FLEXIBLE REAL-TIME ARCHITECTURE FOR HYBRID MOBILE ROBOTIC APPLICATIONS [1]

Houcine Hassan, Alfons Crespo, José Simó

Dept. Computer Engineering, Universidad Politécnica de Valencia, Camino de Vera s/n - 46071 Valencia (SPAIN) T. +3496 3877578 F. +3496 3877579 e-mail: {husein, acrespo, jsimo }@disca.upv.es

Abstract – The latest hybrid architectures proposed in the field of complex mobile robotic architectures have shown the necessity of using real-time algorithms for guaranteeing the combined execution of timing constrained activities with deliberate behaviours. Various fundamental aspects should be considered in the design. One of the main characteristics discussed in this paper is flexibility. A system has to be adapted to the environment and take appropriate actions. This focuses on the ability of the system to select the appropriate activity to be executed (depending on the available time) and to change its behaviour (depending on the environmental information). Researchers in the field of mobile robotic architecture are looking for internal monitoring systems that dynamically measure vehicle quality of service (QoS) and adjusting it for the best vehicle fit. The architecture we present dynamically guarantees the required vehicle QoS – by adapting either the vehicle speed or the mode change of operation. *Copyright© 2000 IFAC.*

Keywords: real-time systems, autonomous mobile robots, robot navigation, adaptive systems, heuristics.

1. INTRODUCTION

Hybrid architectures (Arkin98, Shreckenghost 98, Gat 98, Bonasso 97) have been used to represent advanced robotic systems which operate in uncertain dynamic environments where prior knowledge of the domain may be incomplete. The vehicle dynamics may vary and the information gained from several sensory sources may be incomplete and inaccurate. Reasoning in such dynamic environments must be deliberative in nature, and it must occur fast enough to respond to unexpected events. To function correctly in this kind of environment, the planning systems must be reactive and decisions must take into account, at regular time intervals, information about the current state. Thus, mobile robot architecture must combine deliberative goal-oriented planning with reactive sensor-driven operations.

The latest hybrid architectures proposed in the field of complex mobile robotic architectures have shown the necessity of using real-time algorithms for guaranteeing the combined execution of timing constrained activities with deliberate behaviours (Kortenkamp 98, Shreckenghost 98, Arkin 98, Gat 98). One of the main characteristics of these designs is flexibility.

The proposed real-time architecture has two main objectives: to execute the vehicle plan schedulabilty; and introduce enough flexibility and adaptation in the mobile system to guarantee a high quality of service.

The proposed architecture allows adapting the vehicle system to the environmental conditions. This focuses on the ability of the system to select the appropriate activity to be executed (depending on the available time) and to change its behaviour (depending on the environment information).

Furthermore, the architecture supports analysis of the QoS of the vehicle, a point that is poorly developed in the literature. Many researchers in mobile robotic systems (Arkin 98, Michaud 99, Hexmoor 99, Kortenkamp 98, Bonasso 97) have stated the need to use various empirical metrics to evaluate the utility of the architecture for the tasks accomplished by the vehicle. Indeed, *run-time flexibility* and *performance effectiveness* are quality metrics that are insufficiently developed.

Consequently, a QoS analyser is developed to guarantee the quality of service demanded by the vehicle. The QoS analyser manages two kinds of aperiodic tasks: multiple versions (Audsley 91, Liu 91); and the flexible tasks (Liu 97).

[1] This work has been partially funded by a grant from the Spanish government CICYT TAP99-1226-C02

The multiple versions are based on imprecise computations that use monotone tasks to produce intermediate results as the task is executed. It is usual to define a minimum result needed by the application and this represents the *mandatory* part of the task. The rest of the task is *optional* and will improve the result if executed.

One of the main drawbacks of the model, if used in dynamic environments such as robotic applications, is the impossibility dynamically making decisions for scheduling the best solution. This is because the optional tasks have fixed computational times and fixed priorities. (Garcia 96) proposed a model based on the idea of incorporating different strategies for dynamically planning the most suitable processes.

The paper is organised as follows, after an introduction regarding work in the field, the hybrid architecture of the mobile robot is presented. Sections 3 describe the real-time task model and the system scheduling that supports the planning of the hybrid model. One of the main components of the planner, the QoS analyser, is detailed in section 4. An example of a guided vehicle application is shown in section 5. Simulations are carried out in section 6 to evaluate the various proposed heuristic algorithms. Section 7 presents the evaluation of the QoS architecture improvements. Finally, conclusions and the future work are outlined in section 8.

2. HYBRID ROBOTIC MODEL

According to the requirements discussed in the previous section a hybrid mobile robotic architecture is proposed (Buendia 99, Hassan 99). A scheme of the architecture is shown in Fig. 1. It consists of a set of distributed behaviour entities communicating with a centralised arbiter. The behaviour entities define the task-specific knowledge for the domain. Each behaviour entity runs completely independently and asynchronously, using sensor data and processing algorithms. They provide commands to the arbiter, each at their own rate and according to their own time constraints. The kind of information represented by the commands relates to speed, trajectory, turn, etc. Each behaviour also has a motivation parameter reflecting its importance in performing the control action. Currently, a planner control module is responsible for configuring the motivations, depending of the vehicle state and the environment. The arbiter is then combines the behaviour commands, taking into account their motivation weight, in order to generate the actions that cope with the vehicle guidance. The different elements interacting in the behavioural model of the vehicle are shown in Fig. 1.

The passive elements of the architecture are: the vehicle state, and the environment. They are stored in the blackboard memory (Botti, 95).

The active components are labelled as:

- Behaviour components: reactive and deliberative. The behaviour has a motivation parameter used by the arbiters for deciding which action to perform. The behaviour motivation is assigned and can be reconfigured dynamically by a control module in the intelligent planner.

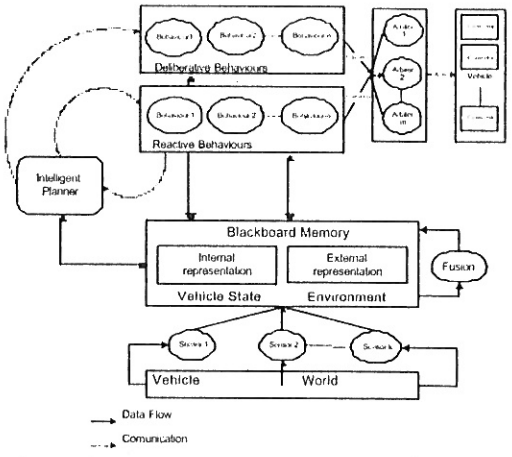

Fig. 1. Components of the behavioural model

- Intelligent planner: This is responsible for analysing the stored data regarding the internal and external state of the vehicle. Based on this information and on the requirements of the quality of service of the application, the QoS analyser decides if the requirements can be met.
- Arbiters: They are responsible for receiving the commands and motivations from different behaviours and evaluating which are the most important.
- Fusion: These processes take from the memory the sensory data generated by the sensory subsystems and fuse them, in order to generate what is termed perceptual information.
- Sensors: They collect data provided by the physical sensors subsystems. These processes are incorporated as real-time drivers.

3. REAL-TIME ARCHITECTURE

The proposed architecture has to guarantee the planning of the missions and the security of the autonomous vehicle. It also introduces a high level of flexibility and adaptation to the mobile system by integrating an intelligent planner able to enhance the vehicle QoS. This new architecture allows the execution of the hybrid model based on behaviours and motivations outlined in the previous section.

The model of the tasks is organised in order to cope with the execution of the reactive and deliberative elements of the hybrid model. These elements are mapped to a task entity.

3.1 Task Model

The architecture is organised in two planning levels. The critical level gives support to the reactive behaviours, and the optional level supports the deliberative behaviours. The tasks that coexist in the architecture have different natures. They can be observed in Fig. 2:

- Fixed periodic tasks: represent the reactive behaviours of the model.
- Variable periodic tasks: implement reactive behaviours which depend on some application parameter, such the *obstacle_avoidance* behaviour which, in turn, depends on the vehicle speed.
- Optional tasks: support the execution of deliberative tasks. These are soft aperiodic tasks.

The fixed periodic task set models the periodic activities of the system. These tasks are contain the following parameters:

$$T_i^f = (C_i, T_i, P_i, \phi_i)$$

where Ci is the worst case execution time, T_i is the period, P_i is the priority, and ϕ_i is the task offset.

The variable periodic task subset is characterised by:

$$T_i^v = (C_i(k), T_i(k), P_i, \phi_i)$$

In this case, the parameters *Ci* and *Ti* depend on parameter *k*.

The variable parameter *k* can, in a robotic system, be the robot speed. If the robot speed is slow, the period of these tasks can be longer, and so reducing the *CPU* load and allowing more optional computation. If the robot speed increases, the period should be reduced to guarantee the response time. These tasks have some specific features such as the variable computation time. If the number of objects in the environment is low, the computation time is shorter than when there are more objects to be recognised. Both aspects are complementary. When the number of objects detected in the environment is large, the computational requirements increase and, consequently, the robot speed must be reduced and the periods have to be increased. On the other hand, if the number of objects decreases, then the computational requirements decrease, so the robot can increase speed and reduce the task periods. The goal of the system is to adjust the system load to achieve the maximum robot speed and process the maximum optional load.

Moreover, the tasks are split in two parts: initial and final. The initial part computes the answer with sufficient quality. The final part sends the final answer to the actuator. However, the answer can be improved during the optional load – thus obtaining a higher quality system response. To allow this, all periodic tasks can be structured into two tasks with the following features: the period and priority are the same for both parts; the computation time corresponds to the computation of each part; and the offset of the final part is delayed to allow for refinement of the answer.

The optional tasks considered in the system are of two types: flexible tasks (Liu 97); and multiple versions of tasks (Audsley 91). Both are soft aperiodic tasks.

In the first set, each task T_k has as its input i_k, data provided by elements of the behaviour architecture such as sensors, behaviour, fusion or arbiter entities. When executed, T_k produces a result o_k.

$$q(o_k) = R_k(r_k, q(i_k))$$

The function R_k trades the resource availability r_k and the input quality i_k, to generate the output quality o_k of the task T_k.

For the multiple version task set, each task To_i is composed of different deepening levels. The first level generates a minimum quality response of the task. Increasing the level, the quality of the response is also increased. This task set is characterised by:

$$To_i = (LI_{i,j}, \mathrm{Im}_i).$$

Where $L_{i,j}$ is the estimated computation time of the j^{th} level of the i^{th} task and Im_i is the importance of the task.

3.2 System schedulers

The group of tasks and activities of the system are basically handled by two types of schedulers that are structured in two levels. The first level guarantees the execution of the reactive tasks while the second level assures the planning of the operations regarding the predetermined quality response.

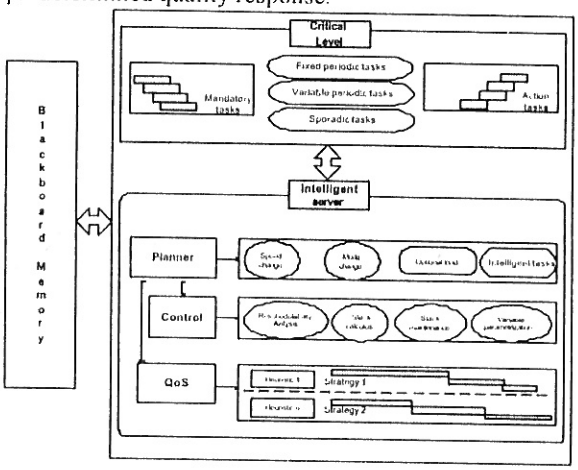

Fig. 2. Real-time architecture

The scheduled tasks of the critical level are periodic tasks. The priority assignation is based on deadline monotonic scheduling, and the schedulability analysis applied are those outlined for fixed priority pre-emptive systems (Lehoczky, 1990).

In the second level, three modules constitute the intelligent server: the planner, the QoS analyzer and the control module.

Optional tasks are executed in slack time by the slack server (Lehoczky 92, Davis 93). This is incorporated in the control module.

The planner module is responsible for analysing the

current state of the system and taking the most appropriate decisions – in co-operation with the rest of modules.
The control module reconsiders the speed depending on the requested load and re-evaluates the slack tables when requested to do so by the planning module.
The QoS analyser is responsible for assuring the execution of the vehicle requirements taking into account the quality of service.

4. QoS ANALYSER

The QoS analyser must take into account different input information to assure the required quality of the vehicle application. The planner tells the QoS module what quality is required. When the system is overloaded, maintenance of expected quality becomes impossible. In this situation, the QoS analyser informs the planner so that an adjustment (reduction/increase) in the speed and/or the load can be made. Consequently, the system relaxes and more CPU time becomes available for assuring the minimum quality required.

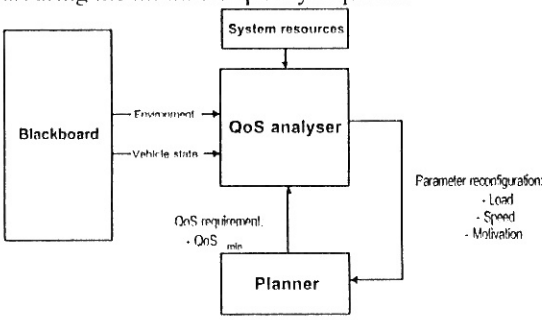

Fig. 3. QoS analyser

The global quality of the vehicle depends on the local qualities of different behaviours of the system.

$$Q_{GLOBAL} = f(Q_1, Q_2, \ldots\ldots, Q_n) \qquad (1)$$

Where Q_{GLOBAL} is the global quality of the vehicle and Qi is the local quality service of the vehicle behaviour.
One of the behaviours that influence the global quality of the vehicle is *obstacle_avoidance* behaviour. The influence of the *obstacle_avoidance* behaviour on the system is analysed below. The quality of service of this behaviour depends in turn on the quality of two processes: the quality of the building local maps, and on the quality of fusing these local maps with odometric and infrared information into global maps.
It can be stated that:

$$Q_{\substack{Obstacle \\ behaviour}} = f(Q_{LM}, Q_{GM}). \qquad (2)$$

- where Q_{LM} and Q_{GM} are respectively the quality service of the local map and the global map algorithms.
The Q_{LM} measures the certainty of existence of obstacles seen from a certain angle. This process is modelled as a flexible task because the quality of the result depends on the data input (angular readings) received from the sonar subsystem.

$$Q_{LM} = R_{LM}(\vec{r}_{cpu}, q(\vec{i}_k)) \qquad (3)$$

Where R_{LM} is the function value that takes one of the n sonar data qualities $q(i_k)$ for generating the quality of the local map.
With regard to the Q_{GM}, the process measures the degree of confidence of the position and orientation of the objects in the map. This information is fused with odometric and infrared information. The process is designed as a multiple version task as it generates global maps of varying quality – depending on the fusion algorithm used.

$$Q_{GM} = \max_{1 \le j \le LL_{max}} (q_j(t_s)) \quad / \quad t_j \le t_s \qquad (4)$$

$q_j(t)$ is the computation time of the quality level j of global map algorithms. LL_{max} is the maximum number of quality levels of each task. t_s is the slack time available

Both Q_{LM} and Q_{GM} depend on the ratio between the real time dedicated to processing the obstacles and the application obstacle processing needs (*Optn*).
The QoS analyser checks if the required application quality is guaranteed – depending on the vehicle circumstances. Specifically, it will make sure that the $Q_{obstacle\_avoidance}$ fits inside the spare space left in the system. Otherwise:

$$Q_{\substack{Obstacle \\ behaviour_k}} = Q_k = f(Q_{LM}, Q_{GM})_k \le S \qquad (5)$$

where S is the slack time available, k represent the circumstances of the vehicle
Previously, when the system was unable to assure a predetermined minimum quality it reduced the desired QoS. However, the architecture is now more flexible, because the planner can reduce the vehicle speed until enough spare capacity becomes available to assure the required quality. When the QoS analyser has enough slack time, it will apply a heuristic strategy for finding the most suitable combination of optional tasks (*Local_Map* and the *Global_map*).
The next section presents an application of the vehicle where the usefulness of the system is more clearly seen.

5. APPLICATION STUDY

The sonar system in the *Yair* vehicle (Benet 98) is used for building cheap and accurate environment maps. The maps are useful for two purposes:

- Obstacle-free navigation
- Improved environment recognition by using vision system information

We will show how the quality of the vehicle is maintained using the sonar system in an unstructured environment with obstacles.

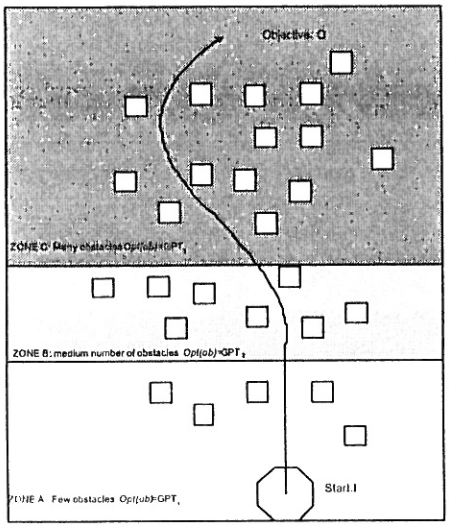

Fig. 4. Scene with obstacles

Fig. 4 shows a possible scene for the vehicle – with squares representing obstacles. In zone A, there are few obstacles, and as the vehicle approaches the objective the number of obstacles increase. For each zone the obstacle processing time needs vary, as they depend on the number of obstacle on the scene:

$$Optn(ob) = GPT_k \qquad (6)$$

k is one of the zones A, B, C. ob is the number of obstacles. GPT is a constant that represents the global processing time for the k zone

The reactive processes of the application are shown in Table 1. These processes depend on the vehicle speed. Three different speeds have been considered: 0.5, 1, and 2m/s. For all these speeds the reactive load is schedulable. Hence, the planner could switch, in run-time, from one speed to another while the schedulability is maintained. The relationship between speed and periods is: $T = \frac{d}{v}$

The attributes appearing in Table 1 correspond to the speed $v_0 = 2m/s$. It is simple to calculate the attributes for the other speeds, taking into account that the distance d is constant.

Table 1. Reactive attributes of the system.

Reactive Task	$Cm_{i,0}$	$Cf_{i,0}$	$D_{i,0}$	$T_{i,0}$
Behaviours				
Obstacle_avoid	2	1	200	200
Fusion				
Odomtric_process	1	1	50	50
Local_min_map	3	1	100	100
Arbiters				
Turn_arbiter	1	1	250	250
speed_arbiter	1	1	275	275

- Obstacle_avoidance behaviour: based on the sonar maps, decides the control actions (turn and speed) for guiding the vehicle by avoiding the obstacles in its path.
- Odometer_process: gives information regarding the position of the vehicle.
- Local_min_map (ultrasounds): builds a minimum environment map for obstacle-free guidance.
- Turn_arbiter: controls the steering wheels of the vehicle. When more than one task generates a turn command, the action with greatest motivation is applied.
- Speed_arbiter: controls the vehicle speed. The action of the task with greatest motivation is applied to the controller.

The Table 2 represents the attributes of the deliberative processes: *Local_Map* and *Global_Map*.

The *Local_Map* processes depend on the number of samples per revolution performed by the sonar system. Different quality maps Q_{LM} have been obtained (Benet, 1998). The computational cost of the map is proportional to the number of samples. For the *Global_Map* process, five quality levels have been considered. Each level identifies the degree of confidence obtained from fusing the odometer data and the sonar map.

Table 2. Optional attributes of the unbounded tasks

Deliberative Task	Quality versions	$L_{i,j}$	$Im_{i,0}$
Local_Map	LM_1 *(10 samples)*	30	*max*
	LM_2 *(20 samples)*	60	
	LM_3 *(80 samples)*	105	
	LM_4 *(120 samples)*	133	
	LM_5 *(180 samples)*	342	
Global_Map	GM_1	32	*min*
	GM_2	58	
	GM_3	90	
	GM_4	265	
	GM_5	375	

The vehicle starts from the initial situation $I=(x0,y0)$ and has to attain the objective $O=(xF,yF)$. It must avoid the obstacles represented by squares in the path and it has to reach the vehicle quality requirements:

$$Q_{obstacle} = \frac{\sum_{i=2n+1}^{2n+2}\sum_{j=1}^{m} L_{i,j}}{Opt(ob)} \cdot 100 \;/\; \sum_{i=2n+1}^{2n+2}\sum_{j=1}^{m} L_{i,j} \le S \qquad (7)$$

Where S is the slack time available, i is deliberative tasks and j is the level of each i task

The quality of service represents the ratio between the *Local_map* and *Global_map* j^{th} levels dedicated to analysing the scene – and the obstacle processing time needs for each k zone.

Although, the number of obstacles increases in the scene, the vehicle will always require the best quality service. If the number of obstacles increase, the *Opt(ob)* function will also increase, and this implies a decrease in quality (E-7). This is because the *Local map* and *Global_map* levels will not change, and the slack time

will also remain unchanged. If we want to counteract this effect, we must increase the QoS. To increase quality, slack time must be increased, and this can be achieved by reducing the vehicle speed.

6. SIMULATION

The experiments carried out focus on measuring the adaptive features introduced by the new architecture in scheduling vehicle plans with guaranteed QoS requirements. Firstly, the usefulness of speed changes for maintaining the required performances are demonstrated. Secondly, different heuristic strategies providing different QoS performances for the vehicle are designed and evaluated.

6.1 Benefit of speed changes

The advantages of speed changes are: the desired dynamic QoS of the vehicle can be guaranteed, and the temporal real-time constraints of the tasks can be relaxed in order to obtain more slack time for improving performance.
Under a traditional architecture, where the vehicle speed cannot be modified as a result of the environment and vehicle state changes, it was impossible to guarantee a dynamic QoS for the system. In these circumstances, if the vehicle load of tables 1 and 2 is executed, the QoS would decrease as the vehicle enters the zones with more obstacles.
The flexibility of the new architecture enables a reduction of the vehicle speed in order to relax the variable periodic tasks; so obtaining more slack time and hence, guaranteeing the dynamic QoS required by the application. Figure 5 shows a simulation of the vehicle load with speed changes. Initially, the speed *v=2m/s* and at this high speed the QoS is achieved. At t=1500, the vehicle arrives in B zone – which contains more obstacles. The QoS will not be reached unless the vehicle decreases speed. Therefore, the speed is changed to *v=1m/s*. In zone C, at t= 4500, the vehicle speed is reduced to *v=0.5m/s* to achieve the QoS requirements

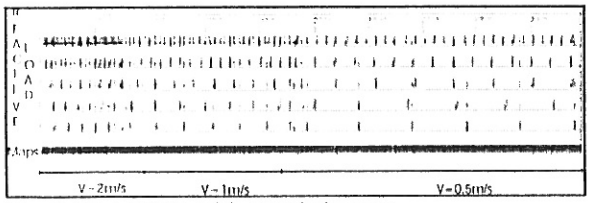

Fig. 5. Simulation with speed changes

Once enough slack for guaranteeing the performance requirements of the vehicle is obtained, which combination of different deliberative tasks will be selected? In other words, which levels L_{ij} of the equation E-7 should be attained?
Different strategies have been designed and evaluated as various combinations of deliberative tasks provide answers of varying quality. The strategies are presented below – together with a study of which provide better performances.

6.2 Boss strategy

This strategy assures the execution of the minimum level of both tasks *Local_Map* and *Global_Map*. However, the *Local_Map* uses all time that it can to execute its maximum level of quality. Any time remaining time is used by the *Global_Map* to execute its best level.
Figure 6 shows the simulation of the application. In the first A-zone, the second level of both processes is executed. Firstly, the boss *Local_Map* task tries to execute its third level. As it cannot, it concedes the option to the deepest *Global_Map* level: the second level. In the B-zone, as more time becomes available, the deeper fourth level *Local_Map* is executed: while the rest of the processing is dedicated to just executing first *Global_Map* level. In the last speed change, in the C-zone, there is yet more computing capacity. The fifth *Local_Map* level is executed before the second *Global_Map* level.

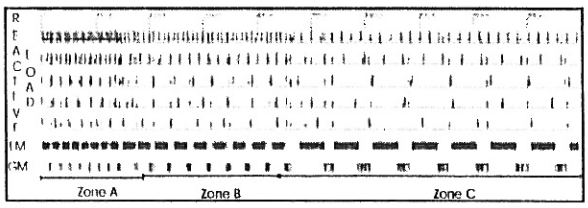

Fig. 6. Boss strategy

6.3 Balancing strategy

Besides assuring the execution of the vehicle QoS, this strategy maintains balance in the execution of the deliberative tasks levels: *Local_Map* and *Global_Map*. The time it takes the *Obstacle_avoidance* behaviour to generate a best quality map is distributed equally among the deliberative tasks.

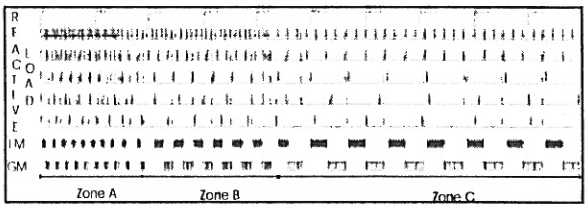

Fig. 7. Balancing strategy

The simulation showing the balanced execution of the *Local_Map* and *Global_Map* tasks can be seen in Figure 7. In zone A, the second level of both tasks is executed. More time becomes available when entering B-zone, and so the third level of both tasks can be processed. In the last part of the simulation, after the 4500 time instant, the slack has been distributed equally between

both deliberative tasks. Hence, the fourth level of the *Local_Map* and *Global_Map* has been executed.

7. EVALUATION RESULTS

The aspects that have been evaluated are firstly, The impact on the vehicle QoS of matching the speed to the environment. On the other hand, as the two strategies designed have different effect on the quality attained, it has been evaluated their influence on the vehicle QoS.

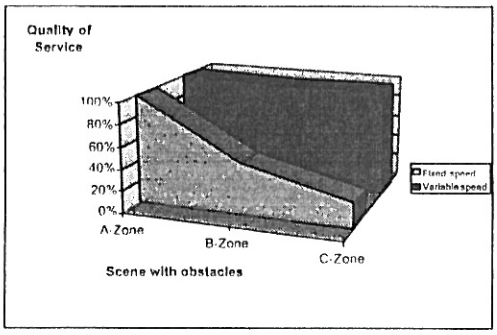

Fig. 8 Speed change influence on the QoS

Initially, QoS is identical, as there are few obstacles and the vehicle dynamics allows processing all the information at the best rate of 100% – as shown in Figure 8. However, if the system does not adapt the speed for creating more slack time for processing the obstacles when B-zone is reached, the QoS drops to 47%. However, when the system allows variable speed, a performance quality of 98% is achieved. The results are clearer when entering the C-zone. If the speed is fixed, the quality drops dramatically and reaches levels of 20%. Yet, if the speed is adapted to the vehicle obstacle processing needs, the QoS is 99%.

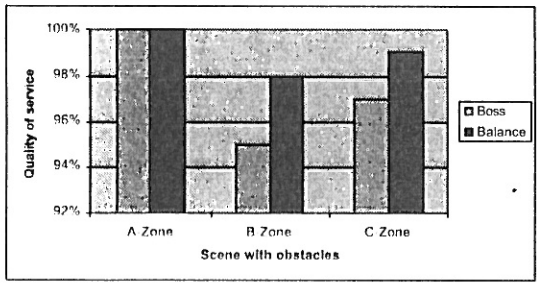

Fig. 9. Strategy effect on the vehicle QoS

Figure 9 compares the effect of the two strategies on the vehicle QoS.

In the less loaded zone, the results are similar using both the strategies. When the environment becomes more loaded, the balance strategy shows a better behaviour than the boss strategy. Besides obtaining better results, the balance strategy distributes the slack time in a balanced way, which is useful from the application point of view. This is because it is better to have medium quality global maps rather than very refined local maps but poor global maps.

8. CONCLUSIONS

An adaptive real-time architecture for complex mobile robotic applications is presented. The task model and the different scheduling levels of the architecture are specified. The different modules composing the intelligent server are detailed. A special emphasis is made on the study of the QoS analyser. The robotic application example presented shows the usefulness of the adaptive system. Different strategies for guaranteeing the vehicle QoS requirements are designed and evaluated.

Future work will pursue the following objectives: the formalisation of the mode changes of the vehicle operations, and the specification of the QoS function values of the remaining deliberative vehicle behaviours. Finally, from the implementation point of view, the system must be embedded under a real-time operating system (RT-Linux).

REFERENCES

Arkin, R. C. (1998). "Behaviour based Robotics" Cambridge MIT Press 1998.

Audsley, N.C., Burns, A., Richardson M.F. and Wellings, A.J. (1991). Incorporating Unbounded Algorithms into Predictable Real-Time Systems. Real-Time Systems Research Group. Department of Computer Science. University of York, UK. Report number RTRG/91/102.

Benet, G. Blanes, F. Martínez, M. Simó, J. (1998). A Multisensor Robot Distributed Architecture. IFAC Conference INCOM'98. Metz-Nancy. June 1998.

Bonasso P. (1997) "Experiences with an Architecture for Intelligent Reactive Agents" Journal of Experimental and Theoretical Artificial Intelligence 1997.

Botti, V., Barber, F., Crespo, A., Onaindía, E., García-Fornes, A., Ripoll, I., Gallardo, D., and Hernandez, L. (1995) A Temporal Blackboard for a Multi-Agent Environment. *Data and Knowledge Engineering*. North Holland Elsevier 15 (3).

Buendia, F. Hassan, H. Simó, J. Crespo, A. (1999) Real-time Systems for Mobile Robot Applications based on Behavioural Model. 23 IFAC/IFIP Workshop in Real-time Programming. Saarland Germany. May 1999.

Davis, R.I., Tindell, K.W., and Burns, A. (1993). Scheduling Slack Time in Fixed Priority Pre-emptive Systems. Proc. Real-Time Systems Symposium, Raleigh-Durham, North Carolina, December 1-3, 222-231

Garcia A Hassan, H. Crespo, A.,(1996) "Strategies for

Scheduling Optional Parts in Intelligent Real-time Environments." IEEE Journal on Systems Architecture. Volume 42. pp. 391-407. 15/12/96.

Gat E. et al(1998) "Three Layer Architectures" Artificial Intelligence and Mobile Robots. The MIT Press. 1998

Hassan, H. Buendia, F. Simó, J. Crespo, A, Benet, G.(1999) "A Real-time Model of Flexible Tasks for Mobile Robotic Applications based on a Behavioural Architecture." 7th. IEEE International Conference on Emerging Technology in Factory Automation. Barcelona'99.

H. Hexmoor, M. Lafary, M. Trosen, 1999. "Towards Empirical Evaluation of Agent Architecture Qualities", In Agent Theoresi, Architectures, and Languages (ATAL-99), Orlando, Florida.

Kortenkamp D. Et al.(1998) "Three NASA Application Domains for Integrated Planning, Scheduling and Execution.". IEEE Symposium on Intelligence in Automation and Robotics. 1998

Lehoczky J. (1990). Fixed Priority Scheduling for Periodic Tasks Sets with Arbitrary Deadlines. Proc. Of the 11th Real-Time Systems Symposium. December, 201 -- 209, IEEE Computer Society Press.

Lehoczky J., and Ramos-Thuel S. (1992). An Optimal Algorithm for Scheduling Soft-Aperiodic Tasks in Fixed-Priority Preemptive Systems. Proc. Real-Time Systems Symposium, Phoenix, Arizona, December 2-4, 110 -- 123, IEEE Computer Society Press.

Liu, J.W.S., Lin, K.J.L, Shih, W.K., Yu, A.C., Chung, J.Y., and Zhao, W. (1991) Algorithms for Scheduling Imprecise Computations. Computer IEEE May 1991.

Liu, J.W.S., Nahrstedt, K. Hull, D., Chen, S., Baochum, Li.(1997) EPIQ QoS Characterization, Internal Report, Dept. Computer Science, Univ. of Illinois, Urbana-Champaign.

Michaud F. (1999) "Managing Robot Autonomy and Interactivity Using Motives and Visual Communication" International Conference on Autonomous Agents. Seattle 1999.

Shreckenghost et al. (1998) "Three Tier Architecture for Controlling Space Life Support Systems" IEEE Symposium on Intelligence in Automation and Robotics. 1998.