Published in Proceeding. Intelligent Autonomous Systems (IAS-4) 1995
which should be used for any reference to this work

A BEHAVIOUR-BASED ARCHITECTURE TO CONTROL AN AUTONOMOUS MOBILE ROBOT

F. Tièche, C. Facchinetti and H. Hügli

Institute of Microtechnology, University of Neuchâtel
Rue de Tivoli 28, CH-2003 Neuchâtel, Switzerland,
E-mail: {tieche,facchinetti,hugli}@imt.unine.ch

Abstract. *In this paper, we present the implementation of an autonomous mobile robot controller based on a behavioural architecture. This architecture is composed of three layers: sensori-motor, behavioural, sequencing. The paper describes its general structure and the function of its main elements. It further analyses the development of an example task presenting the advantages of such an architecture.*

Key Words. Mobile Robotics, Behavioural-Based Architecture, State Automaton Control

1. INTRODUCTION

The ability of a mobile robot to achieve reliably tasks in a real environment depends essentially on the architecture of its controller. We use an behaviour-based architecture that combines the advantages of both the behavioural and the centralised architectures. This architecture distributes distinct competence levels in three layers: the top layer implements tasks as a sequence of behaviours thanks to a state automaton, the intermediate layer implements a set of behaviours, and, finally, the bottom layer servoes the robot. Our architecture extends the behavioural approach discussed for example in (Brooks, 1986) to more complex tasks, by offering the possibility to define and execute the goals as sequences of simple behaviours, and by the use of a world model.

In order to demonstrate our architecture, we choose a task where the robot has to tidy up chairs in a room, by pushing and aligning them, using a sequence of simple vision-based behaviours.

The architecture is realised in the form of a development environment called MANO (Mobile Autonomous robot system NOmad200) which main features are (i) the decomposition of the architecture layers and its implementation in a set of concurrent processes, (ii) a blackboard, handling information exchange between elements of the architecture, and (iii) a virtual robot offering the possibility to control either a real robot or a simulated one.

2. RELATED WORK

There exists two basic kind of architectures to control the robot: centralised architectures and behavioural architectures.

The centralised architectures (Albus, 1991) split up the robot control in three modules responsible for: sensing, planning and acting. First, the sensing module builds a high-level representation from sensed data. Then, using this information, the planning module generates the robot actions which are finally executed in the acting module. These architectures are convenient for high-level planning tasks but are not time-efficient and have difficulties to cope with uncertainties and noise issued from the real world.

The behavioural architectures (Brooks, 1986) separate the robot control in several hierarchical layers. Each layer is responsible for a complete processing from sensing to control and interacts directly with the environment. The layers are organised hierarchically: the modules of upper layers activate or deactivate the modules of the underlying layers. The result is good robot-environment interaction with limited possibilities for the description of the tasks. For real applications, it is often difficult to partition a global task in a set of elementary modules because (i) the decision element is distributed over several modules, and (ii) there is no model of the robot's world.

In this late decade, hybrid architectures or behaviour-based architecture have been proposed. They tend to combine the centralised and the be-

havioural approach in order to take advantage of the quality of both. They usually feature a multi-layered hierarchical architecture (Thorpe, 1992). The lowest layers are organised according to the behavioural architecture, providing strong reactivity between the robot and the environment. The topmost layer is responsible for the temporal or spatial organisation of behaviours. It is based on a model of the world and on data provided by behaviours.

A number of methods have been proposed to orchestrate the behavioural activation: Petri nets (Freedman, 1992), rule-based planning (Slack, 1992; Noreils and Prajoux, 1991), programming language (Coste and Espiau, 1992), state automaton (Duan and Kumara, 1993), contingency table (Connell, 1992).

In the architecture presented here, the sequencing of the behaviour activity is performed by a state automaton.

3. ARCHITECTURE

Our architecture is composed of three layers (Fig. 1) : sensori-motor, behavioural, and sequencing. The layer operates asynchronously with respect to each other. The lowest one, called sensori-motor layer, is based on control theory and on signal processing. It is responsible for the elementary movements of the robot and processes data acquired by the sensors. The second is the behavioural layer. It is composed of a set of behaviours that on one hand control the robot with respect to environmental characteristics, and on the other hand manage measures of the world. On top, the sequencing layer implements tasks which are described as sequences of behaviours. We use a state automaton which receives as its input the status vector $\vec{s}$ corresponding to the status of the behaviours, and which activates elementary behaviours, by means of the activation vector $\vec{a}$.

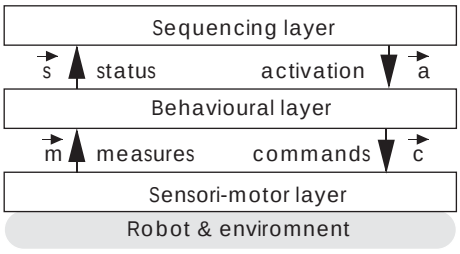

Figure 1 *The architecture is composed of three layers: (i) sensori-motor, (ii) behavioural, and (iii) sequencing.*

3.1 Sensori-motor layer

The sensori-motor layer interacts with the environment by sending command signals to the actuators and receiving signals from the sensors. It is characterised by fast interactions and is mostly hardwired. Typically, the movements of the robot are controlled by servo loops, both for velocity and position. This layer receives commands from the upper-layer in form of a vector $\vec{c}$ and provides processed sensor measures as a vector $\vec{m}$.

3.2 Behavioural layer

This layer (Fig. 2) is composed of a set of N concurrent behaviours b_i performing two main functions. The first function is to control the robot by means of a set of reactive behaviours called *external behaviours*. It is the composition of this set and the variety of the behaviours which define the capability of the robot to interact with its environment. The second function is to manage a database (DB) storing world measures and the parameters of the tasks. Data acquisition and data processing are performed during the execution of a task, by a set of behaviours called by analogy *internal behaviours*.

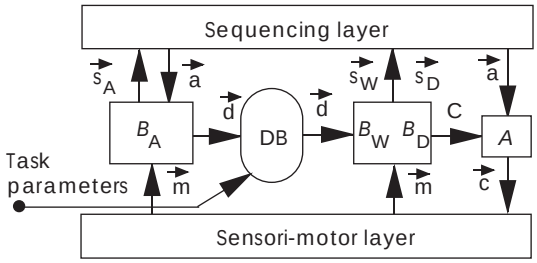

Figure 2 *Interactions of world-driven $\mathcal{B}_W$, data-driven $\mathcal{B}_W$, and internal behaviours $\mathcal{B}_A$ with database DB and acquisition module $\mathcal{A}$.*

External behaviours. The *external behaviours* control the robot either with respect to a real world characteristic (world-driven behaviour), or with respect to information provided by the database (data-driven behaviour). Only one of these behaviours is allowed to control the robot at the same time, so that their output $C = (C_W, C_D) = (\vec{c}_1, \ldots, \vec{c}_M)$ are switched by a selection module $\mathcal{A}$. According to the activation vector $\vec{a} = (a_1, \ldots, a_N)$, the selection module selects the robot command of the active behaviour for controlling the robot.

$$\vec{c} = \mathcal{A}(C, \vec{a}) = C^i \; if \; a_i = activated \qquad (1)$$

The world-driven behaviours are made of the closed loop formed by the world, the sensors, the behaviour module, and the actuators. The module interacts via measures of the world $\vec{m}$ provided by the sensori-motor layer and robot commands $\vec{c}$. Each world-driven behaviour module extracts specific world characteristics, called *sign patterns*, from the measures $\vec{m}$. Each time an expected sign pattern appears, the behaviour is stimulated. It then controls the robot as long as the sign pattern remains present.
Formally, we express a world-driven behaviour module as a vector map.

$$(\vec{s}_W, C_W) = \mathcal{B}_W(\vec{m}) \tag{2}$$

The data-driven behaviours are same except they don't extract a sign pattern from the world measure, but they compute a reference value by means of database information. The data-driven behaviour controls the robot with respect to this reference. When adequate informations are available in the database, the behaviour is *stimulated*. The vector map of the data-driven behaviour modules is defined by:

$$(\vec{s}_D, C_D) = \mathcal{B}_D(\vec{m}, \vec{d}) \tag{3}$$

Each external behaviour provides a signal, called *status* s, which describes its internal stimulation state. This status takes three values : *not stimulated* ($s = 0$), *stimulated* ($s = 1$), *satisfied* ($s = 2$). The presence of a sign pattern (world-driven) or of information in the database (data-driven) stimulates a behaviour. The *satisfied* status is reached when the expected configuration of sign patterns (reference value) appears (world-driven-behaviours) or when the reference value is reached (data-driven behaviour).

Internal behaviours. The internal behaviours are responsible for the acquisition of measures. When activated, the internal behaviour reads a single or a series of measures $\vec{m}$ and stores them in the database $\vec{d}$. These behaviours are always *stimulated* ($s = 1$). Formally:

$$(\vec{s}_A, \vec{d}) = \mathcal{B}_A(\vec{m}, \vec{a}) \tag{4}$$

Activation and status. The status of each kind of behaviours are grouped in a vector $\vec{s} = (\vec{s}_W, \vec{s}_D, \vec{s}_A)$. In the same way each behaviour activation a_i forms a vector $\vec{a} = \{a_1, \ldots, a_N\}$. These two vector are the only information exchange between the behavioural layer and the sequencing layer.

3.3 Sequencing layer

While each behaviour solves a small part of a robot task, the **sequencing layer** composes them to achieve a more complex one. According to a pre-programmed strategy and to the current status of the behaviours $\vec{s}$, this layer activates the suitable behaviours by sending the activation vector $\vec{a}$ to the behavioural layer.

The activation vector is generated thanks to a state automaton we call *Behaviour Activation Automaton* (BAA). It is an extension of both a Finite State Automaton (FSA) and a Moore Machine (MM) (Duan and Kumara, 1993; Harrison, 1965; Hopcroft and Ullman, 1979). This automata extend the FSA formalism by adding output symbols and an output function. The BAA completes also the MM by the introduction of final states. A BAA M is described by a seven-tuple:

$$M = \{Q, q_0, F, \delta, S, \alpha, A\}$$

where:

- $Q = \{q_0, q_1, \ldots, q_n\}$ is a finite set of internal states,
- q_0 is the starting state,
- $F = \{q_f\}$ is a set of final states,
- $S = \{\vec{s^1}, \ldots, \vec{s^k}\}$ is a set of input status vectors,
- $A = \{\vec{a^1}, \ldots, \vec{a^l}\}$ is a set of output activation vectors,
- $\delta : (Q \times S) \rightarrow \wp(Q)$ the set of transition functions.
- $\alpha : Q \rightarrow X$ is an mapping from Q into X.

The BAA input is a series of status vector $\vec{s^k}$ representing the status of all behaviours, and its output is a series of vector $\vec{a}^l$ indicating which behaviour must be activated. The BAA internal states represent the activity of the behaviours level. For each input vector $\vec{s^k}$ correspond a transition from the current state either to itself or to another state. When a state is reached, an activation vector $\vec{a}^l$ is generated. The final states are necessary to indicate that the robot task is achieved.

A graphical representation of the BAA is shown in figure 3 where following notation is used. Circles show the different states q_j with the inner notation indicating the activated behaviours $\{b_i \mid \alpha(q_j) = \vec{a}^j,\ a_i^j \neq 0\}$. The arrows to the same states are ignored and those between two different states (q_i, q_j) are marked with the status which are taken into account. $\{s_i \mid \delta(q_i, \vec{s^k}) = q_j, s_i^k \neq \Phi\}$.

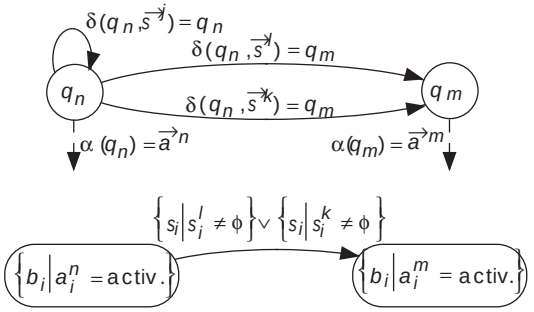

Figure 3 *Element of state automaton BAA.*

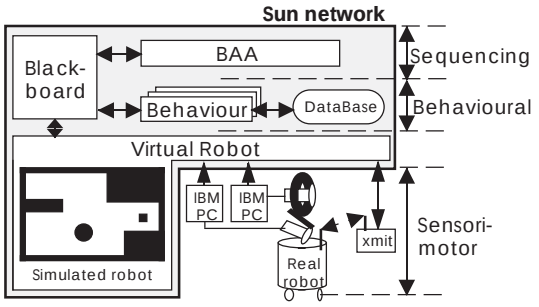

Figure 4 *Development environment MANO.*

The concentric circles are final states: success or failure.

4. DEVELOPMENT ENVIRONMENT MANO

MANO is the development environment (Hügli *et al.*, 1993; ?) for our mobile robot. It implements the principle of our architecture (Fig. 4). The core of this environment is composed of a virtual robot unit and of a blackboard handling the communication between the different layers. The three layers of the architecture are connected to these central elements. The sensori-motor layer is implemented on dedicated hardware located in the robot itself and on additional external units. The two other layers together with the blackboard and the virtual unit are distributed over a network of SUN workstations.

The virtual robot unit links the robot and the blackboard. It offers an interface with equivalent access to both the real and a simulated robot. The transition from real robot to simulated robot is possible at any time by a simple switch. In addition to the simulator, the virtual robot interface provides extended capabilities to monitor the robot, sensor data, commands, position etc.

The blackboard is the communication channel between the virtual robot, the behavioural layer and the sequencing layer. It acts as a server, using a TCP/IP connection protocol. Clients can connect from any point of the network.

4.1 Sensori-motor layer

The robot Nomad 200 — from Nomadic Technologies (Nomadic, 1992)— is a one-meter-tall robot moved by a three wheel synchro-drive motion system; its upper body can be rotate around its vertical axis. It provides sensors of different types: 16 sonars, 16 infrared range-sensors and 20 tactile sensors. The communication between the robot and the virtual robot is established via a serial radio link. The sensori-motor layer is implemented on a number of PC-boards: the servo loops controlling the robot are on board while some vision processing is currently performed remotely

Two active vision-based sensors have been added on top of the robot: a *vision by landmark* sensor and *laser-range* sensor (Hügli *et al.*, 1992). The former uses a light source coupled to a video camera to enhance the contrast of reflecting landmarks distributed in the environment. The bright landmarks are detected, labelled and tracked in a dedicated Transputer system. The latter uses the principle of triangulation to measure the distance of objects in the robot environment. The specific range sensor we use applies triangulation between a plan of light and the line of sight relative to a pixel of the camera. The plane of light of the laser intersects with the environment in a profile line which geometry is finally obtained.

4.2 Behavioural layer

The behavioural modules are implemented as Unix processes and run fully independently. They are client of the blackboard server and read from it (i) the measures $\vec{m}$ provided by the sensori-motor layer and (ii) the activation vector $\vec{a}$ provided by the sequencing layer. The behaviour modules write their robot commands $\vec{c}_i$ and their status s_i on the blackboard.

4.3 Sequencing layer

The sequencing task is implemented in form of a state automaton as a Unix process . It exchanges the activation vector $\vec{a}$ and the status vector $\vec{s}$ with the behavioural layer by communication through the blackboard.

5. APPLICATION: Tidying up chairs

As an example of a task implemented on MANO, we describe here *TidyUpChairs* in a room. It illustrates how a specific task is ported onto our behaviour-based architecture. The robot has to detect chairs located arbitrarily in a room, and to push them up to a virtual line defined with respect to a fixed position of the environment (Fig. 5). This fixed position, called *homing point*, is defined by two landmarks. The virtual line is parallel to the line supporting the two landmarks. This task needs a minimal world representation in form of the homing position and the virtual line position.

Figure 5 shows the *TidyUpChairs* task decomposed in a sequence of simple behaviours. First, the robot performs a wander around behaviour (WA behaviour) until the homing (HO) is stimulated by the two homing landmarks. Then the robot executes the homing (HO). When the homing point is reached, the current position of the robot is stored (GP) in the database for further use. From this point the robot searches chairs by looking around (SC). If a chair is found, it goes towards the selected chair (GC). Then, the robot turns around the chair until it is positioned on the side of the chair opposite to the virtual line (AC) and pushes the chair (PC) until the line is reached. Finally it returns to the homing area (RH) and adjusts its position (HO). The task ends if no more chairs are detected (SC).

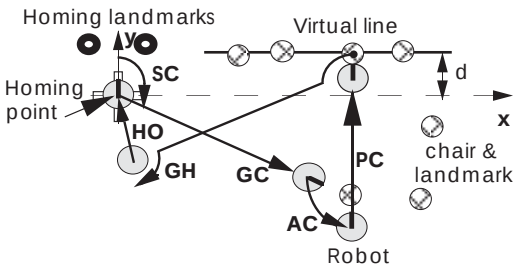

Figure 5 *TidyUpChairs decomposed into behaviours.*

5.1 Sensori-motor layer

Two vision sensors are used: *vision by landmark* detects chairs marked with reflective material and fixed homing landmarks, while *laser-range* sensor detects obstacles in front of the robot. Odometer sensor is used to move the robot to the virtual line and to bring the robot back to the homing area.

5.2 Behavioural layer

The database contains the position of the homing point (X_o, Y_o, θ_o) and the position of the virtual line d with respect to the homing point. The former is acquired during the task even though the latter is a parameter of the task.

The behaviours needed to tidy up chairs are described below:

- Wandering around ($b_{WA} \in \mathcal{B}_W$): this behaviour moves the robot forward and uses the ring of infrared sensor to detect a possible obstacle. If an obstacle is detected, the robot turns away from it and starts moving forward again. Its status is always stimulated ($s_{WA} = 1$).

- Homing ($b_{HO} \in \mathcal{B}_W$): based on vision by landmarks, the homing behaviour (Facchinetti and Hügli, 1993) brings the robot in a fixed configuration with respect to two landmarks. The behaviour is stimulated ($s_{WA} = 1$) as soon as two landmarks are visible and it is satisfied ($s_{WA} = 2$) when the defined configuration of the landmark appears.

- Searching a chair ($b_{SC} \in \mathcal{B}_W$): searching a chairs is a behaviour which is stimulated when a chair landmark is visible ($s_{SC} = 1$). It then turns the robot in the direction of the nearest landmark and stops it ($s_{SC} = 2$).

- Going to a chair ($b_{GC} \in \mathcal{B}_W$): this behaviour moves the robot forward and servoes its move by centring the centermost landmark in the image. It stops when the vertical position of the landmark is below a given threshold ($s_{GC} = 2$).

- Aligning on the chair ($b_{AC} \in \mathcal{B}_D$): based on the orientation of the robot θ_o stored in the database, and on the current orientation θ, this behaviour moves the robots around a chair, until it is oriented perpendicularly to the virtual line.

- Pushing the chair ($b_{PC} \in \mathcal{B}_D$): using the odometry, it move the robot to the virtual line which position is given in the database.

- Returning home ($b_{RH} \in \mathcal{B}_D$): using the current position (X, Y, θ) and the homing position (X_o, Y_o, θ_o), this behaviour brings the robot back to the homing point.

- Obstacle detection($b_{OD} \in \mathcal{B}_W$): using the laser-range sensor, this behaviour detects obstacle in front of the robot.

- Getting position ($b_{GP} \in \mathcal{B}_A$): this internal behaviour stocks the current robot position in the database. It is activated the first time the robot

is on its homing position.

5.3 Sequencing layer

At the sequencing layer, the *TidyUpChairs* task has a pre-programmed structure described by the BAA shown in figure 6. The bold arrows represent the sequence of behaviours [HO-SC-GC-AC-PC-RH] shown in Fig. 5. This cycle is accomplished as long as there are chairs to be tidied up, and no obstacle is detected. The detection of an obstacle leads to others states, indicated by thin arrows.

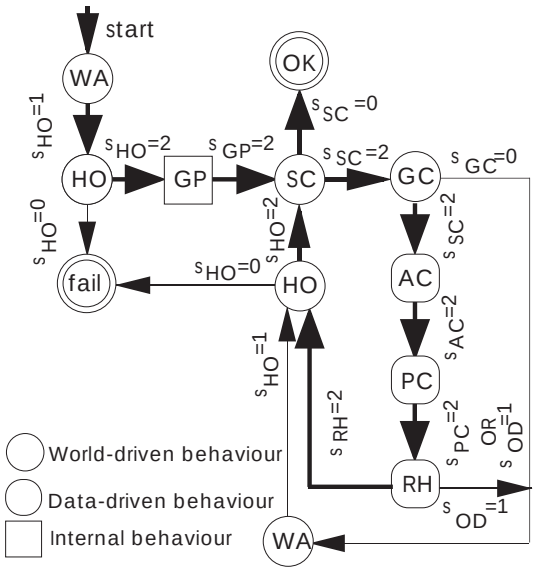

Figure 6 *The TidyUpChairs task represents as an Behaviour Activation Automaton.*

The laser range sensor detects obstacles in a an oblique triangle in front of the robot, provieded by a structured light source. This triangle is defined by its intersection with the ground, 140 cm in front of the robot, and the light source mounted on the to of the robot, at a height of 1 meter. This configuration prevents the detection of close objets. In particular the chairs are not detected when they are near the robot. The OD behaviour is not activated when the robot go towards a chair (GC), since the chair would be consider as an obstacle. On the contrary the OD behaviour is activated when the robot pushes a chair, because they are under the laser plane. In this case, if an obstacle is detected, the robot pushes the chairs up to the obstacle and then returns home. During the way back the presence of an obstacle activates the wander around behaviour until the homing landmarks are visible. The task fails only when the homing behaviour looses the landmarks while it is performing.

The *TidyUpChairs* task performs as desired. Figure 7 shows the robot while performing the *TidyUpChairs* task. Many tests have been run with various chairs and homing landmark configurations: Without obstacle the programmed sequence leads to the kind of path shown in figure 5 (path: HO-SC-GC-AC-PC-RH-HO).

Figure 7 *Robot while performing the TidyUpChairs task.*

6. CONCLUSION

In this paper we present a behaviour-based architecture composed of three layers: sensori-motor, behavioural and sequencing, and describe their structure and interaction. In particular, we define the task to be performed by the robot as a state automaton responsible for the sequencing of the behaviour activity. We illustrate and demonstrate this architecture in a development environment called MANO that runs on a network of workstations, a Nomad200 mobile robot and dedicated vision hardware. It also encompasses various sensing devices at the sensori-motor layer and a large set of behaviour at the behavioural layer. To illustrate the architecture functionality, we present the *TidyUpChairs* task that is expressed in terms of the state automaton.

The experiment demonstrates the succesful implementation of this task using this approach. It shows the advantage of this sequencing approach to describe tasks, which can hardly be expressed in a conventional behavioural architecture. Finally, the use of a simple database at the behavioural level allows some additional flexibilities in the execution of the task.

ACKNOWLEDGEMENT

This work is bound to project 4023-027037 of the Swiss National Research Program "Artificial Intelligence and Robotics" (NRP23), conducted in collaboration with the Institute of Informatics and AI of the University of Neuchâtel, Switzerland.

REFERENCES

Albus, J.S. (1991). Outline for a Theory of Intelligence. In: *IEEE Trans. Sys. Man. Cyb.* pp. 473–509.

Brooks, R.A. (1986). A robust layered control system for a mobile robot system. In: *IEEE Journal of Robotics and Automation.* Vol. 2(1). pp. 14–23.

Connell, J.H. (1992). SSS: A hybrid architecture applied to robot navigation. In: *Proceedings IEEE Int. Conf. on Robotics and Automation.* Nice, France. pp. 2714–2719.

Coste-Manière, E. and Espiau, B. (1992). A tasl-Level Robot Programming Language and its Reactive Execution. In: *Proceedings IEEE Int. Conf. on Robotics and Automation.* Nice, France. pp. 2751–2756.

Duan, N and Kumara, S.R.T. (1993). A Distributed Hierarchical Control Model for Highly Autonomous Flexible Manufacturing Systems. In: *Proceedings of Int. Conf. on Intelligent Autonomous systems IAS-1.* Pittsburgh, Pennsylvania. pp. 532–541.

Facchinetti, C. and Hügli, H. (1994). Using and Learning Vision-Based Self-Positioning for Autonomous Robot Navigation. In: *Proceedings of Third Int. Conf. on Automation, Robotics and Computer Vision.* Singapore.

Freedman, P. (1992). Modelling the actions of an intervention robot. In: *Proceedings IEEE Int. Conf. on Robotics and Automation.* Nice, France. pp. 2697–2701.

Harrison, M.A. (1979). *Introduction to Switching and Automata Theory*, McGraw-Hill, Inc., 1965.

Hopcroft, J.E. and Ullman, J.D. (1979). *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Pub. Co., 1979.

Hügli, H., Maître, G., Tièche, F. and Facchinetti, C. (1992). Vision-based behaviours for robot navigation. In: *Proceedings 4th Annual SGAICO Meeting.* Neuchâtel, Switzerland.

Hügli, H., Tièche, F., Chantemargue, F. and Maître, G. (1993). Architecture of an experimental vision-based robot navigation system. In: *Proceedings of Swiss Vision.* Zürich, Switzerland. pp. 53–60.

Hügli, H., Facchinetti, C., Tièche, F., Müller, J.-P., Gat, J. and Rodriguez M. (1994). Architecture of an autonomous system: application to mobile robot navigation. In: *Proceedings of the NRP 23 - Symposium on Artificial Intelligence and Robotics.* Lausanne, Switzerland. pp. 97–110.

Nomadic (1992). *Nomad 200 User's Guide.* Nomadic Technologies, Mountain View CA.

Noreils, F.R. and Prajoux, R. (1991). From Planning to Execution Monitoring Control for Indoor Mobile Robot. In: *Proceedings IEEE Int. Conf. on Robotics and Automation.*pp. 1510–1517.

Slack, M.G (1992). Autonomous navigation of mobile robots for real-world applications. In: *Interdisciplinary Computer Vision, SPIE, Vol. 1838.* pp. 101–109.

Thorpe, C.E. (1992). Point-counterpoint: Big robots vs. small robots. In: *Interdisciplinary Computer Vision, SPIE, Vol. 1838.* pp. 78–88.