consequent (on and after one second) and varying delays. It is possible [2] to bypass to a great extent, these transmission delays, by graphically simulating (at least in 2D), in advance, the desired action ; yet, even if we know the model of the manipulator, we have to cope with the interaction with an unstructured environment. The superposition of the simulated system with a picture from a CCD camera can help in a great extent the operator's task.

A third approach to the issue consists in taking the variable delays into account and in directly controlling the teleopered system. This involves a structure bringing a dynamic simulation of the remote system and a predictor of the current and future state of it [3].

Our study aims to teleoperate a land vehicle including a manipulator. We have taken the different observations and conclusions of the articles [1], [2] and [3] into account in order to work out, at first, a control strategy that we will comment out in the next chapter and next, to create a setup platform combining a maximum number of the advantages out of these previous three techniques. Then, we will precisely describe this platform and we will eventually comment the setups we have made as yet.

GLOBAL CONTROL ARCHITECTURE

For the moment, we have implemented on the base computer a virtual environment set up with the kinematics models of the vehicle, including its manipulator and its environment. The operator interface consists in a 3D perspective representation (on a Silicon Graphics Indigo), of the vehicle simulated in a structured environment. The operator handles a 6 degree-of-freedom mouse (« a space mouse ») to move either the vehicle or the manipulator. The movements are transmitted to the client which sends them, as orders, to the server through the local network. This client also receives some information about the vehicle (positions, speeds). Figure 1a represents this structure.

In the other hand, the mobile manipulator is fit out with encoders of the manipulator, on the steering wheel and on one wheel (we assume it doesn't skid). Its server transmits the base orders to the controller and sends data from it to the base client. It also signals to the controller whether it has lost contact with the base. The controller locally controls (position or speed) the system in accordance to the desired values sent by the server. When the client signals it a communication rupture, the controller stops the vehicle.

The drawbacks of this simple architecture are that we are victims of transmission delays through the network. Still, we can now measure them and use it as a first approach to a more complete system.

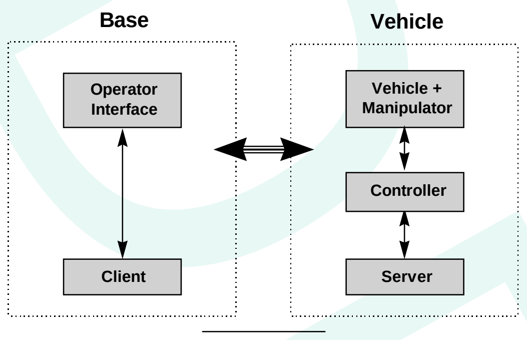

Figure 1a. Current architecture

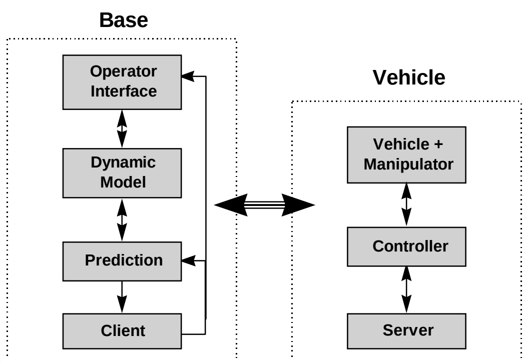

Figure 1b. Future architecture

Future architecture

As in [3], our control will involve a dynamic model of the system (manipulator + vehicle) and a predictive algorithm, so that we ought to cope with statistical delays. The use of the dynamic model, coupled to the prediction block, will allow us, in one hand, to simulate on a computer screen the effects of the orders sent by the operator and, in the other hand, to anticipate the vehicle movements according to the estimated delay (the last delays are measured by the client) of the transmission channel so that the final result will be the one wanted by the operator. The data got from the vehicle will permit to adjust the estimated state of the remote system. In a first approach, the prediction block will simply consist in a *Kalman* filter. We can consider using neuronic networks afterwards.

If needed, we will be able to simulate varying delays with a virtual queue in order to study the behaviour of our control law versus strong delays.

Moreover, the information from the vehicle can be improved by the adjunction of different transducers as accelerometers, magnetic compass, effort transducers, a CCD camera set on the wrist of the manipulator, a laser telemeter and an ultrasound detector in front of the vehicle. The simulator will then be more accurate since it receives richer information and the operator will be able to view simultaneously both the simulation and the reality on his screen.

At this time, when the server signals a communication rupture, the vehicle controller just engages the brakes. An interesting strategy on a loss of contact, may consist in executing a simple procedure that the operator had supplied before starting moving (for instance, « make a U turn », « go 10 m backward », ...). In the future, the controller will also be able to anticipate a collision thanks to the additional transducers (telemeter, ultrasound barrier, CCD camera, ...) and it will be eventually able to manoeuvre as to avoid the obstacles.

THE TELEOPERED VEHICLE

Originally, the vehicle is an electric one dedicated to many tasks on golf grass. It is manufactured by *Andruet S.A.*. It has 6 directing and propulsive wheels It is aimed at some studies in the laboratory (as the generation of coordinated movements for a manipulator on a vehicle, obstacle avoidance with a laser telemeter coupled to a CCD camera, ...)

We have added an automatic mode that makes the wheel converter supervised by the dS*pace* block. We have also added an optic encoder on the left front wheel, that delivers its position to the *DSP*. Concerning the steering, we have coupled a motor and an optic encoder to the steering axle. The motor converter is supervised by the dS*pace* block and the encoder is plugged to the same block. We have modified the wiring of the drive (stop, forward or reverse) so that we can remotely change it.

We have also mounted, at the rear of the vehicle, a platform holding a 6 degrees of freedom manipulator PUMA 560. It is now controlled by a structure we have assembled by ourselves :

- 6 converters for the motors of the Puma,
- a *dSpace® DS1003* block that includes a *TMS320C40 DSP* by Texas Instruments® and analogic in/out cards.

The whole is supervised by the *DSP* in the d*Space* block. A program simultaneously acts (with a PID corrector), in an infinite loop, on every state variable of the system : the position of the 6 axles of the manipulator, the speed or the position of the vehicle and its steering. A second order lowpass filter smoothes the movements by filtering the desired state values of the system.

A *PC-486-DX* laptop is linked to the d*Space* block. It uploads the program to be executed in the *DSP* and it can download some real-time measured values. This *PC* is also including an *Ethernet* radio network card. This card communicates with a transmitter-receiver linked to the *Ethernet* network of the laboratory. Once the card is correctly set up, the radio link is transparent for the rest of the computer and the user. Figure 2 represents the whole vehicle.

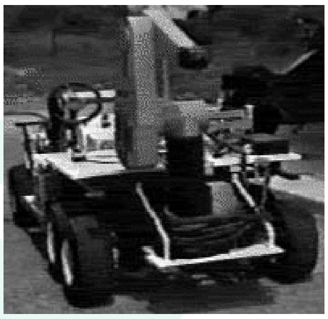
Figure 2. Picture of the vehicle and its manipulator

Figure 3 shows the interactions between the diverse control elements of the whole system.

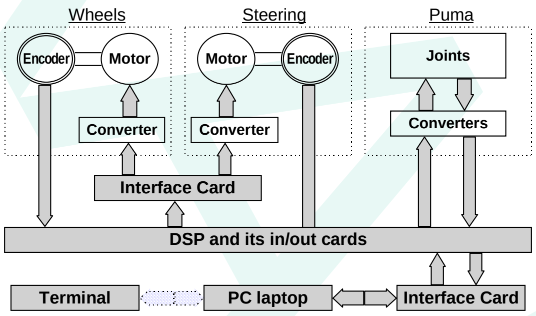

Figure 3 The set of control blocks for the teleoperation

THE TELEOPERATION

Our teleoperation chain consists of :

- a fixed base : a Silicon Graphics® Indigo computer, which a simulation and control software is executed on. It simulates the vehicle in a predefined environment (for the moment, the parking of the laboratory). This software implements a network communication interface (the client).
- the Ethernet network of the laboratory, and its equipment : hubs, gateways, ...
- the radio transmitter-receiver that interfaces the laboratory network with our radio network,
- our radio network that can handle several computer hosts, relays and Ethernet interfaces.
- and the PC laptop located on the vehicle which. a (server) software handles the communication between the base and the DSP on.

As the radio aspect of the communication is transparent to the programmer, the task has consisted in making two softwares communicate over a network, these two softwares being executed in a different environment..

The protocols

Our goal was to control the vehicle through the Internet, so we have been constrained to use a protocol in the *TCP/IP* family (*TCP* or *UDP*). [4] and [5] comment out these *Internet* protocols and the ways to make two softwares communicate across a network. We will note that UDP, unlike TCP, isn't a reliable protocol because one hasn't any mean to be sure that all the sent data has been correctly received by the addressee. Given the nature of our application, it is important that we use a reliable protocol. We have, naturally chosen the TCP protocol.

The client-server model goes on a par with the *TCP* protocol. In our case, the server is the software running on the PC laptop of the vehicle. It provides the ability to drive the vehicle and its manipulator. Our client is the part of the simulation and control software that manages the sending of the orders and the reception of the data from the vehicle.

The server

Our server is iterative : a single client can teleoperate the vehicle at any time. On the other hand, several clients can teleoperate it, each in turn. It is coded for Windows 3.11 and we have included some items that allow a pilote to drive the vehicle by means of a mouse and the keyboard of his laptop.

If there is a loss of communication, the *TCP* protocol doesn't detect the breakdown in every case. So we have added a watchdog for this purpose. When no orders need to be sent to the server (the operator doesn't gave any order), the client sends, all the same, a message which goal is to verify that the communication isn't broken. In the future, it will also be used to evaluate the delays of transmission between the client and the server. Whenever the server detects no message after an arbitrary time, it stops the vehicle in emergency.

The data

For compatibility reasons, the orders are sent as strings of characters. The drawback is that these strings may have variable lengths and the addressee doesn't know in advance the length of the string it is supposed to get. Moreover, a string is likely to be divided in small packets when the network is very busy.

We have programmed two functions « *sendString* » and « *receiveString* » which respectively manage the sending and the reception of the strings in the format of figure 4a. The checksum is redundant with the *CRC* check operated by the *TCP* layer, but it doesn't use much time to compute and it makes the whole system a little more reliable. A sample of communication is given in figure 4b.

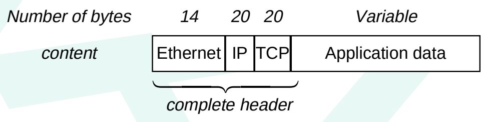

Figure 4a. Format of a data frame

Base ⇒ Vehicle	Header	Length	Checksum	MG 0.0 0.0 -2.0 5.0 -7.11 -3.0 0.5 -2.0 3.0	
Vehicle ⇒ Base	Header	Length	Checksum	Ack	Ok !
Base ⇒ Vehicle	Header	Length	Checksum	AR	
Vehicle ⇒ Base	Header	Length	Checksum	Ack	Reset ... Ok !
Base ⇒ Vehicle	Header	Length	Checksum	LMJFJ F JDSZER	
Vhicle ⇒ Base	Header	Length	Checksum	Ack	Unknown order !

Figure 4b. Sample of communication between the server and the client

The dialogues

The communication takes place in a mode called « handshake » : every order from the client is syntactically analysed by the server which then sends back a reception acknowledgement to the client. For the moment, the syntax is very simple ; common orders are AR (Arm Reset), ST (stop), GA (go ahead), GR (go reverse), GM (global movement) that is followed with 6 increments of position for the manipulator, and by the increment of speed and steering for the vehicle.

SETUPS

Vehicle setups

We have observed the behaviour of the vehicle to some steps of speed (figures 5a). The output voltage of the converter is displayed in figure 5c. The gray curb represents the speed wanted by the operator. The one in dotted lines shows the speed référence after the lowpass filter.

For reasons of security, we have electronically limited the input voltage of the wheel converter to 5V max. This explains that the real speed doesn't fit to the desired speed while the voltage is greater than 5V.

Figure 5b represents the response of the steering. The shape of the voltage applied to the steering converter (figure 5d) tells us that the gain is a little too high. In every case, we can remark a small vertical translation ; it is obviously due to an offset of some CNAs.

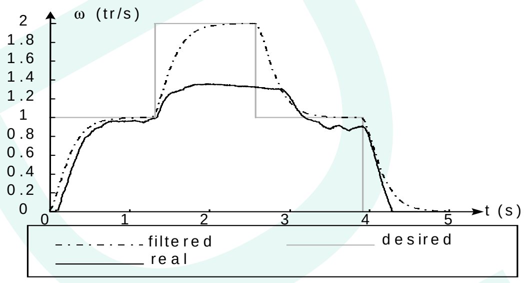

Figure 5a. Speed response

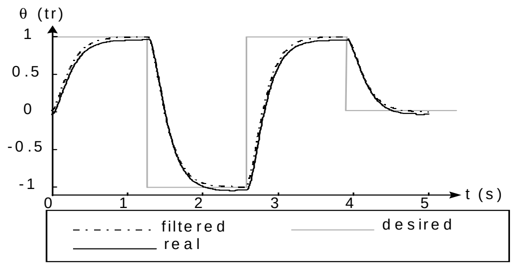

Figure 5b. Steering response

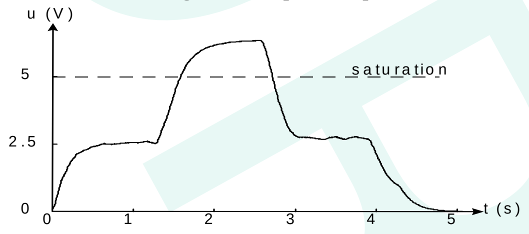

Figure 5c. Voltage for the wheel converter

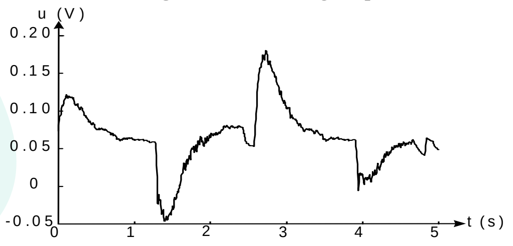

Figure 5d. Voltage for the steering converter

Some other curbs have allowed us to measure the total time of passage from the stopped state to the forward moving state. Its mean is equal to 1,6s. this number represents the time spent between the moment when the server has received the order to go forward and the moment when it has begun to move.

Frame capture

We have captured Ethernet frames exchanged between both the machines PC and Indigo. The connection consists in 3 frames that tell each other its length of buffer and initiate the communication.

The dialogue makes clearly appear the principle of acknowledgement of the TCP protocol ; every message is followed back by a frame without data, dedicated to the validation of the last one. We have noted a different behaviour in the way of sending messages, between the PC and the Indigo. In fact, the *Indigo* sends a message in one piece (as in figure 4a) whereas the PC makes 2 or 3 attempts to send data frames (the 2 first attempts contain truncated data). We suppose that the *Indigo* handles a heavy network traffic (NFS, mount, rlogin, ...) and the unfruitful attempts of the PC client may be due to collisions. In the other way, the PC is just dedicated to this application ; so it doesn't send nor receive any network traffic during a teleoperation. That would explain why every attempt from the *Indigo* is successful.

Timings

Figure 7 represents the distribution of two types of messages sent by the base software during one minute. We can note that, when there is no sending of orders, the sending of test messages isn't regular. It may be due to the coding of this software not in a real-time environment.

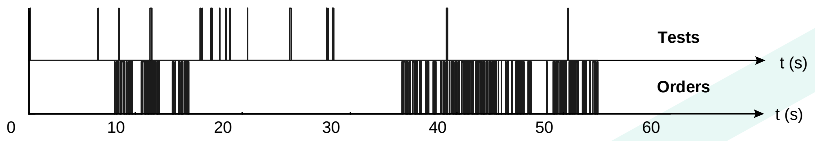

Figure 7. Distribution of test messages and orders

Figure 8 represents the time elapsed between the sending of some data and the receiving of the according acknowledge, in both ways between the PC and the Indigo. The mean value is located around 22 ms with a standard deviation of 20 ms. This figure shows us the speed of communication of the network between the two host computers.

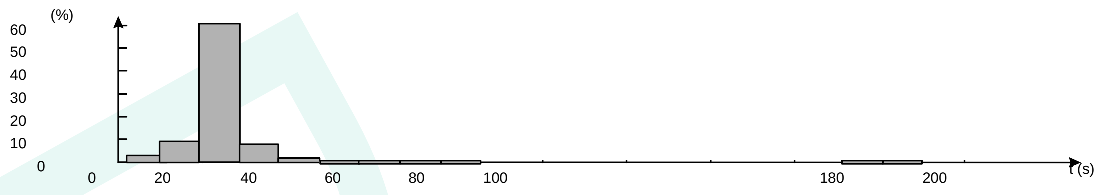

Figure 8. Distribution of response time of the network for data frames

Figure 9a represents the speed orders sent to the vehicle. Figure 9b represents the estimated speed of the vehicle (we have summed the values of figure 9a). This should be always positive because it corresponds to the absolute value of the desired speed of the vehicle. In fact, the sum of the values of figure 9a gives negative values. In the DSP program, there is a routine that filters negative values so that it remains null. We have had to do the same thing for figure 9b. This phenomena tells us that the orders aren't very reliable ; we'll have to take care of it in order to make the whole platform more reliable.

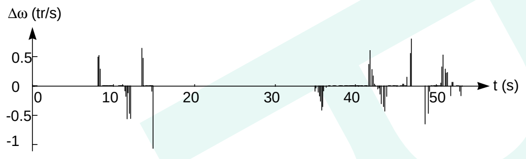

Figure 9a. Speed changement orders

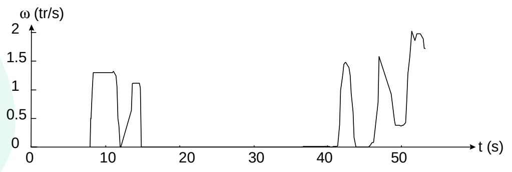

Figure 9b. Estimated speed

CONCLUSION

We have so achieved the building of a teleoperation system. It is quite perfectible but it raises some issues, we couldn't have thought in theory. For example, we will replace the relative values in the order messages by absolute values.

This system now will give us the ability to study some control laws for a reliable teleoperation with variable delays. In a first time, we will study the effects of these delays in a control loop. When we will have it correctly modelled, we will be able to find the most appropriate control law using prediction technics. Figure 1 shows us the final architecture for this project.

REFERENCES

1. P.K. Pool & D.H. Ballard "Remote Teleassistance", Univ. of Rochester, NY, USA
2. A.Rastogi & P.Migram "Augmented Telerobotic Control", University of Toronto, Canada
3. T-J. Tarn & K.Brady "A Framework for the control of Time-Delayed Telerobotic Systems", Washington Univ.
4. R.T. Braden "Requirements for Internet Hosts-Communication Layers" RFC 1122.
5. D.E.Comer & D.L.Stevens . "Internetworking with TCP/IP", Prentice Hall, Englewood Cliffs, NJ