Part II ALICE

Chapter 3

ALICE Structure

The following part of the thesis suggests structures for the spatial modelling level introduced in chapter 2. Navigation, exploration and self-localization in previously unknown environments are the basic tasks which have to be discussed in this context.

In order to focus on the main issues on this field, the robot's world is designed to be simple, but still of practical relevance. The project as well as the mobile, experimental platform itself will be called *ALICE* in the following. This is not an abbreviation, but just a name.

Some basic structures and assumptions are introduced in the following sections, whereas the proposed solutions for the discussed tasks will be presented in chapter 4 "World Modelling", chapter 5 "Navigation", and chapter 2 "Exploration".

3.1. The Assumed World

The world of *ALICE* is constructed on the basis of some elementary assumptions. Practical relevance and sufficient simplicity are two contradictory goals that have to be unified in order to define a universe for the present robot. The practical aspects of *ALICE's* world should allow real world tests, which include all relevant problems of mobile robots not having possibilities of (external) global positioning. On the other hand a certain degree of simplicity is needed to allow plausible simulations and to achieve the possibility of generalizing the results to classes of robots and sensors. Furthermore biologically implausible features should be avoided. The central, resulting approach covering all these streams has to be considered regarding certain categories of the world:

- **Position**
  ALICE is allowed to estimate its relative position by dead-reckoning. This is biologically plausible, when the accuracy does not exceed certain limits. Moreover a system being able to handle significant errors in the position measurement contains a wide practical relevance. This position correlation task has to be solved in any mobile system, not having the opportunities to operate in prepared environments with a dense net of landmarks or to receive global position information (for example *the* global position regarding our planet from the United States military global positioning system (GPS)). Another approach would be to dispense with an explicit position at all. Thus the self localization could be performed on the base of typical sequences of impressions (sensor-readings) as shown in [Tani93].
- **Analog sensor readings**
  ALICE has access to continuously varying features of the outer world, regarding time and movements of the robot itself. According to the biological fact that even the mammal eye is not able to measure absolute physical quantities (in this case light intensities), the sensor models applied for *ALICE* are not

calibrated. Moreover the sensor models are non-linear, contain a large degree of noise, and can fail completely.

- **Binary sensor readings**
  Some impressions from the outer world are of a mainly binary nature as for example simple forms of whiskers or other touch sensors. This is also included in *ALICE's* world, in order to demonstrate that spontaneously varying signals can be handled in the proposed world model.
- **Error Tolerance**
  Two kinds of error tolerance have to be discussed. First the error-tolerance in learning or adaptation phases. Here any kind of "miss" is allowed in *ALICE's* world. This phase can be motivated by any learning phase of mammals, where a huge amount of possibilities is tested in form of (not only children's) games. Due to the fact that the sophisticated mechanics of current robots are not error-tolerant at all in comparison to biological systems, the robot's world itself has to be limited to allow learning in trial-and-error phases. Error-tolerance during the application phase of the system is the second aspect that has to be mentioned. *ALICE* does not distinguish an explicit learning and application phase, i.e. adaptation to the current situation or environment is a lifelong ongoing process in a changing world. This is the first reason for an error-tolerant world at any time. But furthermore the world model as it will be described in chapter 4 includes system-immanent inconsistencies that prevent any globally correct "blind" planning. Any plan may fail in *ALICE's* world and must then be replaced by a more recent and probably better plan. In the opinion of the author, this is not a weakness but an attribute of the real world, which should be included in any robot's universe of practical relevance. Even in simple, completely static environments it is very hard to guarantee a stream of correct respectively consistent sensor readings, which is necessary for a completely correct world model.
- **Structured Environment**
  ALICE's world should be sufficiently "rich" to enable self-localization at positions or areas of a certain density. That means that the sensor readings must change significantly in travelling distances that are small regarding the relative position corruption. The most obvious real example of a "poor" world in this context is a sandy dessert, where the densities of features, applicable for position correlations is too low for most humans. On the other hand the frequency of changing features while moving must be low enough, in order not to generate stochastic signals regarding the robot's sampling frequency. Actually this frequency have to be much lower than given by Shannon's sampling theorem, due to the degree of statistically stability that should be achieved applying fuzzy and unreliable sensor models.

These assumptions about *ALICE's* world lead to the components and structure of *ALICE* itself, as introduced in the next sections.

3.1.1. Platform (Sensors and Actuators)

The physical realisation of *ALICE* and its components will be discussed shortly in this section. For an optical impression of *ALICE* please refer to figure 1. The included features are just sufficient for an autonomous mobile robot operating in realtime in a universe as described in the section above. The platform's kinematics allow an omnidirectional motion control, i.e. *ALICE* is able to go directly towards any direction without any preparatory manoeuvring. The fact that the platform is round and symmetrical regarding the employed sensor devices, skips any movements that would be needed, if certain orientations or perspectives would result in other perceptional or kinematical possibilities. All (six) wheels are driven and steerable. Thus even the propulsion system is completely symmetrical. In order to calculate a relative spatial position the wheel revolutions (measured at the gears) are counted. The achieved accuracy by this dead-reckoning is limited by an (drift-) error of 20 to 25%, related to the passed distance and depending on the kind of performed movements and the condition of the floor. This position error is "sufficient" to ensure that any weak position correlation technique will fail very soon.

The binary sensor system is realized by 24 simple whiskers mounted symmetrically around the border of *ALICE*. Each whisker is 17 cm long and mounted in a small metal tube, giving a binary signal, whenever the whisker is sufficiently pushed to establish a contact between the whisker and the tube. This "sufficiently" is very fuzzy and depends on the angle, force and speed of the pushing object. Therefore the angular resolution detecting for example a straight wall is below 15° or in some configurations even below 30°.

The sensor system giving continuously varying signals is represented by 24 passive light sensors, measuring the light intensity detected from a certain direction, with an angular resolution of approximately 20°,

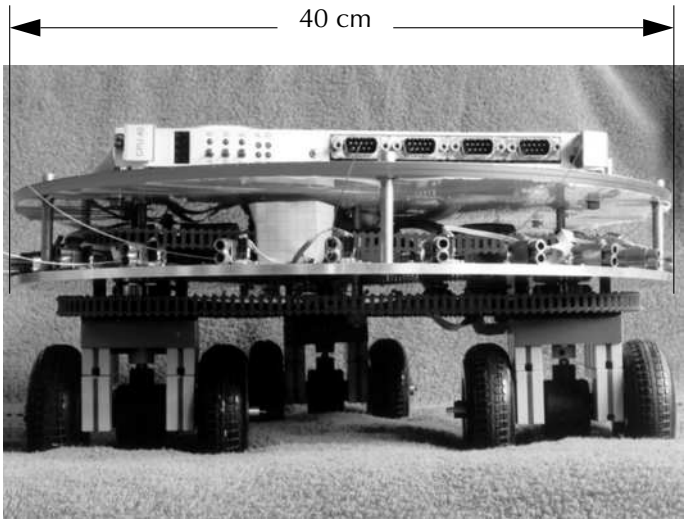

Figure 1: ALICE

i.e. every light impressions in a cone of 20° is integrated into one scalar sensor reading. The light sensors (passive photo resistors) are mounted together with the whiskers and are therefore also distributed symmetrically at *ALICE's* border. Please refer to figure 2, giving an impression of the physical realisation of the light sensor - whisker combination. The upper tube hosts a photo resistor, whereas the lower tube caries the base and the detector for a 17 cm long (steel-) whisker. The tubes are 2.5 cm long.

The on-board computer power is limited to one standard CISC-CPU (currently: MC68040 at 33 MHz equipped with 16 MB random access memory and standard I/O). It is intended (and shown in the following chapters) that this computer power is sufficient for a full realtime implementation of all world modelling algorithms. The term "realtime" can be roughly and pragmatically approximated here by the demand that the machine should be able to go continuously at full speed (25 cm/s) where the world model is adapted continuously without any time-jitter or delay. More details about realtime world modelling may be found in section 4.2.2 whereas general realtime issues are discussion in chapter 5.

The author would like to emphasize that the low reliability and resolution of the employed sensor devices together with the low relative position accuracy and the limited computer power is *not* a weakness of the system but is chosen *intentionally.* A low-precision system like *ALICE* is an adequate experimental platform for any world modelling and control technique, which is intended to be stable and reliable in a real world. The key issue of this part of the thesis is to prove the *ALICE* configuration to be sufficient and efficient to solve some basic mobile robot tasks, like self-localization and navigation. Any sensor and kinematic system with features superior compared to *ALICE* (i.e. almost any sensor system) promises a further improvement in terms of speed and precision but not with regard of the discussed, principal abilities.

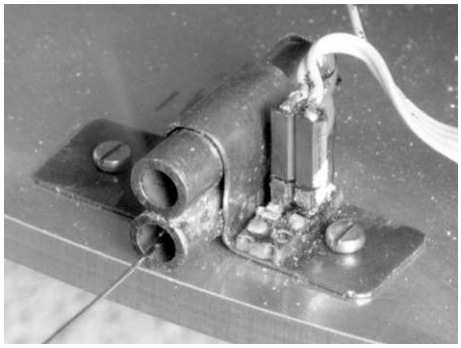

Figure 2: Light sensor - whisker combination

3.1.2. Simulation

A couple of simulation-tools have been developed in parallel to the physical experimental platform. Three phases can be distinguished in the tests. First an environment, simulating the sensor and actor models (including noise and other error sources) is employed in order to prove some principal functionalities of the tested components. After ensuring some stable behaviour in the simulator, the realtime and real world tests on the physical *ALICE* are performed. Finally the gathered information from the test runs is analysed in retracing tools.

Simulation

The simulation has to consider mainly the two employed sensor models (whiskers and passive light sensors) where only a simple 2-d model of light distribution is applied together with a model for the position drift effects. The simulation is implemented including a graphical, interactive user interface to support a immediate impression of *ALICE* moving in its test-environments. For details about the simulation-techniques and a discussion of the user interface please refer to [Smidt94].

Analysing (Retracing)

The simulations are mainly dedicated to the first debugging steps, whereas the retracing tools are applied to analyse the realtime and real world behaviour in all detail. Due to the large memory capacities on *ALICE*, all informations can be stored during a test run. Based on this data-sets the complete processing can be retraced using all (graphical) analysing possibilities of a workstation. The applied algorithms on *ALICE* and the analysing workstation are identical on the object-code level, due to a specially adapted operating system (*ALBATROSS*, see [Wetzler91] and chapter 5) on the physical platform. Therefore the operator analyses the same (physical) phenomena (beside any asynchronous realtime events) on both systems, i.e. not any differences in the implementations respectively compilers. Furthermore it is possible and useful to examine some kinds of changes in the world modelling concepts in the retracing, based on real data, as long as the operator is aware that some manipulations of the algorithms would result in different movements respectively positions of *ALICE* in the real world and therefore in gathering other data from the environment. A complete description of the analysing tools can be found in [Lefèvre94] and [Prüß95].

3.2. Control Structure

The control structure of *ALICE*, as shown in figure 3, is based on the common decomposition of sensor-data abstraction (world modelling) and action generation (reflexes, navigation, exploration). The binding structures between these parts are an explicit world model (here: topologic map), regarding the connection from the sensor data abstraction to the control components, respectively the real world itself, considering the

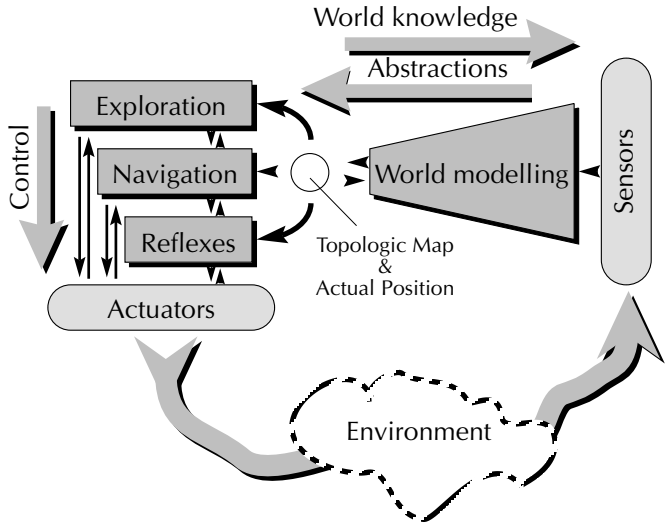

Figure 3: ALICE control-structure

connection in the counter direction. The contact to the outer world is established via a couple of sensors and actors, described in the sections above, where the influence of the actuators regarding the sensor readings cannot be formulated in simple, closed forms, due to the facts that the real world contains certain stochastic and unforeseeable elements and the actual, considered environment is explicitly unknown in advance.

This model is quite general, beside the possibility to overlap the world modelling and control parts, i.e. applying unifying, immediate structures for the action generation, based directly on the sensor data. Examples for such an immediate connection between sensing and acting without an explicit world model, are shown in chapter 5 and extensively in chapter 8 "Visual Search".

The different components, establishing the “inner” connection between sensing and acting in the *ALICE* model are introduced briefly in the rest of this section. For the complete discussion please refer to the following chapters.

World Modelling

Based on the preprocessed and normalized sensor data gathered from the outer world, the world modelling performs a number of abstractions, in order to filter out and integrate the information relevant for the class of applications in mind. The integration process is related to time as well as to different kinds of sensors. The application class discussed here is navigation, i.e. the world model should represent spatial information. The introduced spatial representation is of a qualitative, topological nature, and includes a position information, which is correct and consistent in a local sense. The world modelling process itself is influenced also by the generated world model, i.e. the way of extracting and even the definition of relevant information can rely to a wide extent on the systems's experiences.

A new world modelling concept in introduced in chapter 4, where special emphasize is on stability, simplicity and qualitative, topological modelling together with low computational effort and sensor requirements. The resulting techniques are discussed on the base of algorithmic structures as well as on practical experiments.

Reflexes

The lowest layer in the control hierarchy is responsible for security aspects, and elementary modelling of the robot itself. The security tasks are mainly emergency stops or shutdowns of the robot in critical situations. The second area of responsibilities can be a wide field in complex robots. For example the kinematics and accelerations that have to be modelled can lead to a complex control-application or even an adaptive system, where the relations between motor commands and actual movements are learned or adapted continuously according to the actual or current situation. The chosen configuration of *ALICE* skips these complex modules, due to high accelerations and straight forward kinematics. Nevertheless, position-control including acceleration slopes and dead-reckoning has to be implemented even here.

Furthermore any kind of reflective behaviour, i.e. without need for an explicit spatial representation may be realized in this layer. Due to the fact, that the focus of this part of the thesis is on spatial modelling, the family of reflexes is not discussed here, although a couple of mobile robots are constructed explicitly by combining these “low-level” instincts to a global behaviour (see [Brooks90], or [Mataric90] for examples of pure behaviour-based approaches respectively [Mataric92] for an example of representation integrating behaviour-based approaches).

Navigation

The navigation layer is dedicated to the goal driven control of movements, which are explicitly planned and executed on the base of spatial representations (topologic maps). Two main tasks have to be fulfilled in order to move the robot to a given destination. First a plan is needed, describing the path or the group of paths, which assure certain optimization criteria, like short length, short driving time, or small number of turns. This navigator-component is denoted as “global navigator”, or “planner” (or simply “navigator” in some other robotic-projects). Furthermore the planned path has to be executed. The involved component is denoted as “local navigator”, “pilot”, or “path controller”. Recent information about the current environment is employed in this stage. This new data can differ significantly from the information the plan is based on, due to dynamic environments or errors in the sensor data gathering or abstraction processes. Therefore any plan can fail at some stage during its execution, switching the navigator back to the planning phase, in order to produce a new, refined plan or to indicate a global failure.

In the proposed solution, the execution process is adapted to the current environment, in the sense that motor commands are learned compensating drift effects or which are needed to perform special manoeuvres needed in specific situations. This is an example for an immediate transformation of sensor information (respectively sensor data together with goal specifications) to actuator commands, without an intermediate representation, abstracting the sensor readings.
A concept for a navigator employing a modified A*-algorithm, and introducing an adaptive path execution component based on reinforcement learning methods is discussed in chapter 5.

Exploration

The exploration component is introduced to optimize the manoeuvring strategy for collecting recent data from the environment. Several aspects have to be considered. First, the explorator should build up a new world model optimally regarding several optimization criteria, like speed and accuracy. Second, the already established world model should be "maintained". This could be regularly checking for the presence of already recognized features, a closer analysing of critical areas, or searching for new features, etc. Finally the explorator has to care for specific demands of the world-modelling component. For example establishing a statistically stable representation can rely critically on the order of presenting features to the sensor data abstraction processes, i.e. on the path, the explorator has chosen. Furthermore the stability of the self-localization process can depend strongly on the explorator during the start-up phase of the system.

The exploration module can employ functionality from the navigator respectively from the reflex-layer, in order to optimize its speed and efficiency. Several exploration instincts and their superposition are discussed in chapter 2.

3.3. Other ALICE-Investigations

The *ALICE*-project is mainly discussed on the basis of three interfering levels: world modelling, navigation, and exploration, whereas aspects, not in the focus of the principal discussion are omitted. Thus the reader is asked to refer to the following list of investigations in the context of *ALICE* for any further information:

- Simulation of *ALICE* and dynamic situation clustering: [Smidt94]
- Navigation strategies based on reinforcement learning: [Fischer94]
- Construction of the *ALICE*-platform: [Prüß94]
- Self-localization and stable world modelling under realtime constraints: [Lefèvre94]
- Exploration strategies for qualitative topologic world modelling: [Prüß95]
- Multi-layer self-organizing maps applied for distributed world modelling: [Stranz95]
- Several assistant works regarding "real world problems" performed by Lefèvre and Prüß. A video, documenting aspects of the real world experiments is currently about to be prepared.

All these works were guided and supervised by the author.

Chapter 4 World Modelling

Based on the idea that an autonomous robot should have the ability to create and update "relevant" representations or models of its current environment under hard realtime-constraints in order to be really useful, a couple of "world-modelling" strategies have been introduced. The attribute "relevant" in this context means: "useful for a certain class of tasks". A wide range of symbolic and functional decomposition approaches have been already applied for this class of mobile robot tasks (see e.g. [Brooks83], [Hoppen90], [Kuan85], [Lozano-Pérez83] or [Zimmer94a]), but the success in real environments depends widely on the quality of the employed sensors. Handling high-speed, high-precision sensors means processing high bandwidth information-streams and results in complex multi-processor systems and especially in realtime-problems. Furthermore complex computer-systems limit the lower size and weight for the autonomous mobile platform and therefore the range of applications.

During the last few years some basic tasks like collision avoidance, kinematic modelling and reflective navigation have been approximated by much simpler methods such as behaviour-based approaches, reinforcement learning methods and others. The idea common to all these techniques is qualitative modelling as opposed to high-precision control. Works utilizing qualitative techniques for navigator-world-modelling include the basic article from Kuipers introducing the term "qualitative map" '88 in [Kuipers88], the ultrasonic-clustering techniques published by Kurz in multiple articles (e.g. [Kurz93]), the expansion of the behaviour-based approach from Mataric published '92 in [Mataric92], and the recently published work of Tani based on sensor-sequences rather than on explicit topology ([Tani93]). One project handling multiple representation simultaneously (in this case, topology superimposed on the geometrical model) is the HILARE-project published in multiple papers (see e.g. [Laumond93]).

Nevertheless explicit metric world modelling for navigation tasks is still being employed in most projects. Overcoming the well-known problems of metric mapping, this thesis introduces a new method of robust, qualitative and topological world-modelling (a term to be defined in the next section) usable for navigation-tasks under real-world constraints.

In order to test the real-environment and real-time assumption, the mobile robot "*ALICE*" was built. The performance of the sensor equipment and mechanics, as well as the available computer power, is limited intentionally to a very low level as introduced in the previous chapter (i.e. 24 binary whiskers, 24 passive light sensors, 1 Motorola 680x0 CPU, an omnidirectional platform of 40 cm diameter and odometry with up to 25% drift).

The implementations of the network structures and analysing tools as well as the construction of the real world ALICE are performed by Lefèvre [Lefèvre94] and Prüß [Prüß94].

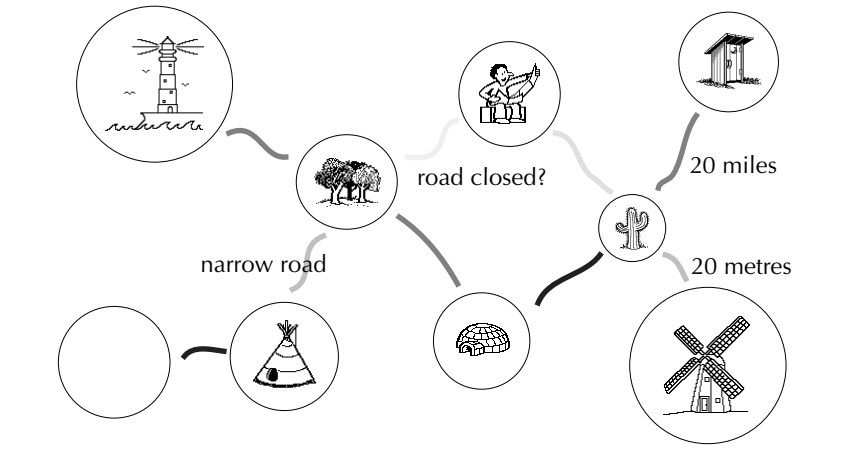

Figure 4: Qualitative topological map

4.1. Topological World-Modelling

The central motivation of **q**ualitative **t**opological world models (**QT**-Models) is the basic mobile robot task: "Recognize places you have seen before!". In this thesis this task will be approximated by extracting "situations" (i.e. recognized places) together with their topological neighbourhood from the current sequence of sensor-samples, rather than modelling the boundaries of the detected obstacles and objects in a metric manner. Assuming a stable situation-recognition-process and a technique for moving between distinct situations, the concept of a qualitative, topological world model (outlined in figure 1) suggests a human-motivated basis for a navigation. The main concept has already been proposed by Kuipers et al. [Kuipers88], but here the construction process was carried out using explicit rules, instead of statistical techniques. Although Kuiper proposes a kind of qualitative reasoning on the current environment, the real-world abilities of the approach are (due to low statistical stability), in the opinion of the author, limited.

The world model proposed in this thesis is based on clustering techniques introduced by Kohonen ("self-organizing-maps", [Kohonen91], [Kohonen90], [Kohonen89], [Kohonen84]) and Fritzke ("growing cell structures", [Fritzke93c]) together with some previously proposed extensions by this research group [Zimmer93b]. Due to a couple of specific autonomous robots-constraints, these structures are modified to cope with realtime-aspects, lifelong learning, "local forgetting", and so on (see chapter 1.2 for details). One important improvement regarding the Kuipers-approach, is the dynamic definition of situations, i.e. concepts of the environment.

To make the term "situation" in this context more precise, figure 2 shows a typical situation generated in *ALICE's* test-runs. The inner circle describes the distribution of light-impressions received from particular directions, the outer circle shows the smoothed touch-values originating from contacts with obstacles in various directions. The two-number pair in the centre gives a rough approximation of the geometric position of this "situation".

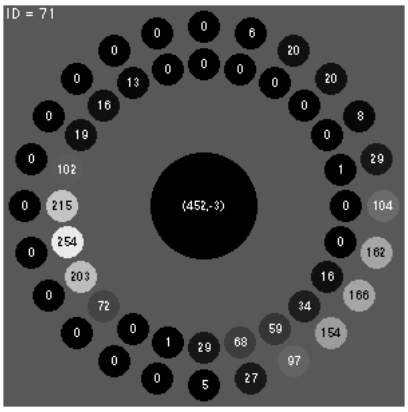

Figure 5: A "situation"

4.2. Methods

This section will introduce the technical details of the proposed topological world model. The algorithms following are expressed in general terms only, ignoring computational details.

4.2.1. Pre-Processing

Following the idea of representing situations, consisting of readings from different kinds of sensors in a way that they can be compared in one step, and by employing a simple norm, the sensor-samples have to be pre-processed to form a vector of unified elements. In the current system passive light and tactile sensors as well as an *(x, y)*-position produced by odometry are available. Considering the fact, that the angular resolution of the tactile sensors is very low, each vector of tactile readings is smoothed by applying gaussian functions. Finally the sensor samples from different types of sensors are weighted and concatenated to produce a "situation-vector", or more briefly a "situation" (consisting of 50 values in the example given).

In the following, sensor situations will be indicated as S; the position or x, y-part of such situations as p.

4.2.2. Adaptation

As a basis for the network model, the euclidian norm is applied to calculate distances between sensor situations, d, and distances between positions, g, respectively.

Consider a network N consisting of a number of cells c_i, which are connected with respect to the topological neighbourhood of the situations $S(c_i)$ attached to each cell. Then at each adaptation step the cell c_{opt} with the smallest situation-distance d_{opt} to the new input situation S_x is determined according to:

$$\forall c_i \in N\colon d_{opt} = d(S(c_{opt}), S_x) \leq d(S(c_i), S_x) \tag{1}$$

In order to limit the effort for this adaptation to a constant the search area is limited by the geometric distance g_{search}. In the current system this is done by applying adequate data-structures to the network-management. The selected cell c_{opt} and all its topological neighbours are then adapted according to:

$$c_{opt}^{new} = c_{opt} - (\varepsilon_o \cdot d_{opt}) \tag{2}$$

$$\forall c_n^j \mid a(c_n^j, c_{opt}) > 0\colon \quad c_{n_j}^{new} = c_{n_j} - (\varepsilon_n \cdot d_{n_j}) \tag{3}$$

where $a(\cdot,\cdot)$ is the adjacency-function of the network. The "classification error" d_{opt} is then added to a total classification error d_{total} attached to the cell c_{opt}.

$$\forall c_i \in N \text{ (after n adaptation steps): } d_{total_i} = \sum_{t=1}^{n} hit_{i,t} \cdot d_{opt_t} \tag{4}$$

$$\text{where } hit_{i,t} = \begin{cases} 1 & ; c_i = c_{opt_t} \\ 0 & ; c_i \neq c_{opt_t} \end{cases} \tag{5}$$

In order to decrease the adaptation speed of a well adapted network, the parameters ε_o and ε_n are controlled by:

$$(\varepsilon_o^{new}, \varepsilon_n^{new}) = \begin{cases} (\varepsilon_o \cdot \varepsilon_\Delta, 0) & ; (d_{opt} \leq d_{acc}) \\ (\varepsilon_o^{init}, \varepsilon_n^{init}) & ; (d_{opt} > d_{acc}) \end{cases} \tag{6}$$

where: $0 < \varepsilon_\Delta < 1$

and ε_o^{init} resp. ε_n^{init} are the initial values of the parameters ε_o and ε_n.

In each adaptation step, where d_{opt} is larger than a tolerated error d_{acc}, a global counter n_{miss} is incremented. This counter will be used as a measurement for the need for change in the network structure. An update-counter u_{opt} attached to c_{opt} is incremented and will be used as an indicator for the stability of this specific cell.

In order to use the high speed of this adaptation process to achieve better adaptation, each situation is presented several times k to the network. A constant delay of l sensor-sample-time-slots before the current sen-

sor situation affects the network is also found useful (see section "Correlation" below). Accordingly a learning set holding $(k \cdot (l+1)) - 1$ situations is implemented.

4.2.3. Growing & Shrinking

At start-up time of the system, there are no cells; the network is empty. So the common problem finding a "good" initial state of the network is avoided, but there is a need for some growing strategy. In the present system, two growing strategies are applied. The first is called "**spontaneous insertion**", the second "**statistical insertion**". In the first, new cells, representing the current sensor situation, are inserted when the distance between the current sensor situation and c_{opt} exceeds a certain limit ρ_s (in the special case of an empty network this strategy produces the first cell). In the second strategy a new cell is inserted in the middle between the cell with the highest "degree of movement" c_{runner} (measured by the cell attribute d_{total}) and its farthest topological neighbour c_{far} every n_{insert} "miss-classifications" (measured by the global counter n_{miss}). The new cell is instantiated with mean-values of c_{runner} and c_{far} for position and light-intensity, but with minimal values for touch-information.

Another aspect of growing relates to the topological connections between cells. Assuming that c_{opt} has just changed from c_{opt}^{old} to c_{opt}^{new} in two consecutive adaptation steps, and that the cell c_{opt}^{new} has m other neighbours c_j $(a(c_{opt}^{new}, c_j) > 0)$, the following changes in connection weights are imposed:

$$a(c_{opt}^{new}, c_{opt}^{old}) = 1 \tag{7}$$

$$\forall j, (1 \le j \le m): \; a^{new}(c_{opt}^{new}, c_j) = a(c_{opt}^{new}, c_j) - (a_{red}/m) \tag{8}$$

with $0 < a_{red} \le 1$

A connection with a weight ≤ 0 is regarded as non existent. Thus the deletion of cells is now straight forward. A cell or a cell-cluster with no connection is removed.

4.2.4. Correlation

Three degrees of freedom out of the internal representation ((x, y)-position and orientation) will be corrupted by drift effects or other errors, if they are not continuously correlated to the world model. Even a stable and error-tolerant network structure will not be able to produce stable world models, if the robot's position is corrupted. The obvious approach is to correlate the internal position continuously with the world model built up so far. On the other hand the robot's position is integrated in the world model. This mutual stabilizing technique is very useful applied to local world models. But in fact each representations are updated with the faults, errors and noise from the other. This principal problem prevents a globally consistent world model of arbitrary size including position when only local information is available.

One strategy, used to stabilize the internal position, is to delay the integration of the current sensor situation S_c by a number of adaptation steps l (see section 1.2.2 above). In this way the internal position is correlated on the basis of the formerly integrated world-knowledge at this point, and not on the base of S_c, which would not make any sense. An estimation p_{est} of the current position is produced using the sensor-distances d_i (to S_c) and the number of updates u_i of all cells c_i in the immediate topological neighbourhood of S_c. The estimates produced in this way, are not necessarily near to the "correct" position in every case. Thus the internal position is not changed to p_{est}, but is only "moved" towards this point by a fraction ε_{pos}. The fact that these estimation errors are normally distributed in the long term, together with the fact that the remaining position-errors are integrated into the world model after a certain delay, leads to a stable behaviour of the local world model.

The third degree of freedom, the orientation of the robot, is also corrupted by drift and other effects. In order to correlate orientation with the internal representation and the world model, it must be assumed that the current position is correct to within a certain tolerance. Based on the geometric distance, the nearest cell c_{near} is determined by:

$$\forall c_i \in N: \; g(S(c_{near}), S_x) \le g(S(c_i), S_x) \tag{9}$$

To obtain a comparable measurement for the orientation based on sensor impressions including position, we interpret the sensor values as polar coordinates:

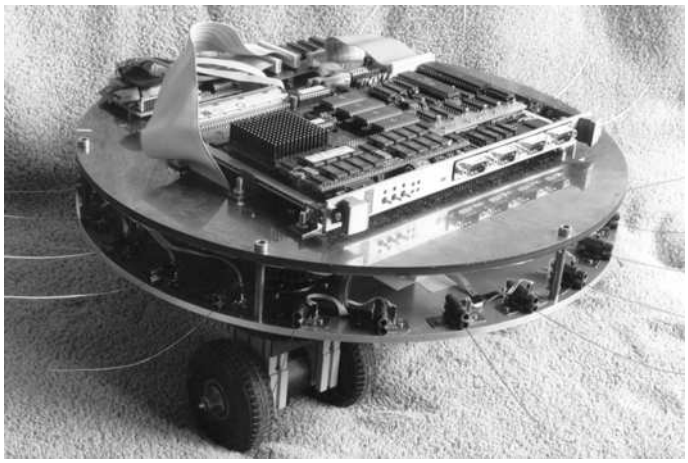

Figure 6: Experimental platform *ALICE*

$$l_i \Rightarrow (l_i \cdot sin\,(i \cdot \frac{360°}{s}), l_i \cdot cos\,(i \cdot \frac{360°}{s})) \tag{10}$$

$$t_i \Rightarrow (t_i \cdot sin\,(i \cdot \frac{360°}{s}), t_i \cdot cos\,(i \cdot \frac{360°}{s})) \tag{11}$$

where l_i are the light-readings, t_i are the tactile readings and $0 \leq i \leq s-1$. A possible derived measurement for the orientation (modulo 180°) is a linear regression O on the $2s$ polar "sensor-points". The angular difference between the orientation of the regression of the current sensor scan O_c, and the orientation of the regression O_{near} of the nearest cell c_{near}, is interpreted as the current orientation-error. As in the case of the position correlation above, the internal orientation is only "moved" by a fraction ε_{or} towards the estimated orientation. In the current system only the light sensors are employed to determine the orientation estimates. The internal orientation is continuous and not quantised according to the angular sensor-resolution. Thus the incoming sensor-readings are interpolated linearly and rotated according to the current orientation of the robot.

Due to the fact that the remaining position-drift is integrated with the world-model, a global drift of the whole model may occur. This is not necessarily a problem if the world model is drifting as a whole, but as the world model expands and some areas are visited in a sporadic manner, different areas will drift in different ways, resulting in an inconsistent world model. Therefore a cell-specific slow-down of the adaptation speed is introduced when cells are "confirmed" at least n_{fix} times (i.e. $u_{opt} \geq n_{fix}$).

4.3. Configuration

In the current configuration, 3 sensor-samples per second are employed to build up a stable world model for a mobile robot moving at a speed of 25 cm/s. All computations are completed in realtime on one CPU from the 680x0-class so that the robot does not need to stop in order to maintain a consistent world model. Even where each sensor-sample is used 10 times, i.e. 30 adaptations have to be completed each second. Figure 4 shows a network-part of approximately 1 by 1 metre as result of a test-run in an environment measures 3 by 3 metres. The information represented here shows only the position-part of each cell together with their topological relations and the light-distribution associated with.

There is sufficient computational power remaining to run the control-loops of the robot as well as the explorator (see next section) using the same CPU. The navigator is not yet tested under realtime-constraints, but considerations based on the computational effort of the navigator in simulated environments suggests that a realtime-integration using the same CPU seems feasible.

4.4. Comparison

In this section the features of the qualitative topological world model described above, are compared to the metric modelling technique as applied in the MOBOT-project (see e.g. [Knieriemen91a], [Hoppen91], [Hoppen90]), as a representative from the range of exact geometric modelling techniques.

Computational Effort

In the QT-approach one standard CISC-processor (680x0) supplies all the computational power needed under realtime constraints, while multiple CPUs are required for the exact geometric modelling in the MOBOT-project. The amount of energy needed for the CPUs, can be also critical, since the weight of the batteries needed has a significant impact on the construction of the robot-chassis. Moreover the control-effort increases with higher weights, leading to a need for further computational power.

More details about the realtime-abilities of the *ALICE*-project can be found in [Zimmer94c].

Stability

In the exact geometric approach, every sensor-sample has to be evaluated for every disturbance. The only possibilities after such an evaluation are to "accept" or "reject" the sample. Due to the fact that the exact world model is very sensitive to noise and any other error source, every slightly disturbed sensor-pattern must be rejected, resulting in a very low noise and error tolerance. On the other hand the QT-model integrates all sensor-samples and smooths the errors by utilising many sensor-readings in a short time/distance. Assuming that systematic errors are reproducible, even they do not lead to system failures in the QT-approach.

Reliability

The benefits of statistical techniques for achieving stability may constitute problems in terms of geometric reliability. A specific situation which occurs only once for example finding a useful pathway through a narrow corridor, is ignored by the QT-technique, but not necessarily by the metric modelling process. The second source of problems originates from the technique of using mean-values instead of stored sensor readings in the QT-model. This may lead to inconsistencies, because these mean values may not correspond to

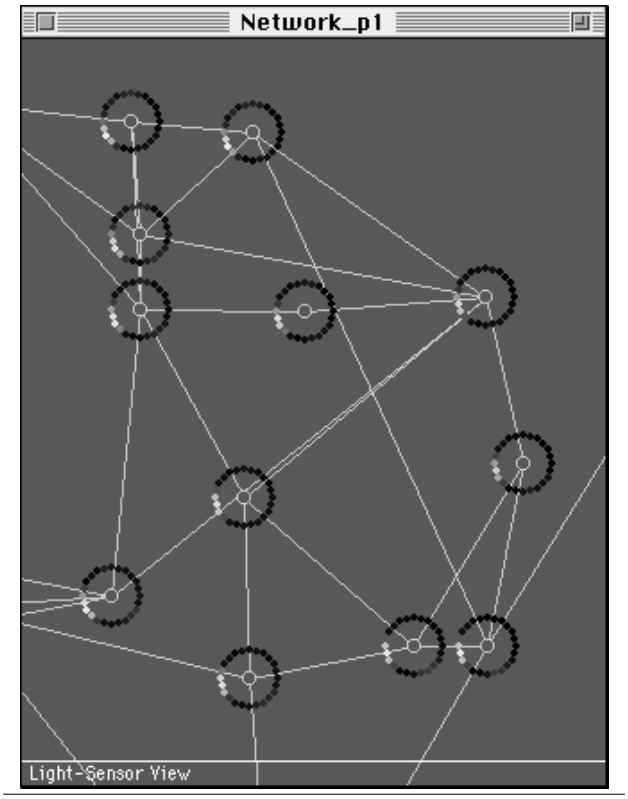

Figure 7: Topological map (light situations)

real points in the environment. If there is a need for a "totally correct" world model, the metric modelling is called for, although even here nothing is guaranteed. The reliability of the QT-approach can only be expressed in statistical terms. On the other hand the reader should keep in mind, that the real world, or at least the gathered sensor-reading, is inconsistent. Therefore eliminating the inconsistencies in the robot's world model is always a process of approximation. Error-tolerance on the other hand should, in the opinion of the author, be a basic requirement for every useful mobile platform.

Local Correlation

The correlation of position- and orientation-drift effects is carried out on a statistical basis in both models (see [Weiß94] for a geometric correlation), achieving some compensation for drift effects in both cases.

Sensor Requirements

The differences in the sensor-requirements are obvious. For an exact geometric model exact distance measurements are needed, as produced by laser-range finders, radar devices, or large-scale video-signal processing. In order to build QT-Models, any kind of short-range sensor may be considered. In the case of *ALICE*, simple passive photo-resistors together with primitive touch sensors prove sufficient. It should be noticed that most topological models in the literature are built "on top" of a geometric model and therefore use the same sensor-information.

Practical Relevance

As a final remark in this comparison, it must be mentioned that any exact geometric modelling as proposed for example in the MOBOT-project is still completely unproven in unknown environments under real world or realtime conditions. Thus the comparison is based on assumptions for the geometric world modelling evaluated in simulations only. Test performed under more realistic conditions, could lead to even worse assurances for example regarding the order of computational power or the assumed stability.

4.5. Experiment

In this section the author tries to emphasize the real world aspect of the *ALICE* project, i.e. the world modelling should be stable regarding the assumed world introduced in section 1.1. The behaviour of *ALICE* is documented under a couple of critical conditions, like inadequate parameters, certain sensor weights, lost correlation, and dynamic environments. The most important, observed feature during the test-runs is the internal position of *ALICE*. Whenever *ALICE* loses its position respectively orientation correlation, the world model is obviously useless for the intended purpose. Furthermore some heuristics regarding the quality of the network can be applied and evaluated (in case of a stable internal position):

- **Generalization**
  The number of generated cells in the network should be small. Thus requirements regarding realtime, memory capacities, and generalization can be fulfilled. Moreover only a sufficiently low density of cells enables the correlation abilities.
- **Topology**
  The generated network should be a topologically adequate representation of the real world. Moreover only a low degree of interceptions between the connections should be accepted. Otherwise the graph search methods of the navigator's planning component will fail (see chapter 5).
- **Consistency**
  The cells as well as the generated connections should remain in areas, reachable for *ALICE*. The adaptation of the network in the form of representation through mean values gives no general guarantee for this feature. Situations represented by the cells in the network are not necessarily observable or existing in the real world. Especially the neighbourhood adaption plays a critical role in the generation of inconsistent network areas.

The tests are performed in a polygonal (roughly rectangular) environment of approximately three times three metres with a round pillar (diameter: 45 centimetres) slightly asymmetrical in the centre. The borders are straight wall segments, each of a length of one metre. Two respectively three light sources are mounted at the border of the test environment producing any kind of reflections and shadows. Depending on the time of the day the sunlight gives a fourth light source, with completely other characteristics than the spots

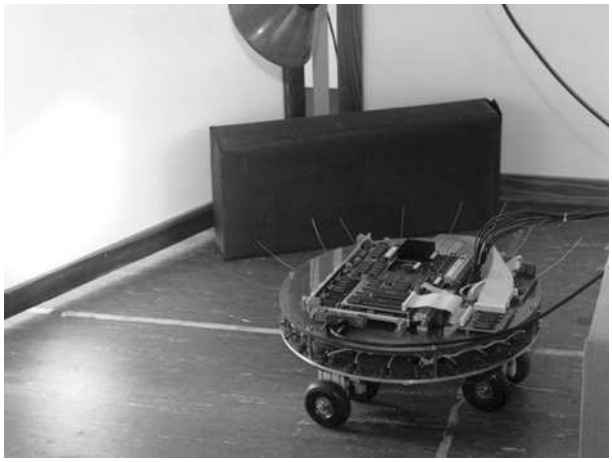

Figure 8: Test scene

with their artificial light. Especially the continuous change of the sunlight makes the test environment dynamic without further engagement. Figure 5 gives an impression of the actual real world environment. The bright spots at the left are light reflections.

The shown results depend critically on the strategy and order of gathering sensor readings from the actual environment. The set of applied strategies (called "exploration") for these experiments, is discussed in chapter 2. Due to the fact that there are mutual relations between the world modelling, the explorator, and the navigator, the presentation of modules cannot be completely hierarchical.

4.5.1. Critical Network-Parameters

As one representative out of the group of network parameters, the critical influence of the parameter ε_o is shown. ε_o controls the adaptation speed of the cells (ε_n is changed proportional to ε_o in this test, i.e. $\varepsilon_o = 10 \cdot \varepsilon_n$). First, ε_o is set to a large value (three times larger then the usual value of 0.03). The resulting world modelling performed by *ALICE* shows a very "lively" behaviour (figure 6a). The cells are moving quickly and only a small number of cells is required to represent the whole environment. But the connections are too muddled to be useful for a navigator and the representation is not consistent for example regarding the pillar in the centre. The counter test of reducing the adaptation speed of the network to one sixth of the usual value results in a even worse situation. A huge amount of cells is inserted, due to the fact that the slowly moving cells cannot integrate the quickly changing, sampled situations. Even the position and orientation correlation fails, when the test as shown in figure 6b is continued.

Obviously ε_o is very critical regarding the whole world modelling process. Therefore it must be carefully calibrated according to the density of sampled features and to the actual speed of the mobile robot. In the current configuration, *ALICE* is driving at a (nearly) constant speed of 25 cm/s, so a constant value for ε_o is sufficient, but in a more general case this parameter has to be adapted proportional to the speed of motion respectively to the sampling frequency.

4.5.2. Sensor Weights

The influence of the sensor weight concerns mainly the kind of produced map. The influence of light and touch information to the sensor situation is given by the distribution and strength of these readings in a typical working environment, but the weight for the position has moreover an additional meaning. This weight controls (beside other parameters) the topologic equivalence of the network and real world regarding the robot's spatial position. If the position weight is chosen too small (figure 7a), the topologic neighbourhoods in the network are controlled by the light and touch information only and the test results in wide, intersected connections, without any geometric equivalence. On the other hand, if the meaning of the position for the sensor situation is set too high (figure 7b), the geometric equivalence seems perfect, but the number of cells needs to be rather high, and even worse, the position correlation is getting very critical. Due to the fact

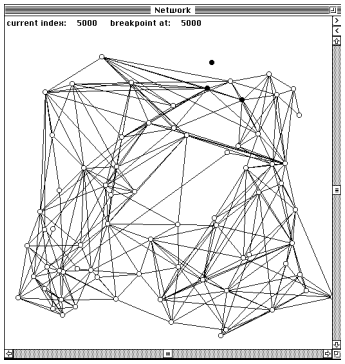

Figure 9a: = 0.1

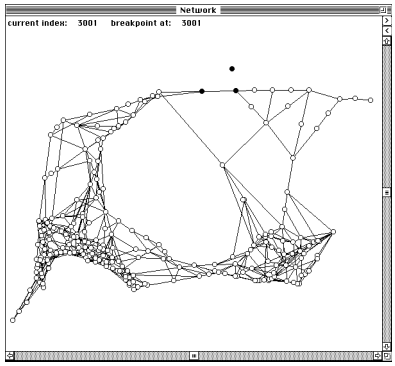

Figure 9b: = 0.005

that the internal position is mostly correlated on the base of other (network-) positions (i.e. not on the base of complete sensor impressions) this world model must fail after a certain time.

The actual setting of the position weight depends on the robot's dead-reckoning precision as well as on the degree of needed geometric consistency.

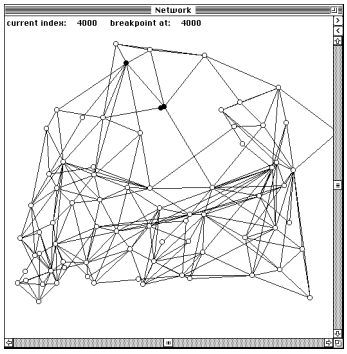

Figure 10a: small position weight

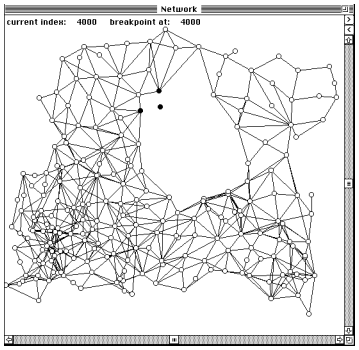

Figure 10b: large position weight

4.5.3. Correlation

The importance of the position- and orientation correlation for the world modelling (and of course for the navigator and explorator) can be shown by two test-runs, where one of the correlation processes is deactivated in each case. The results are shown in figure 8a for a test-run without position correlation and in figure 8b without orientation correlation. Due to the fact, that these experiments without correlation are completely instable, the shown qualitative topologic maps are arbitrary but representative. None of these test-runs could be completed without a total loss of consistency in the world model. This is not surprising, with ALICE's position drift-error of 20 to 25% and rotational drift of up to 0.5° / metre in mind. But the principle loss of consistency does *not* depend on the actual drift rate – as long as there is any drift at all (as in all real world systems) the effect will occur sooner or later.

4.5.4. Static and Dynamic Environments

The final section of the experiments will discuss the development of working qualitative, topological world models in static and dynamic environments. The world model in a static environment reaches an equilibrium state after sufficient exploration of all available features. *ALICE* need approximately 15 to 20 minutes to build up a QT-map of the static test environment as demonstrated in figure 9a to figure 9d. The final state (after gathering 5000 sensor situations) represents the geometric features, the light distribution as well as a

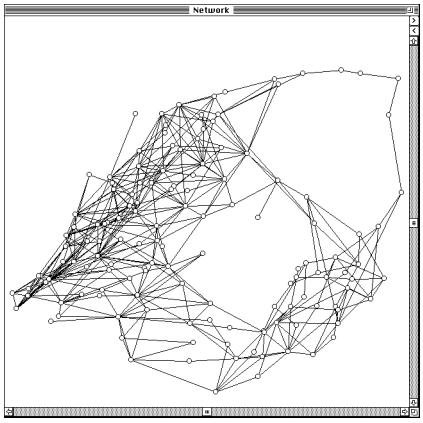

Figure 11a: …without position correlation

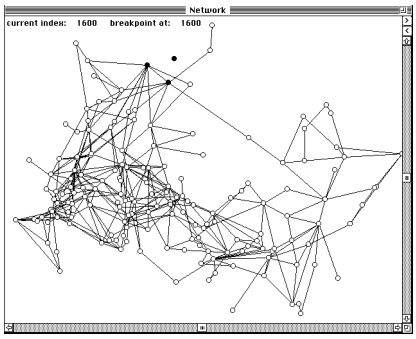

Figure 11b: …without orientation correlation

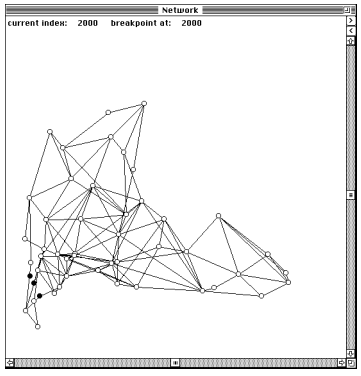

Figure 12a: static, 2000 samples

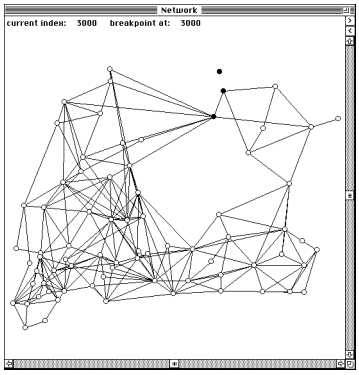

Figure 12b: static, 3000 samples

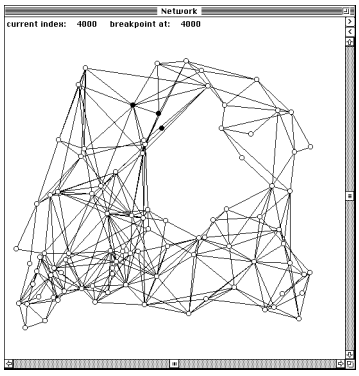

Figure 12c: static, 4000 samples

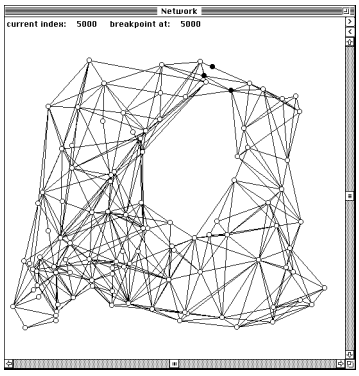

Figure 12d: static, 5000 samples

network graph well suited for the navigator. As introduced in section 1.2, the network holds much more information than shown in these figures. The cells contain whole sensor situations together with statistical values about their history, where the connections are attributed by degrees of confidence. All this information can be employed by the planning or execution (driving) phase of any navigator.

The second world model developing example shown in figure 10a to figure 10d demonstrates the abilities to adapt to a changing world. In order to make the effect obvious, the formerly closed circle in the environment is cut off at the lower end until the gathering of sensor situation 3500 (figure 10c). Up to this situation, the world model shows a clear gap in the lower part. Although the absolute position error between the two sides of the gap is larger than it would be without the splitting wall, then gap is closed smoothly after another 1500 training steps (and of course after removing the introduced obstacle). Some connections at the former gap in the final stage have drastically reduced confidence values and some cells (formally represent-

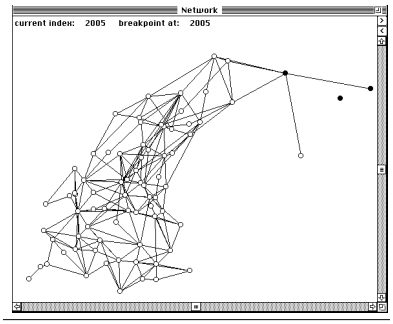

Figure 13a: dynamic, 2005 samples

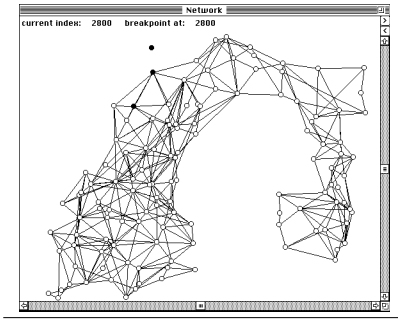

Figure 13b: dynamic, 2800 samples

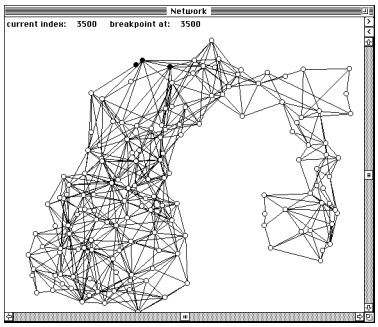

Figure 13c: dynamic, 3500 samples

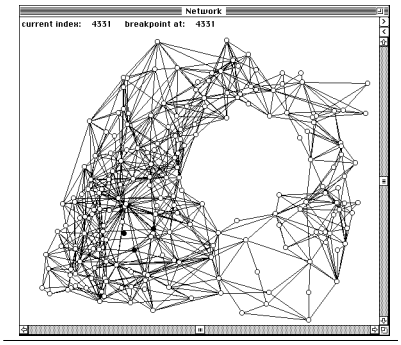

Figure 13d: dynamic, 4331 samples

ing the wall at the gap) are already removed. Due to the careful and smooth removal routines, the two worlds coexists for a certain time, until the world with the gap is completely "washed-out". The navigator may take advantage from the fact that the confidence values of the connections distinguish between most recent and established information.

4.6. Conclusions

As the reader may have expected, the choice of a world model will depend significantly on the task, but some guidelines may be derived from the discussion. One aspect is reliability, for example in a non-error-tolerant environment. If that is to be a central aspect of the robot-task, an exact model may be required to be able to plan safe paths. Similarly, if a guarantee of accuracy when following a path is needed, the exact geometric information may be necessary.

On the other hand, if the main focus is on simplicity, stability or qualitative aspects of the task, the qualitative topological map techniques may be the first choice. Especially the small requirements for computational effort and sensor equipment together with a high degree of robustness is an unique feature. The experiments have shown real world abilities offering sufficient information for navigation purposes (as demonstrated in chapter 5) as well as a stable self-localization method.

Chapter 5

Navigation

The navigation component is responsible for the planning and execution of goal driven manoeuvres, where the underlying world model is a qualitative topological (QT) map introduced in chapter 4, and the goals are situations out of the current QT-map. The task is attacked by two navigation layers, which refer to the path planning task and the problem of translating the planned path into adequate motor commands (path execution). These two layers operate alternating. In some other projects (e.g. [Hoppen91]) these layers are applied simultaneously, but this strategy requires a slowly or even never changing world model, otherwise most plans, generated in parallel to the execution phase are no longer valid, when the manoeuvring stops. Furthermore the planning phases are comparatively fast in the *ALICE* navigation module (see below). Therefore the costs for applying the layers alternately are not critical.

The chapter is divided into three sections, related to the navigation layers and some experiments respectively. While the path planning part is oriented on standard graph search heuristics, the path execution component introduces an example of direct sensor-actuator mappings without an explicit world model. The motor command generation is adaptive according to a reinforcement scheme, and equipped with a fast associative memory.

5.1. Path Planning

Given a certain goal situation, generating a couple of alternative plans reaching this goal is straight forward, because the QT-maps offer immediately graphs of potential ways. The existence and continuous adaptation of a path graph are a system-immanent benefit from the QT-techniques. The usual methods of constructing graphs on the base of extracted convex object boundaries, Voronoi spaces, triangular divisions, or a dozen others can completely be omitted here.

The formal base for the applied graph search heuristics is given in the article from Hart, Nilsson and Raphael from 1968 introducing the A*-algorithm [Hart68]. Due to the fact that two heuristics will be derived from this formal frame, the original A*-algorithm is introduced briefly.

Assuming a graph $G_{QT} = (N, L, g_c)$ given by the current QT-map N_{QT}, where $N = \{n\}$ is a set of nodes (given by the set of cells from N_{QT}), $L = \{l\}$ a set of connections, and g_c: $L \rightarrow \Re^+$ is a function attaching a real confidence value to each connection (determined by the adjacency function $a(\bullet, \bullet)$, see (7) and (8));
a situation $s \in N$ describing the current position of *ALICE*;
a situation $t \in N$ describing the target position;
an evaluation function f: $N \rightarrow \Re^+$ approximating the probability that a node n is part of the solution, where $\sum g_c$ along the path from n to t has a major impact on f (more specifically on h as introduced below);
an expansion function Γ: $N \rightarrow N^*$ giving all topologic neighbours of a node out of N
– then the A*-algorithm can be formulated according to the following steps:

(1) Initialize an "open"-set O of nodes with $\{s\}$; set a "closed"-set C of nodes to {}

(2) $\forall o \in O$ calculate $f(o)$

(3) Find a minimum $o_m \in O$, where $\forall o \in O: f(o_m) \leq f(o)$

(4) If $o_m = t$: terminate the algorithm (successful)

(5) "Expand" o_m: $O = O \cup (n \in \Gamma(o_m) \mid n \notin C)$; move o_m from O to C

(6) If $O = \{\}$: terminate the algorithm (unsuccessful)

(7) Continue with step 2

Usually the evaluation function f: $N \rightarrow \Re^+$ consists of two parts: $f(n) = g(n) + h(n)$, where $g(n)$ denotes the actual costs of the currently optimal path from s to n and $h(n)$ denotes an estimation of the costs for an optimal path from n to t. Thus the heuristic part of the algorithm concerns the function $h(n)$ only, which is based on approximated distances and the confidence function g_c.

Two trivial cases of heuristic search can be derived from the general frame of the A*-algorithm:

a. **Steepest Gradient (Hill Climbing)**
If the already passed distance is not considered for f, i.e. $\forall n \in N: g(n) = 0$, the planning process is driven by the estimated distance to the goal only. Therefore each "dead-end" in the network will be processed until the very end, before any alternative is considered.

b. **Parallel Search (Flooding)**
This is the counter strategy to steepest gradient. In this case $h(n)$ instead of $g(n)$ is set to zero for all nodes. Thus a wave of open nodes is spread out, originating from the starting point s. Due to the impression of a wave rolling of the network, this strategy is also called "flooding".

Hart et al. offer several optimal selections for $g(n)$ and $h(n)$, but due to the fact that even simple assurances, like the triangular inequality cannot be given in the case of QT-maps, any proof given in [Hart68] will fail here. The author is limited to the obvious approximations for $g(n)$ and $h(n)$, based on the geometric distance estimates and the confidence information attached to each connection, but weights control the influence of each function. Depending on the situation any superposition of hill climbing and flooding can be forced.

The strategy chosen in the case of *ALICE* is a continuous variation between both trivial strategies. The underlying idea is that hill climbing will find a solution in a very short time, if the current environment is open and wide. If some first attempts employing steepest gradient search will fail, the environment is obviously not that simple. Parallel search elements are introduced, with the continuous change of weights from hill climbing to flooding. While the heuristic search is still unsuccessful, the assumptions about the environments are changing from "open space" to a kind of labyrinth. This is the reason why the flooding strategy seems to be more promising as a heuristic search, where the main feedback for the planner is the number of unsuccessful search steps. Of course the time-measurement, which the movement of strategies is based on, is scaled according to the estimated overall distance to the goal.

This assumption based on the search time itself has proven reasonable in all test-environments, investigated in the simulations (see section 1.3).

5.2. Path Execution

In contrast to the rather obvious strategies and heuristics, employed in the planning phase, the path execution part introduces some complex restrictions. First, all the geometric positions given from the QT-map or the internal position of *ALICE* are estimates, valid in a fuzzy sense only. These estimates are sufficient for the generation of a locally consistent world model, but once explicit motor commands have to be given, in order to reach a given target, the fuzzy information have to be converted into crisp values considering additional information. Second, *ALICE* is subjected to strong drift effects. Therefore the generated motor commands should consider the expected drift along a certain path in advance. Finally, the connections between situations in the QT-map need not to indicate a potential, straight, and drivable path between them. The connections (and their confidence values) indicate the existence of a (formerly) drivable path only, i.e. the kind of the potential path (straight, curved, complex manoeuvring, etc.) is not described.

Each of these effects influences the selection of promising sets of motor commands, fulfilling the expectations of the planner. One might think of explicit models approximating each phenomena individually and

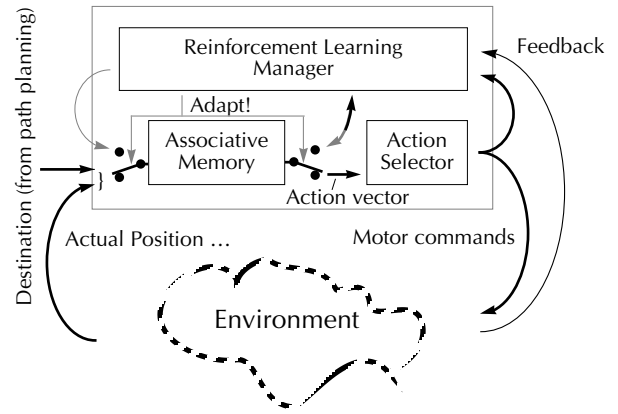

Figure 14: Reinforced learning

superimposing the results. Due to the sparse information such models are based on, and due to the philosophy of giving no explicit models, where these models are only weak assumptions about the expected world, the author has chosen another solution.

Reinforced Learning

Assuming that the selection of an adequate motor command is triggered by the current situation and the desired (sub-) goal only, an immediate mapping of this information to the actuator outputs is forced. The nature of this mapping depends (as introduced above) on a couple of unforeseeable und fuzzy influences. Thus it has to be adapted according to the current working environment and the condition of *ALICE* itself. The evaluating function, giving information about the degree of success for each selected motor command, is calculated on a sparse, but very certain and obvious measurement: the distance to the desired goal after executing the selected motor command. Of course, even this reading is fuzzy, but it is sufficient to support a positive trend, whenever commands are chosen, which are "near" to an optimal manoeuvre. According to this measurement for the "degree of success" of the previously executed motor command, the associations between current situations, desired goals, and proposed motor commands are slightly adapted.

This strategy is a member of the family of reinforcement learning techniques. The basic structure of the resulting path execution component is shown in figure 1. The associative memory module is built on the base of the supervised growing cell structures introduced in [Fritzke93b]. The output of the association process is a set of proposed motor commands together with probabilities of reaching the goal, when applying these actions. The following action selection module picks one of these proposals according to their probability, i.e. *not* always the command set with the highest probability. This technique avoids a straight gradient decent adaptation, following a promising path to a local minimum. Multiple potential solutions are always kept in mind, in case that the most promising solution gets stuck at a certain stage.

Learning Manager

The adaptation of associations as performed by the reinforcement learning manager, based in the feedback from the real experiments, offers another possibility for introducing variance in the trials. Instead of associating a range of possibilities, a history of experiments can be memorized in the learning manager, in order to avoid local minima. Therefore the association can be limited to one dedicated motor command and the action selection module can be skipped completely. This version promises faster learning, in case that a working method of local minima avoidance can be postulated. In any other case the additional effort for the more complex learning manager is questionable.

Performance Aspects

Learning speed is a critical aspect of any reinforcement learning scheme, due to the fact that the feedback is delayed by the physical experiment, and in case of a inadequate initialization of the associative memory module, a long trail of unsuccessful experiments can be forced, before a working solution is approximated. Two measurements attacking these problems are introduced in the *ALICE* project. First the associative memory is pre-learned on the base of a "noisy" world, but free of systematic errors, like drift effects, etc. This gives *ALICE* a chance of proposing actuator setting, which are close to working solutions from scratch on.

Then the selection of supervised growing cell structures as the implementation of the associative memory module, reduces the training times between physical experiments significantly (for example in comparison to any multi-layer backpropagation association system).

Neuro-Fuzzy-System

A further improvement would result in the introduction of a neuro-fuzzy system for the associations. The pre-learning phase could be substituted by a fuzzy system (including a fuzzy rule base, membership functions, a defuzzyfication module, etc.), able to express a wide range of a-priori known and useful driving strategies. In the second stage, the a-prior knowledge could be refined according to the actual experiences, by mapping the fuzzy system to an adaptive neural network model. The *SPIN*-Neuro-Fuzzy-Decision-System (*SPIN-NFDS*) as introduced in section 3.3 respectively [Zimmer93a], is a candidate for this task, but lacks two important features needed in the path execution component. First the adaptation times are too long, in order to be applicable in a closed-loop process. Second, the adaptation phase is limited in time, according to the "cooling down" of the learning constants (in contrast to the applied associative memory system). Any suggestion limiting the amount of training times of neuro-fuzzy-controllers results in strong restrictions for the underlying fuzzy rule system or the applied rules of inference respectively membership functions. The expressiveness of these limited fuzzy systems is not (yet) sufficient for the discussed task.

Some further heuristics, optimizing the reinforcement learning scheme can be found in [Fischer94].

The practical relevance of reinforcement learning schemes depends widely on the achieved adaptation time, until an acceptable approximation is found. Thus some preliminary experiments are discussed, indicating a positive impact of the chosen solution for the path execution task.

5.3. Experiment

The experiments shown in this section are completely performed in simulations. The employed simulator [Smidt94] includes a 2-d model of light distributions as well as a static drift model. Based on the rigid two dimensional simulator environment model, the test-scene (as shown in figure 2) has been constructed, including the following critical features. On the upper left, an asymmetrical narrow corridor prevents any success of straight or imprecise manoeuvres. The QT-model of this corridor offers in most cases only one situation for the whole corridor width, i.e. these cells represent both walls in one situation. On the lower left of figure 2 an narrowing corridor is introduced, interfering with correlation strategies, based on parallelism or other kinds of symmetry. The right wall is rough, in order to reduce the angular resolution of the whiskers. All other outer borders are straight and smooth. The obstacles on the right are of different sizes and equipped with rough border lines, preventing an easy recognition of situations around these objects. One of the three light sources illuminates the whole scene, whereas the others produce sharp shadows in some areas, for example at the asymmetrical corridor. The test-path for the following experiments starts between the obstacles on the right via the open space area through the corridor at the upper left and returns to the starting point. Thus the drift effects have to be considered carefully, when generating the actuator commands. Moreover the manoeuvres around some corners cannot be performed as a straight path.

The employed (static) QT-maps (see figure 3 for an example) are intentionally "bad" according to the evaluation criteria introduced in section 4.5 (large number of intersections, inconsistent areas (gaps) in the network, cell density too low in the corridor area, etc.). The worse-adapted QT-models limit the path execution component to a small working set of very specific actuator commands and the path planning part has to

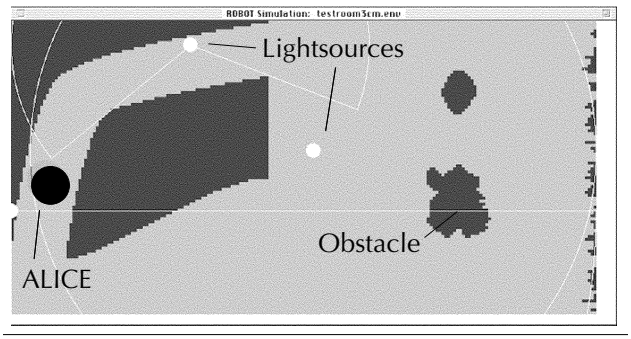

Figure 15: Test-Environment

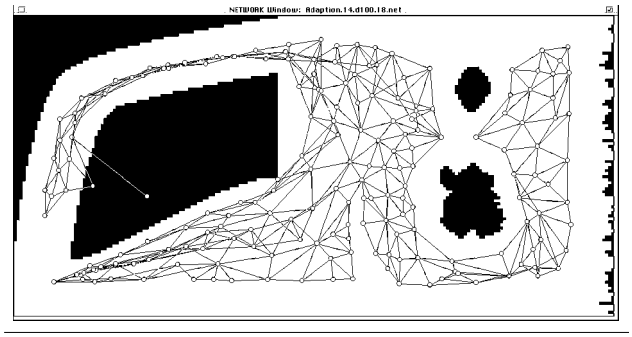

Figure 16: QT-map for the navigation experiments

refine its plans several times, especially in the corridor. All these introduced measurements are trying to imitate some (ugly) real world aspects, that have to be considered in the tests on the *ALICE*-platform.

Due to the fact, that these simulation experiments are performed as a general plausibility check only, the author is limited to qualitative results, i.e. the navigator performance cannot be measured in absolute, real world relevant terms here. Nevertheless the test conditions are quite hard, making some real world relevance plausible.

Drift Compensation

Up to a static drift rate of 5° per meter (even an order of magnitude worse than the drift effects on the *ALICE*-platform), the path execution component is able to compensate this effect completely after 3-5 test-runs.

Performance

The overall performance of the navigator is measured in terms of: the time needed to reach the target, the number of plan refinements, and the number of manoeuvres that must be reversed. The performance is measured at system start with the pre-trained association network and after four test-runs along the described path. The resulting mean improvement factors ranges from 2-3 regarding the time consumption, 3-4 regarding the needed number of refinements, and approximately 5 in terms of revised actions. The improvements are stable regarding the test configuration, but does not necessarily imply stability in a real world test. Finally it should be mentioned that four “training-epochs”, in order to achieve the shown results, indicate a very fast learning behaviour, considering general reinforcement systems.

Realtime abilities

The principal realtime abilities can be evaluated on the base of algorithmic complexities. The complexity of the planner does not allow any general realtime assurance, but due to the applied, simple graph search heuristics and the usually very redundant underlying QT-map, the absolute time spent for the planning phase is in most cases short enough to plan or refine a plan without stopping the platform. Of course, *ALICE* has to stop in case of any complex planning tasks, but this is a reasonable behaviour.

The more important aspect is the realtime ability of the path execution component, due to the fact, that this component has to run in parallel to the world modelling component and explicitly while the platform is moving. Fortunately, all employed structures of the (reinforcement-driven) path execution have a constant complexity, i.e. the association process itself, as well as the adaptation phase (which is limited to a fixed number of learning steps per training sample) fulfil this basic realtime prerequirement. Moreover the absolute, acquired computation time is (as suggested by the simulations) very small in comparison to the manoeuvring-speeds of *ALICE*.

Chapter 6 Exploration

The world modelling as introduced in chapter 4, integrates information gathered from *ALICE's* current environment. The collecting and presentation order of these real world samples influences significantly the stability and efficiency of the desired representation. All applied strategies, considering these aspects, are integrated in a module, called "exploration". Due to the fact that (at least) the exploration and world modelling process have to be applied in parallel, the exploration was in fact already involved in the tests discussed in section 4.5.

The introduced technique consists of a couple of superimposed strategies (or instincts), derived from the central exploration requirements. Some similarities to behaviour-based methods as discussed for example in [Mataric92] are obvious.

6.1. Requirements

The proposed qualitative, topological world model (QT-model) since it is build up continuously and supports livelong learning depends on an exploration, which has the following three requirements:

- **Efficiency**
  Knowledge should be accumulated **efficiently** in terms of speed, number of rotation or others. This requirement is of course true for all world-models. Due to the short-range sensor equipment of *ALICE*, every situation has to be explored by manoeuvring the platform into this situation. Thus the necessary number of movements is rather large, and the efficiency of the exploration strategy is getting even more important.
- **Redundancy**
  Each sensor-situation has to be stabilized using multiple sensor samples taken from different positions. To achieve stability, the gathered data must have a certain degree of **redundancy**.
- **Correlation**
  Due to the need for position- and orientation-**correlation**, the exploration must return the robot regularly to well-established regions of the network. The short range sensor equipment forces this "homing" behaviour of *ALICE*. Applying range finder devices, a "visual contact" to well known regions is sufficient for keeping correlation.

The last two points limit the amount of exploration-efficiency, which can be expected using the QT-approach.

The exploration strategies can be based on local information only, suggesting local manoeuvres ("**local exploration**"), or they can employ the whole world model together with the navigator, resulting in a "**global exploration**". Beside critical realtime aspects that have to be considered, applying global exploration, the

local nature of gathering redundant information for the QT-maps has led to a currently local exploration method, proven sufficient and efficient in all test environments. Nevertheless, further improvements in more complex respectively larger environments might be expected, extending the strategies by some explicit, global behaviours.

6.2. Instincts

From the basic requirements above, a couple of elementary "instincts" (simple manoeuvres, based on local information) can be derived and formulated on the base of the current sensor readings and the information from the currently active part of the QT-map. The notation for the network and the attached attributes as introduced in chapter 4 is applied in the following.

In the current version, only local information is used to determine a promising exploration direction. Therefore the current c_{opt} and its immediate neighbours c_j $(1 \le j \le m)$ are used to extract the following features from the actual (local) world model:

- Degrees of cell-confirmation u_{opt} and u_i
  This is applied as an indicator for the need of stabilizing.
- Number of neighbours of c_{opt}
  When the number of neighbours exceeds a certain limit, it is assumed, that *ALICE* is in a well explored area of the current environment. Additionally, the cell confirmations should be sufficiently high for this assumption.
- Best explored neighbour of c_{opt} with

$$c_{well} \in \{c_j\} \mid \forall j\colon u_{well} \ge u_j \tag{12}$$

  The direction of the best explored neighbour could be a good approximation, when the correlation stability has to be improved.

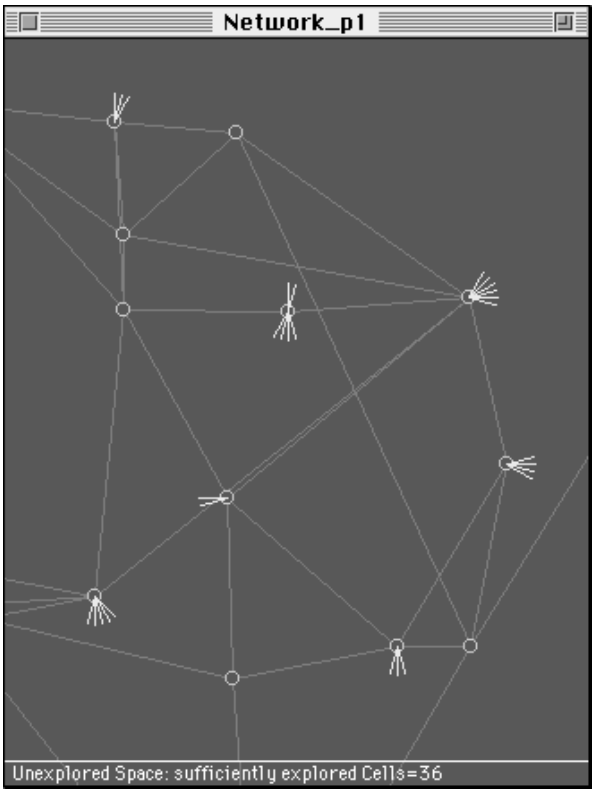

Figure 17: Exploration hints

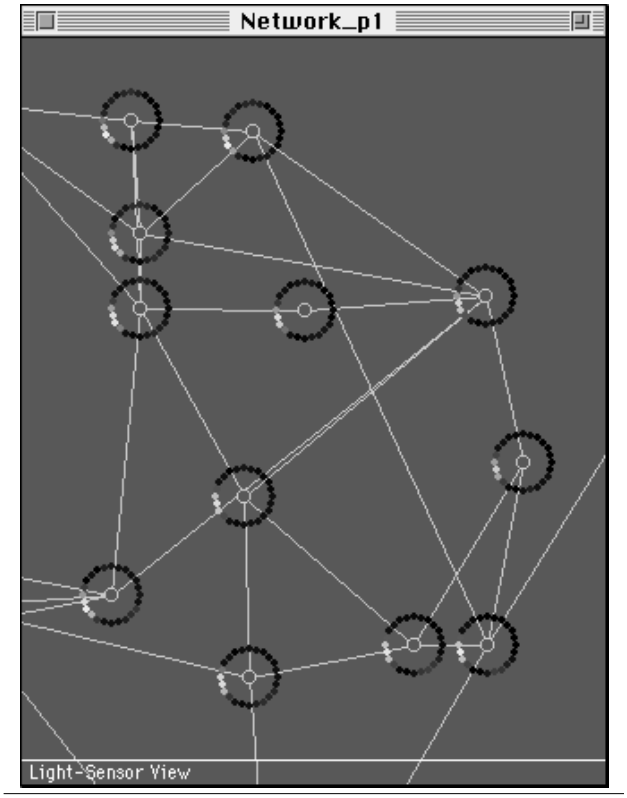

Figure 18: Topological map (light situations)

- Worst explored neighbour of c_{opt} (analogous to (1))

$$c_{worse} \in \{c_j\} \mid \forall j\colon u_{worse} \leq u_j \tag{13}$$

  The worst explored neighbour might indicate a need for stabilizing the world model itself.

- Unknown directions leading away from c_{opt}.
  These directions can be extracted from the topological structure and are memorized as a local attribute at c_{opt} (see figure 1).

Based on these features the following "instincts" are evaluated:

- **Stochastic** movements
  This instinct is applied, when any other instinct fails, or a certain degree of behaviour variance is needed.

- **Straight** movements
  Drive towards a random direction or a direction promising an open area, until an obstacle is touched. This strategy is dangerous regarding any stability consideration, but gathers new, previously unknown areas quickly.

- **Wall-following** (following an obstacle border)
  This strategy assumes, that the structures of the current environment can be sufficiently approximated by the situations near the obstacle borders.

- **Network-stabilizing**
  If there are more than τ_{sat} neighbours and the current c_{opt} is well explored, drive towards the *least explored neighbour*. This cell could be confirmed or rejected by this movement, but the world model itself is stabilized in any case.

- **Position-stabilizing**
  If the current c_{opt} is only less explored, drive towards the *best explored neighbour*. If the position- or orientation-correlation is getting less stable during the previous movements, this strategy promises an immediate or at least fast re-synchronization of *ALICE* and its world model.

- **Network-growing**
  If there are less than τ_{sat} neighbours try to *create a new neighbour*, by driving towards an unknown direction. This is a world model based "curiosity" function. In figure 1 the still unexplored directions from the net-fragment shown in figure 4 are indicated.

By using a kind of subsumption architecture, the several instincts are fused to produce a final exploration strategy (see for example [Asteroth92] for another application of subsumption methods). The resulting behaviour is a continuous variation of curiosity and stabilizing tendencies. *ALICE* is moved towards an unknown area for a certain time, until the unreliably of the internal position and the instability of the world model is getting critical. The resulting movements of *ALICE* appear like a continuous change between "taking heart" and "losing courage".

6.3. Conclusions

An explicit discussion of exploration experiments can be omitted here, due to the fact that all the test discussed in section 4.5, are performed on the base of the strategies introduced above. These tests have shown, that an unknown environment of 3 by 3 metres can be explored (i.e. a stable world model, usable for navigation purposes generated) in approximately 15 minutes. Keeping in mind that no range-measuring devices are employed in this experiment, this result is (in the opinion of the author) of some significance.