Integration of Intelligent Systems and Sensor Fusion within the CONTROLAB AGV

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

[a]NCE - Federal University of Rio de Janeiro, Brazil
[b]IM - Federal University of Rio de Janeiro, Brazil

ABSTRACT

This paper discusses the integration of intelligent systems and the use of sensor fusion within a Multi-Level Fusion Architecture (MUFA) designed for controlling the navigation of a tele-commanded Autonomous Guided Vehicle (AGV). The AGV can move autonomously within any office environment, following instructions issued by client stations connected to the Internet and reacting accordingly to different situations found in the real world. The modules which integrate the MUFA architecture are discussed and special emphasis is given to the role played by the intelligent obstacle avoidance procedure. The AGV detailed trajectory is firstly defined by a rule-based PFIELD algorithm from sub-goals established by a global trajectory planner. However, when an unexpected obstacle is detected by the neural network which performs the fusion of information produced by the vision system and sonar sensors, the obstacle avoidance procedure uses a special set of rules to redefine the AGV trajectory. The architecture of the neural network used for performing the sensor fusion function and the adopted set of rules are discussed. In addition, results of some simulation experiments demonstrate the ability of the system to define a new global trajectory when unexpected blocked regions are detected.

Keywords: Autonomous Guided Vehicles, Sensor Fusion, Distributed Intelligence, Obstacle Avoidance, Trajectory Planning

1. INTRODUCTION

In the development of intelligent autonomous mobile robots, two fundamental concepts have made possible the design of 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 idea of implementing the different robot skills in a distributed fashion[2], which leads to a system consisting of more specialized and optimized parts.

The work discussed in this paper benefits from these two important concepts by merging them in the proposition of a MUlti-level Fusion Architecture (MUFA) for controlling the navigation of a tele-commanded Autonomous Guided Vehicle (AGV). It is focused on the development of an intelligent navigation control system for an AGV with the ability of moving through any type of office environment, having the capability to react 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 scheme of the CONTROLAB AGV operation.

Figure 2 shows a block diagram of the overall MUFA architecture, which consists of the following hardware or software 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. **Client/Server Subsystem:** the client stations use the Internet to issue orders to the AGV and the Request Server organizes the client orders received by the Internet, uses wireless communication to both send the floorplan description and the client requests to the AGV and to receive from the AGV information which allows the remote monitoring of its operation. This received information is sent to the clients by the Internet;

Part of the SPIE Conference on Mobile Robots XIV
Boston, Massachusetts • September 1999
SPIE Vol. 3838 • 0277-786X/99/$10.00

4. **Architect:** a special client, implemented as an object-oriented software tool, which supports the editing of the environment floorplan to be sent to the AGV;
5. **Trajectory Planner:** an on-board software module which is able to establish a trajectory to be followed by the AGV from its current position to the desired destination considering only the previuously defined obstacles in the floorplan;
6. **Intelligent Obstacle Avoidance System:** an on-board sub-system which detects the presence of obstacles close enough to the AGV, using a neural network structure to perform the sensor fusion on information produced by the Sonar System and by the Vision System, and defines the AGV trajectory,when an unexpected obstacle is detecetd,using a set of rules;
7. **Global Position System:** an on-board sub-system that is able to define the current AGV location;
8. **Intelligent Supervisor System:** an on-board sub-system that, based on information coming from 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;
9. **Communication Manager:** transmits the information on the received AGV orders to the Trajectory Planner and receives information on the AGV trajectory, global position and compressed captured image to be sent to the Request Server;

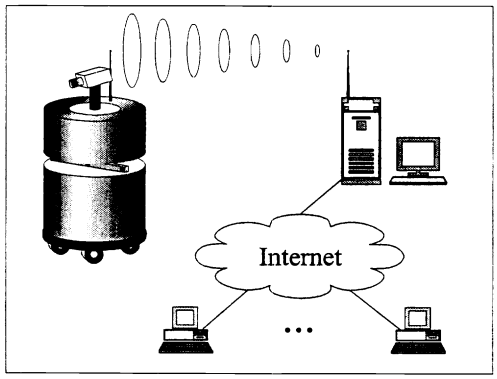


Figure 1: Basic Operation of the CONTROLAB AGV

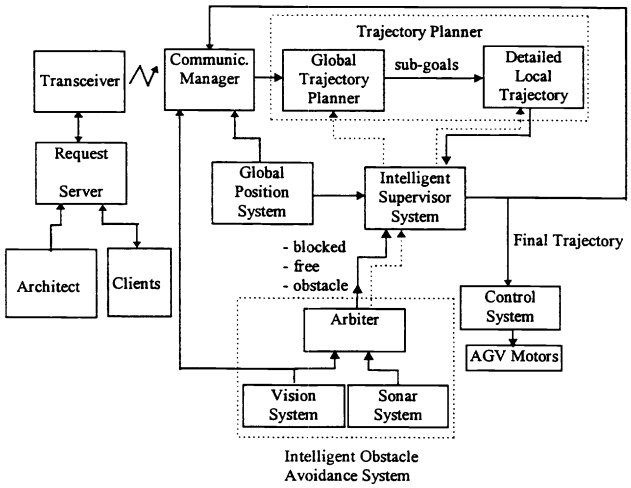


Figure 2: The MUFA Architecture

The AGV has a priori knowledge of the environment in which it should travel. It stores a description of this enviroment consisting of a floorplan and a derived Connectivity Graph. The floorplan description 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.

In this application, the Request Server receives orders to the AGV from the several client stations connected to the Internet. An order is characterized by the identification of the destination room and by the desired final location within that room. Requests are stored in a queue and a new request is only sent to the AGV after completion of the previous one.

In Section 2 of this paper, the AGV hardware components and control system are presented. Section 3 briefly describes the Architect software module and the Client-Server Subsystem. In Section 4, the Trajectory Planner subsystem is described considering the tasks performed by both the Global and the Local Detailed Trajectory Planners. 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 an Arbiter, which uses a neural network structure to perform the sensor fusion function and decide on the presence of obstacles on the AGV way and 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 conclusions of the paper and directions for future work.

2. THE AGV DESCRIPTION

The AGV has a cylindrical body. Its diameter is 50 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 angular position and velocity around the vertical axis.

A black and white camera is placed on top of the AGV body. It is attached to a pan-tilt structure which allows the camera to 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. Currently the AGV is operating with a fixed pan-tilt position (pan angle = 0° and tilt angle = 65°). On top of the AGV there is also an antenna which allows wireless communication between the AGV and 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 ultra-sound transmitter/receiver pairs. The first pair is placed at the middle of the bar and the other two are placed at the bar 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 the AGV 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 described by Aude[3]. In addition, the AGV Control System has additional single-input single-ouput controllers for controlling: the rotation angle of the horizontal bar which holds the ultra-sound sensors; and, in the future, the horizontal and vertical movements of the video camera.

The AGV desired trajectory is described by the velocity vector at its geometric center. The problem of interest to be solved for the AGV Control System is to calculate the values of the velocities at the fixed wheels and the angle to be applied to the steering wheel from this velocity vector.These values define the set-points to be used by the controllers when steering the AGV with a desired velocity along a given trajectory.

3. THE ARCHITECT MODULE AND THE CLIENT-SERVER SUB-SYSTEM

Architect is a special client application which has been designed to produce a description of generic floorplans. A textual format has been adopted to describe practically any kind of indoors environment. To simplify and speed up the process of describing an 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) under Windows 95. In its current version, Architect handles four different types of rectangular elements with integer coordinates: **FREE AREAS, WALLS, DOORS** and **BLOCKED AREAS.**

The client-server sub-system consists of a set of client stations connected through the Internet to a Request Server, which is linked to the AGV through wireless communication. The Request Server sends to the AGV the orders issued by the client stations and the AGV work environment description produced by the Architect. It receives the end of task messages as well as compressed image and position information from the AGV, which may be transimtted to the client stations in order to allow remote monitoring. Figure 3 shows a snapshot of the server screen displaying the AGV trajectory from source to goal.

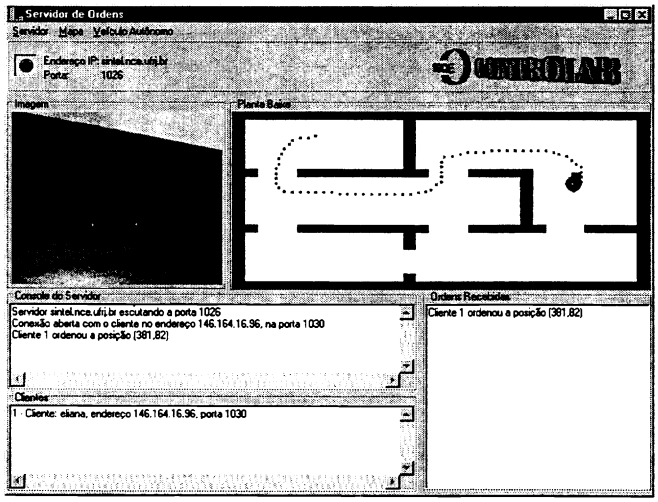


Figure 3: Request Server Screen Snapshot

The Request Server implementation followed the object oriented paradigm within the Microsoft Windows 95 environment through the use of the Borland C++ Builder compiler. It is able to perform multicasting in order to distribute information recieved from the AGV to a group of clients. The implementation of the multicasting capability is based on a protocol proposed by Deering[4] for the Internet. The adopted techniques for implementing reliable and unreliable communication through a radio system or through the Internet and to compress or decompress image data are discussed in detail in previous papers[5, 6]. Figure 4 shows the basic functionality of the Request Server and its interactions with other modules of the MUFA architecture.

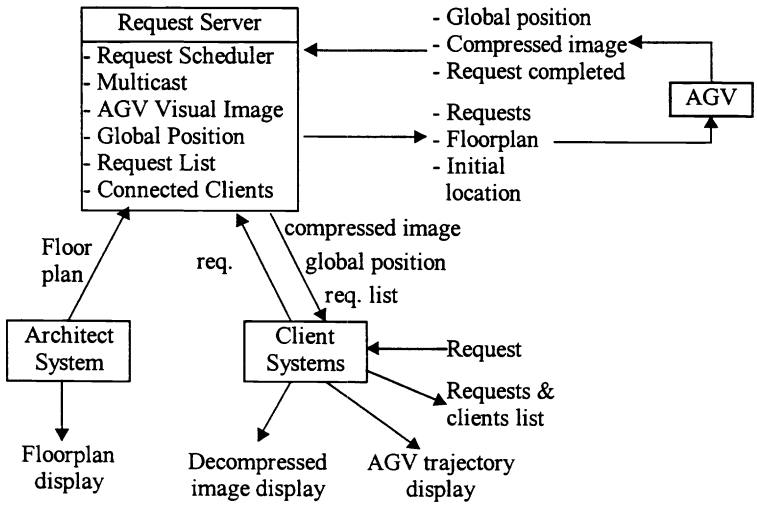


Figure 4: The Request Server Functionality

4. TRAJECTORY PLANNING

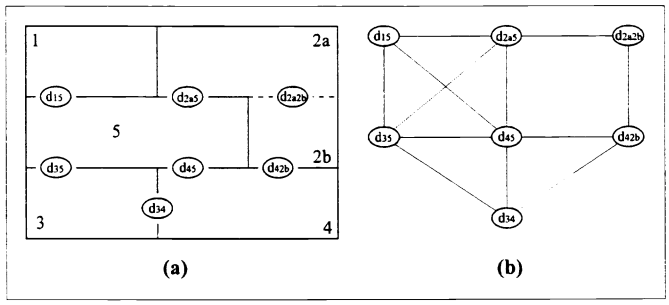
The AGV stores the environment floorplan description generated by the Architect module and transmitted by the Request Server. Once the environment description is reconstructed and the AGV receives information on its initial location within the floorplan, it knows the room in which it is. With information produced by the gyroscope and infomation on the floorplan orientation, the AGV knows its orientation in relation to the floorplan x-axis and, therefore, is ready to receive and execute motion commands sent by the Request Server.

The execution of each motion command 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 Local Detailed 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 final location within a known room. As the Architect module, the Global Trajectory Planner also works with rectangular areas. Threfore, any non-rectangular ROOM will be divided into several rectangular FREE AREAS connected by DOORS. From this structure containing FREE AREAS and DOORS, the Global Trajectory Planner creates a Connectivity Graph. Figure 5 illustrates the FREE AREA and DOOR structure and the Connectivity Graph generated for a particular floorplan. Within the Connectivity Graph, nodes are associated with DOORS and there is an edge in the graph connecting two nodes whenever the DOORS associated with these nodes open to a common FREE AREA. The weight of each edge is given by the product of the distance between the centers of the DOORS and the common FREE AREA weight.

Figure 5: (a) Environment Free Areas and Doors



(b) Connectivity Graph

Let us consider that S(x,y) is the initial AGV location in the FREE AREA a_s, and that G(x,y), located in the FREE AREA a_g, is the destination point of a given request. The Global Trajectory Planner generates then the Connectivity Graph adding two nodes associated with S and G and then calculates the minimum-cost path between S and G. Figure 6 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 - $\boldsymbol{S}$, $\mathbf{d_{15}}$, $\mathbf{d_{2a5}}$, $\mathbf{d_{2a2b}}$, $\boldsymbol{G}$, 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 has to be crossed by the AGV in its trajectory from the source to the 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.

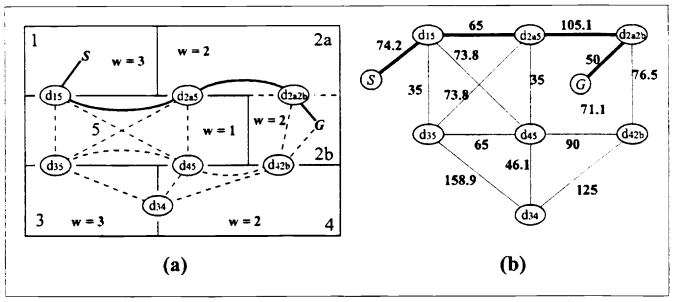


Figure 6: (a) Global Trajectory between *S* and *G*
(b) Connectivity Graph including *S* and *G*

4.2 Local Detailed 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 segment of the global trajectory. The precise trajectory to be followed by the AGV between each of these pairs of points is defined by the Local Detailed Trajectory Planner using a rule-based PFIELD algorithm proposed by Aude[7].

5. THE INTELLIGENT OBSTACLE AVOIDANCE SYSTEM

The Intelligent Obstacle Avoidance System consists of three parts: the Vision System, the Sonar System and the Arbiter. The Vision System works on image information captured by the video camera placed on a pan-tilt structure above the cylindrical AGV body. The camera points to the floor immediately ahead of the AGV and its vision area covers a range between 60 cm and 105 cm from the AGV center.

The Vision System processing to find obstacles ahead of the AGV 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[8]. Therefore, the image processing performed by the Vision System aims at detecting borders which represent obstacles sitting on or close to the AGV path. 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. A vector containing the y-coordinates of the pixels which belong to the detected edges closer to the image lower border is produced as a result of the image processing phase. This vector, consisting of 242 values, one for each x-coordinate, is used as an input to the Arbiter neural network structure which performs the sensor fusion function.

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 and is able to rotate 180° around the AGV central vertical axis in steps of 10°. The use of a pair of transducers, one for transmission and the other one for reception in each set, allows distances as small as 35cm to be measured. Therefore, obstacles as close as 10 cm to the AGV border can be detected by the central pair of transducers. For each AGV position, the Sonar System produces 3 measurements by each pair of transducers for each of the 19 angular positions the sonar bar can be placed. This array of 19 x 3 measurements is also an input to the Arbiter neural netowrk structure which performs the sensor fusion function.

The Arbiter consists of two parts. The first one is the sensor fusion performed by a neural network structure, which analyses the information produced by the Sonar and the Vision Systems and decides whether there is a previoulsy unknown obstacle close enough to the AGV. The second one is a rule-based system which defines the trajectory to be followed by the AGV when an unexpected obstacle or a corner is detected on the AGV way by the sensor fusion system. The following states can be detected as a result of the sensor fusion function performed by the neural network structure: free area (FA); corridor like environment (CO); righ-angle corner (RC); sharp angle corner (SC); obtuse angle corner (OC); perpendicular obstacle (PO); inclined to the right obstacle (IRO); inclined to the left obstacle (ILO). The architecture of each backpropagation neural network used to indicate one of these 8 states is shown in Figure 7.

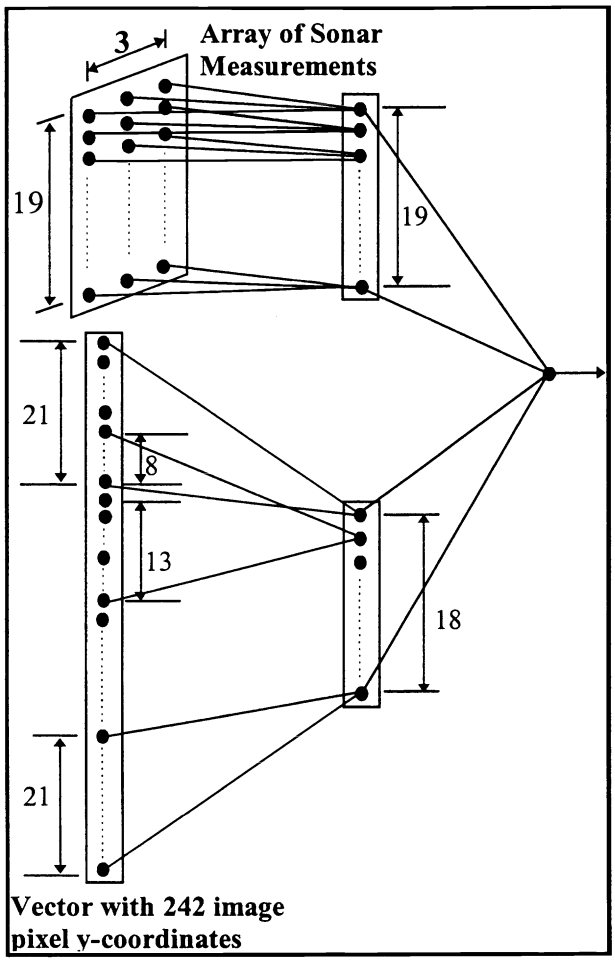


Figure 7: Sensor Fusion Neural Network Architecture

Figures 8 to 13 show the images generated by the Vision System and the measurements produced by the Sonar System when different situations are observed on the way to be followed by the AGV. These situations are analysed by the sensor fusion system to define the system state. In Figure 8, the horizontal line crossing the whole image near the bottom frame border indicates that there is an obstacle perpendicular to the AGV moving direction. The three sonar values when pointed to the movement direction are similar. This confirms the presence of a perpendicular object.

Figure 9 shows the situation when the AGV finds a right-angle corner. The image shows the contour of the righ-angle corner and the sonar measurements show two similar values and one very different one, indicating the presence of the corner. Another situation is shown in Figure 10 where the AGV faces a sharp corner. In this situation the image displays the contour of the corner but the sonar measurements are meaningless when taken in the direction of the AGV movement. Figure 11 shows an internal obtuse-angle corner. Once again the image displays the corner contour. The sonar measurements give the distances to the right and left side walls when they are placed facing them. Meaningless values are produced when the sonar bar is at 0°, pointing to the AGV direction of movement.

Figures 12 and 13 show two interesting situations. In Figure 12, the floor and the obstacle colours are vey similar. Therefore the image information cannot distinguish between them. In this case the sonar sensors are responsible for pointing out the presence of an obstacle. In Figure 13, the colour of the floor changes and the image information shows a false obstacle. Once again, the sonar system is able to indicate the correct information: a free area in the direction of the AGV movement.

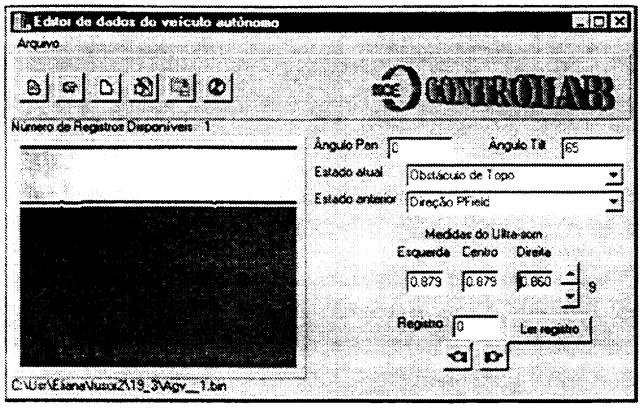


Figure 8: Perpendicular Obstacle

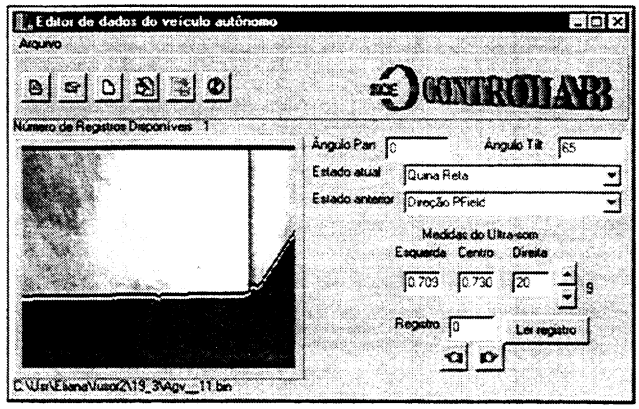


Figure 9: Right-Angle Corner

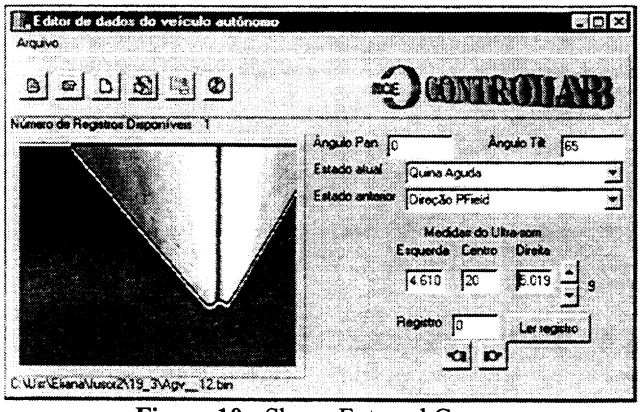


Figure 10: Sharp External Corner

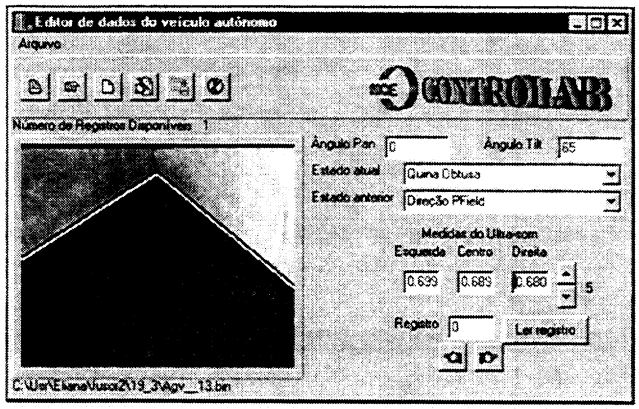


Figure 11: Obtuse Internal Corner

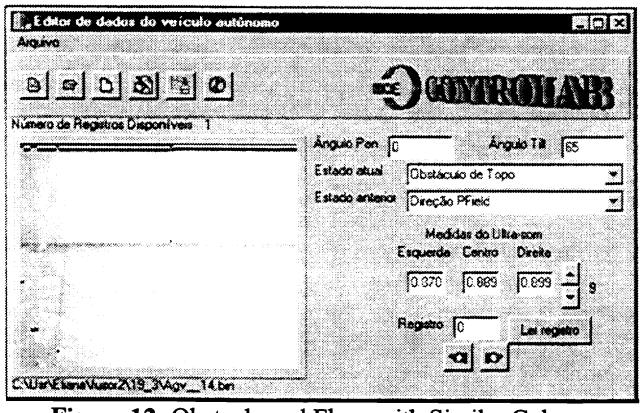


Figure 12: Obstacle and Floor with Similar Colours

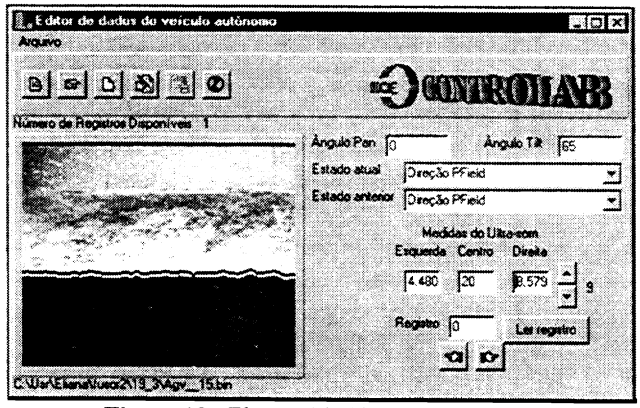


Figure 13: Floor with Change of Colours

When an unexpected obstacle or a corner on the AGV way is detected by the sensor fusion neural network structure, the Arbiter rule-based system is used to redefine the trajectory to be followed by the AGV. These rules are applied considering the sensor fusion state and the rule-based system own state, which informs the movement state. The state diagram shown in Figure 14 gives an overall view on how the rule-based system operates. This diagram consists of 7 movement states: Moving in the PFIELD Direction (MPD); Escaping from an Internal Corner (EIC); Escaping from an External Corner (EEC); Moving Parallel to an Obstacle (MPO); Searching for the Obstacle Angle (SOA); Turning an Internal Corner (TIC); and Turning an External Corner (TEC). The transition between states is in most cases determined by the combination of measurements of the sensor fusion state in a given direction, which is represented in Figure 14 by an expression like SF(direction) = sensor fusion state. For simplicity, the sensor fusion states "inclined to the right obstacle" (IRO) and "inclined to the left obstacle" (ILO) have been merged into a single denomination, "inclined obstacle" (IO). The directions which are used for the verification of the sensor fusion states are the following ones: the PFIELD suggested direction to the goal (PF); parallel to an obstacle (PLO); perpendicular to an obstacle (PPO); facing a corner (CN)

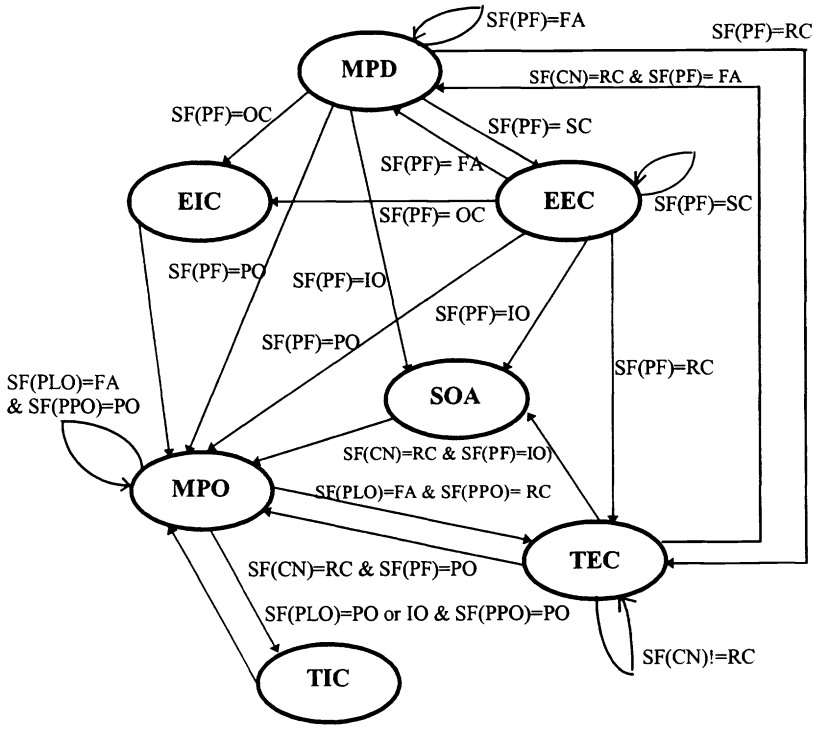


Figure 14: State Diagram of the Arbiter Rule-Based System

The analysis of the Arbiter rule-based system state diagram shown in Figure 14 starts by considering that initially the AGV will try to move in the goal direction as suggested by the PFIELD algorithm. Therefore, the initial movement state is MPD. It remains in this state, in which the AGV is controlled to move along the PFIELD direction, while the sensor fusion state indicates the existence of a free area (FA) in this direction. However, if the sensor fusion state in the PFIELD direction indicates the presence of an obstacle (IO or PO) or a corner (SC, OC or RC), a state transition takes place. If a perpendicular object (PO) is detected, the new movement state is MPO, where the AGV is controlled to move on a parallel line to the obstacle. If the detected obstacle is an inclined obstacle (IO), it is firtsly done a transition to the SOA state, where the obstacle angle is determined by looking at the sonar bar angle in which the three sonar transducers measured roughly the same distance values to the obstacle. Then, a state transition to MPO takes place and the AGV is controlled to move along a parallel line to the obstacle. If within the MPD state, a sharp corner (SC) is detected, the new state movement is EEC, where the AGV is controlled to move along a straight line perpendicular to its previous direction of movement and to the right or to the left depending on the goal position. This is done in an attempt to make the AGV escape from the obstacle corner detected on its way. As soon as the corner is no longer detected by the sensor fusion state, a movement state transisition is performed back to MPD, if the sensor fusion state in the PFIELD direction is indicating a free area (FA), and to MPO or

SOA, if a perpendicular (PO) or inclined obstacle (IO) is detected, respectively. If in the MPD or in the EEC state, an obtuse corner (OC) is detected by the sensor-fusion state in the PFIELD direction, the new movement state is EIC, where the sonar measurement information is used to determine the angles of the two obstacles which meet at the corner. Once these angles are determined, the movement state is set to MPO and the AGV is controlled to move along a parallel straight line to the obstacle. It will move to the right or to the left depending on the goal position. The last possibility to be considered within the MPD and EEC states is when the senor-fusion state indicates the presence of a right-angle corner in the PFIELD direction. In this case, the movement state is set to TEC and the AGV is controlled to move round this corner.

Once in the MPO state, the movement state will remain unchanged while the sensor-fusion state indicates the presence of a perpendicular obstacle (PO) when looking in the direction of the obstacle and indicates a free area (FA) in the current AGV moving direction, a parallel straight line to the obstacle. However, if the sensor-fusion state indicates the presence of an obstacle (IO or PO) when looking in the parallel direction to the obstacle, a state transition is performed to the state TIC, when the AGV will be controlled to turn an internal corner. This is done by determining the angle of the new obstacle using the sonar measurements. After that, the movement state changes back to MPO and the AGV is controlled to move along a parallel line to this new obstacle. Another situation that causes a state transition form the MPD state is the detection of an external corner, which is characterized by two measurements of the sensor fusion states. The first one will indicate a free area (FA) in a parallel direction to the obstacle and the second one will indicate the presence of a right-angle corner (RC) in a perpendicular direction to the obstacle. In this case, the new movement state is TEC, where the AGV will be controlled to turn an external corner. It does a circular turn around the corner and during this movement the sensor-fusion state in the corner direction is measured. The turn is complete when this state is once again right-angle corner (RC). At this point, a new state transition takes place. It goes back to the MPD state if the sensor fusion state in the PFIELD direction is free area (FA). It goes to the MPO movement state if the sensor fusion state in the PFIELD direction is perpendicular obstacle (PO) and to the SOA movement state if the sensor fusion state in the PFIELD direction is inclined obstacle (IO).

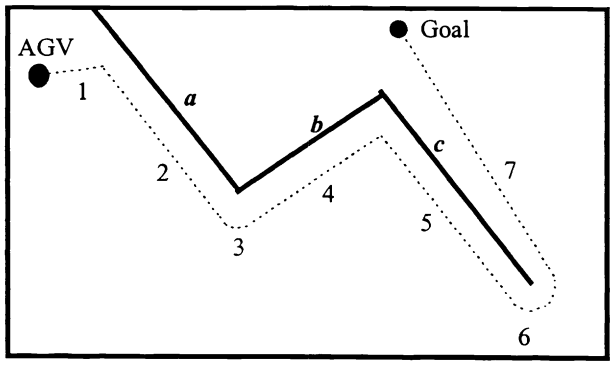


Figure 15: Operation of the Arbiter Rule-Based System

Figure 15 illustrates an example of the Arbiter rule-based system operation. In segment 1, the AGV movement state is MPD and the AGV is moving towards the goal, following the PFIELD suggested trajectory. By the end of segment 1, the sensor-fusion system detects the presence of an inclined obstacle (obstacle *a*) in the PFIELD direction. The movement state changes to SOA and the obstacle angle is determined by consulting the sonar measurements. After that, the movement state is changed to MPO and the Arbiter rules command the AGV to move along a parallel line to obstacle *a* in segment 2. By the end of segment 2, the sensor fusion system detects a right angle corner in the perpendicular direction to obstacle *a* and a free area in the parallel direction to obstacle *a*. According to the state diagram shown in Figure 14, a state transition to TEC is performed. The Arbiter rules command the AGV to turn the corner following a circular trajectory as shown by segment 3. The turn is completed when the sensor fusion state detects a right angle corner again when looking in the corner direction due to the presence of obstacle *b*. At this point the sensor fusion state in the PFIELD direction is inclined obstacle (obstacle *b*). Therefore a movement state change to SOA takes place. The obstacle *b* angle is determined and the movement state is changed to MPO, where the Arbiter rules command the AGV to move along a parallel line to obstacle *b* in segment 4. By the end of segment 4, the sensor fusion state in the perpendicular direction to obstacle *b* will be perpendicular obstacle and in the parallel direction to obstacle *b* will be inclined obstacle (obstacle *c*). Therefore, according to the state diagram shown in Figure 14, the movement state changes to TIC. Then, the Arbiter rules command the AGV to find the obstacle *c* angle and the AGV body rotates to look in a parallel direction to obstacle *c*. After that, the movement state changes to MPO and the

Arbiter rules command the AGV to move on a parallel line to obstacle *c* along segment 5. By the end of segment 5, the sensor fusion state in the perpendicular direction to obstacle *c* is right-angle corner and in the parallel direction to obstacle *c* is free area. Therefore, the movement state changes toTEC and once again the Arbiter rules command the AGV to turn the corner as shown in segment 6. When the turn completes, because the sensor fusion state in the corner direction is again right-angle corner, the sensor fusion state in the PFIELD direction is free area. Therefore, thye movement state changes to MPD and the AGV moves along the PFIELD trajectory in segment 7 until the goal is reached.

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 or corner. 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 within the Intelligent Obstacle Avoidance System as described in Section 5.

All examples shown in Figures 16 to 19 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 described in Section 4. Figure 16 shows all possible trajectories generated by the rule-based PFIELD algorithm. Figure 17 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.

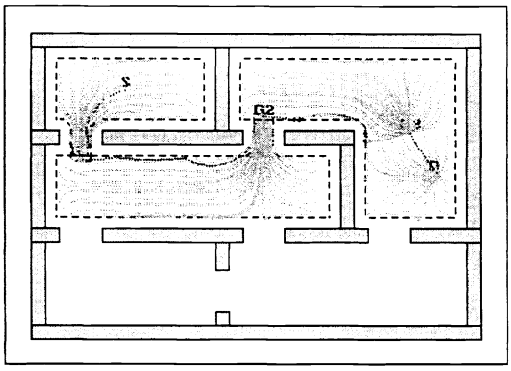


Figure 16: All Possible Trajectories Generated by the PFIELD Algorithm

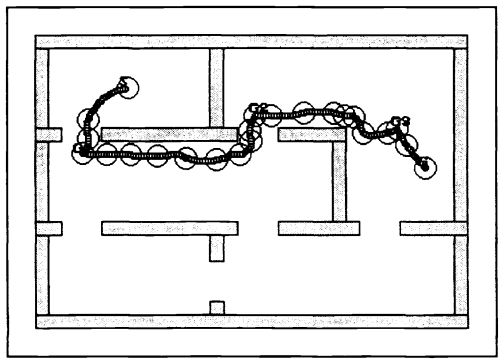


Figure 17: AGV Following the PFIELD Generated Trajectory

Figure 18 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. In Figure 19, 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 19.

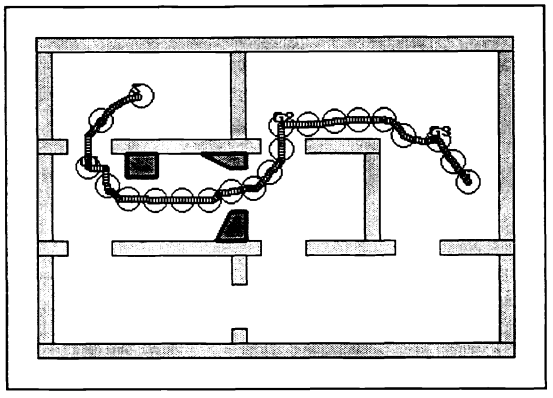


Figure 18: AGV Trajectory Avoiding Obstacles

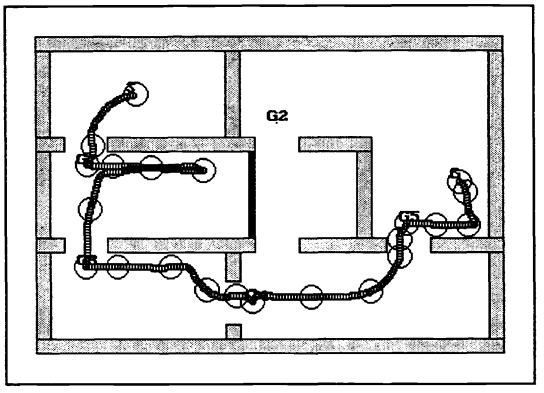


Figure 19: AGV Trajectory when a Blockage is Found

7. CONCLUSIONS AND FUTURE WORK

The benefits of sensor fusion and distributed intelligence are megerd in the proposed architecture for solving the problem of autonomous robot navigation in uncertain environments. The paper discusses the application of these concepts in 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. The neural network based approach for implementing the sensor fusion function as well as the set of rules and state diagram used to define the AGV trajectory when unexpected obstacles are detected have been discussed in detail.

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

ACKNOWLEDGEMENTS

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

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. Carneiro, G.H.M.B., Aude, E.P.L., Sedeira, H., Silveira, J.T.C., Martins, M.F., Lopes. E.P., *An Internet Request Server Architecture for Telecommanding the CONTROLAB AGV through Real Time Data and Image*, Proceedings of the 42nd Midwest Symposium on Circuits and Systems, Las Cruces, New Mexico, USA, August 1999
6. Aude, E.P.L., Carneiro, G.H.M.B., Serdeira. H., Silveira, J.T.C., Martins, M.F., Lopes, E.P., *CONTROLAB MUFA: A Multi-Level Fusion Architecture for Intelligent Navigation of a Telerobot*, Proceedings of the IEEE Internationla Conference on Robotics and Automation, Detroit , USA, May 1999, pp. 465-472

7. 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
8. Gomi, T., Ide, K., *Vision Based Navigation for an Office Messenger Robot*, Proceedings of the IROS'94, Munich, Germany, September 1994