# Navigation of simulated mobile robots in the Webots environment

Richárd Szabó *

**Abstract**

Development of various robotic algorithms can be supported with the aid of robot simulation programs. We present the Webots mobile robot simulator and its applicability to handle machine-learning methods. Furthermore, we show advantages and disadvantages of different navigation paradigms. We focus our investigation on metric navigation and especially on the creation and the usage of occupancy grids. We show a working method on how to use occupancy grid to efficiently navigate in Webots.

**Key words:** mobile robotics, simulation, metric/topological navigation, cognitive map, occupancy grid.

## 1 Introduction

According to some scientifical forecasts robotics can be as important branch of life within reasonable time as it was the automotive industry in the 20th century.

To facilitate the spreading of the discipline engineers research new robot hardwares, while informaticians create new robot controlling softwares. The latter task cannot be imagined without reliable robot simulator environments. These tools may link up powerful algorithms and real-world tasks ([1]).

One of the most important goals of the mobile robot research is the emergence of the ability to efficiently navigate. Since the introduction of artificial landmarks, lights, or characteristic floor map, and other modifications of the environment to ease rapid movement may be too expensive, the algorithms controlling the robot have to be better and better equipped to changing conditions.

In our investigation, we implement robot control algorithms in a simulator environment using the method of metric navigation. The general path of our research follows the ideas described by Thrun ([2]).

## 2 The simulation environment

Webots is a three-dimensional mobile robot simulator ([3]).

The description of the simulation environment based on Virtual Reality Modelling Language, some extensions are introduced for the specific simulation domain. The

*Eötvös Loránd University, Department of General Computer Science, 1117 Budapest, Pázmány P. s. 1/D, and Department of History and Philosophy of Science 1117, Pázmány P. s. 1. e-mail: ricks@inf.elte.hu

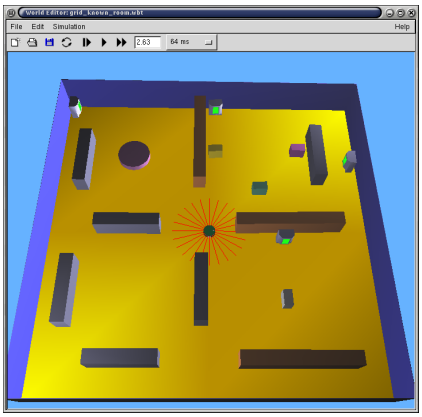

Figure 1: Webots simulation program

appearing robots can be any type of two-wheel differential steering robots which means they are equipped with two indepent motors. Some of the predefined possibilities are Khepera, Koala, Magellan, Pioneer2.

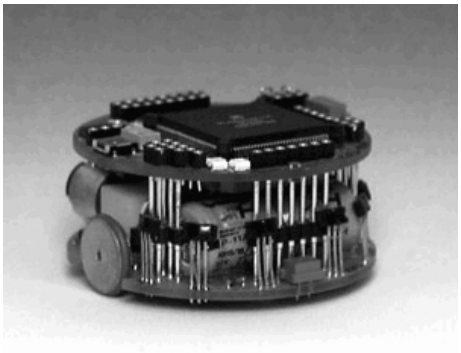

Figure 2: Khepera robot

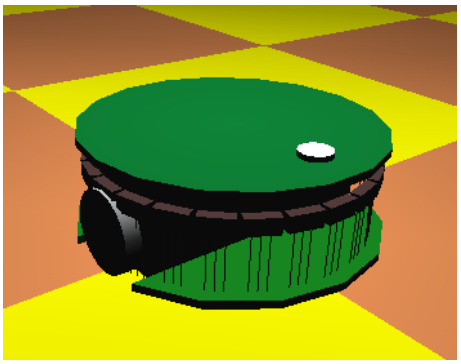

Figure 3: Simulated Khepera robot

The flexibility in the definition of the environment and the robot results that almost every type of mobile robotic task can be evaluated in the Webots.

Robot movement follows the laws of kinematics, dynamics – namely forces – are not handled. In spite of this simplification the movement of the robots has a close resemblance of the real world equivalent. To ameliorate the compatibility, noise is introduced in the sensation, and in the action as well. As a consequence of the above it is possible to control real robots from the simulator.

The control process is mainly influenced by the number, the modality, the position, and the range of sensors. Webots can handle infra-red, laser, or sonar distance sensors, light sensors, touch sensors, and color video cameras ([4]).

# 3 Navigation methods

One of the most important goals of the mobile robot research is the emergence of the ability to efficiently navigate in complex, cluttered, and unknown environments. Without navigation the creation of self-propelled, household machines, guard robots, or planet surveyors is beyond imagination.

To perform this task, momentary, local information collected by the sensors of the robot is not enough. The robot has to create a cognitive map — an acceptable projection of the environment as an inner representation — which can be used for decisioning. The cognitive map is not necessarily a map in a cartographic sense, rather it is a collection of spatial relations, angles, distances, environmental characteristics, and landmarks. Many species of animals — mammals, birds, fishes, bees, or ants, and naturally humans — use different types of cognitive maps ([5]). Used models of cognitive maps can be divided into two major groups: metric navigation and topological navigation ([6], [7]).

Metric, geometric, or grid-based navigation, or survey mapping works with metric spatial properties of objects like distance, angle, and coordinates according to an imagined coordinate system. The resulting map truly can be a "view from above", a proportional map of the environment. During the creation of a metric map the surveyor should be aware of its actual position and all the objects are placed on the map according to this information.

Topological navigation handles topological relations of objects. It focuses on the links between landmarks, the possibility to move from one place to another. The resulting map in this case is a graph of important positions and routes connecting them. The surveyor should recognize important places — the graph nodes — via landmarks.

Table 1 shows advantages and disadvantages of these navigation methods.

Table 1: Properties of navigation methods

| **Metric** | **Topological** |
|---|---|
| + easy to create and maintain | – hard to create and maintain |
| + robust to ambiguous senses | – sensible to ambiguous senses |
| – does not follow world properties | + can adapt to world characteristics |
| – needs large memory | + needs small memory |
| – hard to use for navigation tasks | + easy/efficient for navigation tasks |
| – needs knowledge of exact robot position | + knowledge of exact robot position is unimportant |

An example of topological navigation is presented in [8] and [9], where the authors create a purely vision-based representation of the open environment with a panorama camera. Since the camera has a full view angle non-occluded landmarks can be tracked at every pose of the robot. Displacement in the environment causes contraction of objects in one direction while expansion of objects in the other. This geometrical property makes possible the movement to the goal direction in a small region.

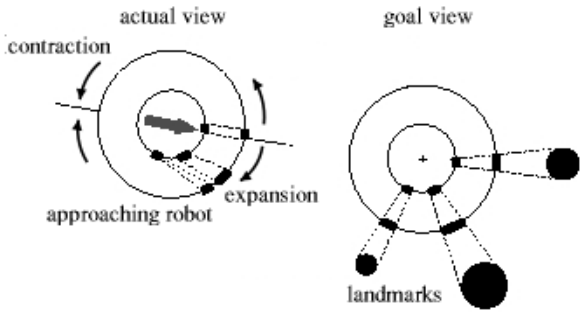


Figure 4: View of the panorama camera

At the borders of these small regions, to avoid ambiguous place-recognition and the lack of landmarks because of occlusion it is necessary to take photos from time to time. These photos are placed in the view-graph nodes while the edges of the graph are determined by the possible displacements.

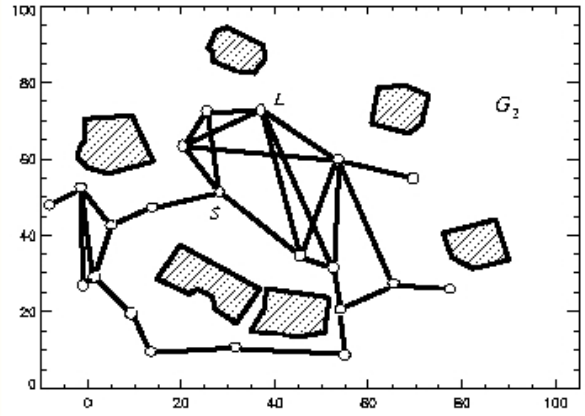


Figure 5: View graph under construction

Example of metric navigation will be shown below, for further reading see [10].

# 4 Our work

Our goal was to create a metric navigation module for a modified Khepera robot in the Webots simulation environment. The developed robot has to build a cognitive map —

"a view from above" — of a square-shaped room in the size of a few square meters while it visits every location.

The selected Khepera robot has a cylindrical shape of 55 mm in diameter and 3 cm in height. The original Khepera has 8 infra-red distance sensors with the sensing range of 2-4 cm depending on the color and texture of the sensed object. In our experiment the robot has 24 sonar distance sensors with the sensing range of 15 cm. These modifications facilitate the fast creation of an accurate map.

Usage of distance sensors has practical reasons. It is much less expensive than video cameras both literally and algorithmically. While digital camera images can deploy more information, the image processing is a significantly more complex task because of the dimensionality of the source.

The selected method of metric navigation is based on the occupancy grid ([11], [12]). This general structure in two dimension manages a tesselation of the plane in cells. Each cell of the occupancy grid contains a probability value which is an estimation that the represented position is occupied by some object. During the navigation process the robot has to continuously calculate the occupancy values of its neighbourhood and has to insert the result into the grid.

The important steps of the map building, in accordance with the work of [2], are the following:

- sensor interpretation
- integration over time
- pose estimation
- global grid building
- exploration

## 4.1 Sensor interpretation

The sensor interpretation is the first phase in the creation of an occupancy grid based navigation.

The 24, circularly placed sonar sensors are fairly dense source of information on the object distance. These items make possible the calculation of occupancy values in a small environment. The calculation is necessarily a transformation of sonar range scans to a set of occupancy grid cells, a translation of 24 scalars to a two dimensional grid of probabilities. That is to say the conditional probability of every grid cell is computed in the vicinity as

$$p(occ_{x,y}|s),$$

where $s$ is a sonar measurement of cell $(x, y)$.

Instead of using an artificial neural network to solve the problem at hand, like in [2], we used a simpler, yet efficient method to estimate. For every sensor the returned value is proportional with the distance of an object in the scan direction. Closer to the

robot on the line of the scan the probability that a grid cell is occupied is small ($\varepsilon$), at the position of the value it is large ($1 - \varepsilon$), and after the scan it decreases linearly to complete uncertainty ($0.5$).

Since the main goal of the grid is to find traversable corridors on the map for the robot it seems natural to broaden the evaluation of the scan on both sides with at most the size of the robot. This enhancement gives a better knowledge of the world from our point of view, because the number of modified cells is increased, leaving less uncertainity locally, as it can be compared in Figure 6 and Figure 7.

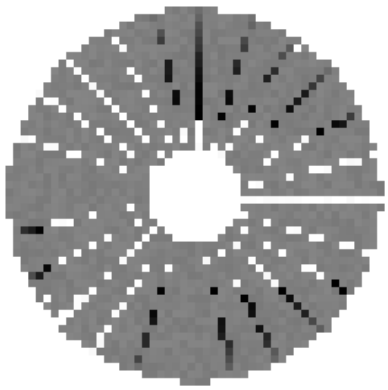

Figure 6: Local grid with narrow environment

Figure 7: Local grid with wide environment

## 4.2 Integration over time

The whole occupancy map of the environment is created during an exploration. This means that several sensor measurements are processed until the end, and several measurements are related to the same grid cell as well. Since circumstances (like pose of the robot, noise, etc.) influence the sensation, different values have to be integrated for every cell. So the conditional probabilities based on sensor interpretations $p\left(occ_{x,y}|s^{(t)}\right)$ of every time step $t$ have to be used to calculate the conditional probability of all measurements, $p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}\right)$. This integration process can be solved by applying Bayes theorem ([13]) if we use the assumption that, given the true occupancy value of a cell, sensations are independent of the time of their collections: $p\left(s^{(t)}|occ_{x,y}\right)$ is independent of $p\left(s^{(t')}|occ_{x,y}\right)$, if $t \neq t'$.

Having the preconditions above we can state that

$$\Pi(x, y, T) = p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}\right) =$$

$$1 - \left(1 + \frac{p(occ_{x,y}|s^{(1)})}{1 - p(occ_{x,y}|s^{(1)})} \prod_{\tau=2}^{T} \frac{p(occ_{x,y}|s^{(\tau)})}{1 - p(occ_{x,y}|s^{(\tau)})} \frac{1 - p(occ_{x,y})}{p(occ_{x,y})}\right)^{-1} \tag{1}$$

where $p(occ_{x,y})$ is the prior probability which if set to $0.5$ the last fraction can be omitted.

Theorem (1) follows from the next proof, using Bayes theorem

$$\frac{\Pi(x,y,T)}{1 - \Pi(x,y,T)} =$$

$$\frac{p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}\right)}{p\left(\neg occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}\right)} =$$

$$\frac{p\left(s^{(T)}|occ_{x,y}, s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}{p\left(s^{(T)}|\neg occ_{x,y}, s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)} \cdot \frac{p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}{p\left(\neg occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}.$$

Using the assumption of indepence of time is equal to

$$\frac{p\left(s^{(T)}|occ_{x,y}\right)}{p\left(s^{(T)}|\neg occ_{x,y}\right)} \cdot \frac{p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}{p\left(\neg occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}.$$

Using Bayes theorem again on the first fraction

$$\frac{p\left(occ_{x,y}|s^{(T)}\right)}{p\left(\neg occ_{x,y}|s^{(T)}\right)} \cdot \frac{p(\neg occ_{x,y})}{p(occ_{x,y})} \cdot \frac{p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}{p\left(\neg occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T-1)}\right)}.$$

Now we reduced the number of sensations on the right side by one, so after induction we get

$$\frac{p\left(occ_{x,y}|s^{(1)}\right)}{1 - p\left(occ_{x,y}|s^{(1)}\right)} \cdot \prod_{\tau=2}^{T} \left(\frac{p\left(occ_{x,y}|s^{(\tau)}\right)}{1 - p\left(occ_{x,y}|s^{(\tau)}\right)} \cdot \frac{1 - p\left(occ_{x,y}\right)}{p\left(occ_{x,y}\right)}\right),$$

and after a rearrangement we get equation (1).

The main consequence of equation (1) is that incremental calculation of occupancy grid values for every cell is possible, sonar scans can be concatenated to previous experiences.

## 4.3 Pose estimation

As the robot moves around in the environment it builds the global occupancy grid employing the local occupancy grid. For merging local grid information into the global map the robot needs to know its position and orientation, alias pose, at least approximately. Since sonar sensor scans are not informative enough to determine the robot pose unambiguously another solution has to be found. If the robot continuously tracks the changes of its pose it can estimate the actual state relative to the starting state. This method is called odometry.

Table 2: Sources of errors in odometry

| **Systematic** | **Non-systematic** |
|---|---|
| – unequal wheel diameters | – uneven floor |
| – irregular wheels | – slippage of wheels |
| – encoder sampling rate limitations | – interaction with objects |
| – encoder resolution limitations | |

In our case wheel encoders of the robot accumulate the movement of each wheels in radians. As the structure parameters of the robot, like axle length and the wheel radius, are given, using equations of coordinate geometry the actual state of the robot can be calculated.

The problem of this approach is that systematic and non-systematic errors may influence the navigation and the calculated pose differs from the real one ([14], [15]). Table 2 shows some sources of odometry errors.

Systematic errors can be compensated after a deep investigation of the working robot, but non-systematic errors cannot be eliminated. As time goes odometry errors are accumulating and may distort pose estimation to a level where they are useless.

Figure 8 shows how the real and the calculated position of the robot diverge with time, because of odometric errors.

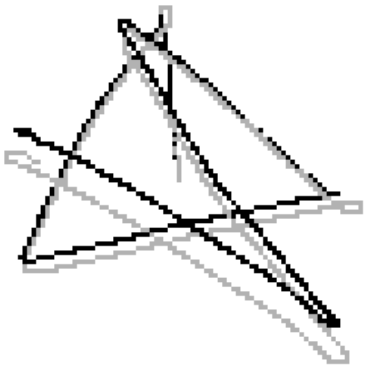

Figure 8: Real and calculated trajectories

Figure 9 displays quantitatively the difference. Error in orientation alternates frequently and it reaches 3 degrees. Error in position increases faster in the direction of the movement and it approaches $10\%$ of the route travelled.

For possible methods of dealing with odometric errors see section 5.

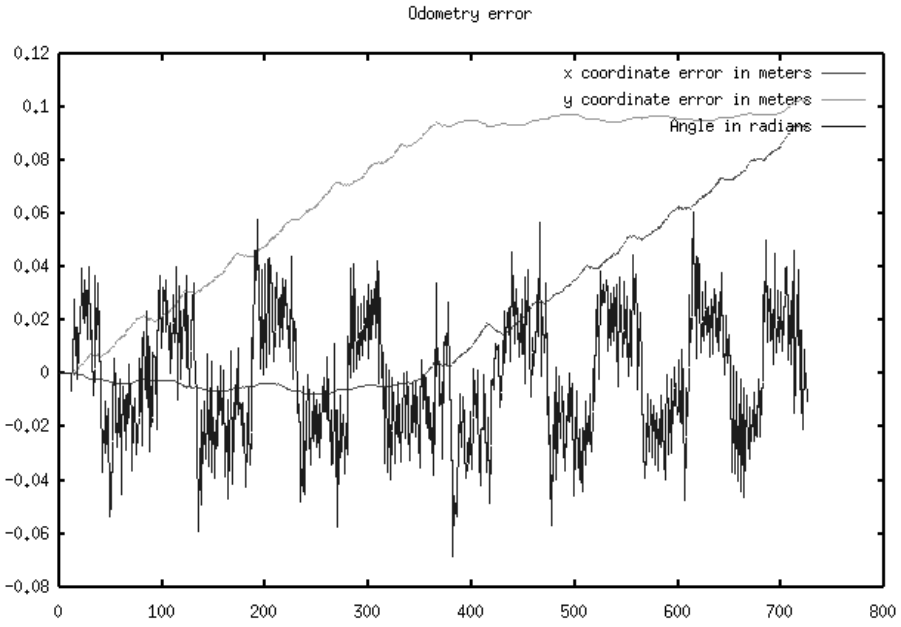


Figure 9: Graph of the odometry errors by coordinate

## 4.4 Global grid building

When the robot can estimate its pose at least approximately local occupancy information needs to be merged into a global map. On one hand this process means the transformation of polar coordinates to Cartesian. On the other hand it is the moment of the sensor integration. As the robot moves around in the environment the revealed territory is growing until a complete discovery as it can be seen in the open environment of Figure 10 and in the maze of Figure 11.

## 4.5 Exploration

After the robot is ready to create a map of its environment, a driving force is needed to urge the robot to explore all the reachable places, otherwise it would wander randomly.

For this reason a variant of value iteration is implemented ([16]). The selected algorithm helps to find the minimum cost-path to unexplored regions of the occupancy grid. A newly introduced cost matrix is calculated iteratively and after convergence for every occupancy grid cell the cost of travelling to an unexplored grid cell from the actual cell is given.

1. The first step of the method is the initialization. Naturally the cost of unexplored

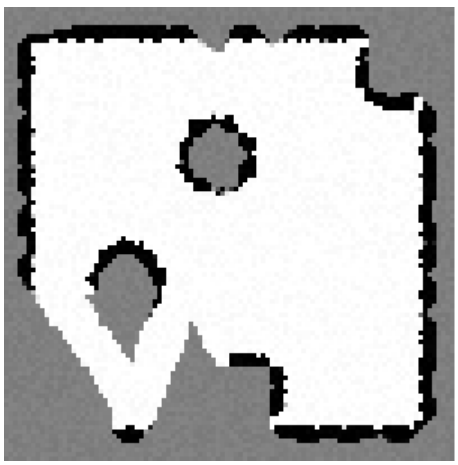

Figure 10: Global occupancy grid of an open area

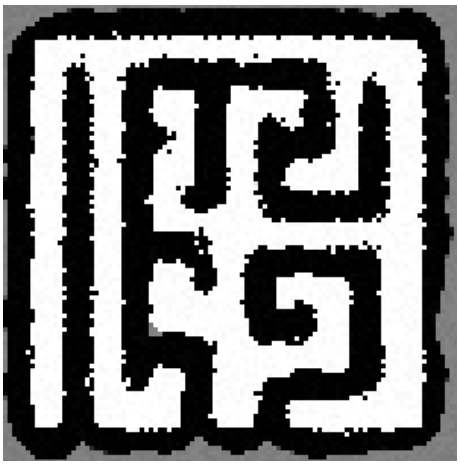

Figure 11: Global occupancy grid of a maze

cells – where occupancy value has not been changed – are 0, while the cost of explored cells is $\infty$.

$$V_{x,y} \leftarrow \begin{cases} 0 & \text{if } (x,y) \text{ is unexplored} \\ \infty & \text{if } (x,y) \text{ is explored} \end{cases}$$

2. During the update loop the value of every explored cell is recalculated. Cells with high occupancy probability get 1 as cost. For others the modification is based on a minimum search in the vicinity, the cost together with the probability of occupancy of the neighbouring cells determine the new value of the investigated cell. The $\varepsilon$ component is necessary to punish the length of the path.

   $\forall$ explored $(x,y)$:

$$V_{x,y} \leftarrow \begin{cases} 1 & \text{if } p\left(occ_{x+i,y+j}\right) > 1-\delta \\ min(1, \min\limits_{\substack{i=-1,0,1 \\ j=-1,0,1}} \{V_{x+i,y+j} + p\left(occ_{x+i,y+j}\right)\} + \varepsilon) & \text{otherwise} \end{cases}$$

   After convergence the cumulative cost of travelling to an unexplored cell is in the matrix for every explored cell.

3. Since the exploration in one region of the environment affects the whole cost matrix a total update of the costs would be necessary every step. With a large number of cells it may make the exploration process extremely slow.

   This is the main reason why total update is not performed continuously. On the other hand regular calculations are necessary to ensure the determination of new exploration direction.

Exploration direction is a consequence of the actual state of the cost matrix. Since matrix cells contain cumulative travelling costs a minimum search is appropriate to find the way to unexplored regions. This procedure has to work in a small environment of the robot, in the interior of a bounding box to avoid the discovery of a cell which has no linear connection with the actual robot position.

Actual orientation of the robot is also taken into account in the selection of new exploration direction to help the generation of a smooth movement.

The evolved exploration strategy is greedy in the sense that it always makes the robot moving straight in the direction of unexplored regions. Normally the environment contains some obstacles which influence the trajectory. In our navigation algorithm front sonar sensors have an important effect on the movement direction since close objects force the robot to turn away. The complementary behaviours of minimum cost search and obstacle avoidance together enables an appropriate navigation.

Figure 12 and Figure 13 shows the cost matrices evolved in the same environments with the occupancy grids in Figure 10 and Figure 11.

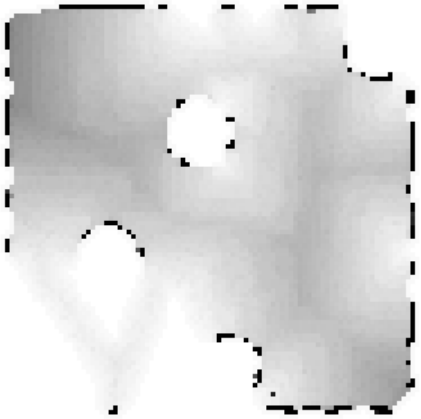

Figure 12: Cost matrix of the open area

Figure 13: Cost matrix of the maze

# 5 Odometric error correction

Various algorithms exist to solve the “chicken-egg” problem of map building and self localization ([17]).

Schiele and Crowley employ a Hough transform on the occupancy grid to extract line segments ([18]). The segments of the recent sensation are matched against the global map. Results of the matching process can be used as parameters of a Kalman filtering technique.

Lu and Milios extend the traditional Kalman method in their research ([19]). They store the history of laser range scans and create a network of pose estimations. The edges of the network defined by odometry and alignments of scan pairs.

Another approach of correcting the odometric error is the expectation maximization algorithm ([13], [20]). This iterative maximum likelihood process has two main steps. In the expectation step the robot pose is recalculated given the actual map. In the maximization step the most likely map is estimated given the current position of the robot.

# 6 Dynamic environments

General robot-working environments change over time. Some objects are so-called "semi-dynamic", they do not change their location only their position like doors, windows. All other objects like people, chairs, desks move in the environment every now and then.

Most of the navigation methods do not support dynamic environments, while occupancy grids handle them only to a limited extent. As it is described in [21] the "integration over time" step of the occupancy grid building ensures that occupancy changes are integrated into the grid.

Our approach has a simple extension to cope better with dynamism: cell probabilities may be influenced more by recent sensor readings than by earlier ones. Introducing a $0 \leq \gamma \leq 1$ decay factor in equation (1) serves this need.

$$\Pi(x,y,T) = p\left(occ_{x,y}|s^{(1)}, s^{(2)}, \ldots, s^{(T)}\right) =$$

$$1 - \left(1 + \gamma^{T-1}\frac{p(occ_{x,y}|s^{(1)})}{1-p(occ_{x,y}|s^{(1)})}\prod_{\tau=2}^{T}\gamma^{T-\tau}\frac{p(occ_{x,y}|s^{(\tau)})}{1-p(occ_{x,y}|s^{(\tau)})}\frac{1-p(occ_{x,y})}{p(occ_{x,y})}\right)^{-1}$$

A more sophisticated method is presented in [22]. The authors create the robot object mapping algorithm (ROMA), which builds a model level above the grid with the characteristic shape of objects. Moving people are not modelled hence the basic assumption is that objects are changing location so slowly that they can be assumed static during an occupancy grid building. In the preprocessing step of the algorithm the robot tries to estimate the number of non-static objects using occupancy grids created at different times with the aid of well-known algorithms of image processing. Having the dynamic objects an expectation maximization procedure determines the model of the objects in the changing grid.

# 7 Conclusions

During our research we created a metric navigation method in the simulation environment of Webots. The presented occupancy grid model resembles the work of Thrun ([2]) with the following modifications.

We use a simpler, yet effective sensor interpretation with broadened scan lines. Odometry of the simulated Khepera robot is determined for later usage for error correction. The value iteration rule of the exploration phase was modified. Instead of a selective reset phase and a prioritized sweeping we use occasional total updates.

The evolved robot controller was tested in three different environments: in an open area (Figure 10), in a maze (Figure 11), and in an office-like room (Figure 1). The open area is 1 $m^2$, while the other two are 2.25 $m^2$. The robot could explore all the three environments. Naturally the open area caused the slightest challenge: full explorations took 8 minutes on an average. Solving the maze and the office was harder, the robot was navigating for 22 and 20 minutes respectively on an average.

# 8 Future work

There are some natural continuations of the research.

- Error correction of position estimation is important.
- Since complete knowledge of the environment is only a tool to perform higher level tasks, after exploration the robot should start a new behavior.
- Modelling dynamic objects would also be an advantage.
- Combination of metric and topological navigation may bring together the desired proprieties of both methods. Building a topological graph as a new control level above an occupancy grid may reduce algoritmic complexity.
- Integration of new sensor types, especially video cameras, may cause a faster and more accurate environment mapping.

**Acknowledgement**
The author wishes to thank György Kampis for his useful suggestions.

# References

[1] R. Szabó. *Mobil robotok szimulációja*. Eötvös Kiadó, 2001.

[2] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[3] Cyberbotics S.a r.l. *Webots 3.2. User Guide*, 2002.

[4] Cyberbotics S.a r.l. *Webots 3.2. Reference Manual*, 2002.

[5] V. Csányi. *Etológia*. Nemzeti Tankönyvkiadó Rt., 1994.

[6] O. Trullier and J.-A. Meyer. Biomimetic navigation models and strategies in animats. *AI Communications*, 10(2):79–92, 1997.

[7] M. Franz and H. Mallot. Biomimetic robot navigation. *Robotics and Autonomous Systems*, 30:133–153, 2000.

[8] M. O. Franz, B. Schölkopf, P. Georg, H. A. Mallot, and H. H. Bülthoff. Learning view graphs for robot navigation. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 138–147, New York, 5–8, 1997. ACM Press.

[9] H. Mallot, H. Bulthoff, P. Georg, B. Scholkopf, and K. Yasuhara. View–based cognitive map learning by an autonomous robot. In F. Fogelman-Souli'e and P. Gallinari, editors, *Proceedings of ICANN'95*, volume II, pages 381–386, 1995.

[10] I. Ulrich, F. Mondada, and J. Nicoud. Autonomous vacuum cleaner. *Robotics and Autonomous Systems*, 19(3-4):233–245, 1997.

[11] A. Elfes. Using occupancy grids for mobile robot perception. *Computer*, 22(6):46–57, 1989.

[12] S. Thrun. Learning occupancy grids with forward models. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS'2001)*, 2001.

[13] T. M. Mitchell. *Machine learning*. McGraw Hill, 1997.

[14] J. Borenstein and L. Feng. Correction of systematic odometry errors in mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 12, 1996.

[15] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning - sensors and techniques. *Journal of Robotic Systems*, 14(4):231–249, 1997.

[16] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. Bradford Book, MIT Press, 1998.

[17] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.

[18] B. Schiele and J. L. Crowley. A comparison of position estimation techniques using occupancy grids. In *1994 IEEE International Conference on Robotics and Automation*, pages 1628–1634, 1994.

[19] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[20] S. Thrun, Burgard W, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots*, 31(5):1–25, 1998.

[21] B. Yamauchi and P. Langley. Place recognition in dynamic environments. *Journal of Robotic Systems*, 14(2), 1997.

[22] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2002.