Theorical Methods for Planning and Control of a Mobile Robotics

Henrik I. Christensen
Centre for Autonomous Systems
Royal Institute of Technology
S-100 44 Stockholm
Sweden
hic@nada.kth.se

Paolo Pirjanian
Lab. of Image Analysis
Aalborg University
DK-9220 Aalborg Ø
Denmark
paolo@vision.auc.dk

Abstract

Through adaptation of theoretical techniques from control theory and computer science it becomes possible to provide well posed expert systems for control of mobile robots. In contrast to other expert systems for mobile robotics the adapted framework enables structured design and verification of implemented systems. This in turn results in a modular system, and a system with a predictable performance. In this paper it is described how a behaviour based system can be controlled by a rule based expert system, where the interaction between behaviours is described in terms of a process algebra, where different processes are modelled as Discrete Event Systems, and compositions are controlled by a supervisory control structure. To verify the described framework experimental results obtained with a prototype system are also outlined.

1 Introduction

Mobile Robotics is an interesting area for use of artificial intelligence (AI), as it is a domain in which large bodies of knowledge is needed to enable tasks like 'intelligent' navigation in a large/complex facility while there at the same time is a need for real-time response to external events. Mobile robotics can thus be used as an interesting test-bed defining a proof-of-concept scenario for empirical verification of AI techniques and investigation of their applicability and feasibility in real-world settings

In many experimental mobile robot systems and almost all commercial systems the use of AI techniques has been abandoned. There are multiple reasons for this. First of all there is a lack of tools and methods for analysis, design, synthesis and finally verification of the performance of a final systems. This is in particular critical for commercial systems. One of the most intellectually challenging problems in this field is thus to devise theoretical frameworks that can help in construction and verification of robot systems with a specified performance and behavior. In addition traditional AI methods have been abandoned due to frequent use of fundamental assumptions that do not hold in real-world scenarios. These assumption include: complete knowledge about the environment, predictable surroundings, perfect sensing, perfect actuation and finally unlimited resources. There is nonetheless a place for expert system technology in mobile robotics. To accommodate use of AI techniques it is, however, necessary that the methods which are developed adhere to two principles:

1. A strict separation between methodological and domain information
2. Use of formal methods for derivation of the methodological component

In many systems that have been developed in particular the first principle is not adhered to in any strong sense. Many might claim that the system they have developed do have a separation between methodological and domain information, but upon closer inspection of the final system, it is evident that the inference engine or the rule base is explicitly tailored for the application at hand. Thus, it is virtually impossible to transfer the same 'software' for use on another platform, and adaptation to a new domain will often result in an 'unexpected' (and usually undesired) behaviour.

In this paper it is described how careful design of the robot architecture and the corresponding deliberative components enable construction of mobile robots that can be used for in-door navigation in cluttered environments. The two above mentioned principles are followed in the design of the example system.

The paper is organised as follows. In section 2 the system architecture is described and it is out-

lined how the low-level components, that facilitate safe navigation, have been designed. In section 3 it is described how the basic components for deliberative task achievement have been designed to accommodate use of formal methods. Section 4 then outlines the associated planning component that is needed for mission planning, mission monitoring and error recovery. Finally sections 5 and 6 provide a few experimental results and a discussion of the resulting system.

2 Robot architecture

In dynamic and changing environments, the mobile robot can usually not assume complete and consistent environmental knowledge, which is necessary to plan *detailed* courses of action ahead of time. Having incomplete knowledge, the robot can only synthesise a *coarse* plan ahead of time and it must then fill in the gaps at the time of execution, when local information becomes available. In this way the robot's goals as well as the current state of the world determine the criteria for making the decision of what action to do next. Apart from long-term planning capabilities, the robot must thus also include facilities that enable sensible reactions to previously unanticipated events. This implies that the robot must be equipped with a suite of sensors that enable real-time modeling of the environment in which it operates. This modeling is typically purposive, in the sense that there is no need for a full reconstruction of the environment, as most structures in the environment can be treated as obstacles and there is only a need for identification of a few selected objects (target structures like a particular machine, or the re-charging station)[9]. To accommodate both achievement of user specified tasks and handling of unexpected events it is convenient to organise the system in a three layered architecture as shown in figure 1. This type of system architecture is denoted a *hybrid* architecture in the literature and is the common way to reconcile AI planning with reactive modules.

The bottom layer (reactive behaviours) is responsible for pre-attentive handling of unexpected events. It includes basic behaviours for local obstacle avoidance and detection of collisions. The behaviors consist of reactive sensory-motoric processes, which operate on raw data and thus enable real-time reactions to run-time contingencies. These behaviours can be activated and deactivated, but are otherwise autonomous and may override commands for higher level behaviours, since this layer is responsible for the actual control of the robot.

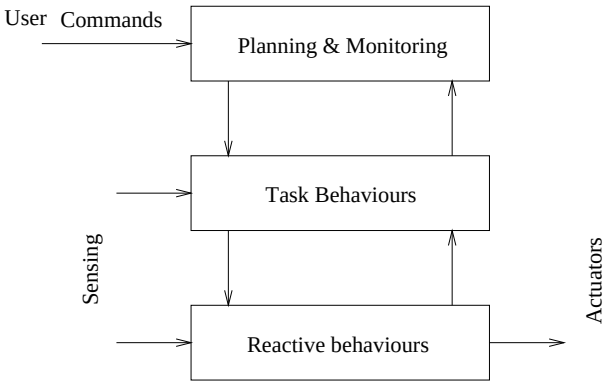

Figure 1 Overall three layer robot architecture

The middle layer 'task behaviours' includes a set of processes that are responsible for achievement of subtasks like 'exploration', 'wall following', 'door traversal', 'door localisation', 'target servoing', etc. These behaviours are invoked with specific parameters to enable achievement of individual parts of a mission. This layer bridges the gap between the top planning layer and the bottom reactive layer.

Finally the top layer is responsible for planning of missions. I.e. based on a user specified goal (the mission statement), the planner transforms the specification into a sequence of actions/tasks to be accomplished. The mission is here specified in terms of a sequence of 'task behaviours' to be executed. From the 'task layer' the planner receives 'success'/'failure' feedback that indicates the completion status for each activated behaviour. If a behaviour has been successful the next step in the sequence is activated. In the event of failure a re-planning is initiated to determine alternative actions that might lead to a successful mission completion.

Some might argue that the best suited architecture for a robot is in terms of the traditional 'sense-plan-act' structure, where the responsibility of the components are well defined. Unfortunately such a breakdown of the system often leads to inefficient systems as all components must be able to operate in real-time and it is not obvious how such systems will scale to complex applications. By choosing an alternative breakdown, where each behaviour has a direct mapping from perception to action it becomes possible to separate the system into reactive/time critical processes and any time processes (like planning). The idea of a behavioural breakdown was initially proposed by Brooks [1], but while he maintains that all actions/tasks can be formulated as behaviours that are organised in a simple hierarchy we maintain that there is a need for traditional planning and dynamic

sequencing of behaviours to enable flexible operation and efficient use of system resources.

Given the architecture in figure 1 the problem is then how each behaviour can be described in a formal framework to enable verification of a robot's activities.

3 Behaviours

In our present system the following set of deliberative behaviours is used:

GoTo Go to a pre-specified location (X, Y, θ). Where θ is the orientation of the platform at the position (X, Y).

Door localise Initiates a vision process that searches for a doorway (to be used for movement into another room).

Door traversal This process traverses a door way, through integrated use of ultra-sonic sonars and visual information. Initially the visual information controls the movement of the robot to bring it to a position that is right in front of the doorway. Using sonars the platforms then moves through the doorway. It should be noted that the robot during the latter part of the 'task' relies entirely on odometry as neither the sonar or the visual system is able to operate at very short range. In practice this does not pose a problem as the robot during this part of the mission is positions very close to the actual door opening.

Wall follow This behaviour makes the robot move at a constant distance from a wall. The behaviour relies on ultra-sonic sonar feedback for the control of the platform.

Hallway navigation For navigation in narrow corridors (where the floor has a homogeneous coloring) a visual behaviour is also available. This behaviour extract the boundary of the hallway and tries to position the robot in the middle of the hallway, using simple projective geometry

Avoid If an obstacle is detected this behaviour uses a local map to guide the robot around the obstacle. The obstacle detection is based on ultra-sonic sonars and/or simple visual processing [8]

To enable description of the interaction between different behaviours each of them are modeled as a finite state machine (FSM), as shown in figure 2 for the 'GoTo' behaviour.

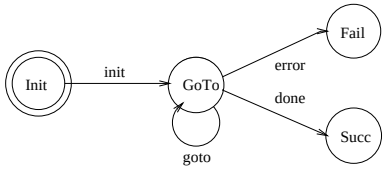

Figure 2 Finite state model of the GoTo behaviour

In this formalism, states correspond to execution of actions/behaviors and events correspond to observations that cause transitions between states.

Each FSM is described as a generator/language using the formalism:

$$G = (\Sigma, E, \Lambda, \sigma_0, \Sigma_m) \tag{1}$$

where

Σ is the set of states

$\mathbf{E}$ is the set of events

Λ is the set of transistions, i.e. $\Sigma \times E \rightarrow \Sigma$

σ_0 is the initial state

Σ_m is the set of marker or terminal states

By adopting the FSM model for each behaviour it becomes possible to describe the combination of behaviours as a discrete event system (DES), as initially suggested by Ramadge and Wonham [10]. This framework was introduced into mobile robotics by Kosecka and Bajcsy [5].

The combined model for a set of behaviours is obtained from the Cartesian product between the elementary models. Through analysis of controllability it is then possible to design an additional FSM, called the *supervisor*, that interacts with and modifies the open-loop behavior of the combined model in order to assure correct behavior. The supervisor achieves its objective by enabling and disabling the events which are *controllable* so that only permisable event sequences are generated by the combined model.

The analysis of combined behaviours can either be carried out off-line (if the possible combinations are known ahead of time) or the combination can be analysed on-line. Unfortunately the analysis of combined behaviours is known to have NP complexity, so for many tasks only the off-line strategy is feasible.

For the description of combination of behaviours several possible methods can be used. We have here chosen to use the process automata model (Robot Schema) suggested by Lyons [7]. In this framework

combinations of behaviours can be described as sequential (;), concurrent (:), conditional (|), disabling (#), and recursive (:;). Through the use of the simple operators it becomes possible to describe complex missions like:

Init; Avoid # (GoTo(2,4,0) ; DoorVerify |
DoorTraverse ; GoTo(10,20,0))

The robot is supposed to move to a position from which it is able to locate a door, through which it will be able to arrive at the goal position. During the entire mission the 'Avoid' behaviour is active to ensure automatic avoidance of encountered obstacles.

Through analysis of this behavioral complex it is possible to specify the necessary control actions, it is further possible to verify how the robot will behave under different circumstances and finally the set of possible error conditions can be enumerated so that this information can be exploited in methods for error recovery.

4 Planning and execution

To enable handling of user defined goals it is necessary to include a planning system into the system. The planning system exploits the following information:

1. Formal models of available behaviours: The DES model for each behaviour, the preconditions required before a behaviour can be initiated, a set of behaviour conditions that remain constant during the execution of a behaviour (e.g it use of resources like sensors), and finally a set of post-conditions that specify the state when the behaviour finishes.

2. An a priori map of the environment. The map can either be a geometric model to be used for reasoning or it can be a simple connectivity model, that simple specifies the topology of the environment.

In our system the basic behaviours enable us to break tasks up into the two categories 'in-room navigation' and 'between room navigation'. Given such a breakdown of tasks it is convenient to exploit a map that encodes the topology of the environment. Consider the layout shown in figure 3.a and the corresponding topology map in 3.b. To plan a mission from one position to another it is necessary to perform a search in the topology map. Based on the results of the search, multiple routes might be possible. The shortest path is then used as the initial plan. The shortest path can then be broken up into parts related to individual rooms. Which implicitly also specifies the set of doorways to be passed during the mission.

The planner consists thus of a rule based system that is able to perform weighted search in general graph structures. The topology map outlined above is then used for the overall planning. The graph is attributed in the sense that each room has an associated description in terms of behaviours for navigation and for each arc the type of passage from one room to the next is specified to enable use of different door traversal behaviours based on the type of the doorway. I.e. for narrow doorways an accurate pre-positioning is necessary, and a more 'sloppy' approach can be used for wide passages.

The resulting sequence of rooms and doorways specifies thus an overall plan for in-door navigation, while the associated attributes specify the behaviours to be invoked in order to execute the task.

If an unexpected situation arises, such as the door is closed (we cannot open doors with the present platform as it is not equipped with any manipulation facilities), the graph is modified in the run-time version to reflect that a particular passage is unavailable. A re-planning from the present position to the goal is then initiated and subsequently executed.

5 Implementation and experiments

The approach outlined above has been implemented on the mobile platform shown in figure 4. The platform is a Robuter 20 robot equipped with 24 ultrasonic sonars, a binocular camera head for image analysis, and a laser systems for use of structured light for obstacle detection and door localisation.

Each of the behaviours outlined above has been implemented as dedicated processes that run under Unix. All of them are interfaced to the knowledge based planner, through socket based communication using the VIPWOB (Vision Programmers Work Bench) system [4]. The planning system is implemented using the CLIPS system (C-Language Interfaces Production System) from NASA [3]. This system has a syntax similar to OPS-5 but as it is implemented in C it is easy to provide interface functions to each of the behaviours.

The rulebase for the CLIPS system contains a generic graph traversal algorithm, an algorithm for analysis of FSM's to generate a supervisory controller, and a small set of rules for communication with behaviours. The topology map of the environ-

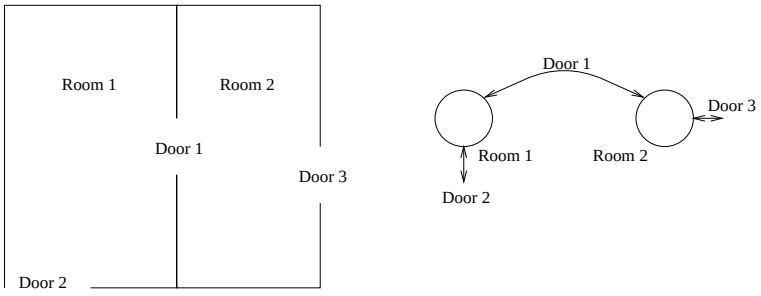

Figure 3 An example layout of the environment and the corresponding topology map.

Figure 4 The ARVID mobile platform

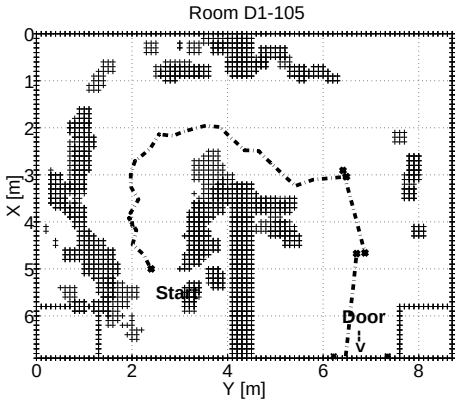

Figure 5 Results from a trial run in a laboratory setting

ment and a description of the 'functionality' of each behaviour is then encoded as declarative knowledge. This implies that it is easy to change the system for application in a different setting and it is also easy to modify the system to exploit new behaviours.

The developed system has been tested extensively in a laboratory setting at the Laboratory of Image Analysis, Aalborg University. A sample results from a run through the lab is shown in figure 3

The robot is here commanded to go from the room D1-105 to the hallway. In order to achieve this the robot must go into the adjacent part of the room and then into the hallway. The 'sharp turns' in the path correspond to places where a different set of tasks/behaviours are invoked. From the illustration above it is obvious that the system is successful in carrying out the mission. The shaded areas correspond to regions with obstacles like tables, chair and computers, which were not in the model provided to the robot. It is thus able to navigate in fairly cluttered environments.

6 Summary

In this paper we set out to demonstrate that expert systems are useful for mobile robot navigation. To enable flexible use of such a technique it is however necessary that the expert system is constructed in such a fashion that it can be subjected to formal verification. To enable this the implemented planner is based on graph search and the execution and monitoring of behaviours is modeled by the use of the Discrete Event Theory. Both components can thus be analysed and verified independently. Further the implementation has been organised in such a fashion that the procedural component only implements basic algorithms while the robot/domain specific knowledge is captured as declarative knowledge.

Using such a strategy it is possible to exploit the inherent flexibility of expert systems while retaining the flexibility necessary for implementation for real systems. In the selection of techniques to use we further decided to exploit a product that has a good run-time performance and a well documented API, as it otherwise is too expensive (in terms of man-power

and computer resources) to exploit these techniques.

One problem that we have encountered is that the rule based paradigm only in a limited fashion enables us to exploit methods for reasoning under uncertainty, an issue that is recurrent in the processing and reasoning about sensory information. Parallel activities have thus explored other paradigms for implementation of our reasoning engine using voting methods [2] and Bayesian Reasoning [6].

In general it is our experience that AI techniques have a tremendous potential for robotics applications, given that a solid theoretical underpinning can be provided.

References

[1] R.A. Brooks. Intelligence without reason. Technical Report 1293, Massachussetts Institute of Technology, April 1991.

[2] J. Faymann, P. Pirjanian, H. I. Christensen, and E. Rivlin. Exploiting redundancy of purposive modules in the context of active vision. Technical Report CIS-9616, Technion, Haifa, Israel, July 1996.

[3] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming*. PWS Publishing Company, 2nd edition, 1993.

[4] N. O. S. Kirkeby and H. I. Christensen. *The Vision Programmers WorkBench*, volume 11 of *Machine Perception and Artificial Intelligence*, chapter 10, pages 195–224. World Scientific Press, Singapore, 1994.

[5] J. Kosecka and R. Bajcsy. Cooperation of visually guided behaviors. In *Proceedings ICCV 93*, Berlin, Germany, May 1993.

[6] S. Kristensen. *Sensor Planning with Bayesian Decision Analysis*. PhD thesis, Laboratory of Image Analysis, Aalborg University, DK, August 1996.

[7] D. M. Lyons. A formal model of computation for sensory-based robotics. *IEEE Transactions of Robotics and Automation*, 5(3):280 – 293, 1989.

[8] H.A. Mallot, H.H. Bulthoff, J.J. Little, and S. Bohrer. Inverse perspective mapping simpilfies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177–185, 1991.

[9] P. Pirjanian and H. I. Christensen. *Modelling and Planning for Sensor Based Intelligent Robot Systems*, volume 21 of *Machine Perception Artificial Intelligence*, chapter Hierarchical Control for Navigation Using Heterogeneous Models, pages 344–361. World Scientific, 1995.

[10] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Contr. Optimization*, 25(1):206–230, 1987.