IEEE International Conference on Robotics and Automation, ICRA 98
Leuven, Belgique, May 1998.

A Multi-Loop Robust Navigation Architecture for Mobile Robots*

José Castro[1], Vitor Santos[2], M. Isabel Ribeiro[1]

[1]*Instituto Superior Técnico/Instituto de Sistemas e Robótica – Av.Rovisco Pais 1, P 1096 Lisboa CODEX, PORTUGAL*
e-mail: {jcastro,mir}@isr.ist.utl.pt
[2]*Universidade de Aveiro, Departamento de Engenharia Mecânica – Campus Universitário, 3810 Aveiro, PORTUGAL*
e-mail: vitor@ua.pt

Abstract

This paper describes a multi-loop, modular navigation architecture for mobile robots whose structure allows the execution of most types of navigation tasks in a highly robust manner. The modularity allows the clear separation of functions according to their complexity and priority. We propose a clear separation between local and global navigation in such a way that both can run independently, but an adequate alternation/competition between them allows accomplishment of trajectory execution including avoidance of unknown obstacles. The multi-loop nature of the architecture ensures adequate stability at different levels yielding safe navigation and accomplishment of higher level tasks. Examples may range from goal reaching in some point of the environment to a 3D-environment mapping application, as this is the case in this work.

1. Introduction

Navigation is a major issue when addressing mobile robotics because the concept is so wide that it includes all aspects of directing a robot's course as it traverses the environment. These issues include path planning, path execution, obstacle avoidance and localisation, just to mention those more frequently addressed by researchers in the area. Navigation is the intermediate level (or sometimes the final one, depending on the ultimate goal) to accomplish task oriented jobs in the mobile robotics domain. The navigation architecture represents the robot internal organisation and the interactions among all the actions that are to be executed to accomplish a task.

In most of the navigation tasks, a place in space is to be reached, that is, the user or another application external to the robot must indicate some type of co-ordinates or references for the robot to reach and/or detect. Another type of navigation may be the roving or wandering mode where not a single co-ordinate or landmark is ever given to the robot. This type of navigation will be mentioned further as a by-product of the proposed architecture. In both cases, obstacle detection and avoidance capabilities are assumed to be executed.

The navigation architecture proposed in this paper is implemented on the AEST (Autonomous Environment Sensor for Telepresence) mobile platform displayed in Figure 1, which was developed in the framework of the RESOLV project. The project aims at the 3D reconstruction of large and complex indoor environments based on range [1] and video data. The transportation of the sensor head (Laser Range Scanner plus Vision Camera) between the successive acquisition positions is done according to a perception plan that defines the next goal to be reached by the AEST [2]. Within a universe where the environment is unknown until it is reconstructed and where autonomous motion is required both in reconstructed and yet unmodelled regions, a robust navigation architecture for the AEST is imperative.

Figure 1 - AEST mobile platform

The proposed architecture is based on previous developments presented in [3] and [4] where obstacle detection and avoidance were implemented in a wandering based approach combined with a reflexive loop that handled emergencies. In the context of the RESOLV project, a final goal has to be reached which requires the execution of a designed trajectory, the avoidance of eventual unknown or unexpected obstacles and the recovery or, should it be necessary, the recalculation of the path to accomplish the final goal. The proposed architecture, composed by three enclosed loops implementing reflexive, reactive and functional procedures, is robust and modular achieving correct

* This work was supported by the project RESOLV-Reconstruction using Scanned Laser and Video of the ACTS Programme, EU.

motion in dynamic environments. The major novelty is the competition of different navigation modes (sets of motion strategies) according to the robot status and task under execution.

2. Proposed architecture

The most well known navigation architectures include the subsumption variant introduced by Rodney Brooks [5] which is characterised by the clear intention of separating priorities, and creating hierarchies of blocks whose actions would inhibit (subsume) other blocks in a lower level according to the system inputs. This inhibition would transfer the responsibility of defining actions to "more intelligent" modules if one of them "decided" to do so. The structure is based in what Brooks called the levels: level zero corresponds to the simplest behaviour (avoid touching obstacles) with successive layers growing vertically on top of this one and having increasing complexity.

The subsumption is a reactive type architecture meaning that actions depend on inputs after some processing upon them, where the processing could be very simple in the lowest level and highly sophisticated in higher levels. A related variant is the purely **reflexive** architecture, where actions are taken after inputs with no further processing, meaning that no other condition than the input value itself is taken into account for the decision. However, this may create stability problems.

Without being exhaustive, most navigation architectures can be divided into two main categories: the **behavioural**, such as the subsumption described before, and the **functional**. The functional type is opposed to behavioural in the sense that no action is taken without the proper analysis of its consequences and/or interactions. Actions tend to be very optimised and the structure of such architectures is fairly complex allowing eventually that communication among its parts is fully bi-directional. They are usually more algorithmic and less reactive. A third type, the hybrid, tries to fuse the pros of both categories to end up with architectures with larger capabilities. Further analysis and comments on these variants can be found in [6].

The envisaged application in the RESOLV project requires an intensive relation with the environment, which could be dynamic, therefore demanding a great deal of reactivity. A navigation architecture for a robot as complex as the one involved in our work cannot be purely reflexive because that would certainly lead to instabilities due to the large number of sensors (24 ultrasonic for navigation purposes) and to the poor reliability of individual measurements. Therefore, a reactive structure, though appearing somehow limited, was a suited point to start defining an alternative. In the other hand, as some navigation actions were to be more elaborate than simple reactive behaviours, the functional component had certainly to be inserted in the architecture. Stated this, it is clear that the final architecture would possibly be of the hybrid type.

However, independently of remaining attached to the **reactive** and **functional** concepts, the main design requirements of the architecture were:

- separation of navigation actions according to some functional independence;
- responsiveness of system in case of high priority events, such as imminent collision but yet, efficiency to avoid entering the previous conditions of emergency;
- stress the importance of local navigation as the closest active contact with the environment;
- count both on local and global information provided by sensors and a priori knowledge, respectively;
- robustness to sensor failure, or irregularity and resolution;
- architecture and architecture modules must be easily modifiable and expandable.

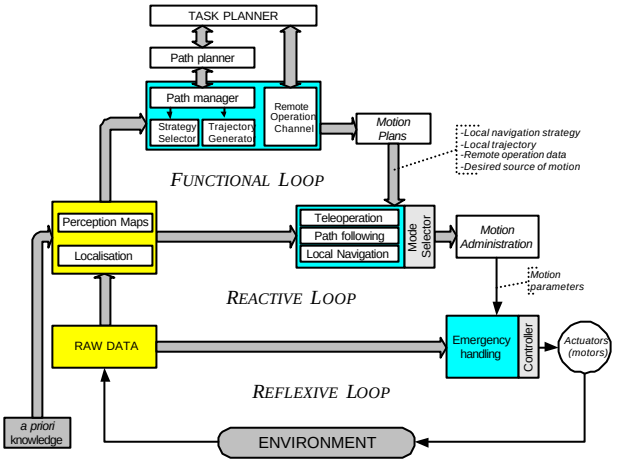

Figure 2 - Proposed Navigation Architecture

The proposed architecture is represented in Figure 2. Earlier developments towards such type of architecture [3] [4] [7] did not implement some of the modules now presented and some were even not so detailed. This architecture may be described as a set of enclosing loops. A loop is a closed circuit of information going from the perception of the status of the environment and/or the status of the robot to actions. The three executive loops are the **reflexive loop** that handles collision avoidance/emergencies, the **reactive loop** responsible for local motion/path following and the **functional loop** that is in charge of local motion strategy and trajectory planning.

2.1. Reflexive Loop

The reflexive loop deals with imminent collision detection, loss of communications with a remote host (if one is used), freshness of sensorial data, and very simple motion commands to evade traps or dead locks. Raw sensorial data is available within this loop.

2.2. Reactive Loop

The reactive loop deals with local motion and path following issues. Actions to be taken after data acquisition are more elaborate than simply reflexive and sensorial data can be more than simply raw. Processed, integrated or fused data is obtained at this level. Also at this level, eventual external *a priori* knowledge (processed data, e.g., a full or partial model of the environment) can be supplied to the system.

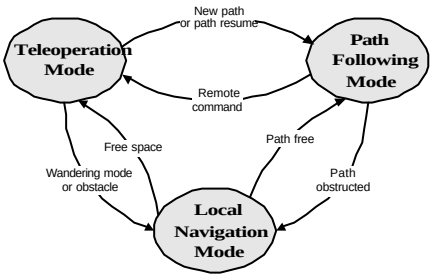

Figure 3 - Transitions of navigation modes

The main issue is to perform safe motion in the environment whatever it may be. Three navigation modes, **teleoperation**, **path following** and **local navigation** are implemented with motion achieved by alternation/competition among them. Each of these modes is permanently available and assured by specific modules. The modules attached to each navigation mode (essentially path follower and local navigation) continuously calculate what would be, from their point of view and with their inputs, the correct motion. Another module, the mode selector acts as a referee by defining which is to be the "winning" motion according to some rules. Teleoperation, by definition, always wins unless path appears obstructed, or other safety concerns are to be taken into account. In the path following mode, robot performs motion along specified curves in order to cover sequentially a list of subgoals. Local navigation wins when the path being executed risks the short term collision with obstacles, or when wandering or similar modes are desired (by the user, for example). Normally, the local navigation module actuates when the projected path is non-feasible due to obstacles. In this working mode the obstacle is avoided (contoured or deviated from, depending on the current local navigation strategy) and as soon as the locally desired path is free, the system returns to path following mode. Figure 3 illustrates the basic navigation modes and the main transitions.

2.3. Functional Loop

The functional loop is responsible for evaluating the path to next subgoal and which strategy is suited for local navigation, should it become active. It still verifies whether remote operator (user/application) is demanding direct control on the robot. Whenever the path following mode is the desired one, the competition among modes occurring at the reactive loop may raise a path-recovery problem, i.e., the definition of a new path to reach the final goal. This problem is not directly solved by the competition among navigation modes given that the reactive nature of the modules involved determines conflicting behaviours competing for the control of the system and path recovering requires co-operation. So, it is only at the functional level that the problem is solved, by computing in real time appropriate motion plans to influence (not decide) the behaviour of the reactive loop.

Among the functions assured by this functional loop there is still the verification whether subgoals have been reached, and the manipulation and rearranging of their sequence in order to fulfil or try to optimise the navigation task. It is worth noticing that in the functional loop, more processing, decision and analysis is performed rather than action upon the real system. This is a characteristic of functional architectures as mentioned earlier.

Expanding this architecture to an outer loop that encloses all these three is not difficult. One could imagine a loop including the user or an application responsible for the task planning issues. That would include path planning or teleoperation or other task planning concerns. Sophisticated perception of the environment could be required, and actions would be of highest level.

2.4. Major novelties

The proposed architecture presents some novelties relative to known solutions. When compared to Brooks' subsumption architecture, behaviours are not predefined but rather built and modelled along a data path. This allows the creation of many, eventually unexpected behaviours. Moreover, the inhibition (subsumption) of actions is not done downwards but upwards: lowest level (and fastest) units of the architecture have priority upon highest levels.

Robustness is achieved with simplicity: when executing a planned trajectory, the path follower monitors it constantly and ensures it is being accomplished within given limits. However, if path faces toward occupied space, local motion is allowed to interfere and deviate or circumscribe the obstacle. If local motion fails for some reason and approximates beyond

safety, the emergency handling actuates imposing speed reduction, or, most of the times, definitely stopping the vehicle. Though likely to occur rarely, further procedures will be carried in case of dead locks, such as "U" shaped obstacles. On the other hand, well-planned tasks are completed successfully without the intervention of lower levels.

The modularity is an important feature of this architecture as well; modules can run independently of the others or even on separate hardware. Another interesting, and somehow novel, concept emerging naturally from this architecture is the so called "**Assisted Navigation**" [4][7]. This means that the user may control the robot directly (joystick, mouse, etc.), but when it approaches more difficult situations, like traversing a door, the automatic activation of the local navigation will assure that part of motion.

3. Main blocks of the architecture

In terms of computational and algorithmic complexity, some blocks of Figure 2 are much relevant, namely: path following, local navigation, perception maps, localisation, emergency handling. A brief description of the principles underlying some of them is presented as follows.

3.1. Emergency handling

This is a key block on the reflexive loop of the architecture, managing the lowest level of procedures. Its actions concern essentially on the detection of **imminent collision** and brake actuation in case the velocity (intensity and direction) is considered unsafe for the free space perceived by one or more sensors. Collision, in the sense of emergency situation, is extended to several cases: obstacles in the path, holes in the ground or hanging obstacles above a certain height. The sensorial data is not processed at all, thus revealing the reflexive component. In the current implementation, an emergency procedure assures some low level motion to escape traps or dead locks as the case of "U" obstacles. The system detects such situations by two means: the frequency of imminent collision detection and also detecting what was called **repetitive stopped rotation**. This last occurs when the robot tries to execute pure rotations to the left and to the right and enters a periodic cycle. In these situations the navigation mode is most certainly the local navigation, which, having simple behaviours to react to free space, may eventually yield that the robot fall trapped in local minima. In any of those situations, the emergency handling block will provide very simple slow motion forwards or backwards in a straight or equally simple line, should sensorial data allows. This temporarily overrides all other motion commands (except a remote "stop" command).

3.2. Local Navigation

The local navigation is a block on the reactive loop, and also the name of a navigation mode. Local navigation provides motion to allow the robot to move with no given references. It exclusively uses the perception of the environment plus some motion behaviour: the **navigation strategy**. The use of raw data from 24 ultrasonic sensors to generate local motion is not suited due to the poor stability caused by erroneous and unstable sensorial measurements. Therefore, data must be processed in order to obtain a more robust and solid representation of space occupancy around the robot. This is achieved by the perception maps, [3] [4], which results from data integration and are organised as a radial robot-centred grid as displayed in Figure 4.

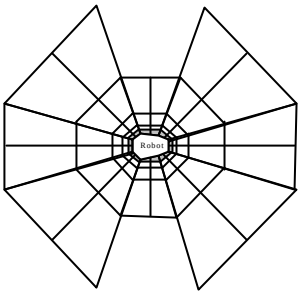

Figure 4 - Perception Map structure

The perception maps are built using sonar data, but can be refined with complementary data (such as laser range or infrared). Among the developed methods to build the map, the most robust found was one based on Neural Networks [3] [7]. With these maps, and obeying some simple strategy, such as "follow the free space", "follow the environment on the left/right", or "contour obstacles", motion is then generated using a dedicated algorithm [7].

3.3. Navigation in the global frame

Some modules are concerned with the issues of navigation in the global (world) frame, namely the **path follower**. This is achieved by sequentially tracking the list of subgoals generated by the path planner, ended with the desired final goal. Each subgoal, $\mathbf{g}_i$, consists of a vector $\mathbf{p}_i = (x_i, y_i, \theta_i)^T$ defining its pose in the global frame, and a flag with three possible status: *stopped* (goal to reach at zero velocity), *forward* or *reverse* (goal to reach with the correct heading). The list of subgoals constrains the path to be executed, but leaves freedom on the navigation between them. The **trajectory generator** module computes, on-line, a smooth trajectory between the robot's current location and the next subgoal. The type of curves used are clothoid pairs [8], but cubic

spirals [9], splines, or other smooth curves could be used, all of them aiming at smoothing motion to reduce the odometry errors produced by undesirable wheel slippage. The referred smooth trajectory is defined by a sequence of vectors $\mathbf{p}_k$ defined in the global frame and an associated curvature, c_k, and velocity, v_k. Whenever the robot deviates more than a given amount from this trajectory a new one is computed from the actual location to the current subgoal.

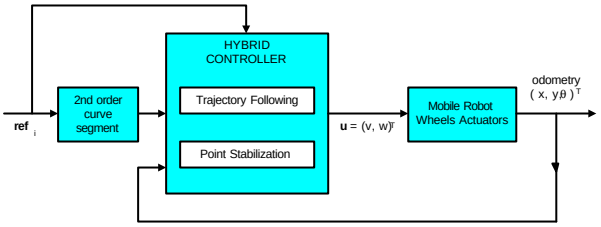

Figure 5 - Path following control block

The **path follower** evaluates the set ($\mathbf{p}_{k-1}$, $\mathbf{p}_k$), in real time, with the smaller distance to the robot's actual location based on an ellipse inclusion criteria. Then, it dispatches $\mathbf{ref}_k = (\mathbf{p}_k^T\ c_k\ v_k)^T$ as the local reference for the control block in Figure 5, where feedback is provided by odometry. The odometric information is always assumed as the most accurate estimate for absolute location, given that it is corrected by an external localisation system. The vector $\mathbf{ref}_k$ locally defines a generic second order curve segment that the hybrid controller takes as reference to implement a trajectory following control law. Due to the non-holonomic nature of the system, the control law will always have singularities in the reference vector space when $v_k=0$. In this case, $\mathbf{ref}_k$ defines a single static posture in the system's state space and the control problem to consider is point stabilisation, instead of trajectory following. This requires implementation of hybrid control laws, [10] [11]. The hybrid controller computes a motion ready to be dispatched to the wheel actuators. This motion is only affected by odometry and does not take into account any other sensor information.

The **path manager** module checks odometry regularly and when the current subgoal is reached, takes the next subgoal as the current one to attract motion to. A path is considered as having been executed when the final goal is reached. Whenever the current sub-goal is blocked by an unexpected obstacle (which is checked out through the combined information of the perception maps and odometry), the **path manager** considers and computes an alternative subgoal.

3.4. Navigation mode selection

Through the interface with the **remote operation channel** the operator/user application chooses the desired navigation mode which, however, might not be always active. In fact, the selection of the active mode is performed by the **mode selector** that takes into account the desired navigation mode and the motions proposed by the modules competing on the reactive loop. Considering the robot dimensions and the motion proposed by the module corresponding to the desired navigation mode, the **mode selector** estimates, within certain tolerances, the spanned area needed to perform the desired motion. The occupancy status of this area is checked by analysing the perception maps. If it is occupied, local navigation is selected. Otherwise, the navigation mode will be the desired one.

3.5. Localisation

In a non-periodically basis, an absolute localisation system based on the detection of natural landmarks of the environment sets odometry system to correct its cumulative errors. Details of the implemented algorithm are in [12].

3.6. Other modules

Given the robot actual location, the already reconstructed model of the environment and the location of the next laser data acquisition (the desired goal), the **path planner** evaluates a set of subgoals. A simple optimisation algorithm defines a set of line segments to the goal, keeping a safe distance from already modelled environment and avoiding, whenever possible, motion on unmodelled regions. Subgoals are taken from the intersection points of consecutive segments. In the framework of the RESOLV project no sophisticated optimisation is required at this level since partially modelled environments are to be dealt with most of the times. Instead, subgoals and trajectories may be dynamically recalculated at different modules and during task execution, as already referred.

4. Results

Extensive results on local navigation can be found in [3] [4] [7]. The new results obtained concern the interaction of the remainder blocks, namely the path follower. Figure 6 illustrates situations of path recovery performed with the mobile robot. A sequence of subgoals, sg_i (position and orientation), is given as a path. Superimposed are the results of two experiments: one, represented by a dashed line, corresponds to the execution of a path exclusively in the path following mode. On the second one, the robot is teleoperated during two time intervals. When teleoperation is no longer active, a new path is evaluated to regain the original one. Yet, one intermediate subgoal is skipped by the path manager.

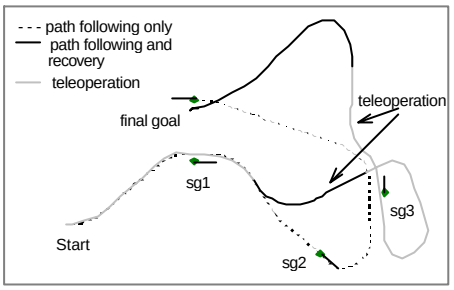

Figure 6 - Path recovery examples

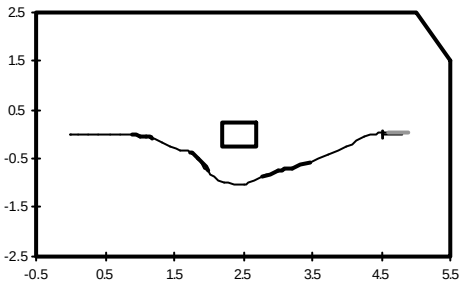

Figure 7 - Avoidance of unknown obstacle and path recovery (scales in meters).

Given a goal straight ahead with an unknown obstacle in between, the robot deviates from the obstacle and resumes its path towards the goal as illustrated in Figure 7. In this figure, the path is made up of a succession of local navigation (thick lines) and path following (thin) tracks. As the robot initially approached the obstacle, the local navigation deviated the robot's path slightly away from it. The path follower took control and continued executing a newly planned smooth trajectory. Local navigation took control two more times whenever path following tried to approach the robot to the obstacle edges. When the path cleared (seen through the perception maps), path following mode controlled the robot towards the goal, shown as a dot (plus heading) in the figure.

5. Conclusions

This paper presents a navigation architecture with great robustness at several levels: safety against collisions, completion of navigation tasks, possibility of distributed processing, dealing with dynamic environments, adequate motion generation, modularity and expandability. The system is highly responsive to emergencies. This is done at the reflexive loop level, which has the highest priority. Navigation tasks are assured by the several loops: the more complex the task the more frequent will be the interactions among the loops and the transitions of navigation modes. However, means exist to fulfil even the more demanding motion related tasks, namely going from one point to another with limited sensorial information and with unexpected obstacles. The architecture can also integrate user intervention as a normal operation, without breaking the normal flow of execution.

The dynamics of the environment has no limits, assuming that moving parts deliberately do not try to collide with the robot in any direction. The way motion is generated is very robust since several motion generators may exist in competition and the "best" motion is selected. Higher levels propose motion plans, but the way these are carried out is normally out of their direct control. If no unexpected situation occurs, the plans are accomplished as expected. Otherwise, the lower level loops will have to intervene, imposing their own proposed motions or motion plans.

Future work includes the thorough testing of the architecture in several types of environments and with different degrees of 3D environment reconstruction completed during previous navigation tasks.

References

[1] V. Sequeira–Active Range Sensing for Three Dimensional Environment Reconstruction, *Ph.D. Thesis*, Instituto Superior Técnico, Technical University of Lisbon, December 1996.

[2] V. Sequeira, J. G. M. Gonçalves, M.I.Ribeiro – Active View Selection for Efficient 3D Scene Reconstruction, *Proc. of the 13th Int. Conf. on Pattern Recognition*, Vienna, Austria, August 1996.

[3] V. Santos, J. Gonçalves, F. Vaz – Perception Maps for the Local Navigation of a Mobile Robot: a Neural Network Approach, *IEEE Int. Conf. on R&A*, San Diego, USA, pp. 2193–2198, May 1994.

[4] V. Santos, J. Gonçalves, F. Vaz – Local Perception Maps for Autonomous Robot Navigation, *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp.821–827, Osaka, Nov 1996.

[5] R. A. Brooks–A Robust Layered Control System for a Mobile Robot, *IEEE J.of Robotics and Automation*, vol.2, n.1, Mar 1986, pp.14–23.

[6] B. A . Langland, O. L. Jansky, J. S. Byrd, R. O. Pettus – The Integration of Dissimilar Control Architectures for Mobile Robot Applications, *J. Robotic Systems*, Vol.14, n.4, April 1997, pp. 251–262.

[7] V. Santos – Autonomous Robot Navigation: Sensorial Data Interpretation and Local Navigation, *PhD Thesis*, University of Aveiro, Portugal, 1995 (Portuguese).

[8] Y. Kanayama, N. Miyake – Trajectory Generation for Mobile Robots, *Robotics Research, The MIT press*, vol. 3, pp.333–340, 1986.

[9] Y. Kanayama, B. I. Hartman – Smooth Local Path Planning for Autonomous Vehicles, *Proc. IEEE Int. Conf. on Robotics and Automation*, pp.1265–1270, 1989.

[10] C. Canudas de Wit, H. Khennouf, C. Samson, O J. Sordalen – Nonlinear Control Design for Mobile Robots, *in Recent Trends in Mobile Robots, World Scientific Series in Robotics and Automated Systems*, Vol. 11, Y. F. Zheng (ed.), Singapore, 1993.

[11] B. d'Andréa–Novel, G. Campion, G. Bastin – Control of Nonholonomic Wheeled Mobile Robots by State Feedback Linearization, *The Int. Journal of Robotics Research*, Vol.14, no.6, pp.543–559, Dec. 1995.

[12] J. Gomes-Mota, M. I. Ribeiro – Localisation of a Mobile Robot using a Laser Scanner on Reconstructed 3D Models, submitted *to 3rd Portuguese Conference on Automatic Control*, pp.186-195, Coimbra, Portugal , September 1998.