WORKSHOP NOTES

‘Service Robotics –
Applications and Safety Issues in an Emerging Market”

14th European Conference on Artificial Intelligence

Thomas Röfer, Axel Lankenau, Reinhard Moratz

Berlin, August 22, 2000

Preface

About two years ago, the accumulated number of industrial robots sold worldwide broke through the mystical one million units boundary. Industrial robots are able to perform well-defined tasks with a very high speed and precision twenty-four hours a day and seven days a week. Despite the tremendous success of these devices, both academic and commercial research has been focusing on the development of a new generation of robots in recent years: service robots.

The IEEE and IPA-FhG database on service robotics provides us with a descriptive definition of the notion "service robot":
"Service robots refill vehicles, reconstruct nuclear power plants, take care of the elderly, observe museums, explore other planets or clean aircraft. So what are service robots? Service robots form an intermediate stage in the evolution from the industrial robot to the personal robot, which might be an important part of our lives in 20 years. Service robots are mobile, manipulative, interact with human beings, or perform tasks autonomously that relieve the human being."

According to a market analysis published by the United Nations Economic Commission for Europe (UN/ECE) and the International Federation of Robotics (IFR) in October 1999, the total number of service robots installed worldwide will have almost quintupled within the next three years. This forecast does not cover toy robots and vacuum cleaning robots. It is estimated that an addition 450000 of these will have been sold by 2002.

The workshop not only intends to present recent work in this rapidly developing field, but also to sharpen the discussion on the relevant topics that have to be tackled in order to ensure a prosperous future of the service robotics domain: navigation, human-machine interaction, safety and reliability.

Thomas Röfer, Axel Lankenau, Reinhard Moratz
Berlin, August 2000

WORKSHOP SCHEDULE

ECAI 2000 – Workshop W20 “Service Robotics”

Berlin, August 22, 2000

Time	Event	Page
$9^{\underline{00}}$	**Opening Talk:** (Chair: Axel Lankenau) *Service Robotics – State of the Art in an Emerging Market* T. Röfer (Univ. of Bremen, Germany)	
$9^{\underline{30}}$	**Navigation of Service-Robots:** (Chair: Axel Lankenau) *A Probabilistic Method for Planning Collision-free Trajectories of Multiple Mobile Robots* M. Bennewitz, W. Burgard (Univ. of Freiburg, Germany)	9
$10^{\underline{00}}$	*Diagrammatic Instruction-Maps for Human-Robot Interaction* R. Moratz, C. Freksa, T. Barkowsky (Univ. of Hamburg, Germany)	17
$10^{\underline{30}}$	**Coffee Break**	
$11^{\underline{00}}$	**Shared-Control in Service-Robots:** (Chair: Reinhard Moratz) *Shared-Control Architecture: Concepts and Experiments* U. Nunes, R. Cortesão (Univ. of Coimbra, Portugal)	21
$11^{\underline{30}}$	*The Role of Shared-Control in Service Robots* A. Lankenau, T. Röfer (Univ. of Bremen, Germany)	27
$12^{\underline{00}}$	**Lunch Break**	
$14^{\underline{00}}$	**Human-Machine Interaction:** (Chair: Thomas Röfer) *Service Robotics and the Issue of Integrated Intelligence: the CARL Project* L. Seabra Lopes, K.L. Doty, F. Vaz, J.A. Fonseca (Univ. of Aveiro, Portugal)	35
$14^{\underline{30}}$	*A Web Based Interface for Service Robots with Learning Capabilities* R. Marín, P. J. Sanz, A. P. del Pobil (Univ. of Jaume, Spain)	45
$15^{\underline{00}}$	**Coffee Break**	
$15^{\underline{30}}$	**Plenary Discussion** (Chair: Thomas Röfer)	

Navigation of Service-Robots

A Probabilistic Method for Planning Collision-free Trajectories of Multiple Mobile Robots

Maren Bennewitz **Wolfram Burgard**

Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany

Abstract. This paper considers the problem of path planning for teams of mobile robots. It presents a decoupled and prioritized approach to coordinate the movements of the mobile robots in their environment. Our algorithm computes the paths for the individual robots in the configuration-time space. Thereby it trades off the distance to both static objects as well as other robots and the length of the path to be traveled. To estimate the risk of colliding with other robots it uses a probabilistic model of the robots motions. The approach has been implemented and tested on real robots as well as in extensive simulation runs. In different experiments we demonstrate that our approach is well suited to control the motions of a team of robots in a typical office environment and illustrate its advantages over other techniques developed so far.

1 Introduction

Path planning is one of the fundamental problems in mobile robotics. As mentioned by Latombe [8], the capability of effectively planning its motions is "eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world." Especially in the context of autonomous mobile robots, path planning techniques have to simultaneously solve two complementary tasks. On one hand, their task is to minimize the length of the trajectory from the starting position to the target location, and on the other hand they should maximize the distance to obstacles in order to minimize the risk of colliding with an object.

In this paper we consider the problem of motion planning for multiple mobile robots. This problem is significantly harder than the path planning problem for single robot systems, since the size of the joint state space of the robots grows exponentially in the number of robots. Therefore, the solutions known for single robot systems cannot directly be transferred to multi-robot systems.

The existing methods for solving the problem of motion planning for multiple robots can be divided into two categories [8]. In the *centralized* approach the configuration spaces of the individual robots are combined into one composite configuration space which is then searched for a path for the whole composite system. In contrast to that, the *decoupled* approach first computes separate paths for the individual robots and then tries to resolve possible conflicts of the generated paths.

While centralized approaches (at least theoretically) are able to find the optimal solution to any planning problem for which a solution exists, their time complexity is exponential in the dimension of the composite configuration space. In practice one is therefore forced to use heuristics for the exploration of the huge joint state space. Many methods use potential field techniques [1, 2, 15] to guide the search. These techniques apply different approaches to deal with the problem of local minima in the potential function.

Other methods restrict the motions of the robot to reduce the size of the search space. For example, [7, 9] restrict the trajectories of the robots to lie on independent road-maps. The coordination is achieved by searching the Cartesian product of the separate road-maps.

Decoupled planners determine the paths of the individual robots independently and then employ different strategies to resolve possible conflicts. According to that, decoupled techniques are incomplete, i.e. they may fail to find a solution even if there is one. [4] consider coarse two-dimensional slices to represent the configuration time-space. [17] applies potential field techniques in the configuration time-space to resolve conflicts. All these techniques assign priorities to the individual robots and compute the paths in decreasing order starting with the robot with highest priority. Whenever a path is planned, these approaches try to resolve the conflicts with the previously determined paths. In this context, an important question is how to assign the priorities to the individual robots. In the approach presented in [3] higher priority is assigned to robots which can move on a straight line from the starting point to its target location. The approach described in [5] does not apply a priority scheme. Instead, it uses sets of alternative paths for the individual robots and determines a solution by applying heuristics to pick appropriate paths from the different sets.

An alternative approach to decoupled planning is the path coordination method which was first introduced in [13]. This method computes the paths of the individual robots independently and then applies scheduling techniques to deal with possible conflicts. The key idea of this technique is to keep the robots on their individual paths and let the robots stop, move forward, or even move backward on their trajectories in order to avoid collisions. Although the coordination method was initially designed for two robots only, [10] recently extended this idea to coordinate more than two robots.

[11] presented a reactive approach for decentralized real-time motion planning. Each robot plans its path towards its target dynamically based on its current position and sensory feedback. Since this method is similar to potential field approaches, it suffers from local minima and may also result in oscillations. Finally there are different techniques based on heuristics like traffic rules to resolve arising conflicts [6, 16].

A general assumption of the planning techniques described above is that the environment is completely known and that it does not change during the operation of the robots. Furthermore, the execution of the navigation plans is generally assumed to be deterministic, i.e. the robots perform all actions with certainty. Especially in real and populated environments these assumptions are generally vi-

olated, since the robots have to use their sensors to react to possible changes of the environment and to unforeseen obstacles. Therefore, the robots often deviate from their previously planned paths.

The method described in this paper is a decoupled and prioritized approach to coordinated path-planning for multiple robots. It incorporates different types of uncertainty into the planning process. First, it computes the path of a robot by trading off the length of the trajectory and the distance to obstacles. Furthermore, the actions of the robots are regarded to be non-deterministic. During planning, our approach therefore considers the possible deviations of other robots from their planned paths to determine the path of a robot in the configuration time-space. The parameters of the deviation-model have been learned in several experiments. Our approach has been implemented and tested on real robots and in extensive simulation runs. The experiments carried out in typical office environments illustrate that our technique is well suited to coordinate teams of mobile robots. They furthermore demonstrate that our technique outperforms the coordination approach described in [10, 13].

2 Probabilistic Path Planning for Multiple Robots

The goal of path planning is to determine a trajectory with the optimal trade-off between the overall length and the distance to obstacles in the environment. To effectively plan the path of a mobile robot, path planning systems need a model of the environment. In our case, the map of the environment is given by an occupancy grid map [12]. The key idea of occupancy maps is to separate the environment into a grid of equally spaced cells. Each cell of such a grid contains the probability that this cell is occupied.

Given such a map our approach uses the well-known A^* procedure to determine the path from the current location to the target point. For each location $\langle x, y\rangle$ the A^* procedure simultaneously takes into account the cost of reaching $\langle x, y\rangle$ from the starting position as well as the estimated cost of reaching the target location $\langle x^*, y^*\rangle$ from $\langle x, y\rangle$. In our approach the cost for traversing a cell $\langle x, y\rangle$ is proportional to its occupancy probability $P(occ_{x,y})$. The estimated cost for reaching the target location is approximated by the straight-line distance $|| \langle x, y\rangle - \langle x^*, y^*\rangle ||$ between $\langle x, y\rangle$ and $\langle x^*, y^*\rangle$. Accordingly, the minimum-cost path is computed using the following two steps.

1. **Initialization.** The grid cell that contains the robot location is initialized with 0, all others with ∞:

$$V_{x,y} \leftarrow \begin{cases} 0, & \text{if } \langle x, y\rangle \text{ is the robot position} \\ \infty, & \text{otherwise} \end{cases}$$

2. **Update loop.** While the target location has not been reached do:

$$\langle x, y\rangle \leftarrow \operatorname*{argmin}_{\langle x', y'\rangle} \Big\{ V_{x',y'} + c \cdot || \langle x', y'\rangle - \langle x^*, y^*\rangle || \Big\}$$

For each neighbor $\langle x', y'\rangle$ of $\langle x, y\rangle$ do

$$V_{x',y'} \leftarrow \min \Big\{ V_{x',y'}, V_{x,y} + || \langle x', y'\rangle - \langle x, y\rangle || \cdot P(occ_{x',y'}) \Big\}$$

In our approach, the constant c is chosen as the minimum occupancy probability $P(occ_{x,y})$, i.e.,

$$c = \min_{\langle x,y\rangle} P(occ_{x,y}).$$

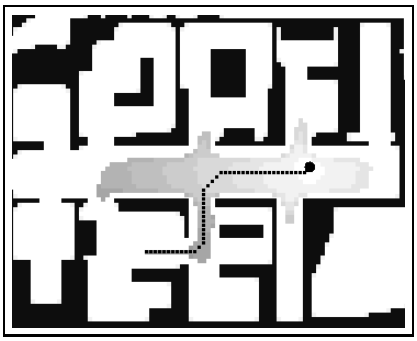

Figure 1. Result of a path planing process for a single robot using A^*. The accumulated costs of the cells considered during the search are indicated in grey (the darker the cell the higher the costs).

This choice of c is necessary to ensure that A^* determines the cost-optimal path from the starting position to the target location. Figure 1 shows a typical space explored by A^*. In this situation the robot starts in the corridor of our environment. Its target location is in the third room to the south. The figure also shows the accumulated costs of the states considered by the planning process. As can be seen A^* only expands a small fraction of the overall state space and therefore is highly efficient. The disadvantage of the A^* procedure lies in the assumption that all actions are carried out with absolute certainty. To deal with the uncertainty in the robot's actions one in principle would have to use value iteration which generally is less efficient than A^*. To incorporate the uncertainty of the robots motions into the A^* approach, we convolve the grid map using a Gaussian kernel. This has a similar effect as generally observed when considering non-deterministic motions: It introduces a penalty for traversing narrow passages or staying close to obstacles. As a result, our robots generally prefer trajectories which stay away from obstacles.

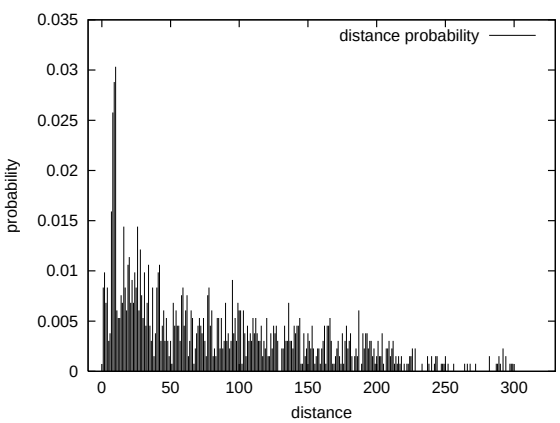

Figure 2. Average deviation of a robot from the originally planned path during plan execution.

As already mentioned above, our approach plans the trajectories of the robots in a decoupled fashion. First we compute for each robot the cost-optimal path using the A^* procedure mentioned above. We then check for possible conflicts in the trajectories of the robots. Whenever a conflict between robots is detected, we use a priority scheme

and determine new paths for the robots with lower priority. More precisely, suppose the k-th robot has a conflict with one or several of the $1, \ldots, k-1$ robots with higher priority. In this case we use A^* to re-plan the trajectory of this robot in its configuration time-space after including the constraints imposed by the $k-1$ robots with higher priority.

While planning in the configuration time-space we take into account possible deviations of the individual robots from their planned paths. For this purpose we use a probabilistic model which allows us to derive the probability that a robot will be at location $\langle x, y \rangle$ at time t given it is planned to be at location $\langle x', y' \rangle$ at that time. To estimate the parameters of this model we performed a series of 28 experiments with two robots in which we recorded the deviations of the robots from their pre-planned paths. In each run we constantly estimated for one robot the closest point on its planned trajectory and determined the distance of the second robot from the corresponding position of its path at the same point in time. As a result we obtained for a discrete set of distance ranges the number of times the second robot deviated from its originally planned path by that distance. The resulting probabilities are depicted in Figure 2. In our current implementation this histogram is approximated by a set of linear functions in order to avoid over-fitting. Given these data, we can easily determine the probability $P_t^i(x, y)$ that robot i is at a location $\langle x, y \rangle$ at time t. This probability is then used to define a cost function which allows us to determine the cost for robot k of traversing cell $\langle x, y \rangle$ at time t:

$$C_t^k(x, y) \quad = \quad P(occ_{x,y}) + \sum_{i=1}^{k-1} P_t^i(x, y)$$

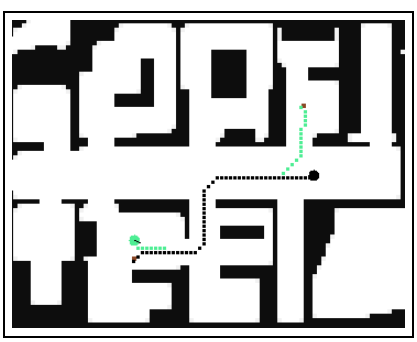

Figure 3. Conflict situation for two robots.

A typical application example of our planning technique is illustrated in Figure 3. In this case, the robot depicted in light grey is supposed to move to the fourth room in the north. The second robot depicted in black starts in the corridor and has its target location close to the starting point of the first robot. Since both paths are planned independently, they impose a conflict between the two robots. After applying the A^* procedure in the configuration time-space for the second robot, the conflict is resolved. The planner decides that the black robot has to avoid the conflict with the grey robot by moving to the north just at the door where the first robot enters the corridor. After this collision avoidance action, the path through the next doorway appears to have less costs, so that it takes a completely different trajectory. The resulting trajectories are depicted in Figure 4.

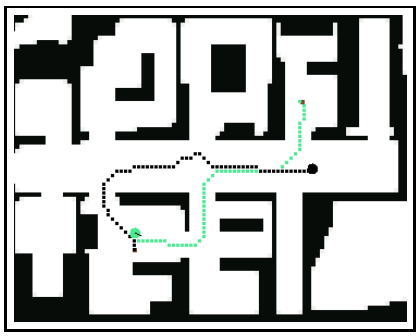

Figure 4. Resolved conflict by choosing a detour for the second robot.

Figure 5. The robots Albert and Ludwig used for the experiments.

3 Experimental Results

The approach described above has been implemented and evaluated on real robots as well as in simulation runs. The current implementation is quite efficient, although there still is a potential for improvements. For the 19×15 m^2 large environment in which we carried out the experiments described here, our system is able to plan a collision-free path in the configuration time-space in less than 6 seconds. The time needed for single robot path planning in the two-dimensional configuration space is generally less than 0.01 seconds. These performance measures were taken on a 500MHz Intel Pentium III running Linux and using a spatial resolution of 20×20 cm^2 for the grid map.

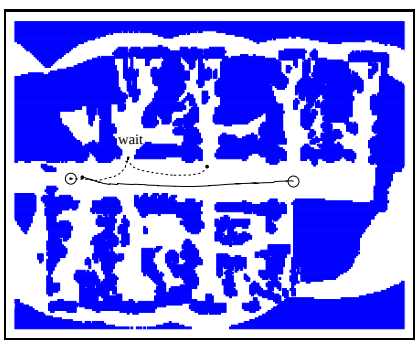

Figure 6. Ludwig moves away in order to let Albert pass by.

3.1 Application Example with Real Robots

The system has been evaluated using our robots Albert and Ludwig which are depicted in Figure 5. Whereas Albert is an RWI B21 robot, Ludwig is a Pioneer I system. Both robots are equipped with a laser-range finder to reactively avoid obstacles. Figure 6 shows one situation, in which both robots have a conflict. While Ludwig starts at the left end of the corridor of our lab and has to move to right end, Albert has to traverse the corridor in the opposite direction. Because of the uncertainty of Albert's actions, Ludwig decides to move into a doorway in order to let Albert pass by. The trajectory of Ludwig is depicted by a dashed line, and Albert's trajectory is indicated by a solid line. The position where Ludwig waited for Albert is indicated by the label "wait".

3.2 Competitive Ratio to the Optimal Strategy

In addition to the experiments using Albert and Ludwig, we performed a series of simulation runs in order to evaluate the applicability of the overall approach. An additional goal of these experiments is to demonstrate that our planner outperforms a prioritized variant of the coordination technique described in [13, 10]. Our current system uses a prioritized version because the joint state space grows exponentially in the number of robots which makes the search intractable for reasonable numbers of robots. The coordination technique described in [10] partitions the overall problem into a set of smaller problems one for each group of robots which have intersecting trajectories and thus is able to consider even huge numbers of robots. In general, however, it cannot be assumed that the resulting groups are small so that a prioritized planning is absolutely necessary. For the

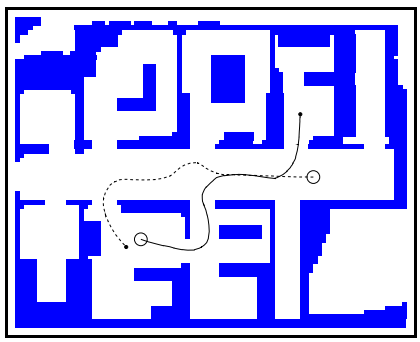

Figure 7. Simulation run with the resulting trajectories for the planned paths shown in Figure 4.

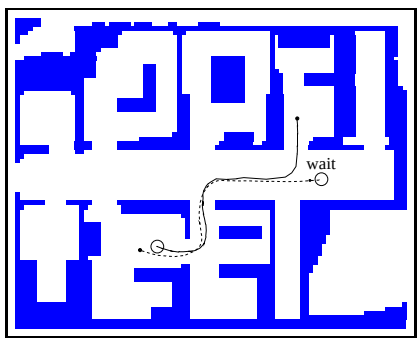

Figure 8. Trajectories obtained using the coordination technique.

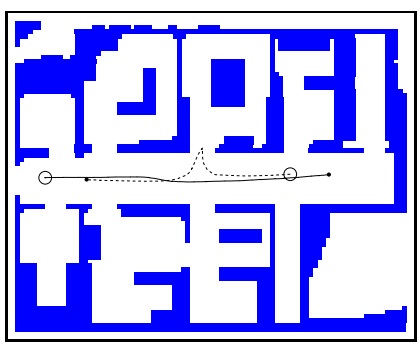

Figure 9. Solution generated by our probabilistic planning technique in a situation in which the coordination method does not find a solution.

following experiments we used the B21 simulator [14] which performs real-time simulations of the robot's actions and of its sensors. To get close to the behavior of a real robot, it adds noise to the simulated sensor information.

Figure 7 shows the trajectories carried out by two robots in the situation depicted in Figure 4. As can bee seen in the Figure, the resulting trajectories in this example are quite close to the planned paths. Figure 8 shows the corresponding paths obtained with the coordination diagram technique. Please note that in this situation our technique is significantly better than the coordination technique. Since the coordination technique does not change the trajectories and restricts the robots to stay on their pre-planned paths, the robot starting in the corridor has to wait until the other robot passed by. Therefore, the time to arrive at its target location is almost twice as long as it would be without any conflict. In contrast to that, the two robots arrive almost at the same time using our technique.

Since the coordination method restricts the robots to stay on their independently planned paths, it does not find a solution in situations in which our technique is able to determine collision-free trajectories. A typical example is shown in Figure 9. Here two robots have to pass each other in a corridor. Whereas the coordination method cannot resolve this conflict, our planner directs one robot to leave its optimal trajectory and to enter a doorway in order to let the other robot pass by.

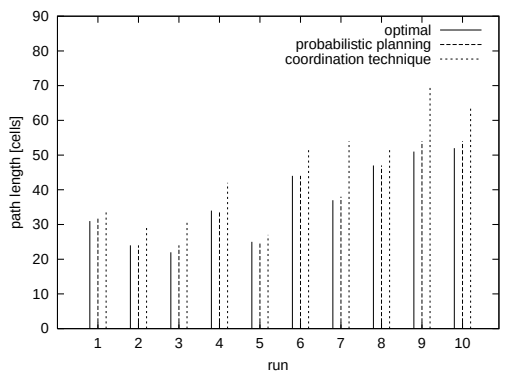

Figure 10. Performance comparison to the optimal solution and to the coordination technique.

To get a quantitative assessment of the performance of our method compared to the optimal strategy and compared to the coordination technique we performed extensive experiments with our simulator.

The first series is designed to compare our probabilistic planning technique to the optimal solution and to the coordination technique. We performed 10 different simulation runs using the environment shown in Figure 3. In each experiment we started two robots at different places and defined target locations for which there is a conflict which can be resolved by the coordination technique. Since our approach is more general than the coordination technique, all three methods were able to compute a solution in these situations. For each solution provided by the individual planners we recorded the sum of the lengths of the two paths, i.e. the number of cells traversed in the map plus the number of time steps each robot waited. In order to be able to compute the optimal solution we had to reduce the resolution of the grid maps to 60×60 cm^2. Figure 10 shows the resulting path lengths for the different runs and the individual planning techniques. Whereas the comparative ratio of our technique relative to the optimal solution was 1.02, the coordination technique needed 1.24 as many steps as the optimal solution. On the 95% confidence level our approach performed significantly better than the coordination technique. On average, the paths generated by the coordination method were 20% longer than the trajectories generated by our method.

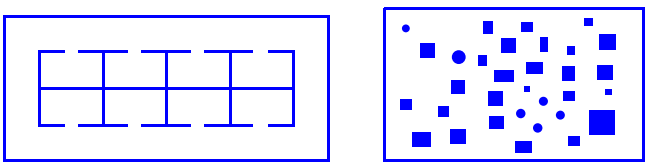

Figure 11. Two different environments used for simulation runs.

3.3 Comparisons for Larger Numbers of Robots

Additionally we performed extensive experiments in two different environments and compared the performance of our probabilistic approach to the performance of the coordination technique for different numbers of robots. Figure 11 depicts the two environments used in the experiments. The first environment shown on the left side of Figure 11 is a typical office environment. The second situation is a rather unstructured environment (see right image of Figure 11) which offers many possibilities for the robots to change their routes. In 9000 experiments we evaluated the path planning techniques for 2 to 6 robots in both environments. The corresponding start and goal positions were randomly chosen from a set of predefined positions.

Figure 12 shows for both environments the average number of conflicts each robot is involved in. Please note that we only evaluated situations in which there was at least one conflict between the robots. As can be seen this number is significantly higher in the office environment than in the unstructured environment because all robots have to travel along the corridor whereas they have a lot more possibilities to choose alternative routes in the unstructured world.

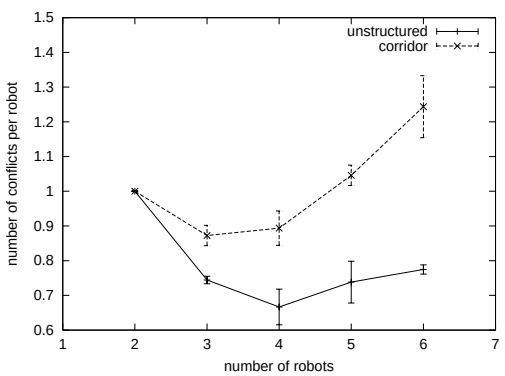

Figure 12. Average number of conflicts.

For each number of robots we evaluated 50 experiments in the structured and 100 experiments in the unstructured environment in

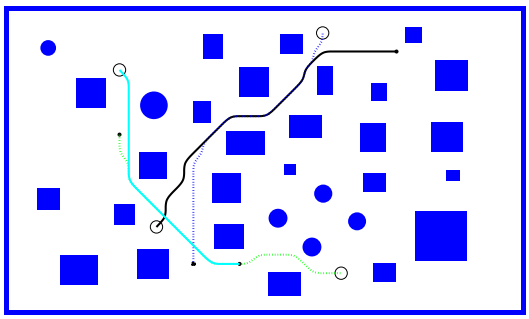

Figure 13. Typical experimental setup with four robots including their independently planned and optimal trajectories.

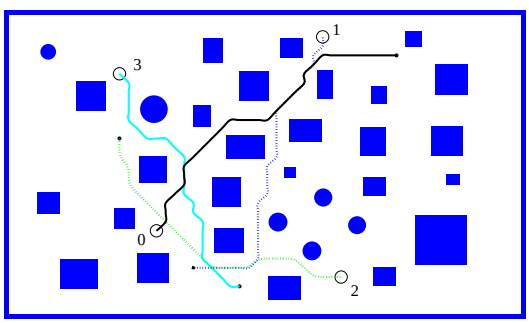

Figure 14. Priorities of the robots and paths computed by our probabilistic technique.

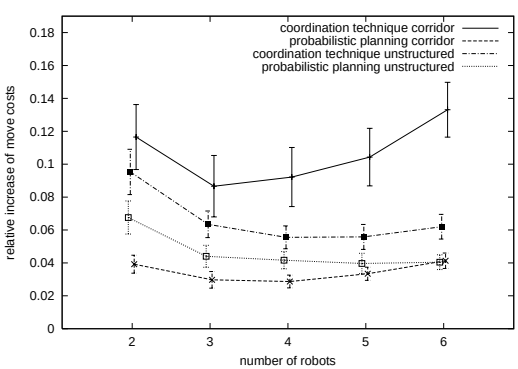

Figure 15. Comparison of the relative increase of move costs of the probabilistic technique and coordination technique.

which there was a conflict between the robots and in which both techniques were able to compute a solution. The priority scheme was to sort the robots according to the optimal move costs between their initial and their goal position. A typical example with four robots is shown in Figure 13. The priorities of the robots and the trajectories computed with our probabilistic planning technique are shown in Figure 14.

In each experiment we measured the sum of the move costs generated by our probabilistic technique and computed by the coordination technique. Since the optimal solutions were not known (and cannot be computed in a reasonable amount of time for more than two robots) we compared the results of the planning techniques with the sum of the optimal move costs for the individual robots if the paths are computed independently, i.e. in independent single robot problems. Thus, in the experiment described above we compared the resulting move costs of the robots (shown in Figure 14) with the corresponding costs obtained with the coordination technique both relative to the move costs of the paths in Figure 13.

As can be seen in Figure 15 our method significantly outperforms the coordination technique in both environments. Especially in the office environment the coordination technique frequently forces the robots to wait in a room for longer periods of time until another robot passed by. Since our probabilistic planning technique allows to robots to choose detours in the corridor, the reduction in the average move costs obtained with our probabilistic planning technique is much higher.

As already mentioned in the experiments described above we used the move costs to determine the priority of the individual robots. To evaluate an alternative priority schemes we performed the same experiments using the number of conflicts each robot was involved in to determine the priority of the robots. It turned out that the results obtained with this heuristic do not differ significantly to those obtained when the robots are sorted according to their move costs.

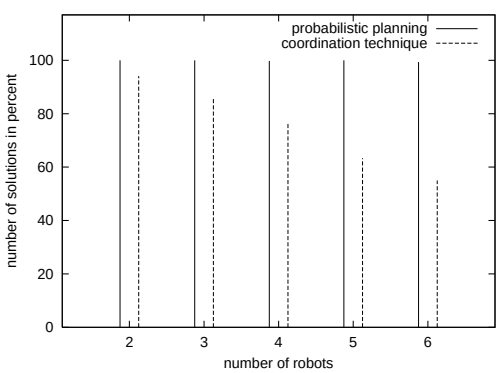

Figure 16. Number of cases in percent where a solution could be found in the unstructured environment.

Another interesting aspect is the number of situations in which the different approaches were able to generate a solution. Figure 16 shows for both methods the number of cases in percent in which a solution could be found in the unstructured environment. Obviously, the coordination technique quite often cannot find a solution as the number of robots rises. For example, for 6 robots only 55% of the planning problems could be solved by the coordination technique whereas our probabilistic technique was able to find a solution in 99.3% of the problems.

Figure 17 depicts one of the two planning problems with 6 robots for which our prioritized planning method is not able to find a solution. Since robots 0 and 2 have higher priority their paths are computed first. As a result, robot 4 cannot "escape" so that no path can be found for this robot. Thus, given the fixed priority scheme there is no way to find a path for robot 4.

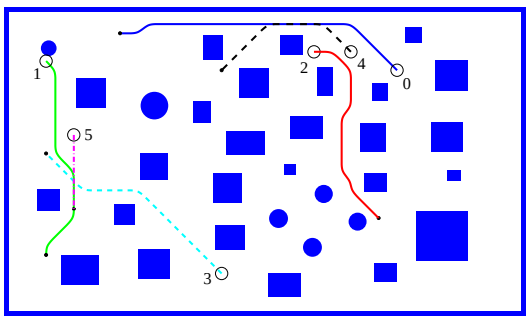

Figure 17. No solution can be found for robot 4.

4 Conclusions

In this paper we presented an approach to decoupled and prioritized path planning for groups of mobile robots. Our approach plans the paths for the individual robots independently. If a conflict between the paths of two robots is detected it uses a priority scheme to re-plan the path of the robot with lower priority in its configuration time-space. Thereby it considers the constraints imposed by the robots with higher priority. Our approach uses occupancy grid maps to plan the motions of the robots using A^*. Simultaneously it trades off the length of the trajectory and the distance to objects in the environment. It furthermore uses a probabilistic model to integrate possible deviations of the robots from their planned paths into the planning process. Therefore, the resulting trajectories are robust even in situations in which the actual trajectories of the robots differ from the pre-planned paths.

Our method has been implemented and tested on real robots. The independent planning of the paths for the individual robots is highly efficient and requires not more than 0.01 seconds. Additionally, the system can rather quickly resolve conflicts. For the examples in the map of our department the computation of a collision-free path in the configuration time-space generally requires less than 6 seconds using a spatial resolution of 20×20 cm^2 and less than 1.5 seconds for a cell size of 40×40 cm^2. Please note that this computation time will not significantly increase in the number of robots, since our approach uses lookup-tables to store the costs introduced by the previously planned robots.

In all experiments our approach showed a robust behavior. Additionally, we performed a series of experiments to compare our technique to the coordination method. These experiments demonstrate that our approach produces navigation plans which are by 17% shorter than those generated by the coordination method. Compared to the optimal strategy in the joint configuration time-space our technique produces paths which are by 2% longer than the shortest paths.

Apart from these promising results, there are different aspects for future research. Our approach currently uses a fixed priority scheme. More flexible assignments of priorities to the individual robots will with high likelihood result in more efficient solutions. Furthermore, our system currently does not react to larger differences during the plan execution and assumes equal constant velocities of the robots. For example, if one robot is delayed because unforeseen objects block its path, alternative plans for the other robots might be more efficient. In such situations it would be important to have means for detecting such opportunities and to re-plan dynamically. On the other hand, the delay of a single robot may result in a dead-lock during the plan execution. In this context, the system requires techniques for detecting dead-locks while the robots are moving and to resolve them appropriately.

REFERENCES

[1] J. Barraquand, B. Langois, and J. C. Latombe, 'Numerical potential field techniques for robot path planning', Technical Report STAN-CS-89-1285, Department of Computer Science, Stanfort University, (1989).

[2] J. Barraquand and J. C. Latombe, 'A monte-carlo algorithm for path planning with many degrees of freedom', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, (1990).

[3] S. J. Buckley, 'Fast motion planning for multiple moving robots', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, (1989).

[4] M. Erdmann and T. Lozano-Perez, 'On multiple moving objects', *Algorithmica*, **2**, 477–521, (1987).

[5] C. Ferrari, E. Pagello, J. Ota, and T. Arai, 'Multirobot motion coordination in space and time', *Robotics and Autonomous Systems*, **25**, 219–229, (1998).

[6] D. Grossman, 'Traffic control of multiple robot vehicles', *IEEE Journal of Robotics and Automation*, **4**, 491–497, (1988).

[7] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, 'Probabilistic road maps for path planning in high-dimensional configuration spaces', *IEEE Transactions on Robotics and Automation*, 566–580, (1996).

[8] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[9] S. M. LaValle and S. A. Hutchinson, 'Optimal motion planning for multiple robots having independent goals', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, (1996).

[10] S. Leroy, J. P. Laumond, and T. Simeon, 'Multiple path coordination for mobile robots: A geometric algorithm', in *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, (1999).

[11] V. J. Lumelsky and K. R. Harinarayan, 'Decentralized motion planning for multiple mobile robots: The cocktail party model', *Journal of Autonoumous Robots*, **4**, 121–135, (1997).

[12] H.P. Moravec and A.E. Elfes, 'High resolution maps from wide angle sonar', in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 116–121, (1985).

[13] P. A. O'Donnell and T. Lozano-Perez, 'Deadlock-free and collision-free coordination of two robot manipulators', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, (1989).

[14] D. Schulz, W. Burgard, and A.B. Cremers, 'Robust visualization of navigation experiments with mobile robots over the internet', in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (1999).

[15] P. Tournassoud, 'A strategy for obstacle avoidence and its application to multi-robot systems', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pp. 1224–1229, (1986).

[16] J. Wang and S. Premvuti, 'Distributed traffic regulation and control for multiple autonomous mobile robots operating in discrete space', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pp. 1619–1624, (1995).

[17] C. Warren, 'Multiple robot path coordination using artificial potential fields', in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pp. 500–505, (1990).

Diagrammatic Instruction Maps for Human-Robot Interaction

Reinhard Moratz [1] and **Thomas Barkowsky** and **Christian Freksa**

1 Introduction

Many tasks posed in the field of service robotics can profit from a scenario in which a human instructor gives a robot tasks in a natural way. Natural interfaces typically refer to human communication as their blueprint. We focus on the task of giving a mobile robot a destination that it has to reach. A natural way is to communicate destinations verbally to a robot (cf. [2]). Another strategy is to show sample routes (cf. [5]). We use simple maps as means of communication here. Humans like to communicate routes and destinations by drawing the route as a sketch map. Therefore, we will look at schematic maps used by humans (see [1] for details on schematic maps). These maps only preserve qualitative spatial relations of the relevant objects, as opposed to true geographical maps which are metrically veridical depictions of a given environment. Also, the representation is reduced to the detail that is relevant for the route instruction.

A human instructor typically can draw a schematic map of a known environment with little effort from memory. The difficulty for the robot comes from the need to establish a mapping from the abstact and coarse map to the more detailed world (or simulated world). We build an experimental system that uses a simulated robot to interpret qualitative instruction maps. The task for the robot is then to reach a target defined on the map in the simulated world.

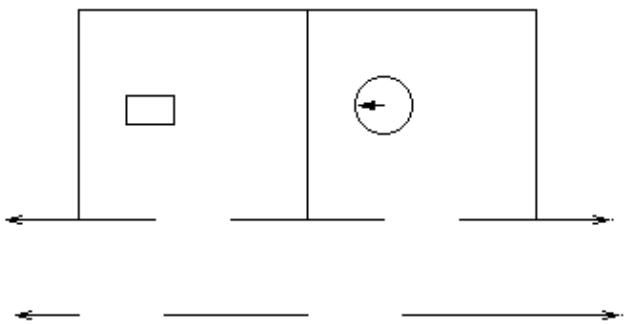

Figure 1. Instruction Map

2 Instruction Maps

In Figure 1 we give a simple example of a qualitative instruction map of an indoor office environment that may be provided by a human instructor to an autonomous robot. It consists of two rooms and a hallway connecting the rooms. The robot that is located in one of the rooms. The adaptation to a robotic communication partner requires certain iconic expressions to make the instruction maps reliably interpretable by algorithmic means. To this end, we allow lines ending in an arrow. The arrow indicates that the respective linear object extends beyond the depicted range in the represented world. This symbol is necessary to make interpretation of the sketch unambiguous.

The schematic map depicts approximate global survey knowledge of the environment. In addition to the robot environment, the map indicates position and orientation of the robot and the location of the goal.

The simulated world in which the robot acts represents our real office environment (see figure 3), with all furniture omitted. A comparison of instruction map and world simulation shows that the dimensions of the rooms are shown only qualitatively. Furthermore, the hallway contains a number of built-in cupboards that were omitted on the schematic instruction map. In the next section we show how an instruction map of this kind is used to give a simulated robot motion instructions.

[1] University of Hamburg, Department for Informatics, Vogt-Kölln-Str. 30, 22527 Hamburg, moratz, barkowsky, freksa@informatik.uni-hamburg.de

3 Path Planning

The robot has survey knowledge that refers to the instruction map and it has only local knowledge in the simulated world. Therefore the robot first uses the survey knowledge supplied by the map to find a path to the target on the map. In the next step for local pieces of the path on the map corresponding entities have to be found in the simulated world. A qualitative description of the path enables the robot to establish a discrete mapping between map and environment. A qualitative path is specified only in terms of the route to be taken, not in terms of metrically specified locations. The basis are landmarks which are salient places which can be identified from different perspectives (cf. [4]).

Our first prototype uses room corners as landmarks during navigation. So the robot matches room corners in the instruction map and in the simulated world. The room corners are easy to detect using a laser range finder. The simulated robot uses two laser range finders for sensors, each covering a field of 180 degrees. Together, these sensors yield a surround view of 360 degrees.

The path in the schematic map from its initial location (according to the instruction map) to the destination is found by a cell decom-

position algorithm ([3]). The first step is to partition the instruction map. The landmarks (room corners) are the end points of connecting lines which segment the free space into convex cells (see figure 2).

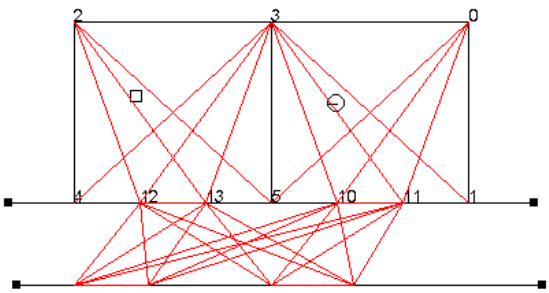

Figure 2. Partitioning

Now a path graph can be constructed. In the path graph all cells are represented as vertices and edges connect vertices if the corresponding cells are neighboured. The robot can use a simple graph search to find a qualitative path from the initial cell to the target cell.

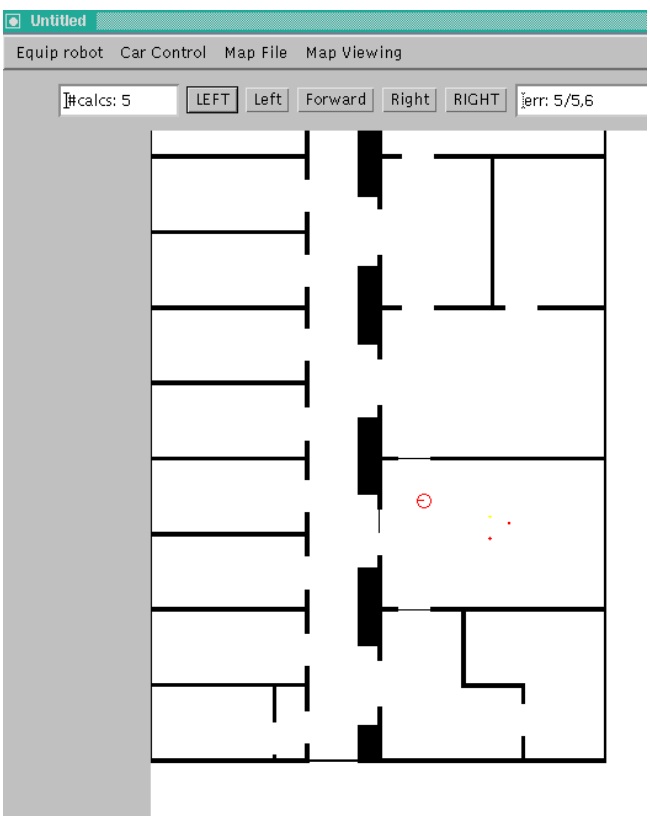

Figure 3. Robot Simulation

To actually traverse this path, the path in the map is now translated into a sequence of pairs of room corners. The midpoint of the connecting line between the landmark pair is then the intermediate destination that the simulated robot can directly attain.

This sequence is another representation of the route, and the order information contained in it enables the robot to obtain pairs of room corners in the simulated world. The qualitative spatial relations used for the correspondence between map landmarks and world landmarks consist of order relations of the visible landmarks (see [6]). Then a local panorama view on the map is matched with a local panorama view in the simulation. Also the collinearity of landmarks is used to perform the mapping. Since not every corner in the simulated world has a counterpart on the coarser map we use an optimizing strategy to account for partial mapping situations.

Crossing a connecting line as intermediate destinations brings the robot one step closer to the destination. With this recursive procedure, the destination area in the simulated environment – which corresponds to the destination marked in the schematic map – is reached.

4 Conlcusion and perspective

We propose the use of schematic maps for human-robot interaction. To explore the scenario we investigate the task of instructing a mobile robot using a qualitative map. Our approach relies on the assumption that schematic maps preserve important ordering information for identifying spatial configurations. Then we can use the order of landmarks to find correspondences between the map and the simulated world.

The presented example is intended to be generalized to a fundamental approach of communicating routes to mobile robots by means of schematic maps. In the future we will also use maps constructed by the robot during an exploration (cf. [7]) for human-robot interaction.

Acknowledgment

The authors would like to thank Jesco von Voss for implementing the simulation and performing the first experiments. This work was supported by the DFG priority program on Spatial Cognition, under grant Fr 806/7-2

REFERENCES

[1] C. Freksa, R. Moratz, and T. Barkowsky. Schematic maps for robot navigation. In C. Freksa, W. Brauer, C. Habel, and K. F. Wender, editors, *Spatial Cognition II - Integrating abstract theories, empirical studies, formal models, and practical applications*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.

[2] C. Habel, B. Hildebrandt, and R. Moratz. Interactive robot navigation based on qualitative spatial representations. In I. Wachsmuth and B. Jung, editors, *Proceedings Kogwis99*, pages 219–225, St. Augustin, 1999. infix.

[3] J-C Latombe. *Robot Motion Planning*. Kluwer, 1991.

[4] T S Levitt and D T Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(2):305–360, 1990.

[5] T. Röfer. Route Navigation Using Motion Analysis . In C. Freksa and D. Mark, editors, *Spatial Information Theory*. Springer, Berlin, 1999.

[6] C Schlieder. Reasoning about ordering. In W Kuhn A Frank, editor, *Spatial Information Theory: a theoretical basis for GIS*, number 988 in Lecture Notes in Computer Science, pages 341–349, Berlin, 1995. Springer Verlag.

[7] S. Thrun. Learning metric-topological maps for indoor moblile robot navigation. *Artificial Intelligence*, 99:21–71, 1998.

Shared-Control in Service-Robots

Shared-Control Architecture: concepts and experiments

Urbano Nunes[1], **Rui Cortesão**, **J. Luis Cruz** and **Paulo Coelho**

Abstract. An hybrid control architecture with reactive, deliberative, and cognitive components appropriate to human-oriented mobile robots is presented. The paper presents globally our control architecture and describes some of its modules in detail: local map-building module, reactive control module and wheel´s velocity controller. Some discussion is made concerning trajectory planning and path-tracking control. The architecture is under development and part of it has been tested in a motorized wheechair equipped with several sensors and a multimodal HMI (Human-Machine Interface). A joystick and a voice unit interface (VUI) are used to steer the wheelchair, giving accessibility to handicaped persons. In its actual stage, the wheelchair navigates autonomously, by means of simple voice commands (or commands issued by the joystick), avoiding static and moving obstacles (like humans moving around).

1 INTRODUCTION

Usually the human is forced to adapt him or herself to the machines in present day manufacturing and process industries. An advanced class of robots and intelligent machines that adapt, extend, and work in a symbiotic way with humans, is required and is under research in University and Industry laboratories. These robots do not only work for humans, but also they work with or assist humans, and share with the users the same environment. Human-oriented robots and intelligent machines are required and are under research. Its development poses many difficult problems, namely concerning HMI, safety and man-machine shared control [10].

In the Intelligent Control and Robotics laboratory (IC&R) at the Institute for Systems and Robotics (ISR-Coimbra), research is being carried out towards the development of an intelligent control system for human-oriented mobile robots (HOMR). We are pursuing an hybrid paradigm composed by a reactive control level and a decision making level supported by a knowledge-based perceptual system, as depicted in figure 1a). The paper presents globally our control architecture and describes some of its modules in detail. The purpose is to achieve an architecture appropriate to integrate different components necessary to a HOMR. For example we can enumerate some research problems faced in this development:

- For a behaviour-based reactive level, what type of behaviours will be necessary? What methods to use for integration/fusion and arbitration of different behaviours? What learning methods can be considered for performing behaviour integration? How to establish the human interaction in this architecture?
- One of the challenges of human-oriented robotics is that robots have to work closely with humans. Can we be inspired by biological behaviours to make robots more friendly and more familiar to humans?
- Besides the interaction capabilities with humans, the robot should integrate and acquire knowledge concerning its own state and environment state. This means it should have other capabilities, such as self-localization, map building and path-planning and task-planning.

[1] Institute for Systems and Robotics, University of Coimbra, - Polo II, 3030 Coimbra - PORTUGAL, email: urbano@isr.uc.pt

2 CONTROL ARCHITECTURE

Figure 1a) presents the major components of the control architecture under development. Parts of it have been tested in a motorized semi-autonomous wheelchair prototype being developed under the RobChair project running in ISR [9].

The main goal of the control system is to overcome user's physical limitations, minimizing his effort. This is achieved sharing the control between the user and the actions provided by the cognitive and sensory systems. A joystick and a VUI (voice unit interface) [7] are used to steer the wheelchair. The VUI is based on a set of voice recognition libraries included in a package that performs the recognition of spoken words (Dragon Voice Tools).

2.1 Conceptual architecture

RobChair architecture is organized as an hybrid architecture combining deliberative reasoning with low-level reactive behaviours. Deliberative reasoning provides the system with the ability to reach proposed goals. Reactive behaviours are indispensable to ensure a safe navigation, enabling the vehicle to react in real time to environment emergent situations. In most robotics applications, a purposeful navigation depends on the integration and interaction of these two control levels. However, there are others where a unique deliberative or reactive architecture ensures a purposeful navigation. The first, deliberative reasoning, can be used in fully deterministic and static environments. However, this doesn't meet the requirements of most real environments. The second, reactive reasoning, usually lacks purposeful goals. The lack of *a priori* information to plan strategies and trajectories can lead to navigation failure. This can be compensated if the goals are always visible. For example, if the goal is a light always detectable, a cue in the floor, or a surface to contour, it will be possible to reach purposeful goals.

RobChair is a specific system integrating closely the human and the machine. The human is a cognitive entity that substitutes parts of the deliberative layer. Presently, without having global environment information, RobChair system is unable of a purposeful navigation without user intervention, so the reason we call it a semi-autonomous system.

The proposed architecture is a four layer distributed architecture: a reactive layer embodying reactive behaviours; a local-action layer for execution of specific tasks dependent of local environment; a deliberative reasoning layer responsible for high-level planning; and finally

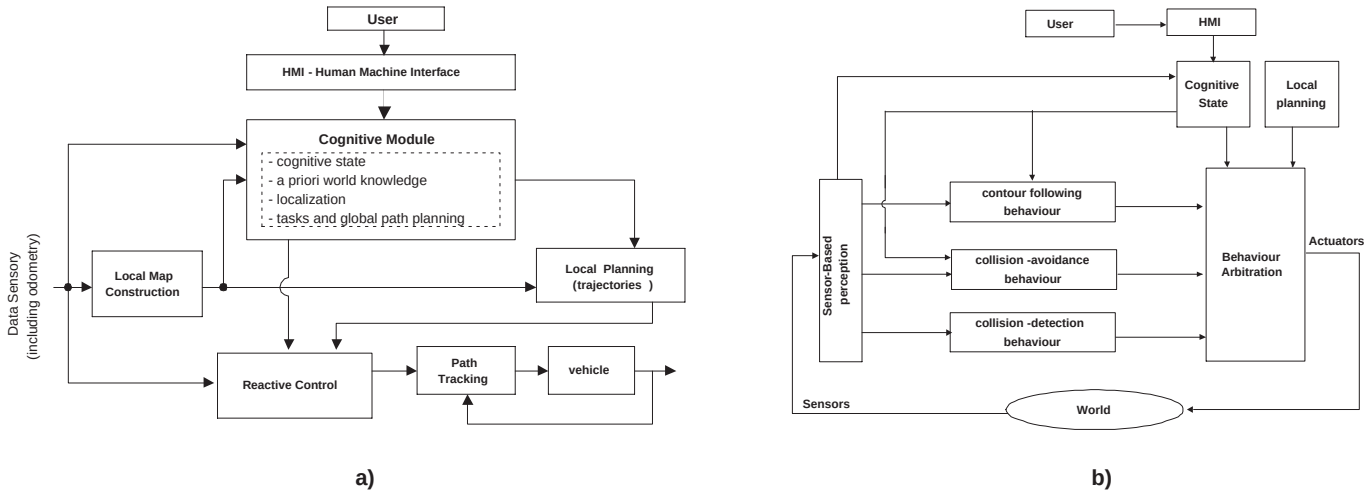

Figure 1. a) Hybrid control architecture; b) RobChair reactive layer

a mission layer where goals are defined. The wheelchair user is part of this layer and he intervenes in the cognitive state. By this way, he can define goals for the deliberative reasoning layer, as well as, depending of system state, goals for reactive control layer guidance. The main modules of the conceptual control architecture and their inter-connections are illustrated in figure 1a):

- **Mission Layer -** In this layer a set of static or dynamic goals are defined by the user or by other human operator. Examples of deliberative goals might be go to room B.
- **Deliberative Reasoning Layer -** It is usually based on *a priori* knowledge of the world. This knowledge takes the form of topological and geometric maps giving, for instance, information of how to go from A to B, or giving the position of objects and landmarks. Based on this information, global path and task planning and execution can be undertaken. This layer relies on long-term memory information, and basically, performs global path-planning providing bottom layers with a set of points defining the path to accomplish the tasks. To perform path-planning it may be required other cognitive capabilities relying on global map updating, integrating over time local maps into the global map, self localisation, etc.
- **Local-Action Layer -** This is an intermediate, short-term memory layer. The plans relies essentially on short-term memory, which integrates sensory information, in terms of a local map, and guidance information from the upper control layers. Two local-action tasks are being implemented: a door-passage and a table/writing desk approaching.
- **Reactive Layer -** This layer is fully implemented [9]. As depicted in figure 1b), this layer embodies three behaviours: collision detection, obstacle avoidance, and contour following. These behaviours rely upon actual sensory information without resorting to environment models. The behaviours are simple, directly coupling perception to action. This layer receives guidance from upper layers. It consists basically on system state information and commands of linear velocity and angular velocity (v, w). An integration/fusion of the guidance variables and data from each behaviour is carried out in this layer.

3 LOCAL MAPPING

A local environment model is acquired iteratively and on-line. Our work is inspired in Thrun's approach [12] in which it is proposed the use of a Neural Network (NN) to interpret sensors readings. Their good results, motivate us to investigate this technique in building local maps. In Thrun's approach the local grid is used to build global geometric maps. In our work the local map is used in planning local trajectories, in triggering and guiding safety mechanisms and as a source of information to a cognitive module in charge of obtaining a cognitive state of the overall human-oriented robot system. The Bayes rule is used to update the local grid map, integrating over time new sensor readings, iteratively and on-line. The local occupancy grid, which consists of $n \times n$ cells, is a local view that moves with the robot. In the next Sections we describe the local map-building process in general terms. A property of this method consists in the possibility of integrating/fusing proximity and distance data from different sensors. This map-building method has given good results when using sonars alone or sonars plus optoelectronic proximity/distance sensors.

3.1 Local map-building process

Figure 2 shows our map-building architecture. In order to update the map cells, the Sector Selector and the Sensors Selector work in a coordinated way. The Sector Selector provides NN with information of the polar coordinates of the cell (x, y), while the Sensors Selector choose the two adequate sensor's readings and provides them to the NN. The function of the NN (see figures 2 and 3) is to provide the conditional probability $P(C_{xy}|o)$ given actual sensory observation o, to the Bayes' update formula. After applying this formula the cell is finally updated.

Figure 3 shows the NN that performs the mapping from two sensor readings to the conditional probability of cell occupancy. For a given cell (x, y), the input layer of NN consists of:

- The observation $o = (ss1, ss2)$ of the two sensors, from the set of sensors, oriented in direction of cell (x, y);

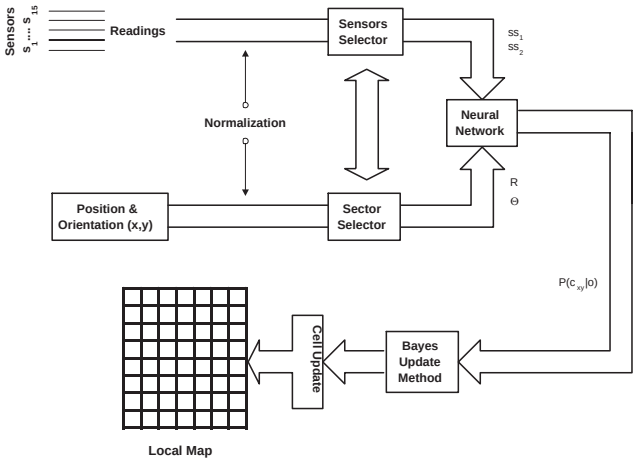

Figure 2. Local map-building architecture. $ss1$ and $ss2$ (the sensor selected 1, 2) are the four sensors that point in cell's direction. (R, θ) represent the polar coordinates of the cell (x, y), to be updated, in relation to the robot coordinates system

- The polar coordinates R and θ, of the center of the cell (x, y) with respect to the robot frame (as illustrated in the example of figure 5 for a circular robot).

The output layer has only one node. This node produces a $P(C_{xy}|o)$, that measures the probability of the cell (x, y) to be occupied given the actual sensory observation $o = (ss1, ss2)$.

The network was trained off-line with a back-propagation algorithm [11]. After training, the network gives values in range $[0, 1]$ that can be interpreted as probability of occupancy. Since the NN is trained based on examples, it can easily be adapted to new situations. Another advantage is the capacity to interpret two sensor readings simultaneously (can be more). Interpreting sensor readings in context of their neighbours generally yields more accurate results [12].

3.2 Sensors selector and sector selector

The Sensors selector and Sector selector work in a coordinated way. The selection of the areas to map with more intensity can be guided heuristically or purposefully. In the last case, a minimum cost function, an objective function or task purposes can drive the selection. For instance, in local navigation more attention might be paid to the areas for which the robot is moving to.

3.3 Bayesian-based cells updating

The local mapping consists to estimate the occupancy of a specific area around the robot that moves with it. Let C_{xy} denotes *"cell* (x, y) *occupied"*. So C_{xy} denotes a discrete random variable defined in the universe $\{0, 1\}$, i.e $C_{xy} = 1$ stands for cell occupied, and $C_{xy} = 0$ stands for cell free. The mapping can be seen as the problem to estimate the conditional probability

$$P(C_{xy}|o^{(1)}, ..., o^{(N)}) \tag{1}$$

where $o^{(1)}$, denotes the first (in time) observation and $o^{(N)}$ the last observation. Based on the Bayes theorem, and after some mathematical manipulation we can express the conditional probability of cell (x, y) to be occupied, given a sequence of observations, as follows [2]:

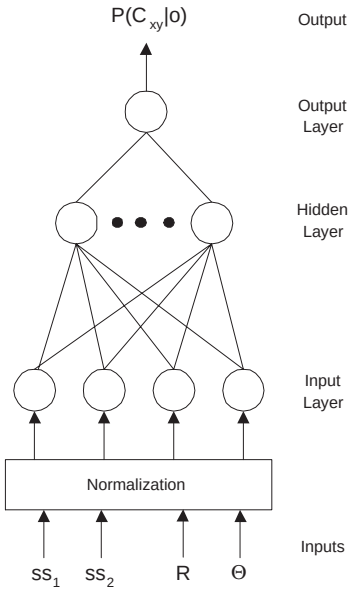

Figure 3. Feedforward neural network

$$P(C_{xy}|o^{(1)}, ..., o^{(N)}) = 1 - (1 + b)^{-1} \tag{2}$$

where

$$b = \frac{P(C_{xy}|o^{(N)})}{1 - P(C_{xy}|o^{(N)})} \cdot \frac{1 - P(C_{xy})}{P(C_{xy})} \cdot \frac{P(C_{xy}|o^{(1)}, ..., o^{(N-1)})}{1 - P(C_{xy}|o^{(N)}, ..., o^{(N-1)})} \tag{3}$$

In equations (2) and (3) $P(C_{xy}|o^{(N)})$ is given by the NN, $P(C_{xy})$ is the initial probability of occupancy of cell (x, y), (equal to 0.5), and $P(C_{xy}|o^{(1)}, ..., o^{(N-1)})$ represents the probability before the actual update. Using equation (2) we can evaluate iteratively the probability of occupancy of each cell, which means that only one value per cell needs to be stored in the local map.

The value of the map cell (x, y) represents the probability of the corresponding space to be occupied (near 1) or free (near 0). Initially all cell values are set to 0.5, i.e unknown case. Every time a cell seems to be occupied, its value increases, on the contrary, its values decreases. Due to the mathematical characteristics of the updating equation (3), if the cell value is 1 or $zero$, in the following iterations, the result remains always 1 or $zero$ respectively, independently of $P(C_{xy}|o^{(N)})$. Thus, the value of the cells is in the range $]0, 1[$. In the experiments described in Section 3.5, it was used the range $[0.01, 0.99]$.

3.4 Cell update algorithm

1. Initialization: $P(C_{xy}) = 0.5$
2. For each cell (x, y) and for each new observation $o^{(N)}$ (selected for this cell) the NN gives as output $P(C_{xy}|o^{(N)})$
3. Cell's value update:

$$P(C_{xy})[k] = 1 - \left(1 + \frac{P(C_{xy}|o^{(N)})}{1 - P(C_{xy}|o^{(N)})} \cdot \frac{P(C_{xy})[k-1]}{1 - P(C_{xy})[k-1]}\right)^{-1} \tag{4}$$

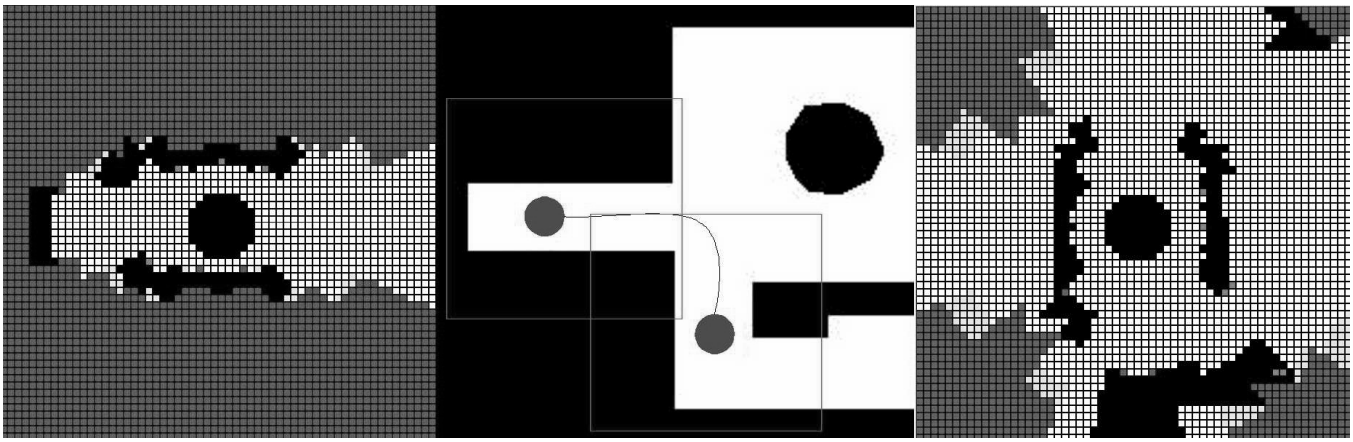

Figure 4. Left image: Grid map seen by the robot in the initial location. Center image: Simulation environment. Right image:Grid map seen by the robot in the final location.

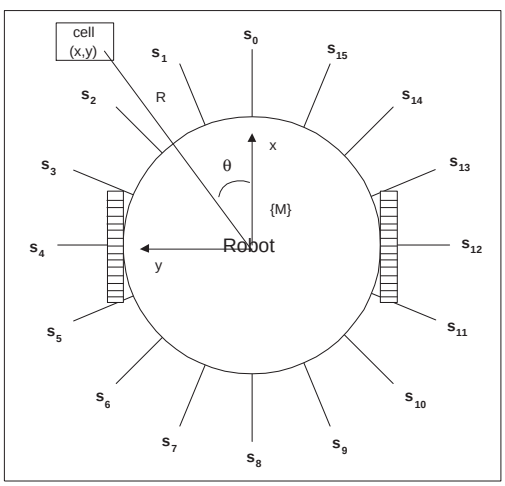

Figure 5. The robot and sensors orientation.This geometry refers to the case of a circular robot (in the simulation of the algorithm we have used the Nomad robot). $\{M\}$ defines the robot's coordinates system. R and θ are the polar coordinates for the cell (x, y) related to $\{M\}$

where $P(C_{xy})[k]$ denotes the actual cell's value, that is $P(C_{xy})[k] = P(C_{xy}|o^{(1)}, ...o^{(N)})$.
Equation (4) is equation (3) after the following operation. Since in equation (3) $P(C_{xy})$ denotes the cell's initial value that is 0.5 (unknown case), then $\frac{1-P(C_{xy})}{P(C_{xy})} = 1$.

3.5 Example of local grid maps

Figure 4 shows an example of two local grids built by using our mapping process shown in the schematic of figure 2. These experimental results were obtained using the "Nomad 200 Simulator" and its 32 proximity/distance sensors: 16 sonars and 16 infrared proximity sensors. The sensors are disposed around the Nomad as described in figure 5 (top view of the robot). The sonars and the infrared sensors have the same orientation, which means that in figure 5, each S_i represents a pair of sensors (1 sonar plus 1 proximity infrared sensor).

In the simulation reported in figure 4, for each S_i pair it was used a simple heuristic rule to choose between the readings from the sonar and the infrared proximity sensor. In the short-range, preponderance is given to the infrared data. Good results were achieved, and the inclusion of the infrared sensors improved substantially the results.

In Robchair we are integrating sonars, proximity infrared sensors and infrared triangulation-based distance sensors (Sharp GP2D12) whose disposition around the wheelchair is not regular as in the case of Nomad. The map-building architecture (figure 2) is applied without changes. Only the strategy of choosing or preprocessing the readings of a sector can be different, in order to get the desired inputs to the NN.

Let us to interpret the results reported in figure 4. In the center is shown the simulation environment where the robot navigates corresponding to an area of $6 \times 6\ m^2$. Left and right images show grid maps. To each map corresponds an area of $3 \times 3\ m^2$ divided in cells of $5 \times 5\ cm$, giving a total number of cells per map of 3600.

In the central image the frames around the robot (the robot is represented by a circle and it is shown in two different locations), symbolize the space of the environment that is mapped in the grid maps of the adjacent images.

The simulation consisted in the following: the robot made a straight movement from the left position, along the corridor, and next described a curve to the final position, as shown in figure 4 (central image). The left image shows the first map built by the robot located in the initial position (with only one update). The right image shows the local map built by the robot after successive updates made during the trajectory execution accomplished by the robot to the final position. As can be seen the local maps are very good.

4 PATH AND TRAJECTORY PLANNING

Finding a path from a start position to a goal position, avoiding collisions, is the purpose of a robot path planner.

The path-planning problem's complexity depends not only on the assumptions on the environment (*a priori* knowledge, static or mobile obstacles,...) but also on the robot models used to represent the robot motion. Kinematics and dynamic constraints on the robot motion should be taken into account.

4.1 Control architecture

The control architecture that is under development in the IC&R laboratory at ISR-Coimbra, namely being tested in the Robchair, aims to fulfill three scenarios:

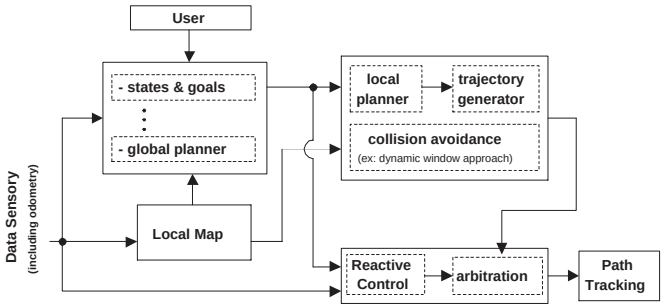

Figure 6. Path and trajectory planning module

1. In the first scenario, there is the possibility to navigate the Robchair with commands or instructions, given by voice or user's joystick. This navigation is supported by a reactive control layer that receives sensory information from the surrounding environment. This layer enables collision avoidance. However, in this scenario the navigation is far from being optimal, and does not solve the purpose of navigation on several situations. For instance, to pass through a door it may be necessary to give a big number of instructions, by joystick or by voice, which is quite inconvenient for the user, and it is only possible for users with dexterity. This reactive layer is already working [9].
2. The second scenario includes not only the first scenario but it carries out additional local map information (referred in Section 3) in order to plan local target points (sub-goals). It should be stressed that such planning happens only if the system is in certain states, activating a "local goal window". For instance to pass through a door overcoming the drawbacks of the first scenario. This scenario has a collision avoidance processing mode with a higher degree of intelligence. An interesting approach is presented in [5], where it is used the "Dynamic Window Method" that computes the velocity values to achieve in a certain time interval. These velocities produce a trajectory in which the robot is able to stop safely, i.e it reduces the search space to a dynamic window. This method is valid for synchronous robots incorporating its dynamics. Moreover, it enables to know in the dynamic window: the robot localization related to a certain target; the trajectory velocity; and the distance to the nearest trajectory goal.
3. In the third scenario the goals are global (e.g. go from room A to room B). Thus there is a need of a global planner that calculates the global route (goal points in a global perspective), using local map and sensory information. The global path planner implements the algorithm that finds the optimal obstacle free route. Methods of this type include the well-Known road-map, cell decomposition and potential fields (see [6] for an overview and further references). The global path planner must build an obstacle-free path, and conduct the robot to visit each of the sub-goals defined by the mission. This global path, is an approach of the final trajectory followed by the robot and does not take into account the details of the vehicle's local environment. When an obstruction on the global path occurs, the local planner comes into action to calculate, function of the robot's local environment, the trajectory to be followed. Usually, the global path can be calculated off-line, but the local planning is required to run on-line.

Figure 6 shows the proposed path and trajectory planning module that observes the three scenarios. Presently, our objective is to analyze the performance of the different solutions (working alone) and to design the final planning module optimized to navigate the robot, namely in the three scenarios above defined.

As concerns the trajectory and path planning problem an interesting approach has been developed by Muñoz et al. [8]. The path is defined by a sequence of points (w_i) that correspond to locations to be visited by the robot in a given order. Let $W = \{w_i\}, i = 1, ...N$ define a set of N sparse locations computed by the path planner, defining a path between locations A and B. In the Muñoz approach a path generation method, to join the intermediate points satisfying certain continuity conditions (in heading, curvature,...), is proposed based on cubic β-Spline curves. β-Splines where chosen due to their properties for fast computation. Finally the navigation system must include a path-tracking module (see figure 6). The path tracker generates the vehicle's steering commands to track the path. These steering commands, as well as the speed command, are sent to the low-level controllers of the wheels.

5 WHEEL'S VELOCITY CONTROL

A new velocity controller is applied in the RobChair. Each wheel has an independent velocity input, enabling the RobChair to have the desired angular or linear velocities. The wheel actuator (DC motor) model is of the type

$$G_{\mathrm{p}}(s) = \frac{1}{1 + T_{\mathrm{p}} s} e^{-sT_{\mathrm{d}}}, \tag{5}$$

where the time delay T_{d} is the deadtime and has a key role in the discrete state dimension. T_{p} is the velocity response time constant. Its equivalent temporal representation is given by:

$$\dot{y}(t) = -\frac{1}{T_{\mathrm{p}}} y(t) + u(t - T_{\mathrm{d}}), \tag{6}$$

which is of the form

$$\dot{x} = Ax(t) + Bu(t - t_{\mathrm{d}}). \tag{7}$$

Discretizing the system of equation (7) using the new concept of Active Observers [3], with sampling time h and dead-time $t_{\mathrm{d}} = (d-1)h + \tau'$, with $0 < \tau' \leq h$, the equivalent discrete time system of equation (8) is obtained,

$$\begin{bmatrix} x_k \\ u_{k-d} \\ \vdots \\ u_k \\ u_{k-1} \\ \hat{p}_k \end{bmatrix} = \begin{bmatrix} \Phi & \Gamma_1 & \Gamma_0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ u_{k-d-1} \\ \vdots \\ u_{k-3} \\ u_{k-2} \\ \hat{p}_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{bmatrix} u'_{k-1} + \xi_k. \tag{8}$$

Φ, Γ_0, Γ_1 are given by equations (9) to (11) respectively [1], and ξ_k is a noisy vector with properly chosen statistics [3].

$$\Phi = e^{Ah} = \phi(h), \tag{9}$$

$$\Gamma_0 = \int_0^{h-\tau'} \phi(\lambda)\, d\lambda\, B, \text{and} \tag{10}$$

$$\Gamma_1 = \phi(h - \tau') \int_0^{\tau'} \phi(\lambda)\, d\lambda\, B. \qquad (11)$$

x_k is the output velocity, u_k the command input and $\hat{p}_k$ the active state. Figure 7 depicts a block diagram of the discrete state-space based controller applied in each wheel. The reference input r_k is limited to the maximum velocity slop that the wheel can handle. The active state performs a feedforward compensation of all the disturbances, estimating the error e_k referred to the system input, permitting it to have the ideal behavior. Thus, it enables the RobChair to have always the same dynamic behavior, specified by the designed closed loop poles, regardless environment conditions as ramps to rise, different user weights, and even dropping or carrying objects during the process. Moreover, it can also give useful information if the RobChair crashes with "invisible" obstacles (e.g. a stone in the floor), warning the end-user about it. A Kalman Active Observer was designed as a Stochastic Active Observer example [4] . The robustness of the system is accomplished through optimal noise processing embedded in the control strategy.

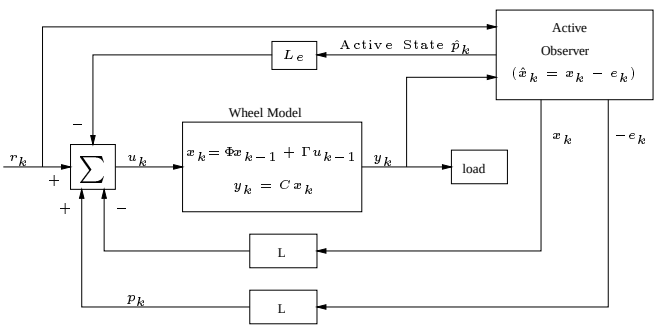

Figure 7. State-space based controller applied in each wheel, using the active observer architecture. The wheel's actuator model was obtained in a "free space" environment, i.e., the wheels' experiments were done without floor contact.

6 CONCLUSIONS AND FUTURE WORK

In this paper it is presented an hybrid architecture suited for Human-Oriented Mobile Robots. Currently the Robchair can navigate in dynamic environments in the presence of humans, commanded by simple voice or joystick instructions. The navigation is currently supported by a pure reactive layer enabling the obstacle avoidance and the contour-following. An efficient map-building method was described to extend potentialities of higher level reasoning capabilities. At the low-level control the Robchair is equipped with an independent wheel controller based on a state-space approach embedding a stochastic active-observer. This controller is robust to external disturbances very common in Robchair activities.

As future work new algorithms are being developed to accomplish path and trajectory planning. Learning techniques are to be explored in order to give progressive autonomy in the execution of "similar" tasks (e.g. learning to pass a door and then generalize it to different doors in different environments).

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support provided by FCT under the research project MENTOR PRAXIS/P/EEI/13141/1998.

REFERENCES

[1] K. J. Astrom and B. Wittenmark, *Computer Controlled Systems: Theory and Design*, Prentice Hall, 1997.

[2] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W Steiner, and S. Thrun, 'Esperiences with an interactive museum tour-guide robot', Technical Report CMU-CS-98-139, (June 1998).

[3] R. Cortesão, R. Koeppe, U. Nunes, and G. Hirzinger, 'Explicit force control for manipulators with active observers', in *(Accepted) IEEE Int. Conf. on Intelligent Robots and Systems, (IROS'2000)*, Japan, (2000).

[4] R. Cortesão, R. Koeppe, U. Nunes, and G. Hirzinger, 'Force control with a kalman active observer applied in a robotic skill transfer system', *Int. J. on Machine Intelligence & Robotic Control*, **2**(2), (2000). To Appear.

[5] D. Fox, S. Burgard, and S. Thrun, 'The dynamic window approach to collision avoidance', in *IEEE Robotics and Automation Magazine*, volume 4:1, (1997).

[6] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[7] C. Lopes, N. Honório, F. Perdigão, and U. Nunes, 'Voice interface for an intelligent wheelchair', in *Third Portuguese Conference on Automatic Control (Controlo98)*, pp. 841–843, (1998).

[8] V. Muñoz, A. Ollero, M. Prado, and A. Simon, 'Mobile robot trajectory planning with dynamic and kinematic constraints', in *Proc. of 1994 IEEE International Conference on Robotics and Automation*, volume 4, pp. 2802–2807, (1994).

[9] G. Pires, R. Araújo, U. Nunes, and A.T. Almeida, 'Robchair: A powered wheelchair using a behaviour-based navigation', in *IEEE 5th Int. Workshop on Advanced Motion Control*, pp. 536–541, (1998).

[10] T. Röfer and A. Lankenau, 'Ensuring safe obstacle avoidance in a shared-control system', in *7th IEEE Int. Conference on Emerging Technologies and Factory Automation (ETFA'99)*, pp. 1405–1414, Barcelona, (1999).

[11] D. Rumelhart, G. Hinton, and R. Williams, 'Learning internal representations by error propagation', in *Parallel Distributed Processing*, eds., D. Rumelhart and J. McClelland, MIT Press, (1986).

[12] S. Thrun, 'Learning metric-topological maps for indoor mobile robot navigation', *Artificial Intelligence*, **1**, 21–71, (1998).

The Role of Shared-Control in Service Robots

The Bremen Autonomous Wheelchair as an Example

Axel Lankenau, Thomas Röfer[1]

Abstract. This paper intends to emphasize the importance of taking into account potential shared control problems when designing a service robot application. A brief overview of shared control and the resulting mode confusion problems is given. So far, the subject is primarily discussed in the avionics domain. This paper motivates how to transfer these experiences into the service robotics community. Two recently developed specification techniques are introduced. Finally, a concrete example is presented: How to avoid mode confusion in the Bremen Autonomous Wheelchair "Rolland".

1 INTRODUCTION

The Bremen Autonomous Wheelchair "Rolland" is a rehabilitation service robot, that realizes intelligent and safe transport for handicapped and elderly people. The system consists of dedicated modules each of which adds certain skills to the platform. The vehicle is a commercially available power wheelchair Genius 1.522 manufactured by the German company Meyra. For the work described here, it has been equipped with a control PC and a ring of sonar proximity sensors (see Fig. 1).

Since such an assistive device does not only operate in the direct vicinity of people but also carries a person, who is often entirely dependent on the correct behavior of the system, demanding safety requirements have to be satisfied. Service robots in general and the wheelchair Rolland in particular have to be considered as safety-critical systems according to [12]. Therefore, during the development of Rolland, much attention has been paid to the dependability of the software, among other things by using formal methods to prove certain properties of the wheelchair's braking behavior.

In contrast to other (autonomous) service robots, Rolland is jointly controlled by its human operator and by a so-called safety module. Depending on the active operation mode, either the user or the automation is in charge of driving the wheelchair. Conflict situations arise if the commands issued by the two control instances contradict each other. How to detect and subsequently avoid such shared control conflicts is the subject of this paper.

2 SAFETY-CRITICAL SYSTEMS AND SHARED-CONTROL

A common approach in developing safety-critical systems is to introduce a controller into the system that interacts with the environment and the so-called *equipment under control* (EUC) via sensor and actuator channels, respectively (see Fig. 2). The EUC comprises the basic version of the technical system, i.e., a naive implementation that does not pay attention to the safety requirements. The behavior of the EUC in the absence of any controller is specified in the *physical model* which comprises the physical laws, parameters of the system, etc. The physical model is fundamental in the verification and validation process in that it provides the reference document that describes the reactive behavior of the system. If there are modeling errors in the representation of the EUC, the meaning of a future verification is rather limited.

Figure 1. The Bremen Autonomous Wheelchair Rolland: A control PC and a ring of sonar sensors for obstacle detection are mounted on a commercially available power wheelchair

Considering Rolland for instance, the EUC consists of the basic power wheelchair without any sensors or computer. The physical model describes the braking and accelerating behavior of the wheelchair, the character of the environment, etc. The controller is a computer program that processes the data it receives from the sonar sensors as well as from the wheelchair itself (current speed and steering angle).

If the EUC is not a purely technical system but one that involves a human operator, the physical model is insufficient to adequately describe the system's behavior. This is because there are no physical laws prescribing which command the human operator will issue in a certain situation; in the worst case his or her decisions are completely nondeterministic. Figure 3 shows the human operator as an unpredictable additional "environment". There is no direct interface between the user and the controller. Instead, potential warnings or

[1] Bremen Institute of Safe Systems, University of Bremen, P.O.-Box 330440, 28334 Bremen, Germany, {alone,roefer}@tzi.de

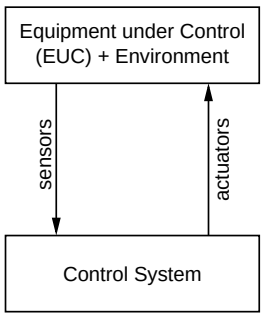

Figure 2. Basic idea of a safety controller in a common embedded system

status reports about the system state are communicated via the standard interface which is part of the EUC. Employing these means, the human operator must be enabled to track the system's state in order to have at least a chance to choose the commands in accordance to the specific situation.

However, in complex embedded systems which offer a huge variety of operation modes, problems often occur due to the fact that the technical system is in a different mode to that the human operator expects it to be in. In addition to the engineering task to be solved when designing a shared-control safety-critical system, the *mode confusion potential* has to be given careful consideration.

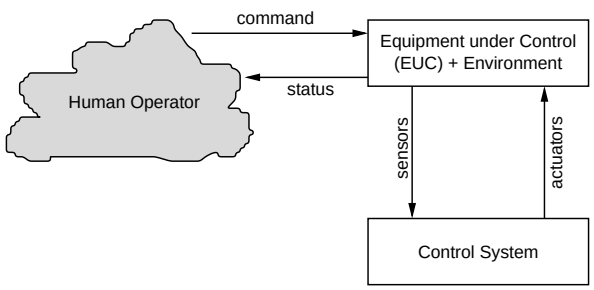

Figure 3. Problem of a shared control embedded system

In order to find and to eliminate neuralgic points in the system as early as possible during the design process, the approaches presented later have to cover various kinds of conflict situations, most of which result from mode confusions.

3 MODE CONFUSION

In recent publications (e.g., [5, 11]), a variety of causes of problems in shared-control systems has been described. These problems occur if the (controlling) human operator and the (potentially also controlling) technical system have divergent assumptions about the actual system state. Such situations arise, if the mental model of the operator does not match the model of the technical system about the behavior of the whole system.

The following list intends to introduce the important items that are especially relevant to the wheelchair application presented later.

Interface interpretation errors occur if the human operator takes an inappropriate action because he or she misinterprets the actual system output, often due to bad interface design. As an example consider the modes for inserting and overwriting text in a word processor. When changing between these modes, often only a small detail of the word processor's user interface is changed.

Inconsistent behavior of the automation causes problems because the system does not work as expected. A basic requirement is that the automation must work deterministically so that the same input command from the operator in the same state results in the same output.

Indirect mode changes happen if the automation changes modes autonomously (without previous operator instruction). In such situations, the operator's mental model is prone to lose track of the real state of the system.

Operator authority limit. If the automation is authorized to override operator commands under certain conditions, the human operator may suffer from this limitation in unforeseen situations. The standard example is the "escape-from-fire" scenario: if the proximity sensors of a wheelchair robot misinterpret dense smoke as obstacles, the automation might hinder the human operator's escape.

Lack of appropriate feedback. The automation should indicate any important mode transitions that it performs. It is very difficult to define exactly which mode transitions are "important"; indicating too many transitions probably results in a careless operator after a while, whereas confining the indication of mode transitions to the really important ones may cause the operator's mental model to diverge from the actual system behavior sooner or later.

4 TACKLING SHARED-CONTROL PROBLEMS BY USING FORMAL METHODS

Recent research aims at finding solutions to the shared-control problem with the help of formal methods (e.g. [2]). The basic idea is to check the (formal) specification of the system (including the interaction with the user) for potential conflict situations. The challenging part is to model and to formally specify the behavior of the human operator correctly. As a number of plane crashes were caused by mode confusion problems that occurred between the pilot and the automation, there is a huge industrial interest in these techniques.

Two prominent approaches will be briefly sketched here: The specification language SpecTRM-RL by Nancy Leveson [5] and a model checking method proposed by John Rushby [10].

4.1 Detecting Mode Confusion Potential with SpecTRM-ML

In [5] the authors present a method to detect potential mode confusions in system specifications. They employ the formal language SpecTRM-RL (Specification Tools and Requirements Methodology Requirements Language) to describe the technical system as well as the human operator. SpecTRM-RL is based on state machine semantics. It has the advantage that it is formally analyzable as well as readable by humans.

The approach works as follows: first, Leveson and colleagues identify a list of errors that are typically made by humans in shared-control systems. Afterwards, the black-box behavior of the automation is analyzed in order to find out which situations are error-prone. Finally, suggestions can be made as to how to improve the system design to avoid the mode confusions. Even though the specification language is automatically processable, the authors are convinced that human experts should do the analysis.

4.2 Model Checking for the Detection of Mode Confusions

In [10] the authors make use of the generally accepted assumption that the user of a technical system develops an individual mental model which describes the behavior of the technical system. Being a kind of internal representation of the behavior of the automation, such a mental model is the basis for decisions about interaction with the system. Today, it is rather well-known how to formally model a technical system. There are graphical techniques such as Petri Nets, Statecharts etc., and there is a large variety of formal specification languages such as the Duration Calculus, CSP, etc. If one could model the behavior of the human operator, i.e., if one could formally specify the mental model of the user with the same description technique, both specifications should be comparable. It would be rather straightforward to check whether the human's mental model can lose track of the system behavior. If so, a mode confusion situation would have been detected.

In order to find out whether the design of the technical system as specified has potential mode confusions, the model of the automation is not compared to the mental model of an individual but to a general one. Such generally applicable mental models can be derived from operator instruction material. Rushby and colleagues use (finite) state machines to specify the operators mental model as well as to describe the technical system. In a subsequent step, they are able to employ a standard model checking tool to search for potential mode confusions.

The results presented in [10] are encouraging in that the approach ensures that any mode confusion potential will be detected automatically, provided that both the technical system and the user's mental model were correctly specified.

5 SHARED-CONTROL IN SERVICE-ROBOTS

Both papers that were discussed in the previous section draw their "benchmark" scenario from avionics: a human pilot has to cope with the automation of an aircraft — and vice versa. Even though these systems are much more complex than most service robots, the principles remain the same in both application areas.

In order to illustrate the role of shared-control in service robots such as the wheelchair Rolland, its operation modes and the resulting confusion potential are discussed in the following subsections.

5.1 Operation Modes of the Bremen Autonomous Wheelchair

The user controls the commercial version (no control PC, no sensors) of this power wheelchair with a joystick. The command issued via the joystick determines the speed and the steering angle of the wheelchair. In the current state of development of Rolland, the joystick is the only interface between the technical system and its human operator. So far, there is no means to indicate mode changes to the user. Thus, most of the potential interface related errors listed in section 3 had to be considered during the development process. In the future, a speech recognition and a voice output system will serve as additional means of communication.

The idea of Rolland's safety layer (cf., [4, 6]) is to wiretap the control line from the joystick to the motor. Only those commands that won't do any harm to the wheelchair and its operator are passed unchanged. If a command turns out to be dangerous (e.g., approaching too close to a wall), the safety layer intervenes and decelerates the vehicle. Thus, this fundamental module ensures safe traveling in that it guarantees that the wheelchair will never actively collide with an obstacle.

The specification of the safety layer was deduced with the help of a fault-tree based hazard analysis [3]. A stepwise refinement of the threat *Collision with an Obstacle* leads to a set of safety requirements that must hold to ensure that the wheelchair never collides with an object in its surroundings. For instance, it can be derived that the safety module must never overestimate the distance to an obstacle the wheelchair is approaching. As collision avoidance is the only task of this module, it makes a binary decision: either the command issued by the human operator with the joystick is safe inasmuch as an emergency brake is not necessary at the current point in time, or the wheelchair would inevitably collide with an obstacle if the command of the user were executed, so an emergency brake is required immediately. Parts of the safety module have been proven to be correct with the help of the model checking tool FDR [3].

Above the safety module, higher-level skills provide additional functionality: obstacle avoidance (i.e., smoothly detouring around objects in the path of the wheelchair), assistance for passing the doorway, behavior-based traveling (wall following, turning on the spot, etc.) and others. These modules have been combined to the *driving assistant* (cf., [7, 8]). It provides the driver with various levels of support for speed control and for steering. Since this module averts collisions with obstacles and facilitates difficult maneuvers such as doorway passage, it is useful for most people confined to a wheelchair. Depending on the need of the individual user, the driving assistant could only realize a kind of emergency brake if the human operator overlooked an obstacle. This happens rather often, especially in the back of the vehicle which heavily swings out in curves. In an alternative scenario, the driving assistant takes a rough indication of the travel direction desired by the driver and converts it into a safe motor command. As a consequence, the human operator remains in control of the "large-scale navigation", and controls the wheelchair as he or she used to, e.g., via the joystick.

Depending upon the amount of control left to the human operator, Rolland's operation modes can be classified in a hierarchy as displayed in Fig. 4.

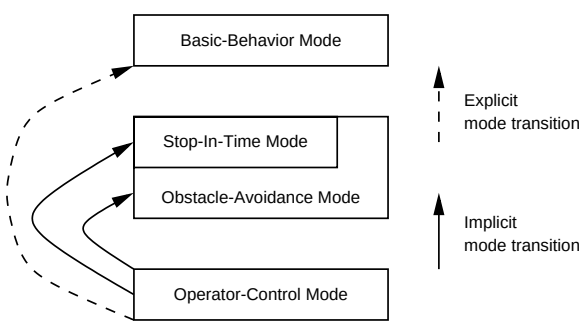

Figure 4. Hierarchy of operation modes of the Bremen Autonomous Wheelchair

The following paragraphs focus on the functionality of these modes and potential mode confusion scenarios.

5.1.1 Operator-Control Mode

In the *operator-control mode*, the wheelchair only monitors the commands issued by the human operator via the joystick. If there is no

obstacle close to the wheelchair, the safety module does *not* alter these commands. If there is an obstacle dangerously close to the wheelchair, the automation performs a transition into the *stop-in-time mode*. The notion "dangerously" refers to a situation in which there is an object in the surroundings of the wheelchair that would be hit, if the vehicle was not decelerated to a standstill immediately.

5.1.2 Stop-In-Time Mode

In the *stop-in-time* mode, the safety module overrules every command given by the user and sets the target speed to zero. In addition, it freezes the steering angle to the current value at the moment the stop-in-time mode was invoked. The underlying idea is the fact that you can travel faster in cluttered environments if you ensure that the steering angle remains constant during an emergency brake. Note that in this mode the human operator cannot control the wheelchair in any kind of way. As the driving comfort significantly suffers from frequent activations of the stop-in-time mode, it is basically enclosed within the so-called *obstacle-avoidance* mode.

5.1.3 Obstacle-Avoidance Mode

The obstacle-avoidance mode ensures that emergency braking maneuvers are avoided whenever possible. If in this mode, the wheelchair smoothly decelerates the velocity when approaching an obstacle. During this deceleration, the user is still in control of the steering angle. If the automation realizes that the projected path of the vehicle is free again, it again accelerates up to the speed indicated by the human operator via the joystick. In addition to the smooth speed control, this mode causes the wheelchair to detour around obstacles in its projected path: If there is an object close to the wheelchair *and* the user indicates a travel direction that points to the left or to the right of the object, the automation reduces the speed and changes the steering angle accordingly.

Whereas the transitions between the modes presented so far are implicit (i.e., the technical system autonomously changes operation modes without any indication to or feedback from the user, there is an additional *basic-behavior* mode that has explicitly to be chosen by the driver.

5.1.4 Basic-Behavior Mode

If the system carries out a basic behavior, such as wall following or turning on the spot, it ignores the position of the joystick completely. The only way of intervening during the execution of the autonomous behaviors is to interrupt the behavior with a certain joystick movement. After the automation successfully carried out such a behavior, the user regains control within the operator-control mode.

5.2 Solving a Shared-Control Problem: Obstacle Avoidance

When analyzing the problems of both experienced and unexperienced users of the wheelchair, we found out that a profound knowledge of the automation is indispensable in order to decide which action is best (i.e., in terms of matching the operator's intention) in a certain situation. However, training of the operator cannot be the only means to reduce the mode confusion potential. Instead, the automation itself has to be adapted to the human understanding of driving a power wheelchair.

As an example of such an adaptation, the history of the implementation of the obstacle avoidance mode of the Bremen Autonomous Wheelchair is briefly discussed here.

Similar to the Vector Field Histogram (VFH) method described in [1], the first obstacle-avoidance approach [4, 9] used for Rolland tried to figure out which steering angle would be the most promising in a given situation. Therefore, for each steering angle the projected path for a certain distance was checked for potential obstacles. If it turned out that the current steering angle was inferior, the automation took control and set the steering angle to that with the longest possible travel distance. The major drawback of this method is that the human operator has no chance to directly approach an obstacle. If a certain distance to the object was reached, the algorithm would have steered to the left or to the right. As the mode transition to the obstacle-avoidance mode is not indicated via any interface to the driver, he or she might get confused if the wheelchair behaves contra-intuitively. For instance, consider a situation in which the user wants to approach a wall in a narrow corridor in order to turn round. The obstacle-avoidance algorithm would try to "detour" around the wall. As a result, the user would not be allowed to drive close enough to the wall to make use of the whole shunting space that is required to be able to turn in the narrow corridor.

As a consequence, a new obstacle avoidance method was developed [7]. It realizes an intelligent shared-control behavior by projecting the anticipated path of the wheelchair into a local obstacle occupancy grid map. Depending on the side on which the projected path indicated with the joystick passes the obstacle, the algorithm decides how to change the steering angle in order to best serve the user. If instead, the driver directly steers toward an obstacle, the algorithm infers that the he or she wants to approach the object and does not alter the steering angle. As a result, both requirements are fulfilled: obstacles are smoothly detoured if desired, but they can be directly approached if need be.

As depicted in Fig. 4, the transition to the obstacle-avoidance mode is an implicit one, i.e., the mode is not invoked by the user by purpose. The resulting fact is that the driver does not adapt to the new situation after an obstacle has been detoured, because he or she did not notice that the operation mode changed from operator-control to obstacle-avoidance. It is very likely that the user would not react immediately after the avoidance maneuver and steer back to the original path. Instead, he or she would probably not change the joystick commando. As a consequence, the wheelchair would follow the wrong track.

This mode confusion problem motivates an additional feature of the new obstacle-avoidance algorithm of Rolland: It is able to steer back to the projection of the original path after the obstacle was passed. If the user does not adapt to the new situation, i.e., he or she does not change the joystick position after a detouring maneuver, the algorithm does interpret the command in the frame of reference that was actual when the maneuver began. As a consequence, it is able to navigate through a corridor full of people or static obstacles by simply pointing forward with the joystick. If there is an object that has to be detoured, the user keeps the joystick in an unchanged position and thereby enables the obstacle-avoidance algorithm to steer back to the projection of the original path.

6 FUTURE WORK

In the future, the human-machine interface will be improved. By means of a speech module, the wheelchair will indicate mode changes. As a consequence, confusing situations in which, for in-

stance, the driving assistant tries to circumvent an obstacle that cannot be seen by the human operator will occur less often. In such a scenario, it is likely that the operator will issue commands that contradict those set by the driving assistant.

In addition, a formal approach, similar to the model checking technique shown in section 4.2, will be used to guarantee the non-existence of mode confusion potential and thereby ensuring the safety of the shared-control service robot Rolland.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft through the priority program “Spatial Cognition”.

REFERENCES

[1] Borenstein, J., Koren, Y. (1991). *The Vector Field Histogram – Fast Obstacle-Avoidance for Mobile Robots*. In: IEEE Journal of Robotics and Automation. Vol. 7, No. 3. 278-288.

[2] Butler, R. et al. (1998). *A Formal Methods Approach to the Analysis of Mode Confusion*. In: Proc. of the 17th Digital Avionics Systems Conference. Bellevue, Washington.

[3] Lankenau, A., Meyer, O. (1999). *Formal Methods in Robotics: Fault Tree Based Verification*. In: Proc. of the Third Quality Week Europe. Brussels, Belgium.

[4] Lankenau, A. et al. (1998). *Safety in Robotics: The Bremen Autonomous Wheelchair*. In: Proc. of AMC’98, Fifth Int. Workshop on Advanced Motion Control. Coimbra, Portugal. 524-529.

[5] Leveson, N. et al. (1997). *Analyzing Software Specifications for Mode Confusion Potential*. In: Proc. of the Workshop on Human Error and System Development. Glasgow, UK.

[6] Röfer, T., Lankenau, A. (1998). *Architecture and Applications of the Bremen Autonomous Wheelchair*. In: Wang, P.P. (Ed.): Proc. of the Fourth Joint Conference in Information Systems 1998. Association for Intelligent Machinery. 365-368.

[7] Röfer, T., Lankenau, A. (1999). *Ensuring Safe Obstacle Avoidance in a Shared-Control System*. In: J.M. Fuertes (Ed.): Proc. on the 7th Int. Conf. on Emergent Technologies and Factory Automation. 1405-1414.

[8] Röfer, T., Lankenau, A. (1999). *Ein Fahrassistent für ältere und behinderte Menschen*. In: Schmidt et al. (ed.): Autonome Mobile Systeme 1999. Informatik aktuell. Springer. 334-343.

[9] Röfer, T., Lankenau, A. (2000). *Architecture and Application of the Bremen Autonomous Wheelchair*. In: Wang, P. (ed.): Information Sciences Journal. Elsevier Science BV. To appear.

[10] Rushby, J. et al. (1999). *An Automated Method to Detect Potential Mode Confusions*. In: Proc. of the 18th AIAA/IEEE Digital Avionics Systems Conference. St. Louis, Montana.

[11] Sarter, N., Woods, D. (1995). *How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control*. In: Human Factors. Vol. 37

[12] Storey, N. (1996). Safety-critical computer systems. Addison-Wesley.

Human-Machine Interaction

Service Robotics and the Issue of Integrated Intelligence: the CARL Project

L. Seabra Lopes ([1]), **K.L. Doty** ([2]), **F. Vaz** ([1]) and **J.A. Fonseca** ([1])

([1]) Departamento de Electrónica e Telecomunicações, Universidade de Aveiro / P-3810 Aveiro - Portugal
([2]) Machine Intelligence Laboratory, University of Florida, Gainsville, FL, USA

Abstract
It is argued in this paper that designing and building intelligent robots can be seen as the problem of integrating four main dimensions: *human-robot communication*, *sensory-motor skills and perception*, *decision-making capabilities* and *learning*. Some of these dimensions have been thoroughly studied in the past. However, their integration has seldom been attempted in a systematic way. This integration is crucial for Service Robotics: it will allow the development of robots that, instead of being programmed in the classical way, will accept, via a friendly interface (preferably by speech communication), instructions in terms of the concepts familiar to the user. The integrated intelligence issue is the subject of the CARL project, described in this paper. The project addresses the integration challenge paying particular attention to symbol grounding. As applications, the project is envisioning a "tour guide" and a "mobile manipulator".

1. Introduction

The development of systems that don't have to be programmed in the classical way and, instead, can accept instructions at the level of concepts of the human user will be a major breakthrough.

In manufacturing, flexibility implies a significant degree of reconfigurability and reprogrammability of the production resources. In order to implement reconfigurability, the modularity of the manufacturing hardware must be enhanced, such that, by combining resources in different ways, different operations can be performed. One of the few examples of this philosophy, already implemented in industrial practice, is the possibility of a robot changing the end-effector. Eventually, modular design will be commonly applied to other categories of resources, including fixtures and even arms. When the structuration constraints are reduced, the reprogrammability requirement can only be addressed by enhancing the autonomy of the manufacturing system.

An analogy can be established between service robots and flexible industrial robots. In fact, if a flexible manufacturing system is supposed to produce a variety of products and in small quantities, then industrial robots will tend to play the role of craftsmen. Therefore, both service robots and industrial robots will need to use sensors extensively in order to develop a high level understanding of their tasks.

Robot decision-making at the task level is, therefore, a central problem in the development of the next generation of robots. In manufacturing, as the modularity and reconfigurability of the hardware are enhanced, the number of action alternatives at the task-level increases significantly, which makes autonomous decision-making even more necessary.

The development of task-level robot systems has long been a goal of robotics research. It is of crucial importance if robots are to become consumer products. The use of the expression *task-level* is due to Lozano-Perez *et al.* (1989). The idea, that was already present in automatic robot programming languages, such as AUTOPASS and LAMA, developed in the 1970's, has been taken up in recent years by other researchers (Haigh and Veloso, 1996; Seabra Lopes, 1997).

The authors of the present paper are currently involved in CARL (= "*Communication, Action, Reasoning and Learning in robotics*"), a project aimed at contributing to the development of task-level robot systems. This paper outlines the adopted approach and planned demonstration.

To a large extent, the approach followed in CARL is based on the hypothesis that it is possible to build an intelligent robot by designing at least the general structure and organisation of the needed representations of the world as well as of the execution modules, from low-level control to high level decision-making. In such architecture, symbols play an important role. The problem of defining symbol meanings is the classical robot programming problem. The incorporation of learning mechanisms in robot architectures will be explored in order to simplify the programming problem. While most of the research on robot learning has been centred on control tasks using sub-symbolic learning techniques, CARL also addresses robot learning at the task level.

2. Robot Programming Paradigms

The economical viability of a robot system is often determined by the time required to program the robot to perform the desired tasks. Programming time is a function

of the features supported by the programming system, the ease in specifying the tasks and the support for program development and debugging. In some approaches, the planning phase, is significantly automated. In other cases, detailed planning is manual and the executable plan must be hand-coded step by step. On the other hand, depending on the execution skills of the robot controller, detailed planning of each operation may be simplified, or may require extensive mathematical treatment.

In which concerns robot programming, the main approaches that have been pursued until now are the following:

Industrial Practice — The most common programming method in industrial practice is a combination of robot guiding and textual programming. Most commercially available robots require position variables to be initialised by human guidance, with help of a teach pendant. The flow of execution is then specified by means of a robot-level programming language.

Automatic Programming — As task planning techniques become available, they are slowly being integrated in robot languages, leading to the *automatic programming* paradigm, which meets the industrial need for more flexible and powerful programming systems. The general goal is to allow the user to program in his/her domain of concepts. Usually, the language is a formalism for describing the task in terms of abstract operations and the underlying model supplies a method for handling the details.

Programming by Human Demonstration — This paradigm has been attracting increasing attention of robotics researchers. Applications range from low-level skill acquisition to task-level planning (Asada and Asari, 1988; Kang and Ikeuchi, 1995; Kuniyoshi, *et al.*, 1994; Morik *et al*., 1999). Human demonstrations can be performed in a virtual environment or in the physical setup.

The combination of guiding with robot-level programming languages, that is common in industrial practice, has many limitations. In fact, as the robot working environments become less structured, the variety of tasks increases and the robot users (either in manufacturing or in the service sector) become less aware of the technical details (their focus of attention is on the tasks), the traditional methods will no longer be practicable. Automatic programming and programming by demonstration are, currently, the main alternative approaches being investigated. In addition, for the same reason that we would like robot users to be able to describe tasks in their own domain of concepts (at the task level) it would also be desirable to solve the whole programming problem by considering also *human explanations*. This is especially important in which concerns the handling of unexpected situations.

Robot programming in the next generation of robots must, therefore, combine high-level task specifications, demonstrations in physical or virtual environments and human explanations. Of course, as will be pointed out next, in order to make full use of all these categories of feedback from the user, the robot must also be able to make decisions concerning actions and, very important, to learn upon experience.

3. Robot Autonomy and Intelligence

As the degree of structuredness and predictability of the environment decreases and the interaction with non expert humans becomes more intense, robots will need a deeper understanding of what is going on around them. In some way, they will have to be intelligent.

The nature of intelligence has been studied by philosophers since ancient times. In our days, researchers from other fields, including psychology, cognitive science and artificial intelligence, joined the debate. Unfortunately, while some abilities related to intelligence, such as learning or reasoning, can be defined precisely, the word *intelligence* seems to have different meanings for different people. Humans are intelligent, everybody seems to agree. But, are animals intelligent? Is a computer running a medical expert system intelligent?

Artificial intelligence researchers have been characterizing intelligence along two dimensions: *thinking* and *acting* (Russell and Norvig, 1995). In either case, success can be measured by comparison to *human performance* or to an ideal concept of intelligence, that may be called *rationality* (a system is considered rational if it does things in the best way).

The traditional viewpoint in artificial intelligence research used to be to evaluate an agent's intelligence by comparing it's thinking to human-level thinking. In 1950, Turing proposed a test to provide an operational definition of intelligent behavior: an agent would pass the test if it could be taken as a human during a dialogue with another human. This would involve natural language understanding, knowledge representation, reasoning and learning.

Even if emulation of human intelligence is not a priority, and emphasis is placed on rational action, these capabilities will be necessary for agents whose goals are defined at a high-level of abstraction. In recent years, human-level intelligence has been attracting increasing attention in the robotics community. For instance, Fukuda and Arai (1995) consider that an intelligent robot should have three main abilities: (1) reasoning ability; (2) learning ability; and (3) adaptation ability. In parallel, the development of humanoid robots is gaining momentum (Kanehiro *et al*., 1996; Brooks,1996).

A characterisation of intelligent systems in terms of *acting* is more interesting for robotics. The system is, in this case an *agent*, i.e. *something that perceives and acts*. One possibility is to characterise an intelligent agent as one that *acts rationally*, meaning that, given its perception of the world, it achieves its goals in the best way. Rational thinking (deductive reasoning) is not enough for acting rationally, because sometimes there isn't a way or the time to prove what is the best action to take and still something must be done. *Acting rationally therefore needs things like empirical reasoning and reactivity*.

In the late 1960's, the first robots showing some degree of intelligence and autonomy were presented. The mobile robot Shakey, developed by this time at the Stanford Research Institute, is generally considered as a milestone in autonomous systems. Shakey could already perform some of the functions that are considered, still today, as fundamental for autonomous systems, namely perception, representation, planning, execution monitoring, learning and failure recovery. By the same time, Winograd (1972) presented SHRDLU, a natural language understanding system for a physically embedded agent, a robot manipulating blocks.

Unfortunately, AI techniques and representations, traditionally tested only in toy problems like the blocks world, did not extend easily to complex dynamic environments (Brooks,1986; 1991; 1996; Maes, 1993; Arkin, 1998). It was found difficult to maintain world models and to connect the symbols in these models to numerical variables of the outside world. Furthermore, AI systems tended to be very slow, due to the sequential processing of information and to the cost of maintaining the world model. They had difficulty in reacting to sudden changes in the environment. This is partially related to the combinatorial explosion problem in planning, which also limited the decision-making capabilities of the system. Finally, AI systems were brittle, in the sense that they failed in situations only slightly different from those they were programmed for.

The realisation of these problems motivated substantially new approaches, for instance reactive reasoning and planning, multi-agent systems and distributed AI, as well as a revival of the connectionist paradigm. The integration of learning mechanisms in robot architectures was considered important in order to reduce the programming effort.

Finally, in the 1990's optimism seems to have returned to those who are developing intelligent robots and systems. Reactivity, reasoning and symbol grounding are major issues, that will now be briefly analysed.

3.1. Reactivity and Emergent Functionality

In the tradition of STRIPS, classical AI planners [Tate *et al.*, 1990] rely on problem-solving techniques. A set of operator schemata characterise the specific application domain, and the problem is usually to determine a sequence of operator instances that transform a given initial state of the world into a given goal state. Being purely deliberative, classical planning is adequate for strategic planning problems.

In real applications, however, executing a plan also requires some ability for improvisation. The plan executive will need to make short-term decisions continuously, in order to adapt to a dynamic and uncertain environment. This is especially true in robotics. Working at a symbolic level, classical planning systems typically have difficulty in reasoning about sensing and coping with uncertain information. On the other hand, combinatorial explosion in problem solving slows down the mapping of sensing to action. This led to the development of purely reactive systems.

The decomposition in functional modules, common in classical AI approaches, did not exclude a hierarchical organisation between them. Even in reactive planning systems, hierarchies have been used. For robotics applications, where many levels of abstraction can be identified, several architectures were proposed in which a hierarchical or layered organisation is emphasised. The most famous and influential of these proposals is probably the *subsumption architecture* (Brooks, 1986): organised in various layers, it supports levels of competence which go from basic reflexes (reactions) to reasoning and planning. The more abstract control layers use the functionality implemented by the lower layers, but can also use directly sensor data as well as send commands to actuators. Each layer is implemented by a set of modules which run in parallel, without any kind of centralised control.

The main goal of Brooks was to build robots that can survive for long periods in dynamic environments without human assistance. In more recent work, Brooks has taken this "decentralisation" even further by completely removing explicit internal representation and reasoning from robot control architectures: *intelligence without reason* (Brooks, 1991). A centralised world model is not necessary because the w*orld is its own best model*. The functionality of the robot emerges from the interaction among its components and with the environment. Intelligence is not explicitly encoded. The robot is simply observed to be intelligent, depending on its emergent functionalities and the situation of the world.

More recently, Brooks (1996) proposed a second paradigm shift, from behavior-based to cognitive robotics. The basic idea seems to be to scale up the behavior-based approach to generate human-like behavior and intelligence. Since 1993, Brooks has been building Cog, a humanoid robot, to demonstrate his new ideas.

However, the emergence of complex human behavior and intelligence in the humanoid does not seem to be an easy task. Brooks is the first to acknowledge that the behavior-based approach has been demonstrated almost exclusively on navigation tasks, where the problem of motivation for engaging in other activities does not even arise. Sensing, action, reasoning and learning are deeply rooted in human bodies, which are impossible to reproduce with available technology. At most the human appearance can be reproduced. Furthermore, even if the technology for building good approximations of human bodies was available, there is the whole history of mankind evolution. Human intelligence emerged from that evolution process. Many scientists are therefore skeptical about purely behavior-based or cognitive approaches and prefer to design explicitly the reasoning and control mechanisms of the robot or system (Hayes-Roth,1990; Haigh and Veloso,1996; Seabra Lopes, 1997).

3.2. Combining Reactivity with Reasoning

Reactive and behavior-based systems addressed with some success the problem of situated activity, coping with dynamically changing environments. In particular, the display of emergent intelligent-looking behavior by

reactively controlled robots is considered one of the successes of the approach. However, despite the modular and incremental design that is often used, reactive systems seem to be difficult to build, requiring a great engineering effort to hard-code the desired functionality.

Being limited to applications requiring no explicit representation of the world is a major limitation of purely reactive systems. For instance, in mobile robots, the classical path planning problem requires some representation of space. Any solution superior to random motion necessitates an internal model of the robot's current location.

A related fundamental limitation of this type of systems is that they are generally unable to display adequate goal directed behavior. On the other hand, when new situations arise, the ability to plan ahead could avoid time-consuming and possibly dangerous explorations of the environment. Moreover, a search in an internal model is orders of magnitude faster than the same search performed on the environment. There is a great difference between emergent clever-looking behavior and truly clever behavior. It should be noted that, despite the efforts of the emergent computation community, research in *emergent mind* remains purely speculative (Havel, 1993).

While purely reactive systems continue to be investigated, many researchers started to develop hybrid systems. Typically, reactivity is employed in low-level control and explicit reasoning is employed in higher-level deliberation (Georgeff and Lansky, 1987; Ambros-Ingerson and Steel, 1988; Connell and Mahadevan, 1993; Mataric, 1992; Firby *et al*.,1995; Seabra Lopes, 1997; Arkin, 1998). Horswill (1995) presented one of the firsts prototypes performing a classical AI task, from language through vision, without assuming a central model.

3.3. Symbol Grounding

Most of the initial AI research took a strong logical and symbolic perspective. This trend was encouraged by failures of connectionism, such as the inability of the initially proposed neural architectures to learn non linear problems. Although many of these problems have been solved with appropriate network structures and update rules, connectionist models continue to fail in symbol manipulation and language-level problems. Another weakness of connectionist models is their inability for providing explanations.

The hypothesis on which most AI research is based is, therefore, that the mind (or in general an intelligent system) is a symbol system and cognition is symbol manipulation.

A symbol system is a set of tokens that are manipulated on the basis of explicit rules. These rules are also tokens or strings of tokens. The application of rules in symbol manipulation is strictly syntactic (i.e. based on the shape of symbols, not on their meaning) and consists of combining and recombining symbol tokens. The symbol tokens and, therefore, the entire symbol system, are semantically interpretable: there is a way of systematically assigning a meaning to syntax. The problem, in this case, is how to define the symbol meanings, i.e. how to link the symbols to the outside world: the *symbol grounding problem* (Harnad, 1990; MacDorman, 1999). In connection to this problem, other more or less philosophical questions have been discussed, namely those relating to *consciousness* and *intensionality* (Russell and Norvig, 1995).

While the more philosophical questions still need a lot of debate and can be overlooked when developing practical applications of AI, the same cannot be said about symbol grounding. This is especially true in the robotics domain. Steels (1999) explores visual grounding in robotic agents.

Intelligent robots will need high-level decision making capabilities that are essentially based on symbol manipulation. However, the meaning of symbols must be obtained from sensors which provide essentially non symbolic data. Connectionism, with its ability to discover patterns, seems to be a good paradigm for implementing the sub-symbolic layers of a cognitive architecture. For this reason various authors have been proposing hybrid symbolic-connectionist approaches (Harnad, 1990). The exact architecture of these hybrid systems is the topic of discussion.

Sometimes it is argued that symbol grounding should be a bottom-up process. However, like humans, machines will benefit from using both grounding directions, i.e. symbols externally communicated will be grounded in a top-down fashion while other symbols will be discovered in a bottom-up fashion. Symbol grounding is intimately connected to learning: supervised learning is the most promising approach for top-down grounding while clustering is appropriate for bottom-up grounding.

4. Integrated Intelligence

The CARL project, here presented, is based on the hypothesis that a combination of reactivity with reasoning is more likely to produce useful results in a relatively near future than the purely reactive or behavior-based approaches. This is especially true for robots that are expected to perform complex tasks that require decision-making.

The integration of reactivity with reasoning has proven to be difficult to achieve. Horswill (1995) wrote: «Traditional architectures have focused on traditional problems (reasoning, representation, NLP) and alternative architectures have focused on problems such as real-time perception and motor control. There have been few if any satisfying attempts to integrate the two». The position and (driving hope) of the CARL project is that most of the encountered difficulties are the result of not addressing properly the learning and, especially, the interface issues.

The reference architecture of the project is shown in figure 1. It emphasises the integration of the following four major dimensions of the problem of building intelligent robots:

- *basic sensory-motor skills*,
- *decision-making capabilities*,
- *learning*, and
- *human-robot interaction*.

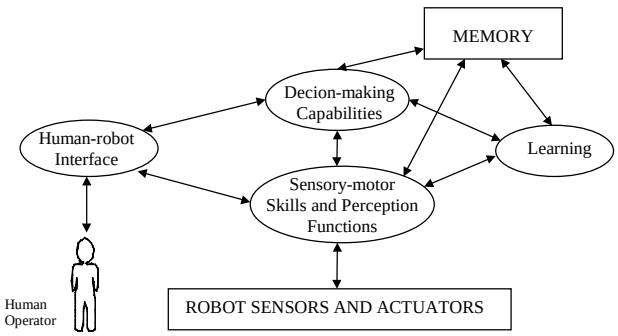

Fig. 1 - Reference architecture for the CARL project

4.1. Human-Robot Interface

A robot executing a task for humans is a process that involves making decisions concerning the task progress and controlling the hardware in the desired way. Decision making is primarily based on symbol manipulation. Even the primitive operations the robot is able to perform are viewed as symbols. As already pointed out, symbol grounding is a difficult problem to solve. However, to correctly address symbol grounding in the context of task execution, the first thing to notice is that most symbols are inherent to the tasks. The human user, who defines the tasks, will be a primary source of information for the symbol grounding process. The communication interface between human and robot is, therefore, of primary importance (Connell, 2000).

Fukuda and Arai (1995) point out that one of the most delicate and important factors to take into consideration for the success of such innovative devices, like service robots, relates to the psychological aspects and to the implementation of techniques for human-robot interaction in unprotected environments, such as a house.

In the traditional approach to building intelligent systems, the human devises a formal language and specifies symbol semantics with it. As the application becomes more and more complex, the task of the programmer becomes overwhelmingly difficult. The most pervasive approach to robot programming still today in industrial practice is a combination of guiding and textual programming. A teach pendant is used to teach the positions referenced in the program. This is probably the simplest and oldest form of symbol grounding in robotics. Efforts in developing automatic programming languages got stuck in some of the typical AI problems, namely automated reasoning and symbol grounding.

Ideally, the human-robot interface should be multi-modal (Flanagan, 1997). Certainly, it may still be useful (at least in a research robot) to specify some information textually using a formal language. A physical demonstration of the task, or of certain operations in it, is another form of communication. In some cases, demonstrations in a virtual environment can be a better alternative.

Meanwhile, if we are developing intelligent robots with significant decision making capabilities, the use of natural language seems to be the most flexible type of interface for exchanging task-level concepts and, certainly, a very comfortable one for humans (Jayant, 1997; Edwards, 1997). But, it is unavoidable because no other alternative is practical enough. The common (naïve) user does not want to learn a formal programming syntax. To our knowledge, the only project that has been consistently working in this direction is JIJO-2 (Asoh et al., 1997; Fry et al., 1998).

Teaching a task, for instance, should be an interactive process, in which the human explains one or two steps of the task, the robot tries them out and then the human explains a little more, and so on (Huffman and Laird, 1993). Natural language seems also to be the best way for easily guiding the robot in recovering from failure situations. In general, natural language seems to be the easiest way of presenting new symbols (representing physical objects, failure states, operations, or whatever) to the robot. Grounding these symbols depends essentially on the robot sensors and learning abilities.

4.2. Learning Module

Learning, defined as the process by which a system improves its performance in certain tasks based on experience, is one of the fundamental abilities that a system must possess in order to be considered intelligent. Learning can be seen as the combination of inference and memory processes (Michalski, 1994). The application of learning techniques to robotics and automation is recognised as a key element for achieving truly flexible manufacturing systems (Morik *et al*., 1999).

The problem with implementing robot autonomy for manufacturing or service is that it is not easy to design and program from scratch all the necessary sensory-motor and decision-making capabilities. The major contribution of learning is to generate models and behaviors from representative cases or episodes when such models and behaviors cannot practically be designed and programmed from scratch. The use of machine learning techniques for acquiring these capabilities in a framework of programming by human demonstration and explanation seems a very promising approach (fig. 2).

Until now, research in applying learning in robotics and automation has focused on learning control functions, adaptation and dealing with uncertainty in real-world parameters. Behavior-based robotics has investigated the integration of learning mechanisms in robot architectures for navigation and environment exploration (Connell and Mahadevan,1993). In this kind of applications, the external feedback used for learning is either scalar feedback or control feedback. In learning based on *scalar feedback*, the robot receives a positive or negative reward signal evaluating its performance. These rewards can be provided more or less frequently. In a fully unsupervised mode, only when the robot achieves the goal, will a reward be provided, confirming success. In between supervised and unsupervised learning, reinforcement learning is based on scalar feedback received at the end of every performed action, or at least when critical intermediate states are reached.

In learning based on *control feedback*, the robot receives information about the appropriate actions to take in different situations. This corresponds to supervised learning. It is especially suited for top-down symbol grounding and very useful in learning at the task level. Task-level robot systems will also need analytic feedback, meaning that the robot must be provided with explanations of why certain actions are (or not) able to achieve certain goals (Evans and Lee, 1994; Seabra Lopes, 1997, 1999).

4.3. Decision-making Capabilities

Initially, the human user explains the task in his/her own domain of concepts. For instance, the user might ask: "Please, get me a coke from the fridge". Executing this task involves reasoning about the task and the environment and making several decisions. To start with, the robot must plan a strategy to achieve its goals. Then, while executing the devised strategy, the robot may encounter unexpected situations, that eventually lead to failure. It is then necessary to analyse and recover from that failure. These decision-making activities can be addressed with AI techniques. Instead of hand-coding the knowledge that must support decision-making, learning and human-robot interaction is used to generate it.

Task planning — This is an instance of the action sequence planning problem (Hutchinson and Kak, 1990) which, in real-world situations, with many action alternatives, becomes very difficult due to search complexity. Having an understanding of why solutions to past task planning problems were successful may be a considerable help in new similar problems. Explanation-based learning (Kedar-Cabelli and McKarty, 1987) and case-based reasoning (Kolodner,1993) techniques may be of great help in robot task planning. Analytic feedback seems to play the major role in learning for nominal task planning.

Execution monitoring — The goal of execution monitoring is to detect unexpected situations during task execution. Monitoring certain goals is trivial. For others, learning may help, for instance by generating classification knowledge to discriminate between different situations. Learning in execution monitoring typically involves scalar and control types of feedback.

Failure diagnosis — A diagnosis function is called when unexpected situations arise (execution failures). In general, failure diagnosis can be divided into four main sub-problems, namely failure confirmation, failure classification, failure explanation and status identification (Chang *et al.*, 1991; Meijer, 1991; de Kleer et al., 1992; Camarinha-Matos, Seabra Lopes, Barata, 1996). The first two sub-problems are basically classification problems. Classification knowledge for diagnosis can be acquired using inductive concept learning methods (Seabra Lopes and Camarinha-Matos, 1998, 1999). Failure explanation uses a domain theory to explain the execution failure. Deriving explanations based on a domain theory and observations or classifications of aspects of the robot system is, like task planning, a problem that may involve a certain search complexity. Knowledge about previous explanation tasks may guide failure explanation. Learning by analogy and by deductive generalisation seems to play the main role in failure explanation. Finally, state identification can be decomposed into a set of classification tasks for recognising the effects of the initial exception, as determined in the explanation phase. Learning in failure diagnosis therefore primarily uses control and analytic feedback.

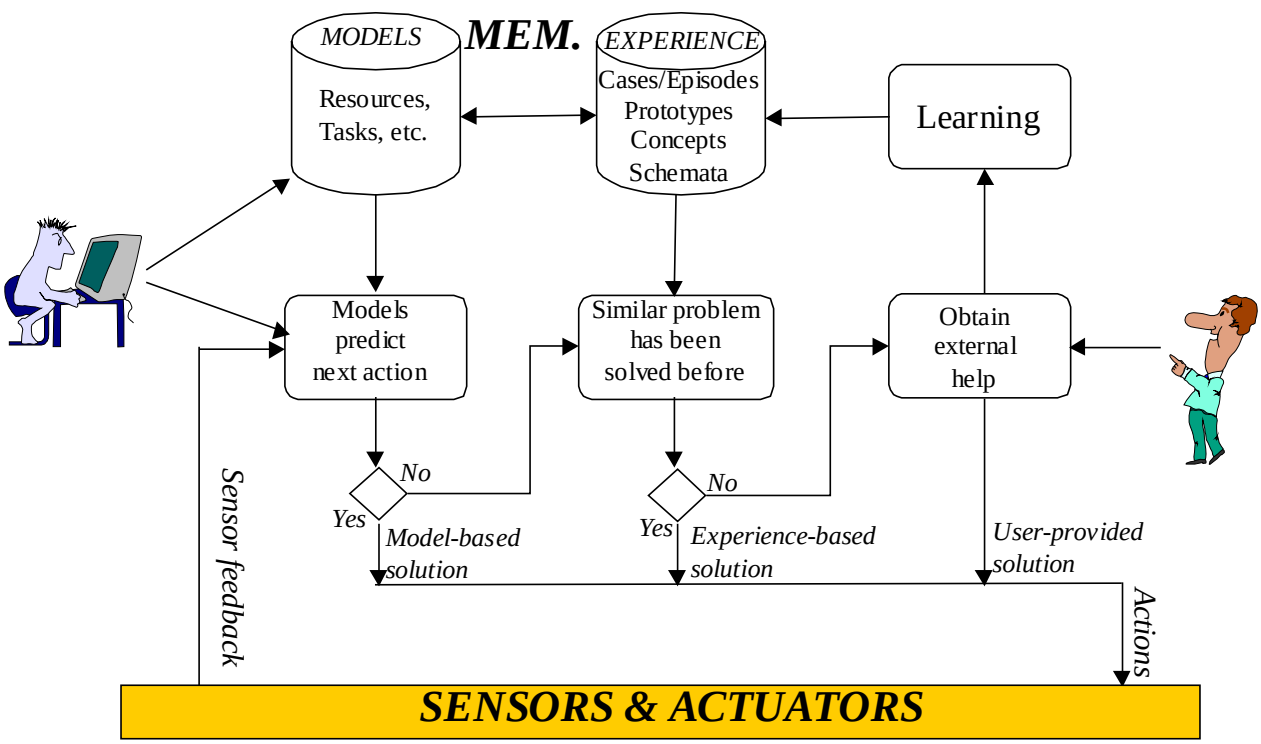

Fig. 2 – Robot programming as human-supervised robot learning

Failure recovery planning — After diagnosing a failure, the robot must plan a strategy to overcome the situation and achieve its goals. In principle, this strategy will make some corrections so that the robot may resume execution of the initial plan in the point of failure or some steps after. Pre-programming failure recovery routines and search-based planning are two approaches that cannot handle this complexity. The only solution seems to be adapting plans provided by some external agent (e.g. the human) for previous failure recovery problems (Seabra Lopes, 1999). Like in task planning and in failure explanation, analytic learning techniques seem to be the most promising in failure recovery.

4.4. Basic Skills

The interaction of the robot with the environment and, in particular, the execution of tasks is supported by a set of sensory-motor skills, each of them involving a tight connection of perception to action. A classical example, from the robotised assembly domain, is the peg-in-hole insertion skill, first studied, from a learning point of view, by Dufay and Latombe (1984). In a mobile robot, navigation with obstacle avoidance is also a basic skill. Developing skills has been addressed with different approaches, including model-based approaches, programming by demonstration and learning. The application of various types of neural networks, due to their ability to handle non-linearity and uncertainty, has already delivered interesting results. The use of reinforcement learning in on-line skill learning or refinement is also being investigated (Bagnell, Doty and Arroyo 1998; Nuttin and Van Brussel, 1995). Skill learning involves typically scalar and control types of feedback.

Developing robot skills for specific applications has been termed *skill acquisition*. An even more difficult problem could be called *skill discovery*. In this case, the robot must be actively exploring its environment. Such exploration involves the generation of sensory-motor interactions with the environment and the identification of their effects. Once in a while, some of the experimented sensory-motor interactions will produce effects that, from the point of view of the tasks and goals of the robot, are interesting to "keep in mind". This kind of learning, which necessarily involves all types of feedback (scalar, control and analytic), while very relevant for true learning robots, remains almost completely unexplored.

6. Current Work

Until now, the CARL team has been working mainly on robot construction (starting from a Pioneer 2-DX mobile platform) and on the human-robot communication interface (Seabra Lopes and Teixeira, 2000).

6.1. Carl, the robot

Carl is the name of the robot of the CARL project (fig. 3). It is based on a Pioneer 2-DX indoor platform from ActivMedia Robotics, with two drive wheels plus the caster. It includes wheel encoders, front and rear bumpers rings, front and rear sonar rings and audio I/O card. The platform configuration that was acquired also includes a micro-controller based on the Siemens C166 processor and an on-board computer based on a Pentium 266 MHz with PC104+ bus, 64 Mb of memory and a 3.2 Gb hard drive. The operating system is Linux.

We have installed a compass (Vector 2XG from Precision Navigation Inc.), a PTZ104 PAL Custom Vision System and a Labtec 7280 microphone array.

With this platform, we hope to be able develop a completely autonomous robot capable, not only of wandering around, but also of taking decisions, executing tasks and learning.

6.2. "Innate" capabilities

Our idea for Carl is to integrate, in the development phase, a variety of processing and inference capabilities. In contrast, the initial body of knowledge will be minimal. After this phase is concluded (after the robot is born!), a life-long learning process can start. Carl learns new skills, explores its environment, builds a map of it, all this with frequent guidance from the human teacher.

Some of the "innate" capabilities / knowledge, that will be integrated in Carl during the construction phase are:

- Wandering around in the environment while avoiding obstacles; this is the only "innate" behavior.
- Natural language processing, supported by a fairly comprehensive vocabulary of English words; the meanings of most words are initially unknown to Carl.
- Basic speech processing.
- A small dictionary of words and their meanings for identifying the robot's sensors and basic movements; these are the initially ground symbols over which Carl will incrementally build his knowledge.
- Ontologies for organizing and composing behaviors, map regions, dialogues, task plans, episodic memory, etc.
- Knowledge of basic mathematical functions, that the teacher can use for teaching new concepts or behaviors.
- Logical deduction (in a Prolog framework)
- Capabilities for task planning and execution monitoring.
- Capabilities for learning numerical functions.
- Capabilities for learning symbolic classification knowledge.
- Capabilities for explanation-based learning and case-based reasoning.

Part of these capabilities can be implemented by adapting prototypes previously developed by the research team (Seabra Lopes, 1997 and 1999ab).

6.2. Envisioned applications

The project is envisioning two demonstrations, one for the short-term and the other for the long-term. The short-term application is a "tour guide". This is an application in which, ideally, the robot enters in dialogue with the tourist or visitor, answering the questions that the tourist asks, and skipping the aspects in which the tourist does not seem to be interested. That means that the robot must really maintain a knowledge representation about what there is to see, adapt to the user and, at the same time, do all other things that robots are expected to do in the physical world, particularly navigation and localization. A

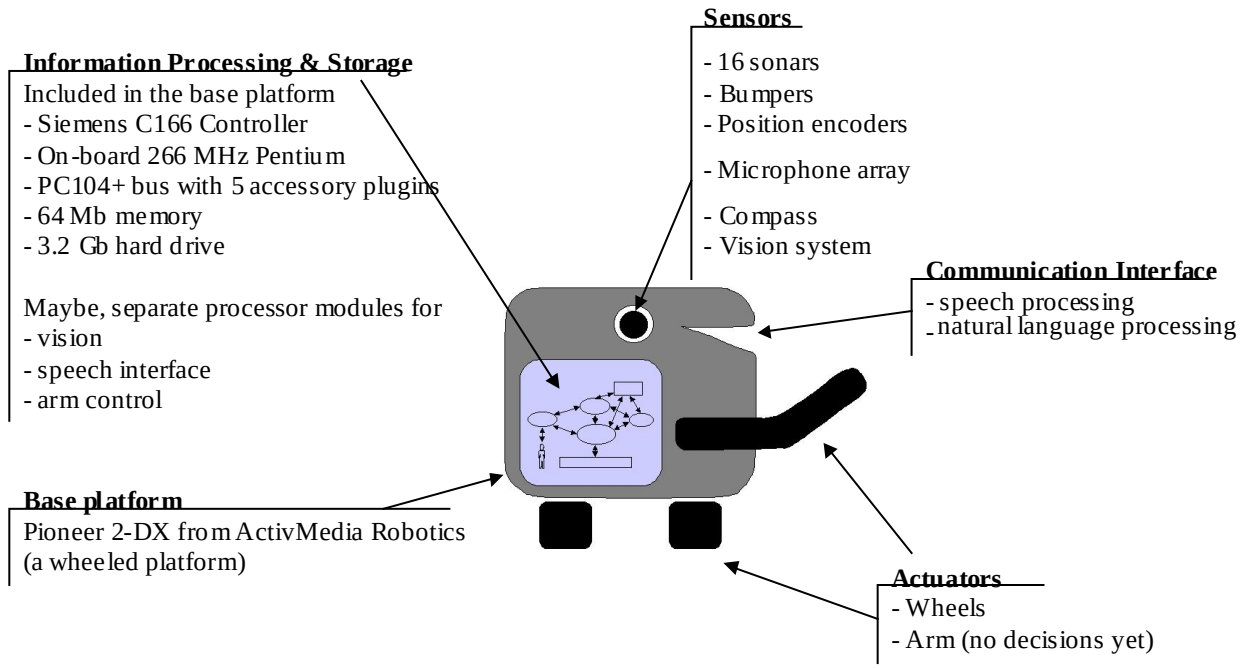

Fig. 3 – Carl, the robot

robotic tour guide for a museum in Bonn has been built at Carnegie-Mellon University (Burgard et al., 1999), but this robot only has pre-recorded messages and does not enter any kind of dialogue. We therefore want to see how far dialogue can go in such application with available speech technology.

The longer term demonstration is the "mobile manipulator" (Khatib, 1999) for materials handling in an unstructured environment (Doty and Van Aken, 1993). With this, the CARL project will study in greater depth the integration of human-robot communication and robot learning with the traditional reasoning and real-time action capabilities.

5. Conclusion and Current Work

The CARL project aims to contribute to the development of task-level robot systems by studying the interrelations and integration of the four major dimensions of the problem: *human-robot interaction*, *sensory-motor skills and perception*, *decision-making* and *learning*. A lot of literature has been produced on general decision-making (Russell and Norvig, 1995), on decision making for robots (Haigh and Veloso, 1996), on learning for robot decision making (Seabra Lopes, 1997), on different approaches to skill acquisition (Asada and Asari, 1988; Morik *et al*, 1999), on active sensing (Crawley,1995) and on human-machine interaction (Flanagan,1997; Jayant,1997). However, being all these aspects fundamental for developing intelligent robots, their integration in the same robot has seldom been attempted.

This lack of integration efforts in robotics is explained by various technological limitations. However, thanks to the increasing availability of compact hardware at reasonable cost and to the dramatic increase in computational power that was observed in recent years, the proposed integration work is now becoming feasible (Stork, 1997).

Within CARL, care is being taken to avoid developing every specific functionality that would be required, if the robot was a final product. Attention is, therefore, focusing on a small set of functionalities at the various levels of the architecture and on the problems that their integration raises. Particular attention is paid to symbol grounding in connection with the learning and communication modules.

Acknowledgements

CARL ("*Communication, Action, Reasoning and Learning in Robotics*") is a project funded by the Portuguese research foundation under program PRAXIS XXI, reference PRAXIS / P / EEI / 12121 / 1998.

References

Ambros-Ingerson, J. and S. Steel (1988) Integrating Planning, Execution and Monitoring, *Proc. of the 7th National Conference on Artificial Intelligence (AAAI-88)*, St. Paul, Minnesota, pp. 735-740.

Arkin, R.C. (1998) *Behavior-Based Robotics*, MIT Press.

Asada, H. and Y. Asari (1988) The Direct Teaching Tool Manipulation Skills via the Impedance Identification of Human Motions, *Proc. IEEE International Conference on Robotics and Automation*, pp. 1269-1274.

Asoh, H., S. Hayamizu, I. Hara, Y. Motomura, S. Akaho and T. Matsui (1997) «Socially Embedded Learning of the Office-Conversant Mobile Robot Jijo-2», *Proc. 15th Int. Joint Conf. on Artificial Intelligence (IJCAI-97)*, Nagoya, p. 880-885.

Bagnell, A., K.L. Doty and A.A. Arroyo (1988) "Comparison of Reinforcement Learning Techniques for Automatic Behavior Programming", *CONALD 98*, Carnegie Mellon University, Pittsburg, PA.

Brooks, R.A. (1986) A Robust Layered Control System for a Mobile Robot, *IEEE Journal on Robotics and Automation*, vol. RA-2 (1), pp. 14-23.

Brooks, R.A. (1991) Intelligence without Reason, *12th International Joint Conference on Artificial Intelligence* (IJCAI-91), Sydney, Australia, pp. 569-595.

Brooks, R.A. (1996) Behavior-Based Humanoid Robots, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS'96), Osaka, Japan, pp. 1-8.

Burgard, W., et al. (1999) Experiences with an Interactive Museum Tour-Guide Robot, *Artificial Intelligence*, 114, 3-55.

Camarinha-Matos, L.M., L. Seabra Lopes, J. Barata (1996) Integration and Learning in Supervision of Flexible Assembly Systems, *IEEE Transactions on Robotics and Automation*, vol. 12, p. 202-219.

Chang, S.J., F. DiCesare and G. Goldbogen (1991) Failure Propagation Trees for Diagnosis in Manufacturing Systems, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, 767-776.

Cole, R.A., S. Sutton, Y. Yan and P. Vermeulen (1998) Accessible Technology for Interactive Systems: a New Approach to Spoken Language Research, *Proc. ICASSP*.

Connel, J. (2000) Beer on the Brain, *My Dinner with R2D2: Working Notes of the AAAI Spring Symposium Series on Natural Dialogues with Practical Robotic Devices*, Stanford, CA.

Connell, J.H. and S. Mahadevan (ed.) (1993) *Robot Learning*, Kluwer Academic Publ., Boston.

Crowley, J.L. (1995) Integration and Control of Active Visual Processes, *Proc. of the Workshop on Vision for Robots*, Pittsburgh, Pennsylvania, pp. 80-85.

de Kleer, J., A.K. Mackworth and R. Reiter (1992) Characterizing Diagnoses and Systems, *Artificial Intelligence*, vol. 56.

Doty, K. L. and R.E. Van Aken (1993) "Swarm Robot Materials Handling Paradigm for a Manufacturing Workcell", *IEEE Int. Conf. on Robotics and Automation*, May 2-6, 1993, Atlanta GA., pp. 778-782.

Dufay, B. and J.-C. Latombe (1984) An Approach to automatic robot programming based on inductive learning, *International Journal of Robotics Research*, vol. 3, pp. 3-20.

Edwards, J. (1997) Voice-based Interfaces Make PC's Better Listeners, *Computer*, August 1997.

Eicker, P.J. (1995) An Approaching Golden Age of Intelligent Machines, *IEEE International Conference on Robotics and Automation: Opening Cerimony. Opening Keynote Lectures. Industry Forum*, pp. 37-39.

Evans, N.Z. and C.S.G. Lee (1994) Automatic Generation of Error Recovery Knowledge through Learned Reactivity, *Proc. IEEE Int. Conf. on Robotics and Automation, San Diego*, CA, 2915-2920.

Firby, R.J., R.E. Kahn, P.N. Prokopowicz and M.J. Swain (1995) Collecting Trash: a Test of Purposive Vision, *Proc. of the Workshop on Vision for Robots*, Pittsburgh, Pennsylvania, pp. 18-27.

Flanagan, J.L. (1994) Technologies for Multimedia Communications, *Proceedings of the IEEE*, v. 82, 590-603.

Fry, J., H. Asoh and T. Matsui (1998) «Natural dialogue with the JIJO-2» ofiice robot, *Proceedings of the IROS'98*.

Fukuda, T. and F. Arai (1995) Intelligent System and Intelligent Human Interface, *Proc. 1995 IEEE International Conference on Robotics and Automation / Workshop WM2: Task-Driven Intelligent Robotics: Methods and Technologies*, Nagoya, Japan, pp. 3.1-3.8.

Georgeff, M.P. and A.L. Lansky (1987) Reactive Reasoning and Planning, *Proc. 6th National Conference on Artificial Intelligence* (AAAI-87), Seattle, Washington, pp. 677-682.

Haigh, K.Z. and M. Veloso (1996) Interleaving Planning and Robot Execution for Asynchronous User Requests, *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS'96), Osaka, Japan, pp. 148-155.

Harnad, S. (1990) The Symbol Grounding Problem, *Physica D*, vol. 42, pp. 335-346.

Havel, I.M. (1993) Artificial Thought and Emergent Mind, *Proc. Int. Joint Conference on Artificial Intelligence*, Chamberri, France, 758-756.

Hayes-Roth, B., R. Washington, R. Hewett, M. Hewett and A. Seiver (1989) Intelligent Monitoring and Control, *Proc. 11th Int. joint Conf. on Artificial Intelligence*, pp. 243-249.

Horswill, I. (1995) «Integrating Vision and Natural Language without Central Models», *AAAI Fall Symposium on Embodied Language and Action*, Cambridge.

Huffman, S.B. and J.E. Laird (1993) «Learning Procedures from Interactive Natural Language Instructions», *Proc. 10th International Conference on Machine Learning*, Amherst, MA, pp. 143-150.

Hutchinson, S. A. and A. C. Kak (1990) SPAR: A Planner that Satisfies Operational and Geometric Goals in Uncertain Environments, *AI Magazine*, vol. 11, pp. 30-61.

Jayant, N. (1997) Human Machine Interaction by Voice and Gesture, *Proceedings of ICASSP*, Munich, 175-178.

Kaelbling, L.P. (1991) Foundations of Learning in Autonomous Agents, *Robotics and Autonomous Systems*, vol. 8 (1-2), p. 131-144.

Kanehiro, F., M. Inaba and H. Inoue (1996) Development of a Two-Arm Bipedal Robot that can Walk and Carry Objects, *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Osaka, Japan, p. 23-36.

Kang, S.B. and K. Ikeuchi (1995) Toward Automatic Instruction from Perception: Recognizing a Grasp from Observation, *IEEE Transactions on Robotics and Automation*, vol. 11, p. 670-681.

Kedar-Cabelli, S.T. and L.T. McCarty (1987) Explanation-Based Generalization as Resolution Theorem Proving, *Proc. 4th Int'l Workshop on Machine Learning*, Irvine, CA, pp. 383-389.

Khatib, O. (1999) Mobile Manipulation: the Robotic Assistant, Robotics and Autonomous Systems, 26 (2-3), 175-183.

Kokkinaki, A.I. and K.P. Valavanis (1995) On the Comparisson of AI and DAI Based Planning Techniques for Automated Manufacturing Systems, *Journal of Intelligent and Robotic Systems*, vol. 13, pp- 201-245.

Kolodner, J.L. (1993) *Case-Based Reasoning*, Morgan Kaufman Publ., San Mateo.

Kuniyoshi, Y., M. Inaba and H. Inoue (1994) Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance, *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 799-822.

Lozano-Pérez, T., J.L. Jones, E. Mazer and P.A. O'Donnell (1989) «Task-level Planning of Pick and Place Robot Motions», *Computer*, vol. 22, n.3 (March), pp. 21-29.

MacDorman, K.F. (1999) Grounding Symbols through Sensorymotor Integration, *Journal of the Robotics Society of Japan*, 17 (1), 20-24.

Mitchell, T. (1997) Machine Learning, McGraw-Hill.

Maes, P. (1993) Behavior Based Artificial Intelligence, *From Animals to Animats 2* (Proc. 2nd International Conference on Simulation od Adaptive Behavior, Honolulu, Hawaii, 1992), J.-A. Meyer, H.L. Roitblat and W. Wilson (ed.), MIT Press, pp. 2-10.

Mataric, M. (1992) Integration of Representation into Goal-Driven Behavior-Based Robots, *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 304-312.

Meijer, G.R. (1991) *Autonomous Shopfloor Systems. A Study into Exception Handling for Robot Control* (PhD Thesis), Universiteit van Amsterdam, Amsterdam.

Michalski, R.S. (1994) Inferential Theory of Learning: Developing Foundations for Multistrate-gy Learning, *Machine Learning. A Multistrategy Approach*, vol. IV, Michalsky, R.S.; Tecuci, G (eds.), Morgan Kaufmann Pub., San Mateo, CA.

Morik, K., M. Kaiser and V. Klingspor [edrs.] (1999) *Making Robots Smart. Behavioral Learning Combines Sensing and Action*, Kluwer Acedemic Publishers.

Narendra, K.S. and S.T. Venkataraman (1995) Adaptation and Learning in Robotics and Automation, *Proc. IEEE International Conference on Robotics and Automation*, Nagoya, Japan, pp. 1244-1249.

Nuttin, M. and H. Van Brussel (1995) Learning an Industrial Assembly Task with Complex Objects and Tolerances, *Proc. 4th European Workshop on Learning Robots*, Karlsruhe, Germany, pp. 27-37.

Rumelhart, D.E., G.E. Hinton and R.J. Williams (1986) Learning Internal Representations by Error Propagation, *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McLelland (ed.), MIT Press, Cambridge, Massachusetts, vol. 1, chap. 8.

Russell, S.J. and P. Norvig (1995) *Artificial Intelligence. A Modern Approach*, Prentice Hall, New Jersey.

Schraft, R.D. (1994) Mechatronics and Robotics for Service Applications, *IEEE Robotics and Automation Magazine*, vol. 1, n. 4, pp. 31-35.

Seabra Lopes, L. (1997) *Robot Learning at the Task Level: A Study in the Assembly Domain*, Universidade Nova de Lisboa, Ph.D. Thesis.

Seabra Lopes, L. (1997) *Robot Learning at the Task Level: A Study in the Assembly Domain*, Universidade Nova de Lisboa, Ph.D. Thesis.

Seabra Lopes, L. (1999a) Failure Recovery Planning in Assembly Based on Acquired Experience: Learning by Analogy, *Proc. IEEE. Int. Symp. On Assembly and Task Planning*, Porto, Portugal (to appear).

Seabra Lopes, L. (1999b) «Learning a Taxonomy of Failures in Assembly», in Morik, Kaiser and Klingspor [edrs.] *Making Robots Smart. Behavioral Learning Combines Sensing and Action*, Kluwer.

Seabra Lopes, L. and A. Teixeira (2000) Teaching Behavioral and Task Knowledge to Robots through Spoken Dialogues, *My Dinner with R2D2: Working Notes of the AAAI Spring Symposium Series on Natural Dialogues with Practical Robotic Devices*, Stanford, CA.

Seabra Lopes, L. and L.M. Camarinha-Matos (1998) «Feature Transformation Strategies for a Robot Learning Problem», *Feature Extraction, Construction and Selection. A Data Mining Perspective*, H. Liu and H. Motoda (edrs.), Kluwer Academic Publishers.

Stork, D. [edr.] (1997) HAL's Legacy: 2001's Computer as Dream and Reality, MIT Press.

Tate, A., J. Hendler and M. Drummond (1990) A Review of AI Planning Techniques, in J. Allen, J. Hendler and A. Tate (edrs.), *Readings in Planning*, Morgan Kaufmann Pub., pp. 26-49.

Turing, A.M. (1950) «Computing Machinery and Intelligence», *Mind*, 59, p. 433-460.

Winograd, T. (1972) Understanding Natural Language, *Cognitive Psychology*, 3 (1).

A Web Based Interface for Service Robots with Learning Capabilities

R. Marín[1], P. J. Sanz[1] & A. P. del Pobil[1]

Abstract. One of the essential points in the design of a telerobotic system is the characterization of the Human-Robot interaction to be implemented. In most telerobotic systems user interaction is still very computer-oriented, since input to the robot is accomplished by filling forms or selecting commands from a panel. Very little attention has been paid to more natural ways of communication such as natural language, or gestures. Besides this, in certain situations a voice-controlled user interface is not enough to specify exactly a task to be performed by a robot. Consider for example a doctor that have to operate a person through a telerobotic system. It should be necessary to specify the different tasks in a more accurate manner. This introduces the convenience to have a mixture of both approaches, the natural language and controls that allow the specification of a task with exactitude.

1 INTRODUCTION

In this paper we present a real application whose goal consists of the design and implementation of a telerobotic system that includes learning capabilities and aims at controlling a robot by means of a subset of the natural language. Besides this, the user interaction is complemented with a set of graphical controls that can be manipulated through a mouse interaction and allow the user to specify a task precisely.

The system, which provides control over a vision-guided robot called "Jaume" (see fig. 1), follows an Object Oriented Distributed Arquitecture by means of the CORBA standard [1]. This allows the system to interconnect multiple modules running on different platforms and implemented using distinct programming languages (C++, Java, etc). Besides this, the user interface is defined by using the Java programming language, allowing it to be run on multiple platforms or event better, over the Internet.

Figure 1. The robot "Jaume".

[1] Department of Computer Science, Jaume-I University, E-12071. Castellón, SPAIN. E-mail:{rmarin,sanzp,pobil}@inf.uji.es

2 RELATED WORK

Many different telerobotic systems have been reported since Goertz introduced the first teleoperator at the Argonne National Laboratory four decades ago [2]. Nowadays, the expansion of the World Wide Web has allowed a increasing number of user interfaces that control remote devices not only for robots, but also cameras, coffee pots, and cake machines, to name a few.

The first telerobotic systems with this kind of interface were presented by researchers from the University of Southern California (USC) and the University of Western Australia (UWA). The Mercury Project [3], carried out at the USC, led to the development of a system in which the manipulator was a robotic arm equipped with a camera and a compressed air jet, and the interface consisted of web page that could be accessed using any standard browser.

In our system, we aim to allow operators to use their voice and their mouse input to control the robot movements. Besides this, the system is able to learn from experience and the user interaction. Thus, it will accept high-level commands such as "Jaume, pick up the small pen", that require the use of some visual identification to be carried out. By the other side, the recognition algorithm has been selected in order to allow it to be used as a real-time application.

The Robot response must come up in less than a few seconds. In fact, the problem to control a robot via the web will have many additional difficulties because is a very uncertain environment where the velocity of data transmission can not be guaranteed, and the delay is always present. A very recent discussion about this topics in the telerobotics domain can be found in [4], [5], [6], and [7].

However, we have selected the web as the best way of communication between the user interface and the robot because it allows the manipulator to be accessed from any computer connected to the internet. It makes this connection cheap and easy, and we consider these reasons are sufficient to prefer the Internet to a dedicated network that would allow us a much better performance. Besides, as new broadband Internet connections are coming up (fe. ADSL or Satellite) this web bottleneck will fade out too.

3 OVERALL SYSTEM DESCRIPTION

The telerobotic system is based on Computer Vision and the robot called "Jaume", programmed in real time by a remote interface using a subset of natural language specification and/or a mouse input. The system is able to learn from the user by means of its interaction, making then the knowledge base more extensive as the time goes by.

A camera is used to obtain the environment images with the objects the robot can access to. These images are transferred to the computer running the user interface, using Internet as access medium, and then showed to the user in order to allow him to know the real environment state. The restricted natural language input gets into the system via the user interface, which translates in real time these commands into another language that the robot can understand.

Based on a previous work [8] we have designed a Server application that offers low level grasping and camera services. We will refer to this subsystem as "Robot Server". The Robot Server is in charge of setting up the robot and the camera, and controlling them. The Robot Server capabilities are used remotely by the user interface running on the Internet through a distributed object oriented interface implemented with CORBA (Common Object Request Broker Arquitecture) [1]. See fig. 2 to appreciate the different system components.

The natural language command can get into the system by typing it directly into the User Interface or by means of the North-bound interface that allows the system to interconnect a speech recognition tool, such as "Dragon System", through the standard CORBA.

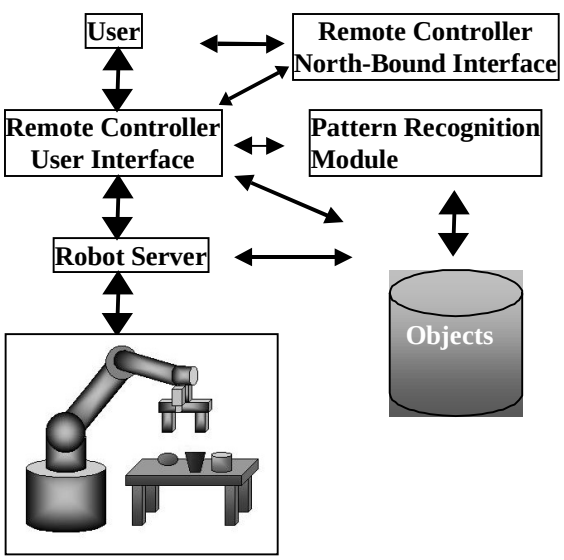

Figure 2. Overall System Architecture.

4 USER INTERFACE

Basically, the user interface consists of a Java application that can be run on any kind of computer with a Java Virtual Machine installed on it. The Java application allows users to obtain images of the robot's workspace and configuration information about the robot and the camera, send commands to the Robot Server for controlling the robot and access the object database in the Database Server to read and update the knowledge robot database.

The user interface, which is shown in fig. 3, allows the use of a subset of natural language for specifying the orders to be sent to the Robot Server. This means users can employ the same names and adjectives they use to refer to objects in the real life, as well as the same constructions for expressing location (above, at the left hand of, etc). Besides this, as we can see at the left side of the "scene window" in fig. 3, the system includes a set of buttons than allows the user to manipulate the objects directly into the scene. At the moment the interaction is based on 2D scenes, but we are starting to use the Java 3D specification, in order to allow the user to interact with 3D scenarios.

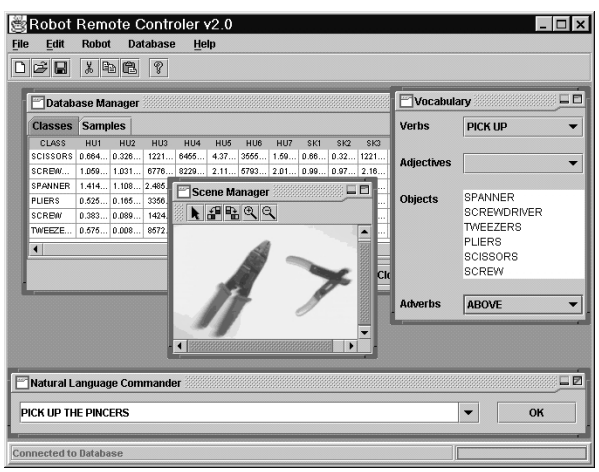

Figure 3. User interface (Remote Controller).

4.1 Integrated object recognition techniques

We have implemented in our system object recognition skills, by using a very fast classifier [9], based on automatic vision processing of the scene, permitting to the user assign labels to a new kind of possible object, if the system do not match the corresponding class for this object in its model data base.

The object recognition and learning tasks are performed by a user interface module, which performs an analysis of the real images received from the Robot Server or provided by the user, and then computes a set of descriptors that identify uniquely the objects contained in the different scenes. These descriptors, known as "HU descriptors", are based on the invariant-moments theory [10]. Once we compute the "HU descriptors" for a given object sample, we apply the learning recognition procedure to know if this object belongs to an already learned class or whether it identifies a category that must be added to the knowledge robot database.

Before the classification process and the HU descriptors extraction of an object the image is preprocessed and segmented in order to identify the objects belonging to the scene. This action is accomplished through a previous binaryzation of the scene and then a simple segmentation algorithm that searches from top to bottom and from left to right the different objects to be treated. The idea is to make it simple enough in order to obtain as much performance as possible.

4.2 Integrated natural language processing

The user interface includes a natural language processing module that translates user commands into commands for the Robot and the database servers. This module has access to the database of known objects, in order to check if the objects users are referring to are or not manageable by the system. When an object cannot be recognized by the module described above, the user will be asked for a name for that object.

The system accepts both voice and keyboard entered commands. A speech processing module is being developed that translates voice commands into a string of text, which, in turn, is translated into a sequence of commands for the Robot and the database servers.

5 CONCLUSIONS AND FUTURE WORK

Future steps in the project will be oriented towards completing and improving the set of services that each component of the system must provide. In the user interface the richness of the language the system can understand must be increased. We also plan to extend the learning capabilities, implementing facilities to define tasks, so that the user can define new tasks as a sequence of other already-known ones.

Finally, by reviewing the last researches about telerobotics, two important challenges are pending. The first one is the design of flexible and autonomous systems able to accomplish a certain amount of high level operation with a minimum of supervision from the operator. These systems are used in controlled environments protected against uncertainties, and they are able to learn high level tasks, a priori, by means of simulation packages. By the other hand, a second line of research appears that is related to the evident problem of using a low-bandwidth medium between the teleoperation user interface and the robot that is placed far away from it. The user is unable to control the robot properly because exists a delay for the feedback to reach the user interface. Obviously this topics could intersect with ours in a near future due to the fact that are very closely related to the challenge of the project here presented.

ACKNOWLEDGEMENTS.
This paper describes research done at the Robotic Intelligence Laboratory, Jaume-I University. Support for this laboratory is provided in part by Generalitat Valenciana under project GV97-TI-05-8, by CICYT under TAP98-0450, and by Fundació Caixa-Castelló under P1B97-06.

REFERENCES.

[1] OMG (1999): The Common Object Request Broker, Aquitecture and Specification (CORBA): http://www.omg.org/corba/cichpter.htm
[2] Goertz, R. and Thompson, R. (1954): Electronically Controlled Manipulator: Nucleonics, 1954.
[3] Goldberg, K. (1995): Desktop Teleoperation via the World Wide Web: Proceedings of the IEEE Conference on Robotics and Automation: Nagoya, Japan, 1995.
[4] Fitzpatrick. Live Remote Control of a Robot via the Internet. IEEE Robotics and Automation Magazine Vol. 6, Nº 3, pp.7-8. 1999.
[5] Baldwin, J., Basu, A., Zhang, H. (1998): Predictive Windows for Delay Compensation in Telepresence Applications: Proceedings of the IEEE Conference on Robotics and Automation: Leuven, Belgium, 1998.
[6] Everett, S.E., Isoda, Y., Dubey, R. V. (1999): Vision-Based End-Effector Alignment Assistance for Teleoperation: Proceedings of the IEEE Conference on Robotics and Automation: Detroit, Michigan, 1999.
[7] Jägersand, M. (1999): Image Based Predictive Display for Tele-Manipulation: Proceedings of the IEEE Conference on Robotics and Automation: Detroit, Michigan, 1999.
[8] Sanz PJ, del Pobil AP, Iñesta JM, Recatalá G. "Vision-Guided Grasping of Unknown Objects for Service Robots". In *Proc. IEEE Intl. Conf. on Robotics and Automation*, 3018-3025, Leuven, Bélgica, 1998.
[9] R Marín, PJ Sanz, AJ Jimeno, & JM Iñesta. Design of a Telerobotic Interface System by using Object Recognition Techniques. Next to appear in Proc. of SSPR'2000, Spain.
[10] Hu, M.K. (1962): Visual Pattern Recognition by Moment Invariants: IRE Transactions on Information Theory, vol. IT-8, pp. 179-187, 1962.