Preprints of the Fourth International
Symposium on Experimental Robotics, ISER'95
Stanford, California, June 30–July 2, 1995

Real-Time Programming of Mobile Robot Actions Using Advanced Control Techniques

Roger Pissard-Gibollet, Konstantinos Kapellos, Patrick Rives, Jean-Jacques Borrelly

INRIA-centre de Sophia Antipolis,
2004 Route des Lucioles, 06565 Valbonne, France
e-mail : {name}@sophia.inria.fr

Abstract

In this paper, we present a robotic application performed by sequencing visual servoing tasks. The theoritical framework used is the sensor-based control for the continuous parts of the application and discrete events system theory for its logical aspects. The design and analysis of the whole system are coherently handled using ORCCAD concepts. We focus our attention on the programming aspects of these theories and concepts, from application level specification up to real-time implementation and results analysis. An effective experimentation on our mobile hand-eye robot validating and illustrating this approach is fully detailed.

1. Introduction

Programming mobile robots in order to achieve a desired objective in a reliable way, operating in a structured and partially known environment, requires to solve many appealing problems. They range from task planning, reactive behavior synthesis up to control law choice and design. Even though, at the high level, problems of path planning and off line programming have been broadly addressed, the state is not so clear at the control level, where the interaction of the mobile robot with its environment must be considered. In most cases, the trajectory provided by the path planner is directly played at the servo level without taking into account perturbations due to real interactions between the robot and its local environment. As a consequence, the control law works in open loop with regard to this environment. A manner to perform a robotic task in a more robust and reliable way, is to control explicitely the local interactions between the robot and its environment. It can be efficiently done by using a sensor-based control approach. Furthermore, we claim that a complex robotic task can be successfully performed by sequencing elementary sensor-based control tasks.

The paper is organized as follows. In the first part we briefly recall the theoritical framework used to model elementary tasks and their composition. The second part describes the real experiment on a robotic task of *reaching a target* with our experimental testbed constitued by a mobile robot carrying a hand-eye system using dedicated vision hardware. A special attention will be put on the programming of the application ranging from specification to real-time programming and results analysis.

2. Continuous Control Laws Design

We assume that, in order to correctly complete a task, we need a low level of task specification which explicitely integrates the interaction between the robots and its local environment. When the task can be defined as a servoing on a local part of the environment, we use a previously developped framework in vision-based control. It allows us to perform elementary visual servoing tasks by means of robust closed loop control laws using data vision. Let us give a brief overview of the approach, while a more detailed presentation can be found in [1, 2, 3].

The basic assumption about the sensors is that the vector signal $\underline{s}$ provided by the sensor is a function of the relative position and orientation $\bar{r}$ between the sensor, associated to a frame F_S, and the target, associated to a frame F_T. We may thus write :

$$\underline{s}(\bar{r}, t) = \underline{s}(F_S, F_T)$$

A Jacobian matrix L^T of the vision feature $\underline{s}$ with regard to the relative displacement (velocities screw T_{ST}) between the camera and the environment can be computed to get : $\dot{\underline{s}}(\bar{r}, t) = L^T.T_{ST}$

Of course, the formal expression of L^T depends on the type of geometrical primitives (point, line, ellipse...)

and their parametrization [1].

To specify a task and to select the visual signal, we use the notion of *virtual linkage* between the robot sensor and its environment. It is characterized by the velocity screw T^* which leave $\underline{s}$ invariant during the motion :

$$\underline{\dot{s}}(\bar{r}, t) = L^T.T^* = 0$$

Using the general formalism of task function [3], we can express the goal of a task in term of the regulation of an output function $\underline{e}(\underline{r}, t)$. Applying this formalism to the sensor based task, this function can be written like : $\underline{e}(\bar{r}, t) = \underline{s}(\bar{r}, t) - \underline{s}_d(t)$ where $\underline{s}_d$ is the desired visual signal. It has been shown in [1] that a very simple *gradient based approach* is sufficient to ensure an exponential regulation of $\underline{e}$ by using the following desired velocity screw T_d as control input. For a positioning task we have :

$$T_d = -\lambda . L^{T^+}_{\underline{s}=\underline{s}_d} . (\underline{s}(\bar{r}) - \underline{s}_d) \qquad (1)$$

The transposition of such a scheme to nonholonomic mobile robots is not straightforward. One way to overpass this problem consists in adding some degrees of mobility to the camera by mounting it on a motorized device like a manipulator or a "head". Considering the whole mechanical system (cart + manipulator) like a single kinematic chain, it becomes possible to fully control the motion of the camera without beeing limited by the cart nonholonomic constraints [4, 5].

On the other hand when we do not, or can not, take into account sensory data from the environment to design the control laws, it is possible to directly design it in the cartesian space. But it is now well established, that it does not exist a smooth continuous state-space feedback able to stabilize a nonholonomic mobile robot around a fixed point. So we used recent results in mobile robot control [6] and we have implemented a control law using a *time-varying* feedback technique. Using the state error vector $(x \; y \; \tilde{\theta})^T$ the control can be written :

$$\left\{ \begin{array}{l} v = y^{\frac{1}{3}} \; sin(\omega t) + g_1 \; x \; cos(\tilde{\theta}) \\ \dot{\theta} = \frac{2}{3} \; g_2 \; y^{\frac{1}{3}} \; v \frac{sin(\tilde{\theta})}{\tilde{\theta}} + g_3 \; \tilde{\theta} \end{array} \right. \qquad (2)$$

3. Robotic Actions Concepts

3.1. The Robot-task (RT)

Control laws as (1),(2) without ambiguity characterize, in continuous time, the physical motion of the robot during the $[0, T]$ interval of their validity. Nevertheless, when we want to execute this motion in a realistic environment we need to take into account and react in time to various situations, at least for ensuring the integrity of the robot. These two tightly coupled aspects of a robotic action are coherently captured by the *Robot-task* (RT) definition as proposed by ORCCAD concepts [7]. Let's remind that a RT models an elementary robotic action. It is formaly defined as the entire parametrized specification of a control law, and a logical behavior associated with a set of events which may occur just before, during and just after the task execution.

The behavior of the system is handled by the framework of reactive systems theory: it consists of the legal sequences of input/output signals received/emitted by the system. Its specification is methodic; events are typed in pre-conditions, exceptions with three types of reaction and post-conditions. The control-law activation starts at the instant that all pre-conditions are satisfied. During its execution exceptions are monitored; they are handled either localy changing in-line a control parameter or globaly asking from the application to interrupt the current RT and activate a recovery program or imposing the total application interruption driving the robot in a safe position. Finaly the action stops when the set of post-conditions is satisfied.

3.2. The Robot-Procedure (RPr)

A more complex robotic application is therefore seen as the composition of the RTs necessary to accomplish the desired objective. Composition is obtained using dedicated operators as the sequence, the conditional, the iteration, and different levels of prehemption. In ORCCAD system the Robot-procedure (RPr) formalism is proposed to methodicaly specify, verify and implement RTs controllers in order to design complex robotic actions. It clearly separates the composition of the actions driving to a nominal execution from those required to recover from an exception not handled localy by the RTs. To the whole can be associated pre and post-conditions driving to the specification of an entity which can be used to compose other ones; structural programing is therefore obtained.

RPr formalism is translated to adequate languages (ESTEREL and TIMEDARGOS) providing the behavior controller with nice semantics. These languages may be compiled into a wide class of models, usually labeled transition systems. This allows to methodicaly verify a large set of behavioral and quantitative temporal properties, including crucial properties of liveliness and safety, as well as the conformity with applications requirements. From a practical point of view we use the ESTEREL compiler to obtain the program that schedules the evolution of the systems behavior.

4. The Robotic System

We have developped a versatile testbed in order to validate sensor-based control approaches in real experiments. This robotic system ([8]) uses a nonholonomic wheeled cart carrying a two d.o.f head with a CCD camera and recently equiped with a belt of eight sounders (see figure 1).

The on board computer architecture is built around a VME backplane. The robot control is assumed by a Motorola MVME 167 board and a 6 axes custom-made servocontrol board. The management of the sounder belt is done by an other Motorola MVME 162 board.

Difficulties in image processing are induced by strong real time constraints and the processing of large dataflow from the sequence of images. To overpass them we have developped a vision system [9] which is characterized by its modularity and its real time capabilities. Its architecture is based both on VLSI chips for low level processing and on multi DSP's processors for more elaborated processings. For facilities of development, the machine vision has been implemented in an independant VME rack outside of the robot. However, for experiments which require high autonomy, these boards can be plugged into the on board rack.

During the development step, an external link is used both for power and communication (ethernet thin link, three video channels and two serial links).

Figure 1. View of the Robotic System

5. A Robotic Application Specification

5.1. Description of our Application

Our robotic system is used to validate and illustrate the theoritical aspects previously presented. In this application, our long term objective is to explore an indoor environment.

The application, presented in this paper, consists in *reaching a target* and it is performed by our robotic system. Informally it is specified as follows: the robot must cross the room by following a wall, reach a region of interest, find a target in this region and go in front of it. To perform this application, sophisticated control laws must be sequenced; their activations are conditionned by the occurence of various types of events determining the end of the wall, the presence of the target, some failures. A nominal execution of this robotic application is illustrated in figure 2.

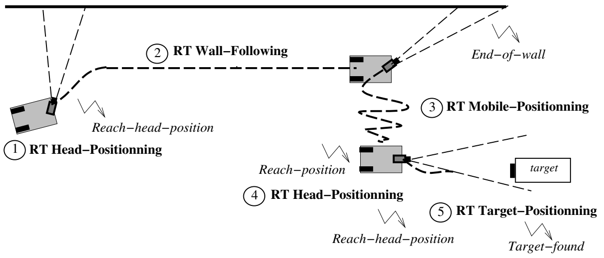

Figure 2. Description of the Application

5.2. Decomposition in Robot-task

Once the application is described, the next step towards its realization consists in identifying all the elementary tasks (RTs) needed to perform it. The basic idea is to associate a control law to a sub-objective and to identify the set of events related to its execution. Therefore a RT is constructed by the characterization of these elements. In this experiment, using only vision and odometry, we have defined four RTs.

Head Positioning

In a priori known environment, the robot must be able to move its head (two axis on pitch and yaw) in order to look at an object of interest. For example, using *MoveHead*, the robot can look up to see the final target or look toward the floor to find a skirting board. This elementary action is finished when the head reaches the desired position (post-condition event *Reach-head-Position*).

Visual Wall Following

To cross the room the robot can follow a wall. The parallel lines corresponding to a skirting board at the bottom of a wall (see the two first images on figure 3) are used for its control. This visual-servoing task is handled by the vision-based control reminded in the theoritical part. For this task *WallFollowing*, thanks to the vision system, line parameters are extracted for the visual servoing control loop and events are detected as the *End-of-wall* (post-condition) or the exception *Target-lost*.

Visual Positioning in front of a Target
To position the robot in front of the target we have implemented the visual-servoing task *Pos-Target* (control law (1)). The selected $3D$ target is a cone which projects onto the image frame in two ellipses corresponding to the two circles bounding the cone (see the two last images on figure 3). The positioning task consists in aligning the cone axis with the optical axis of the camera and to tune the distance between the camera and the target along with this axis [4, 5]. Like in the previous visual servoing Robot-task, an event *Target-lost* monitors a possible failure. The correct end of this action is managed by an event indicating the end of task duration.

Cartesian Mobile Robot Positioning
To drive the robot in an area and to stabilize it around a fixed point, we use the time-varying feedback expressed in control law (2). Despite that the cartesian position of the mobile robot is reconstructed using only odometry, the precision of robot positioning is sufficient. When the robot reaches the desired position, the post-condition *Reach-position* is satisfied inducing the end of this Robot-task.

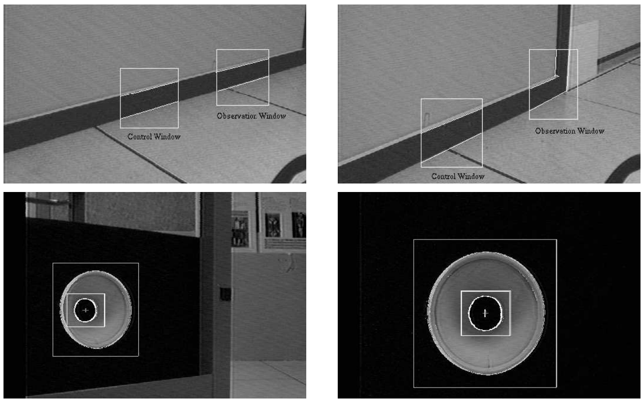

Figure 3. Images during the two visual-servoing RT

5.3. Application Synthesis with RPrs

The RPr *ReachTarget* (see below) is designed to specify the evolution of the application as the composition of the previously defined RTs. It simply states that the application will start after user confirmation, pre-condition *start*; its nominal execution consists on the sequencing of five actions. Initialy the RT *MoveHead* fixes the head to the indicated position. It is followed by the *SafeWallFollow* (see bellow) which, after detecting the visual motif drives the mobile accross the room until the end of the wall avoiding eventualy the obstacles. Afterwards, *MoveCart* drives the mobile to reach the target location and *MoveHead* directs the robot look toward the target. Finally, *PosTarget* brings the robot in front of it. During the execution *Target-Lost, Robot-Fail...* events are handled as failure exceptions asking for application interruption.

SafeWallFollow (see below) is active continuously, since the end of the wall detection, the *OneWallFollow* which, using *WallFollow* and *ObstacleAvoidance* displace the mobile along the wall and handles the case of obstacle presence.

The translation of *ReachTarget* RPr to a dedicated language, ESTEREL in the particular case, gives us both, the model describing its evolution and the program that controls it. Figure 4 gives an abstract view of the controller considering only the activation and the end of the RTs and the most significant events. On the other hand the control program uses the services of a real-time software that we developped to control RTs.

```
PrR ReachTarget
[
 Pre-cond:  Start
 <main_program>
                MoveHead(-45,-25);
                SafeWallFollowing();
                MoveCart(0,1,0);
                MoveHead(0,0);
                Pos-Target();
                End.
exc_T3:         (Robot-Fail), (Sensor-Fail), ...
]

PrR OneWallFollowing
[
 <main_program>
                WallFollowing();
                End.
exc_T2:        (A-head-obstacle,ObstacleAvoidance().)
exc_T3:       (Target-lost), (Robot-Fail), ...
Post-cond: (End-wall-following)
]

PrR SaveWallFollowing
[
 <main_program>
                Loop
                     OneWallFollowing();
                EndLoop.
exc_T3:        (Target-lost), (Robot-Fail), ...
Post-cond: (End-wall-following)
]
```

Figure 4. RPr design

6. Real-Time Programming

A real-time program must be logically correct to produce the correct outputs but also it must be temporally correct to produce them at the correct time ; control laws performance together with robot safety highly depends on it.

The real-time software used to realize this experiment

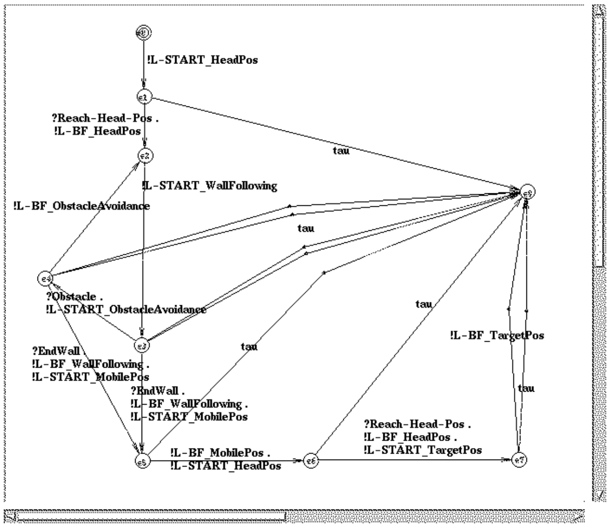

Figure 5. Automaton of the Application

has two main parts : the first one assumes the robot control ; the second concerns the vision processing. Their communication is based on a client/server architecture ; the robot controller manages the application and sees the vision program as an *intelligent sensor* (see 6.2). Let us briefly describe this software before presenting the experiment results.

6.1. Robot Control Software

The aim of our robot control software is to support robotic applications constructed as a composition of RTs . It is naturally divided in three parts [10] concerning :

• The computation of the continuous part of the system (control law and observers) : the tasks are usually periodics, communications and message passing between them are asynchronous of producer/consummer type.
• The discret event controller : synchronous approach is used to compose the behaviors of the RTs as well the coordination of RTs switching. This permits to enhance the reliability of our system by formally analyzing/proving interesting properties of the controller.
• The interface between synchronous and asynchronous parts of the application : save communications, preserving the order of messages, are established in a systematic way between the asynchronous observers and the synchronous controller.

The use of adapted programming and debugging tools for each part of the application is of prime necessity : object oriented language (C++) for programming facilities, synchronous language (ESTEREL) use for critical controllers and efficient real-time operating system with large environment capabilities (WINDRIVER tools : VxWorks 5.1.1, WindView).

6.2. Vision Software

To perform the vision processing, we use a dedicated machine vision of which the hardware and software architecture are described in [9]. It implements the concept of active window. An active window is attached to a particular region of interest in the image and has in charge to extract a desired feature in this region. To each active window is associated a temporal filter able to perform the tracking of the feature along the sequence. Several active windows can be defined, at the same time, in the image and different processings can be associated to each window.

In our application, the two visual servoing RTs use services coming from the vision software application, allowing to launch/stop active windows in images. These windows (see figure 3) return parameters for the visual servoing (on lines or ellipses) but also events (*target-lost, End-of-wall*). Services of target recognition in image and automatic windows positioning initialization are not currently developped but one way is investigated in [11].

7. Experimental Results

Let us present the effective experiment which consists of the RPR *ReachTarget* nominal execution by our robotic system. Actually, a set of tools allows the user to analyze the real execution of the robotic application from a continuous and discrete time point of view. They are used off-line after the execution and allows its analysis at different levels.

Firstly robot trajectory can be visualized using a 3D animation tool to estimate roughly the robot evolution (see figure 6). For a finest estimation from the automatic control point of view, the evolution of continuous variables, such as robot state or sensors data, can be plotted. In addition occurences of events and activation of RTs can be associated to these plots. For example, in figure 7, we verify that, during the RT1 *MoveHead* execution, the robot wheels velocities are null until the head is positioned which is signaled by the occurence of *Reach-head-Pos* event. Its presence activates the controller which transits from state e_1 to e_2 activating the RT2 *WallFollowing* which induces the wheels velocities evolution. Finally, all the real-time mecanisms involved in the application execution can be analyzed; in figure 8, the switch between the RTs *WallFollowing* and *MoveCart* is visualized.

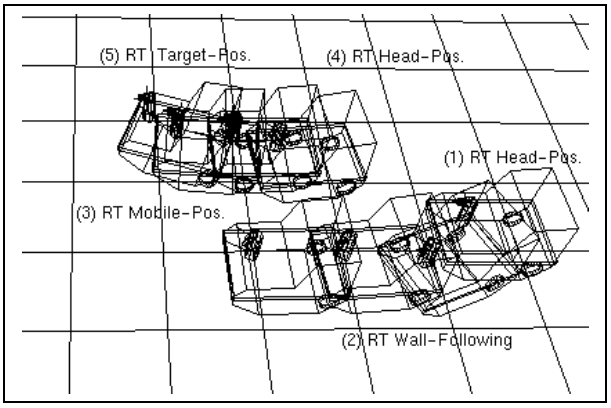

Figure 6. Mobile Robot Trajectory

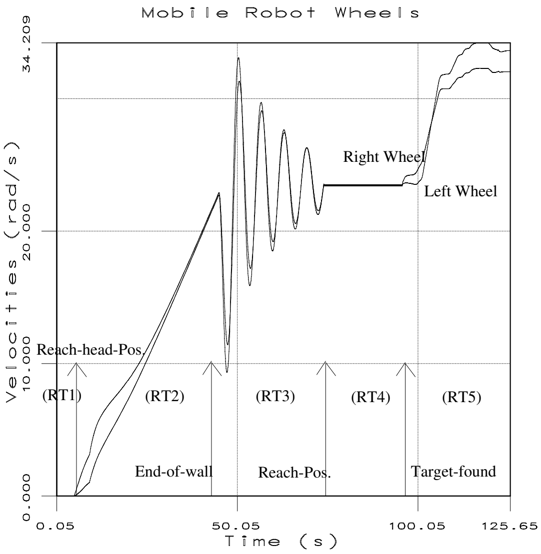

Figure 7. Velocities on Robot Wheels

8. Conclusion

Using a generic framework given by ORCCAD lying on continuous and discrete approaches, we have shown experimentaly how a "complex robotic task" can be successfully performed by sequencing elementary sensor-based control tasks. A special attention has been given to the programming/debugging/analyzing tools and real-time performances of the system. The results show that work remains to be done for robustifying the experimental aspects. For example, more capabilities must be taken into account with regard to events failures or obstacles. Concerning the sensor aspects, we have to implement more elaborated vision algorithms and to integrate sounder-based tasks. In the future, we expect to achieve more complex application concerning automatic exploration and cartography of partial known structured environment.

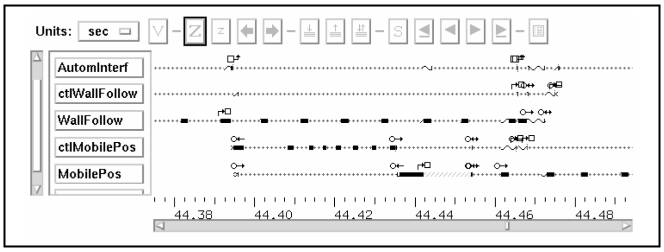

Figure 8. Timing of a Robot-Task Switching

References

[1] F. Chaumette, *La commande référencée vision: une approche aux problèmes d'asservissements visuels en robotique.* PhD thesis, Université de Rennes, Juillet 1990.

[2] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 313–326, June 1992.

[3] C. Samson, B. Espiau, and M. L. Borgne, *Robot control: the task function appproach.* Oxford University Press, 1990.

[4] R. Pissard-Gibollet and P. Rives, "Applying visual servoing techniques to control a mobile hand-eye system," in *IEEE Int. Conf. on robotics and automation*, (Nagoya, Japan), May 1995.

[5] R. Pissard-Gibollet, *Conception et Commande par Asservissement Visuel d'un Robot Mobile.* PhD thesis, Ecole des Mines de Paris, December 1993.

[6] C. Samson, "Time varying feedback stablisation of a car like wheeled mobile robot," *The International Journal of Robotics Research*, vol. 12, pp. 55–64, February 1993.

[7] D. Simon, B. Espiau, E. Castillo, and K. Kapellos, "Computer-aided design of generic robot controller handling reactivity and real-time control issues," *IEEE Transactions on Control Systems and Technology*, Fall 1993.

[8] P. Rives, R. Pissard-Gibollet, and K. Kapellos, "Development of a reactive mobile robot using real time vision," in *ISER Third International Symposium on Experimental Robotics*, (Kyoto, Japan), October 1993.

[9] P. Rives, J. Borrelly, J. Gallice, and P. Martinet, "A versatile parallel architecture for vision based control applications," in *Workshop on Computer Architecture for Machine Perception*, (New-Orleans USA), December 1993.

[10] E. Coste-Manière, "A synchronous/asynchronous approach to robot programming," in EUROMICRO *Workshop on Real-Time Systems*, (Oulu Finland), pp. 268–273, June 1993.

[11] D. Djian, P. Probert, and P. Rives, "Active sensing using bayes nets," in *to appear ICAR Int. Conf. on Advanced Robotics*, (Spain), September 1995.