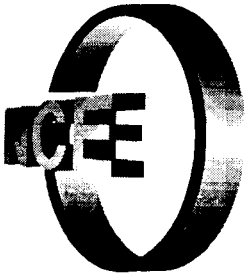

Relatório Técnico

Núcleo de Computação Eletrônica

CONTROLAB MUFA: A Multi-Level Fusion Architecture for Intelligent Navigation of a Telerobot

E. P. L. Aude, G. H. M. B. Carneiro
H. Serdeira, J. T. C. Silveira
M.F. Martins, E. P. Lopes

NCE - 06/99

Universidade Federal do Rio de Janeiro

CONTROLAB MUFA: A Multi-Level Fusion Architecture for Intelligent Navigation of a Telerobot

E.P.L. Aude*, G.H.M.B. Carneiro**, H. Serdeira*
J.T.C.Silveira*, M.F.Martins*, E.P.Lopes***

NCE/UFRJ* IME/RJ** IM/UFRJ***

NCE/UFRJ, P.O. Box 2324, Rio de Janeiro - RJ, 20001-970, Brazil,
e-mail: elaude@nce.ufrj.br

Abstract

This paper proposes a MUlti-level Fusion Architecture (MUFA) for controlling the navigation of a tele-commanded Autonomous Guided Vehicle (AGV). The architecture combines ideas derived from the fundamental concepts of sensor fusion and distributed intelligence. The focus of the work is the development of an intelligent navigation system for a tricycle drive AGV with the ability to move autonomously within any office environment, following instructions issued by client stations connected to the office network and reacting accordingly to different situations found in the real world. The modules which integrate the MUFA architecture are discussed and results of some simulation experiments are presented.

1 Introduction

Two fundamental concepts have made possible the design of intelligent autonomous mobile robots which are able to act in a dynamic system such as the real world. The first one is the use of sensor fusion techniques [1] which enable the system to have more consistent environment information and to achieve better reasoning based on this information. The second important contribution is the implementation of the different robot skills in a distributed fashion [2], which leads to a system consisting of more specialized and optimized parts.

The MUlti-level Fusion Architecture (MUFA) combines these two important concepts for controlling the navigation of a tele-commanded Autonomous Guided Vehicle (AGV). In this paper, this architecture is used in the development of an intelligent

navigation control system for an AGV capable of moving through any type of office environment and of reacting accordingly to different unexpected situations found in the real world. The AGV moves autonomously and follows instructions issued by any station connected to the office network. Figure 1 shows the basic operation scheme of the CONTROLAB AGV.

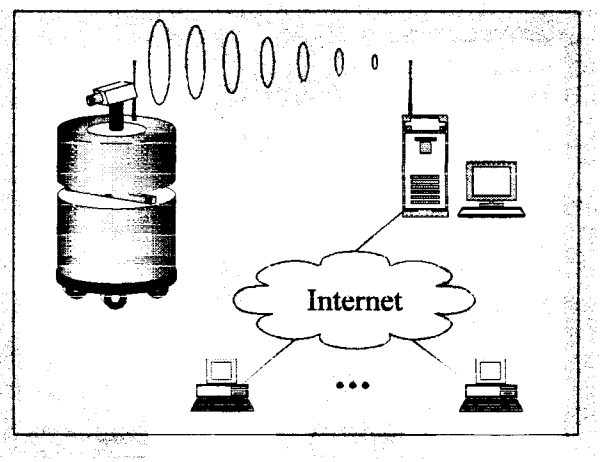

Figure 1: Basic Operation of the CONTROLAB AGV

Figure 2 shows the overall MUFA architecture of the AGV intelligent navigation system. It consists of the following modules:

1. **Autonomous Guided Vehicle**: a moving robot equipped with a radio transceiver, a video camera, sonar sensors and hardware/software resources for storing and processing information;

2. **Control System:** integrates all the controllers used for commanding the direction and speed of the AGV movement and the movements of the AGV structures holding navigation sensors;

3. **Architect:** an object-oriented software tool which supports the editing of the environment floorplan to be sent to the AGV;

4. **Request Server:** responsible for: organizing requests for the AGV operation sent by client stations; sending to the AGV the floorplan description and the client requests; receiving from the AGV image information on its operation; and sending this information to the client stations;

5. **Trajectory Planner:** an on-board software module which establishes a trajectory to be followed by the AGV from its current position to the desired destination considering the

floorplan description;

6. **Intelligent Obstacle Avoidance System:** an on-board sub-system which detects the presence of obstacles close enough to the AGV, using sensor fusion and arbitration on information produced by the Sonar System and by the Vision System, and defines, in these cases, the AGV trajectory based on rules;

7. **Global Position System:** an on-board sub-system that defines the current AGV location;

8. **Intelligent Supervisor System:** an on-board sub-system that, using information produced by the Intelligent Obstacle Avoidance System and the Global Position System, takes decisions that affect the AGV navigation depending on whether the desired path is free, has unknown obstacles or is blocked;

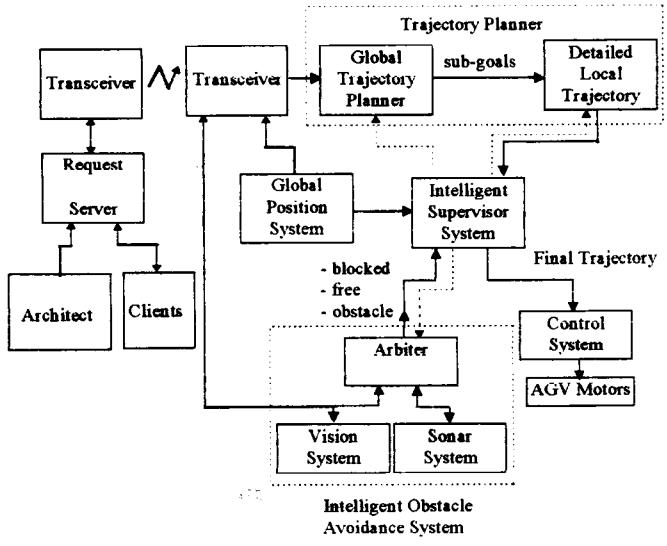

Figure 2: The MUFA Architecture

The AGV has a priori knowledge of the environment in which it should travel. It stores its description as a floorplan and a derived Connectivity Graph. The floorplan is produced by the Architect and is supplied to the AGV by the Request Server. At the start of its operation, the AGV also receives information on its initial room within the environment floorplan.

The Request Server receives from the several client stations orders for the AGV. An order consists of the destination room identification and the desired final location within this

room. Requests are stored in a queue and a new request is only sent to the AGV after completion of the previous one.

In the application currently under consideration, the CONTROLAB AGV receives from the Request Server a request to go to a particular location within the environment. The Trajectory Planner then defines the best way to go from the AGV current position to the desired destination. During the trip, the AGV Intelligent Obstacle Avoidance System uses vision and sonar sensor information to detect the presence of previously unknown obstacles. When this occurs, the Arbiter uses a set of rules to change the original route in oder to avoid the obstacles. In addition, when a given segment of the global trajectory to be followed is found to be blocked, the Global Trajectory Planner is used to redefine an alternative route starting from the AGV position when the blocking situation was detected

In Section 2 of this paper, the AGV mechanics, hardware components and control system are presented. Section 3 briefly describes Architect as an object-oriented tool for the generation of the AGV work environment description. It also discusses the implementation of the Request Server, the techniques used for wireless communication between the AGV and the Request Server and the adopted approach used for compressing image information transmitted by the AGV to the Request Server.

In Section 4, the Trajectory Planner subsystem is described. Both the procedure used for defining a global trajectory on a Connectivity Graph derived from the environment floorplan and the operation of the Detailed Local Trajectory Planner are discussed. Section 5 discusses the implementation of the Intelligent Obstacle Avoidance System which consists of a Vision System capable of detecting the presence of obstacles ahead of the AGV; a Sonar System which may give more precise information on the distance of an obstacle to the AGV; and a Sensor Fusion and Arbiter Subsystem which takes a final decision in relation to the presence or not of obstacles on the AGV way, considering the information produced by the other two subsystems, and applies a set of rules to re-define the AGV trajectory in the presence of unexpected obstacles.

Section 6 describes the Intelligent Supervisor System and presents results of simulation experiments showing the behaviour of the AGV in several distinct situations. Finally, Section 7 presents the main conclusions of the paper and directions for future work.

2. The AGV Description

The AGV has a cylindrical body. Its diameter is 40 cm long and its height is 91 cm. It is a tricycle drive with two fixed wheels and a steering wheel. The two fixed wheels have a common axis but are driven by independent DC motors which provide independent velocity control. Angular velocity is measured through incremental encoders. The steering wheel can rotate around a vertical axis and is commanded by a DC motor to define the AGV movement direction. An incremental encoder is also used to measure the angular velocity around the vertical axis.

A black and white wide-angle camera is placed on top of the AGV body. It can rotate around a vertical and a horizontal axis under the command of two step motors. This freedom of movement allows the camera to "see" low and lateral obstacles located near the AGV body. On top of the AGV, there is also an antenna to allow wireless communication with the Request Server.

Around the AGV waist, which is located 50 cm above the AGV basis, there is a 6 cm wide opening. Inside this opening,there is a 30 cm long horizontal bar holding three 40 Khz sonar transmitter/receiver pairs. The first pair is placed at the middle of the bar and the other two at its ends. This set of sensors allows the detection of obstacles around the AGV since the horizontal bar can move around the AGV body vertical axis under the command of another step motor. Finally, the AGV is also equipped with a fiber optic gyroscope for the measurement of its angular rotation.

The AGV Control System controls the angular velocity of the two fixed wheels and the angular position of the steering wheel through a multivariable LQG control algorithm [3]. In addition, the AGV Control System has additional single-input single-ouput controllers for the rotation angle of the horizontal bar which holds the sonar sensors and the video camera horizontal and vertical movements.

The AGV desired trajectory is described by the velocity vector at its geometric center. From this information, the values of the velocities at the fixed wheels and the angle to be applied to the steering wheel are determined and used as set-points by the controller when steering the AGV.

3 The Architect Module, the Request Server and Communication Techniques

3.1 The Architect Module

Architect has been designed to produce a description of generic floorplans. A textual format has been adopted to describe practically any kind of indoors environment. Architect works as a floorplan graphic editor, supporting commands for creating, removing and modifying floorplan elements and for naming the floorplan rooms. Architect has been implemented in C++ (Borland C++ Builder) for Windows 95. In its initial version, Architect handles four different types of rectangular elements with integer coordinates: **FREE AREAS, WALLS, DOORS** and **BLOCKED AREAS.**

3.2 The Request Server Structure and Operation

The Request Server is responsible for sending information to the AGV on the orders issued by the client stations and on the environment description produced by the Architect. It receives the end of task message as well as image and position information from the AGV which may be transmitted to the client stations in order to allow remote monitoring. The implementation of the required multicasting capability wthin the Request Server is based on a protocol proposed by Deering [4] for the Internet. Figure 3 shows the basic functionality of the Request Server and its interactions with other modules of the MUFA architecture.

The Request Server has also been implemented in C++ (Borland C++ Builder) for Windows 95. Figure 4 shows a summarized diagram of classes and objects of the client/server system. Both the client and the server have similar diagrams.

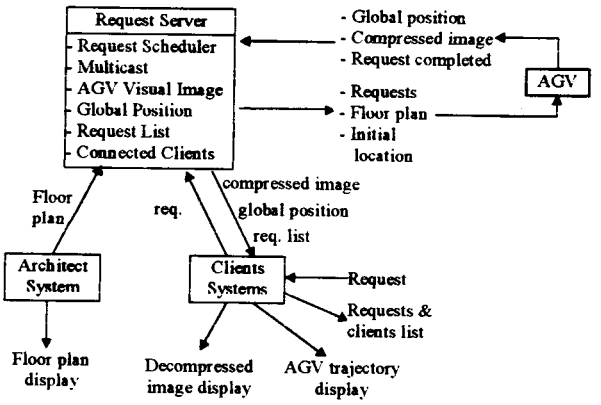

Figure 3: Request Server Functionality

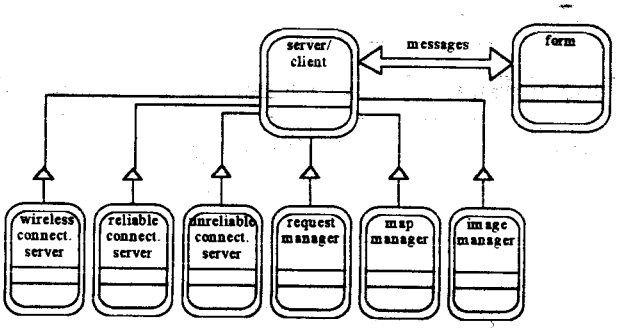

Figure 4: Diagram of Classes and Objects

In Figure 4, *form* represents the screen with text windows, image windows, buttons and menus. The *wireless connection server* performs the full communication protocol with the AGV. The *reliable connection server* manages the connections with each client implemented through sockets using TCP/IP. The *unreliable connection server* manages faster connections that cannot ensure that correct messages will arrive at the destination and that the original message order will be preserved. Due to its low overhead, this implementation may be suitable for some real time applications. The *request manager* accepts the inclusion of new requests by the clients and also the elimination of requests which have not yet been sent to the AGV. Currently, requests are served in FIFO order. The *map manager* is able to read a floorplan produced by the Architect module and to represent on it the current AGV position, which is sent by the AGV. This information is also sent through multicasting to the clients. The *image manager* is activated whenever a compressed image is received by the server through wireless communication with the AGV or by the client through the Internet (image sent through multicasting by the server).

Figure 5 shows a snapshot of the system presentation screen on the server.

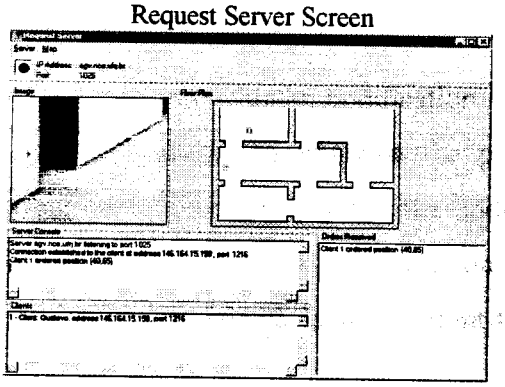

Figure 5: Snapshot of the Server Screen

3.2 Communication Techniques

This section describes the techniques used for wireless communication between the Request Server and the AGV and for image compression by the AGV and decompression at the server and client stations before display.

3.2.1. Wireless communication

The equipment used for wireless communication, Radiometrix RPC-433-A: IC+BIM, can reliably transmit data in packets of up to 27 bytes within a range of 30 m at 40 kbits/s half-duplex.

RPC is not fault-tolerant. When an error is detected, it simply does not send the packet to the host. To try to overcome this limitation, an ARQ (Automatic Repeat Request) stop-and-wait protocol [5], which detects an error and asks for data retransmission, has been used in the Data Link Layer. With this technique reliable and low cost communication is achieved.

3.2.2 Real time image compression and decompression

The approach adopted for implementing real time video image compression and decompression follows the H.263 specification [6].

The AGV image is represented by a 242 x 199 array of pixels with gray levels in between 0 (black) and 255 (white). This format is transformed into QCIF (Quarter Common Intermediate Format) [7] with 176 x 144 pixels by discarding the leftmost columns and the bottom lines of the image array. There are three types of frames which can be produced by compression techniques: the I (intra mode) frame, which uses only spatial compression; the P (predicted) frame with time and spatial compressions using the previous encoded image (I or P) as a reference; and the B (bi-directional) frame, which is based on two P frames, the previous encoded P frame and the current one under codification.

From the different options available to improve the performance of the image compression process, only the arithmetic coding [8] and the codification of PB frames (P frame + B frame) proved to be amenable for real-time applications. In addition, the use of I frames or P frames only has also been considered. It is important to note that as both P and PB frames depend on previous images, it is not advisable to transmit only these frames since if any error takes place, all following frames may be corrupted. To overcome this problem an

I frame is sent from time to time to synchronize the receiver.

Table 1 shows experimental results for the evaluation of compression schemes on a Pentium@233 Mhz computer. All input images are compliant with QCIF and in all measurements the same sequence of 50 images produced by the AGV camera when moving inside a room has been used.

Considering that the wireless communication in use transmits data at 40kbits/s, Table 1 shows that this is sufficient to cope with the transmission rates produced by all compression schemes. However, for real-time applications, a key factor is the image transmission rate. Regarding this aspect, the compression technique based only on I frames is the one that produces by far the best result. Therefore, it has been adopted within the CONTROLAB AGV regardless of the bad results it produced for the compression rate.

	1I+4P	**1I+9P**	**50 I**	**PB**	**Ar.Cod.**
Elapsed Time (s)	120	129	34	195	120
Compr. rate	31,18	44,34	21,64	64,52	30,80
Tx. rate (bits/s)	4063	2659	20664	1209	4111
Images/s	0,47	0,39	1,47	0,26	0,42

Table 1: Evaluation of Image Compression Schemes

At the reception end of the image transmission, a variable length buffer is used to store the compressed image since its size is not known a priori.

The image decompression module is multithreaded, operates under Windows 95 and has been implemented using Borland C++ Builder. When the Request Server process starts, three threads are created. They work cooperatively and share data between them. To solve synchronization problems and avoid race conditions, semaphores are used under the producer-consumer scheme. The first thread becomes ready to run whenever a new packet arrives. It stores in a buffer all arriving packets until an end-of-message packet is received. The buffer size is big enough to store several compressed image frames. The second thread becomes ready to run whenever the buffer holds at least one compressed frame. It performs the H.263 decompression algorithm. Finally, the third thread becoems ready to run whenever a new decompressed image frame is produced. Its function is to show s this frame on the screen.

4 Trajectory Planning

The execution of each motion command by the AGV consists of two phases: the definition of the trajectory to be followed and the AGV movement along this trajectory avoiding unknown obstacles detected on the way. The trajectory definition task is described in this section and is performed by two sub-systems: the Global Trajectory Planner and the Detailed Local Trajectory Planner.

4.1 Global Trajectory Planning

The Global Trajectory Planner is responsible for generating and analysing all possible paths from the AGV original location to the desired final location.

As the Architect module, the Global Trajectory Planner also works with rectangular areas. Threfore, any non-rectangular ROOM is divided into several rectangular FREE AREAS. DOORS are generated to connect these FREE AREAS. From this structure containing FREE AREAS and DOORS, the Global Trajectory Planner creates a Connectivity Graph. Figure 6 illustrates the FREE AREA and DOOR structure and the Connectivity Graph generated for a particular floorplan.

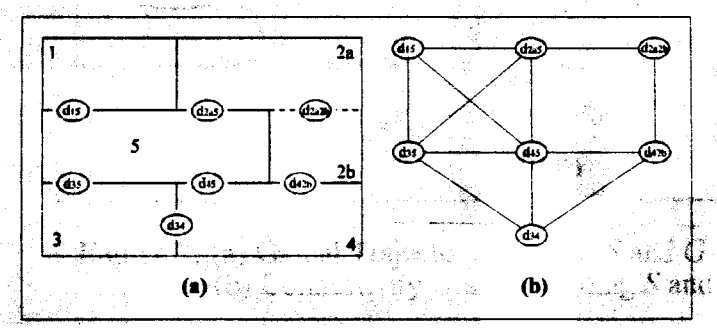

Figure 6: (a) Environment Free Areas and Doors
(b) Connectivity Graph

Within the Connectivity Graph, nodes are associated with DOORS. An edge connects two nodes whenever the DOORS associated with them open to a common FREE AREA. Edge weights are given by the product of the distance between the centers of the DOORS and the common FREE AREA weight.

Let us consider *S* as the initial AGV location in the FREE AREA 1, and *G*, located in the FREE AREA 2b, as the destination point. The Global Trajectory Planner generates the

Connectivity Graph adding two nodes associated with *S* and *G* and then calculates the minimum-cost path between *S* and *G*. Figure 7 illustrates the previous floorplan with the inclusion of *S* and *G*, placed in the FREE AREAS 1 and 2b, respectively. The assigned weights indicate that FREE AREAS 1 and 3 should be avoided and that it should be given priority to using FREE AREA 5, which represents a corridor. All possible connections between DOORS are shown both in the topological scheme and in the Connectivity Graph. The best path - ***S***, $\mathbf{d_{15}}$, $\mathbf{d_{2a5}}$, $\mathbf{d_{2a2b}}$, ***G***, with a total cost of 294.3 - is emphasized.

During the operation of the AGV, the Global Trajectory Planner may be requested to recalculate the global trajectory by the Intelligent Supervisor System whenever the AGV sensors detect that previously unknown obstacles are completely blocking one of the FREE AREAS that have to be crossed by the AGV in its trajectory from source to destination. In this case, the Global Trajectory Planner redefines the Connectivity Graph by introducing a BLOCKED AREA element, which behaves like a WALL in the floorplan and finds again the minimum-cost global path connecting the AGV current location to the desired destination point.

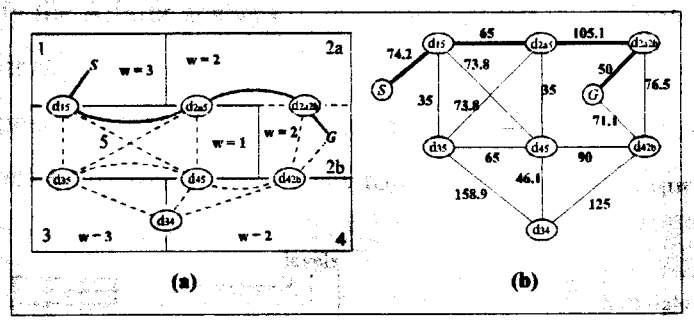

Figure 7: (a) Global Trajectory between *S* and *G*
(b) Connectivity Graph including *S* and *G*

4.2 Detailed Local Trajectory Planning

Each edge of the path defined by the Global Trajectory Planner in the Connectivity Graph defines the start and end points of a global trajectory segment. The precise trajectory to be followed by the AGV between each of these pairs of points is defined by the Detailed Local Trajectory Planner using a rule-based PFIELD algorithm proposed by Aude [9]. This trajectory is followed by the AGV whenever unexpected obstacles are not detected.

5 The Intelligent Obstacle Avoidance System

The Intelligent Obstacle Avoidance System consists of three parts: the Vision System, the Sonar System and the Arbiter.

5.1 The Vision System

The Vision System works on image information captured by the video camera. It is based on the principle that objects near the lower border of the image frame are closer to the AGV while those near the top border are farther away [10]. The image processing performed by the Vision System aims at detecting borders which represent obstacles sitting on or close to the AGV path and organizing this information efficiently to simplify the Arbiter work.

This processing starts with the application of an intelligent threshold which is able to take into consideration the scene illumination conditions [3]. Following, edge detection is performed with the use of a Sobel filter which is applied bottom-up and from left to right to the image. The next step is to divide the image in 5 vertical regions as shown in Figure 8. The goal is to simplify the Arbiter analysis of the image information. Each region has a fuzzy meaning within the image. Region 2, the central one, represents the straight ahead path. Regions 1 and 3 represent areas slightly to the left and to the right, respectively. Finally, regions 0 and 4 are related to areas farther away to the left and to the right, respectively.

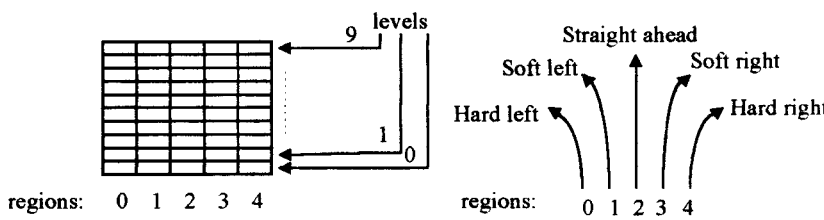

Figure 8: Image division inregions and levels

Each vertical region is divided in 10 horizontal levels, which give information on the proximity of an edge to the AGV. When level changes occur within the same vertical region, this information is stored in a linked list for the Arbiter use. For each level change, a new element is added to the list.

The linked list is filled by scanning the image from left to right and determining the region and level of every image dot. A new node is created whenever a level change occurs.

The node structure indicates the initial column where that level is present. There are as many nodes as level changes within the region.

5.2 The Sonar System

The Sonar System consists of three pairs of sonar transducers attached to a horizontal bar (the sonar bar) which is driven by a step motor to be able to rotate up to 360° around the AGV central vertical axis. The use of a pair of transducers, one for transmission and the other for reception in each set, allows distances as small as 35cm to be measured. Therefore, obstacles as close as 25cm to the AGV border will be detected by the central pair of transducers.

5.3 The Arbiter

This system analyses the information generated by the Sonar and the Vision Systems and decides whether there are previoulsy unknown obstacles close enough to the AGV. Three types of decisions can be taken:

- there is no obstacle and, threfore, no trajectory change takes place;
- there is an obstacle and the new trajectory is defined by the Arbiter according to some rules;
- there is a total blockage and the task to redefine a new global trajectory is handed to the Global Trajectory Planner by the Intelligent Management System.

S1	S2	S3
211	212	214

Figure 9: Blockage detection

Figure 9 shows the image generated by the Vision System and the measurements produced by the Sonar Systems when a blockage is found. The horizontal line crossing the whole image near the bottom frame border indicates that there is no free of obstacle path to be followed. This indication is reinforced by the very similar measurements produced by the three sonars.

When the AGV approaches an obstacle folowing the trajectory dictated by the PFIELD algorithm, the Vision System will report the presence of an edge near the bottom of the image frame. If the sonar bar is parallel to the obstacle, the Sonar System will be able to measure the distance to the obstacle. Otherwise, the Sonar System fails and the Arbiter requests the rotation of the sonar bar until it gets parallel to the obstacle. From then on, if along the PFIELD flow direction the path is blocked, the AGV will be commanded to move along a parallel line to the obstacle in the direction indicated by the Vision and the Sonar Systems as the preferential one. This verification is performed before each AGV moving step. When the AGV is close to a corner, the Arbiter has to find out if there is room for the AGV to turn the corner. Several situations may occur as shown in Figure 10.

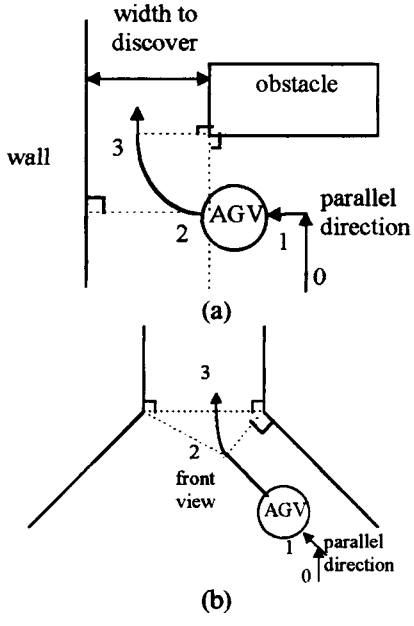

Figure 10: The AGV Turning an Obstacle Corner

In Figure 10(a), both the Sonar and the Vision Systems detect the presence of the wall and the distance to it is supplied by the Sonar System. Therefore, the Arbiter can decide if the AGV can turn the corner.

On the other hand, in Figure 10(b), the Sonar System is unable to measure the distance to the opposite wall. So, this task is assigned to the Vision System by the Arbiter. To do that, the AGV is commanded to turn round the corner. At each step, the AGV camera is required to rotate around a vertical axis. When the Arbiter detects both corners on the same horizontal line within the image, the Vision System is required by the Arbiter to calculate the entrance width.

In any case, if there is room for the AGV to turn the corner it does it on a circular trajectory until a point is reached where the Sonar System can measure the distance to the same corner again. From then on, the AGV follows a parallel route to the obstacle side.

6 The Intelligent Supervisor System

The Intelligent Supervisor System is responsible for defining the final trajectory to be followed by the AGV. It consults the Intelligent Obstacle Avoidance System to get information on the situation of the trajectory currently followed by the AGV. It can be reported as free, blocked or partially blocked by an obstacle. In the first case, the final trajectory is dictated by the Detailed Local Trajectory Planner using the rule-based PFIELD algorithm. In the second case, the Intelligent Supervisor System requests the Global Trajectory Planner to reroute the global trajectory. Finally, in the third case, the final AGV trajectory is defined by the Arbiter of the Intelligent Obstacle Avoidance System.

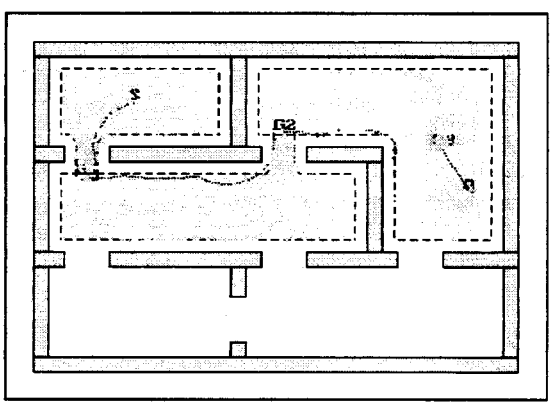

Figure 11: All Possible Trajectories Generated by the PFIELD Algorithm

All examples shown in Figures 11 to 14 are related to the task of defining the AGV trajectory to go from the start position (S) to the goal position (G) placed in the L-shaped room. The sub-goals along the trajectory have been defined by the Global Trajectory Planner

as decribed in Section 4. Figure 11 shows all possible trajectories generated by the rule-based PFIELD algorithm. Figure 12 shows the trajectory followed by the AGV when no unexpected obstacle is present on its way. Therefore, this trajectory is totally defined by the rule-based PFIELD algorithm.

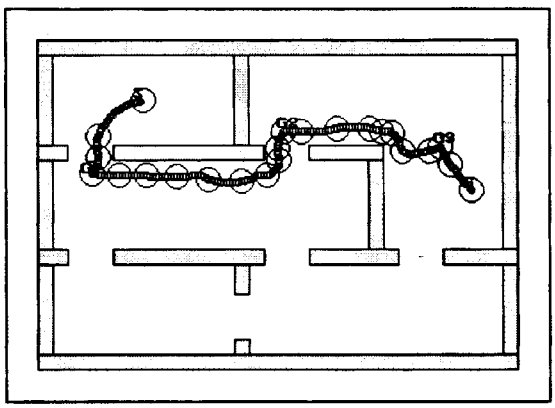

Figure 12: AGV Following the PFIELD Generated Trajectory

Figure 13 shows the trajectory followed by the AGV when obstacles are present along its previously defined trajectory. In this case, the Arbiter modifies the original trajectory by using the sensor fusion techniques and the rule-based decision scheme discussed in Section 5.

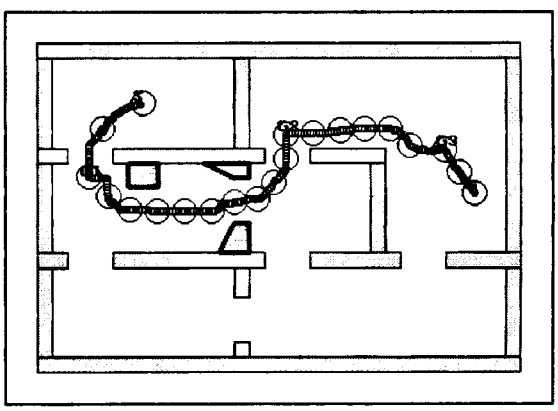

Figure 13: AGV Trajectory Avoiding Obstacles

In Figure 14, the previously defined trajectory for the AGV is found to be totally blocked when the AGV is already trying to reach the goal. In this case, the Intelligent Supervisor System requests the Global Trajectory Planner to redefine the global route between the current AGV position and the goal. The resulting trajectory is shown in Figure 14.

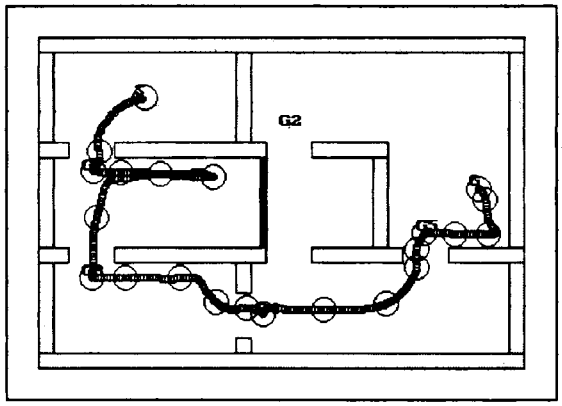

Figure 14: AGV Trajectory when a Blockage is Found

7 Conclusions and Future Work

A MUlti-level Fusion Architecture (MUFA) for solving the problem of robot navigation in uncertain environments is proposed. The benefits of sensor fusion and distributed intelligence are merged in this architecture. This architecture is applied to the development of an intelligent navigation control system for a tele-commanded AGV which is able to avoid unexpected obstacles in its trajectory using both vision and sonar sensor information.

Simulation experiments have shown promising results in the ability of the system to cope with different real world situations. Future work will focus on the use of this architecture with multiple-robots and on the introduction of evolutionary algorithms in most of the architecture sub-systems.

8 Acknowledgements

The authors would like to thank the support given by CNPq/RHAE, Brazil, to the development of this reasearch work.

9 References

[1] Kam, M., Zhu, X., Kalata, P., *Sensor Fusion for Mobile Robot Navigation*, Proceedings of the IEEE, Vol. 85, No. 1, January 1997, pp. 108-119

[2] Brooks, R.A., *A Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation, RA-2, March 1986, pp. 14-23

[3] Aude, E.P.L., Silva, F.A.B., Lopes, E.P., Serdeira, H., Martins, M.F., *CONTROLAB: An Integrated System for Intelligent Control of Robot Arms*, Proc. 1995 IEEE Conference on Robotics and Automation, Nagoya, Japan, May 1995

[4] Deering, S., *Host Extensions for IP Multicasting*. STD 5, RFC 1112, Stanford University, August 1989

[5] Bertsekas,D.P., Gallager,R., *Data Networks*. 2nd edition. Prentice Hall, Inc. 1992

[6] ITU - International Telecomunication Union. *ITU-T Recommendation H.263. Line Transmission of Non-Telephone Signals. Video Coding for Low Bitrate Communication*, 1995

[7] *CCIR (International Radio Consultative Committee) Recommendation 601 (Also Resolutions and Options)* Volume XI - Part 1 - Broadcasting Service (Television). pp 95-104. 1990

[8] Witten, I.H., Neal, R.M. and Cleary, J.G., *Arithmetic Coding for Data Compression.*, Comm. ACM, vol. 30, no. 6, 1987, pp. 520-540

[9] Aude, E.P.L., Silveira, J.T.C., Silva, F.A.B., Lopes, E.P., Serdeira, H., Martins, M.F, *CONTROLAB: Integration of Intelligent Systems for Speech Recognition, Image Processing and Trajectory Control with Obstacle Avoidance Aiming at Robotics Applications*, Proc. SPIE's Int'l Symposium on Intelligent Manufacturing, Pittsburgh, Pensylvannia, USA, October 1997

[10] Gomi, T., Ide, K., *Vision Based Navigation for an Office Messenger Robot*, Proceedings of the IROS'94, Munich, Germany, September 1994