UPB Sci. Bull., Series C Vol.70, No.1, 2008 ISSN 1454-234x

TELEOPERATION CONTROLLER ARCHITECTURE FOR MOBILE ROBOTS

Ctin NEGRESCU[1], D. M. POSTELNICESCU[2]

Articolul prezintă un sistem de control al roboților mobili. Arhitectura modulară a controllerului de teleoperare include comunicațiile interproces, localizarea, edificarea hărților, autoprotecția, managementul senzorilor şi sinteza vorbirii. Controllerul suportă multiple nivele de cooperare dintre operator şi robot, de la controlul direct la cel de supervizare. Autorii acestui articol prezintă arhitectura de control robotic, omisă în referințele legate de această implementare.

The paper presents a control system for mobile robots. The teleoperation controller has a modular architecture and includes interprocess communications, localization, map building, safeguarding, sensor management, and speech synthesis. The controller supports varying degrees of cooperation between the operator and robot, from direct to supervisory control. The authors of this article presents an new approach of robot control architecture which is not represented in other papers.

Keywords: human robot interaction, mobile robots, vehicle teleoperation

1. Introduction

Various researchers have addressed the problem of designing control systems for teleoperation. Teleoperation controllers encompass an extremely varied range of designs and techniques. The most, can be described within the framework of one or more existing *robot control architectures* [1].A parallel, three-layered control architecture for teleoperation of mobile robots is described in [2].The design uses a network of low-level control behaviors switched on-off by a high-level symbolic layer. A mobile robot control system with multisensor feedback is presented in [3]. Cooperation between human and robot implies *command fusion* witch enable operators to share control with a safeguarding system on-board the robot [4].

[1] Eng., PhD Student, Dept. of Control Engineering Industrial Informatics University Politehnica of Bucharest, e-mail:negrescu55@yahoo.com
[2] Eng., PhD Student, Dept. of Control Engineering Industrial Informatics University Politehnica of Bucharest, e-mail: daragoshi @yahoo.com

2. Robot Hardware

The controller was designed to operate Pionner mobile robots: both shown in Fig. 1 are 4-wheel skid-steered and capable of traversing moderately difficult terrain. Both have a microcontroller for servo and hardware control.

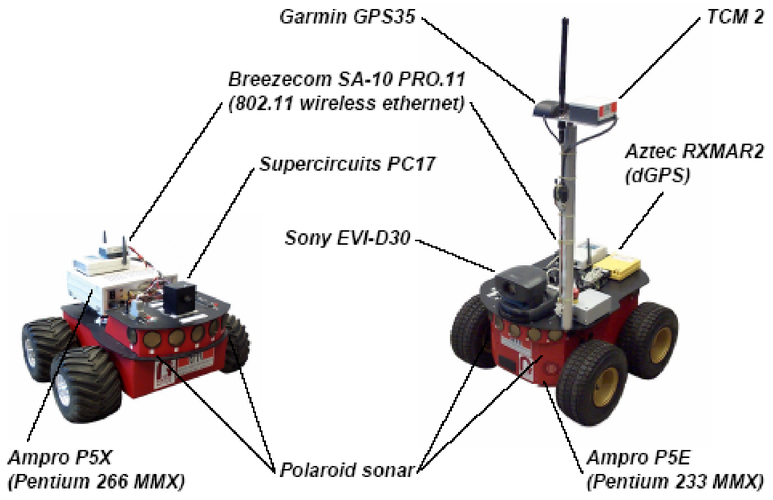
Garmin GPS35
TCM 2
Breezecom SA-10 PRO.11 (802.11 wireless ethernet)
Supercircuits PC17
Aztec RXMAR2 (dGPS)
Sony EVI-D30
Ampro P5X (Pentium 266 MMX)
Polaroid sonar
Ampro P5E (Pentium 233 MMX)

Fig. 1. Left Pionneer –AT, right Pionner2-AT

3 Design
3.1 Requirements for teleoperation controller

The most teleoperation interfaces incorporate tools and displays to help the operator perceive the remote environment, to make decisions and to generate commands. An good teleoperation controller provide resources to support these tools and displays. In particular the controller must supply capabilities and sensory feedback which make the interface work well and the remote task easy to perform. To support navigation, the controller must provide :

- spatial feedback (sensor-based maps)
- visual feedback (still images and / or video)
- situational feedback (robot health, position, command status)

Because navigation is dependent on perception, the controller is required to provide capabilities for *sensor management* and *sensor processing*. (e.g. sensor-based map building require range data processing.)

3.2 Controller architecture

The teleoperation controller is implemented as a distributed set of modules. Fig. 2 shows where the modules are resides and how they are connected.

This kind of structure look like a network of computers with a server (here notated – *FPC Server*) and a lot of workstations (here the modules with different names.)

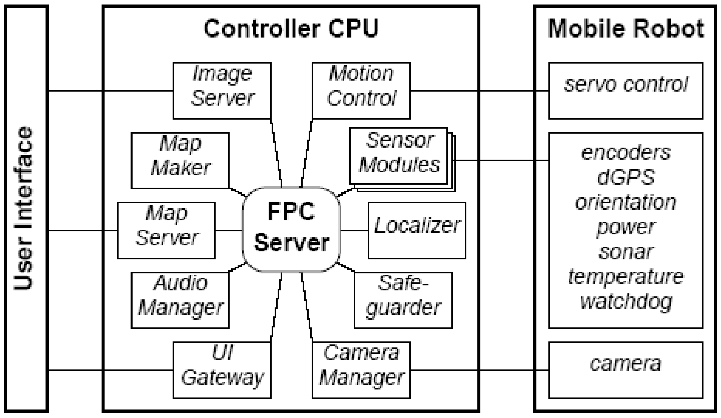

Fig. 2. Teleoperation controller architecture

It is partially true because some of modules run standalone and operate asynchronously and, others modules, particularly those witch process sensor data or operate robot hardware, have precise timing or data requirements. These last modules operate in the SAPHIRA system.

- The *system architecture* provides a micro-tasking operating system and functions for communicating with and operating robot hardware.
- The *robot control architecture* contains representations and routines for sensor processing, for environment mapping, and for controlling robot actions.

SAPHIRA is a good choice for several reasons:

1. it is a mature system and works well with Pionner robots
2. it provides efficient command fusion through fuzzy behavior.
3. it is extensible, modular and portable
4. the micro-tasking operating system is synchronous and interrupt – driven, thus making it easy to implement modules with precise timing.

SAPHIRA (see fig.3) is a framework for constructing mobile robot controllers and contains both a system and a robot control architecture [5]. It was first developed in conjunction with the Flakey mobile robot project, as an integrated architecture for robot perception and action. The software runs a reactive planning system with a fuzzy controller and a behavior sequencer.

There are integrated routines for sonar sensor interpretation, map building, and navigation. At the center of the architecture is the Local Perceptual Space (LPS). It accommodates various levels of interpretation of sensor information, as well as *a priori* information from sources such as geometric maps. The main system consists of a robot server that managest6he hardware, and SAPHIRA which is a client to this server.

SAPHIRA has been implemented on a number of different operating systems and in addition an API across a number of languages has been developed, in particular to support different types of experiments from low-level control to task planning. There are several coding methods used in the SAPHIRA architecture. The core of system is programmed in the C language. A special high-level interpreted language has been designed called Colbert (Konolige, 1997). It has a C-like syntax with semantics based on finite state machines. A part of SAPHIRA is written in LISP.

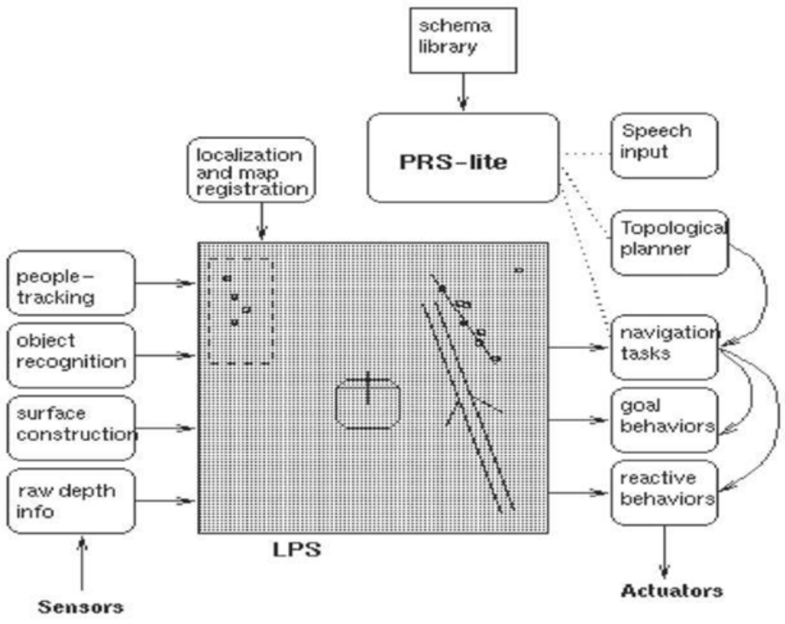

Fig. 2 The SAPHIRA system

Table1 lists the modules and describes functions, execution style and implementation of each.

3.3 Interprocess communications

In the past, all robot software was designed as a single monolithic block. Modern robotic system, however, are constructed as a group of modules each of which performs distinct processing functions. Modular design provide many benefits including:

- encouraging team development
- facilitating module implementation
- enabling distributed computation

At the same time, however, this approach requires that some mechanism be used to integrate modules and to distribute data between them.

A *network-based interprocess communication toolkit* is the most common mechanism for this purpose. Interprocess communication toolkits have been used to support distributed and parallel computing. Thus numerous interprocess communication toolkits have been developed for robotics including –IPT, NDDS, NML, TCA/TCX/IPC and RTC [6]

For implementation was selected the Fourth Planet Communicator (FPC) toolkit [7] (FPC's design was inspired by both message-based (e.g. TCA/TCX/ IPC) and information-based (e.g. NDDS) systems. Reasons for choice:

1. It provided both reliable (for message sequences) and unreliable (for fast data) delivery.
2. Its performance (message rate and latency) is well suited to the needs of this controller modules.
3. It facilitates integration of diverse modules with multiple language interfaces and support for multiples operating systems.

Table1

Teleoperation Controller modules

Name	*function*	*execution style*	*implementation (language)*
UI Gateway	proxy server for user interfaces	synchronous (varies)	standalone -C
Task Modules	task-specific functions	asynchronous	standalone -C
Motion Controller	low-level motion	synchronous (10 Hz)	SAPHIRA (C, fuzzy behavior)
Safeguarder	health monitoring motion safeguards	synchronous (10 Hz)	SAPHIRA (C, fuzzy behavior)
Map Maker/ Map Server	map building map generation	asynchronous	standalone -C
Localizer	position estimation	synchronous (10Hz)	SAPHIRA (C)
Camera Manager	camera control	asynchronous	standalone -C
Image Server	image capture	asynchronous	standalone -C
Audio Manager	sound playback speech synthesis	asynchronous	standalone -C
Sensor Modules	sensor processing	synchronous (10Hz)	standalone -C

4. Modules

4.1 User interface gateway

UI Gateway is a proxy server for user interfaces. It provides access to controller services (e.g. motion control) and uses a simple message protocol (text-based and asynchronous (to reduce latency). whenever an interface is connected, the UI Gateway continually monitors the connection. If it detects a communication problem (e.g. network outage), or that the interface not longer responding. The module immediately stops the robot and close the connection to ensure that operator commands are only executed while the interface is active and functioning.

4.2 Task modules

Each Task Module performs a specific perception or decision making function to meet the needs of a particular teleoperation application. A task module may add an autonomous capability to the robot or may offer assistance to the human. In this case two task modules have been developed; *MotionDetector* and *RockFinder*. The *MotionDetector* is used for autonomous visual surveillance. It detects motion in the environment by acquiring camera image sequences and computing interframe differences. This task notifies the human whenever occur a motion.

4.3 Motion Controller

Motion Controller execute and monitors motion commands. It generates position and rate setpoints (translation and rotation) for the robot's low-level servo-controller. The MotionController sequences and executes motion commands using SAPHIRA fuzzy behaviors. This allows all vehicle motion to be safeguarded (i.e. via *command fusion* with Safeguarder behaviors.)

The MotionController continuously monitors the progress and status of each executing motion behavior. Whenever it detects lack of progress (e.g. due to safeguarding) or failure, the MotionController contacts the operator.

4.4 Safeguarder

This module maintains the robot's safety. each safeguard is implemented as a SAPHIRA fuzzy behavior and may be individually activated or disabled by other controller modules. The most important behaviors are:

- Collision avoidance
- Rollover avoidance
- Health monitoring
- Clutter monitoring

4.5 Map Maker

Image-based driving is an efficient command mechanism, but it may fail to provide sufficient contextual cues for good situational awareness. Maps can remedy this by providing reference to environmental features, explored regions and traversed path. In addition, maps can be efficiently used for collision avoidance. The MapMaker builds maps using range sensors.

4.6 MapServer

The MapServer provides maps as images. Whenever it receives a request, the MapServer queries the MapMaker for the relevant grid region and converts certainty values to graylevels. Clear areas appear as white, obstacles as black, and unknown as light-gray. The maps are generated in either the world or local frame, with arbitrary resolution (sub/super sampling) and of any extent (regions outside the grid are marked "unknown")

4.7 CameraManager

The CameraManager operates the robot's camera system, arbitrating and sequencing camera commands from other modules. The functions are :

- to control steereable CCD cameras
- to configure imaging parameters (gain, exposure compensation, magnification, etc)
- to output a message describing the camera's state (position, magnification, field-of-view) so that modules making use of the camera can act appropriately.

4.8 ImageServer

In most vehicle teleoperation systems, video is the primary source of visual feedback., but high-quality video, requires significant communication bandwidth. For applications with low bandwidth and/or high transmition delay, video may not be practical. For this reason the controller provides an event-driven ImageServer. This server captures a frame, compresses it into a JPEG image, and sends it whenever:

- the operator issues a request
- the robot stops
- an obstacle (static or moving) is detected
- an interframe timer expires.

Event-driven imagery is a flexible mechanism for visual feedback.

4.9 Sensor Modules

A sensor Module interacts with a single sensor or a group of related sensors. each module works like an operating system driver: it communicates with the device using sensor-specific protocols, processes the sensor data, then delivers the results for other controller modules to use. There are currently five Sensor Modules in the teleoperation controller: GPS, Health, Odometer, Sonar and TCM2.

- *The GPS Module* acquires GPS position fixes and transforms the data from geodesic coordinates to a regional Cartesian user grid.
- *The Health module* monitors vehicle health sensors: power(battery voltage) temperature(environment) and watchdog (hardware controller)
- *The Odometer module* processes wheel encoder data by computing differential position and velocity based on encoder pulses.
- *The Sonar module* controls a ring of ultrasonic sonar used to enable/disable transducers and to configure polling order (to minimize crosstalk)
- *The TCM2 module* acquires orientation (roll, pitch, compass heading) and external temperature data from the TCM2.

4.10 AudioManager

When robot must operate around or with humans, audio plays an important role in human-robot interaction. Audio is a highly effective mechanism for conveying the robot's intent and for communicating information to humans. Thus the AudioManager is designed to perform two functions: sound playback and speech synthesis.

Sound playback is used to produce informative signals. Speech synthesis is used for information which cannot be conveyed by simple sound playback, such as status messages.

5. Three-layer architecture

We have observed from the Fig. 2 that there is no true robot control architecture and the figure represents only the interprocess communications. In our opinion, it is important to represent the three layers architecture in a new figure that should comprise the data flow, specific to the robot control.

In the ***bottom layer***, the controller provides *reactive feedback –control*: it uses behaviors for safeguarding and robot motion. In contrast to other three-layer architecture, these behaviors maintain internal state (activation time, number of attempt, etc) to facilitate human-robot interaction and to assist in failure detection. Modules involved: *Localizer* and *Safeguarder*

In the ***middle layer***, the controller performs sequencing; it selects which primitive actions should be in use, then handle behavior activation and deactivation. (e.g. *low-level motion control* and safeguard behaviors are sequenced during execution of waypoint commands. Similarly, task-specific modules(e.g. autonomous survey) generate control sequences to exercise control over the robot. Modules involved: *Motion Controller* and *Camera Manger*

In the ***top layer,*** the controller integrate human input via user interface.

- In autonomous systems, the top layer contains *high-level planning* and *decision making* modules.

- In teleoperation the human typically performs these tasks (with collaborative control, both the robot and human share responsibility for producing plans and for responding to queries from the other controller layer. Task Modules implements this layer. All the other modules are utility modules. The schema is illustrated in Fig. 4

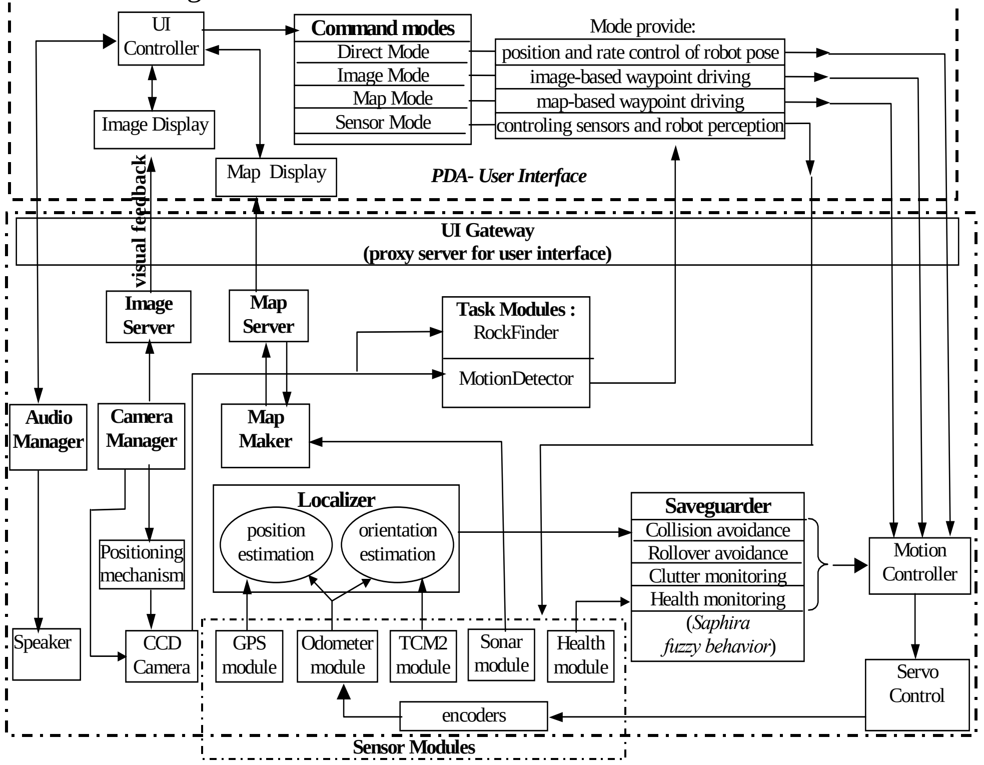

Fig. 4- Robot control architecture

6. Conclusions

The architecture represented in Fig. 4 points out the architecture of the controller that we have been talking about, from the point of view of the system organized on three layers, offering in this way a better understanding of the functioning and from here on, finding future solutions for improving the actual system.

R E F E R E N C E S

[1] *J-M. Hasemann*, “ Robot Control Architectures : Applications, Requirements, Approaches, and Technologies”, SPIE Intelligent Robots and Manufacturing Systems, Philadelphia, PA, 1995

[2] *A. Maslowski, et al.”*Autonomous Mobile Robot Controller for Teleoperation System” ISMCR Prague, Czech Republic 1998

[3] *I. Lin et all* “ An Advanced Telerobotic Control System for a Mobile Robot with Multisensor Feedback ” IAS-4 IOS Press 1995

[4] *E. Krotov* “Safeguarded Teleoperation for Lunar Rovers: From Human Factors to Field Trials”, IEEE Planetary Rover Tech. and Sys. Workshop 1996

[5] *K. Konoligue and K Mayers “*The SAPHIRA Architecture for Autonomous Mobile Robots” in AI and Mobile Robots, (Bonasso, and Murphy R eds) MIT Press Cambridge MA 1997

[6] *J. Gowdy* “ A Qualitative Comparison of Interprocess Communications Toolkits for Robotics” CMU-RI-TR-00-16 Carnegie Mellon University 2000

[7] *T. Fong* ”FPC: Fourth Planet Communicator , Fourth Planet, Inc Los Altos Ca 1998