DÉPARTEMENT D'INFORMATIQUE
LABORATOIRE DE MICROINFORMATIQUE
LAMI - DI INF
CH- 1015 LAUSANNE

Philip Mächler Email: philip.maechler@epfl.ch
%021 - 693 39 07 Téléfax 021 - 693 52 63

Submitted to IAV '98 Madrid, Spain

WITHOUT A CLUE:
UNSUPERVISED ROBOT NAVIGATION WITH UNKNOWN SENSOR CONFIGURATION

Philip Mächler (philip.maechler@epfl.ch)

LAMI-DI, EPFL Swiss Federal Institute of Technology, Lausanne
Tel: ++41 21 693 39 07 Fax: ++41 21 693 52 63

Abstract: The aim of this research project is to perform robot localization and navigation without prior information about the environment. A method for the interpretation of different kinds of unknown but significant sensor signals as "landmarks" is proposed. These landmarks are used to create a topological map containing navigation information to get from one landmark to another. A navigator steers the robot toward "learned" landmarks to synchronize the actual robot position with the position previously associated to the landmark. The results show that a robot can learn to stabilize its odometry error without any prior information about the environment or its sensor ability. It extracts only adequate environment features which make the algorithm suitable to new and unknown surroundings.

Keywords: Autonomous mobile robots, navigation, position estimation, classification.

1 INSPIRATION

The position and direction determination of an autonomous robot is done in many different ways. An often used method is to detect and recognize predefined structures (landmarks) in the environment, used for deducing position and orientation. The characteristics of the landmarks and the strategy to detect them are often known in advance and directly implemented into the algorithm. A "run time" adaptation can only be done in a limited manner by adjusting parameters compensating some foreseen influences like the change of ambient light or distorted shape of the landmark. Nevertheless, the algorithm remains fixed on the pre-programmed landmark characteristics and the strategy to find them.

1.1 Adequate concepts

A real outdoor environment offers a huge quantity of information like colors, light, direction, distance, etc. Human beings and superior animals are able to process this flood of information and abstract *concepts* like door, road and object, which are used to handle the environment on a higher level. Thus, the concepts depend strongly on the task and the sensor abilities and are therefore different for humans and robots.

1.2 Use of concepts in different layers

The application of concepts is not new and many recognition algorithms (specially in Artificial Intelligence) leave the creation of concepts to the machine in order to increase the flexibility and to reduce prewired and therefore unsuitable influences by the designer. Automatic concept creation is normally applied on a quite high abstraction level for example to recognize objects or situations, based on sensor and status information which are preprocessed in a more conventional manner. Such preprocessing always leads to a loss of (probably important) information reducing the power of automatic concept creation.

The purpose of this paper is to tackle this problem by moving the automatic concept creator as close as possible to the raw sensor signals, see also [Kuipers and Pierce, 1997]. In an idealistic case, such a system is completely independent of the type of sensors, because any restrictive preprocessing is suppressed. Every kind of statistically rare event perceivable by the sensors can be used as a landmark, developing a very high adaptability of the autonomous robot system to the environment. We want to investigate the limitations and how close an experiment can approach this theoretical case by applying only one but very simple concept in a experiment.

2 DEFINITION OF THE EXPERIMENT

This chapter gives a overview of the structure and algorithm used to carry out the experiment.

2.1 Aim

A robot moves randomly in an unknown static environment, repelled by obstacles and recording information from its sensors in order to recognize significant signals as landmarks. The robot has no prior knowledge about the environment nor the type and character of its sensors. The robot should learn to recognize significant signals as landmarks, store them into a memory and compare them with new landmarks in order to recognize again familiar situations and positions.

The experiment shows that a robot is able to stabilize the odometry error allowing continuous movement with a constant precision. Moreover, the robot can use its own self-discovered landmarks to create a topological map, allowing it to navigate in an unknown environment.

2.2 Experience set-up

We use a Khepera simulator written by Olivier Michel [Michel, 1996] to test the algorithm in its first state. In the future a real Khepera robot (see fig. 2-1) will be used to prove the feasibility of the experiment. The simulated robot is equipped with eight short distance sensors (about 5 cm range), eight ambient light sensors, a compass and a linear camera which reads a horizontal line of 64 pixels corresponding to 36 degree in front of the robot. This can be used to recognize obstacles and wall contours in front of the robot.

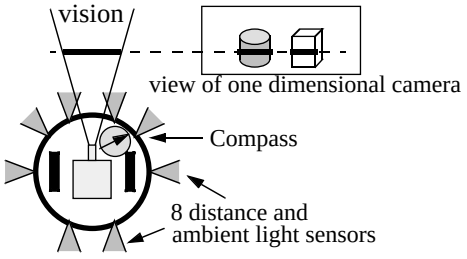

Fig. 2-1: Equipment of the extended Khepera robot

The environment consists of a field of about 15 x 15 times the size of the robot, containing obstacles and light sources.

2.3 Algorithm overview

The signals coming from different sensors are partially mixed together and directed into several unsupervised classifiers, creating *feature* regions of the input space (see fig. 2-2). All classifiers are neural networks but of different types like ART or LVQ producing different feature distributions. These features are analyzed by the statistics module, which selects only rare and “usable” features and stores them with the estimated robot position. These entries are compared with new incoming features. In case of a sufficient match, the robot position will be corrected to the position previously stored with the landmark.

3 NORMALIZATION

Our aim is to use raw sensor signals and avoid signal preprocessing as far as possible. We nevertheless have to normalize some signals as described below.

3.1 Distance and ambient light sensors

Because of the poor quality of the distance and ambient light sensors, a simple two state value is used to distinguish far and close obstacles or bright and dark regions. The threshold is selected in an empirical manner.

3.2 Camera

The image of the camera was simulated and calculated by a pseudo reflectance image model [Song and Choi, 1996], assuming a diffuse light source and Lambertian object surface (matte surface, no texture).

$$c \approx \frac{r \; a \; \cos\alpha}{d^2}$$

d Object distance
a Ambient light
r Object reflectivity
α Angle surface-camera
c Light intensity

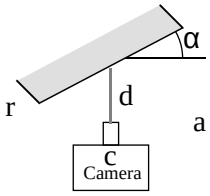

Fig. 3-1: Calculation of light intensity, first approximation

The measured light intensity (c) can be approximated by the equation in fig. 3-1, using as variables object distance (d), ambient light (a), object reflectivity (r) and the angle of the surface to the camera (α). Each object transition causes a strong change in d and α, which is very suitable for edge detection.

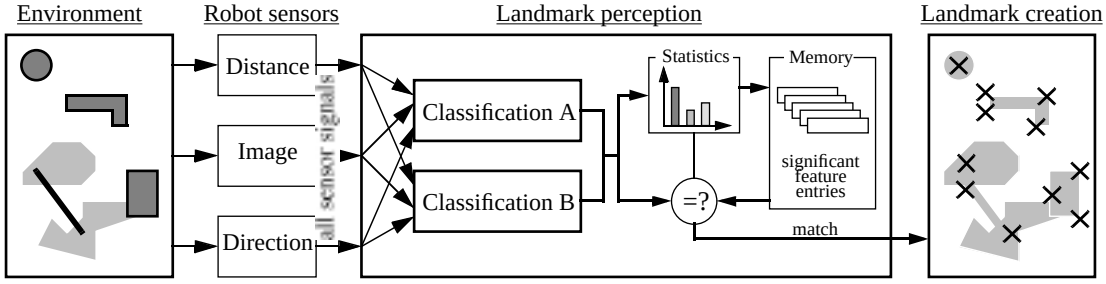

Fig. 2-2: Diagram of the sensor signal data processing and recognition of significant features

Edge detection is done by a discrete double spatial derivation and normalization of the obtained image (see fig. 3-2). After that, the image is stretched until the extreme edges reach the border of the image. This conversion makes the image stable against rotation and any kind of movement, as long as the same obstacles (contours) are visible to the camera.

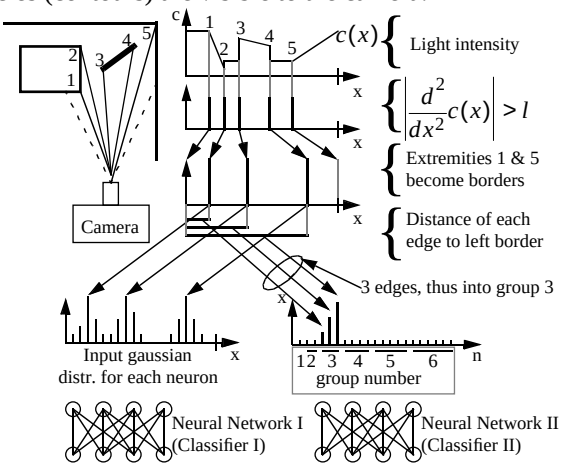

Fig. 3-2: Image processing of the camera view: Double derivation, normalization and two different methods for preparing the data for the classifiers

The result is prepared in two different ways to feed two neural networks (NN). The NN-A gets the position of the edges, filtered by a gaussian to smooth out minor position changes of the contours. The input neurons of NN-B are grouped according to the number of observed edges. Only one group can be activated at the same time. The individual neuron activity depends on the distance of the corresponding edge to the left wall. This preparation makes the NN-B very sensible to a change in the number of edges, which is often neglected by the NN-A if the edges are close to each other.

4 CLASSIFICATION

In order to recognize unknown and rare events, unsupervised classifiers like neural network must take care of small classes with very few entries. Such classifiers must neither forget classes nor combine them with each other. Two unsupervised classifiers are used in this project up to now. See also [Hertz et al., 1991] for a complete overview.

4.1 ART

The Adaptive Resonance Theory (ART) neural network was designed by S. Grossberg [Grossberg, 1988] and is well known for the mentioned capability (see also [Fausett, 1994] [Pandya and Macy, 1996]). Nevertheless we do not know about ART application extracting rare classes for landmark classification in mobile robot. The ART NN is designed for clustering vectors, independently of the order of input patterns. The degree of pattern similarity in a cluster and therefore the amount of clusters can be controlled by a vigilance test. A new cluster (neuron) will be created, if a new input pattern fails the vigilance test with the existing clusters. This helps to overcome the *stability-plasticity dilemma*, this means the ability to learn constantly new experiences without ignoring old ones.

4.2 Modified LVQ

An other method for unsupervised classification is the Learning Vector Quantization (LVQ), designed by T. Kohonen [Kohonen, 1990]. It is here slightly modified to fit the conditions described at the beginning of this chapter.

- The amount of classes (neurons) are not determined from the start. Each input pattern “sufficiently” different from the existing clusters creates a new class (similar to the ART theory).
- The stability (flexibility to new input patterns) of each existing class is controlled by the amount of entries (the more entries, the more flexible).

4.3 Feature generation

Each classifier produces an independent and continuous stream of *class-numbers* (called *class stream*) during the movement of the robot. Our experiment uses four classifiers, so we get four asynchronous class streams. Classifier III and IV process distance and ambient light information. We call *features* the four classes which are generated at the same time; a feature represents a certain sensor situation (see fig. 4-1).

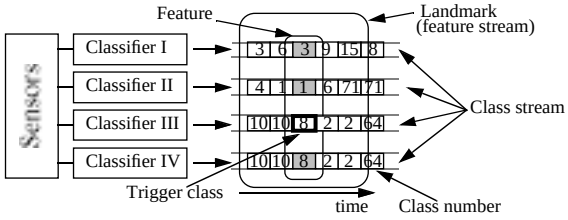

Fig. 4-1: Expression definitions

5 CLASSES BECOME LANDMARKS

The classification module produces about 50 features per second. It is quite impossible to discover a definite feature corresponding to a significant robot situation. The class streams are always a little bit shifted with respect to each other and they contain extra noise as well.
Therefore significant features have to be discovered by independent analysis of each class stream. This is done by a simple statistics module, selecting a *trigger class* (see fig. 4-1) in the following way:

- The occurrence of a class has to be small (<0.01%).
- A class must not appear more than three times in a stream.
- A class must not appear again within a certain distance to its previous occurrence. The distance is calculated by odometry.

Such trigger-classes are quite rare and they generate a *landmark* containing a stream of features (30 in our experiment) and other informations like estimated robot position, time, etc. Our simulated robot environment contains about 300 landmarks; however not every landmark will be used.

6 COMPARISON OF LANDMARKS

The classes (4 x 30 in our experiment) stored in a landmark are absolutely not identical when generated several times by the same cause. However, the *Weighted Levenshtein Distance* measure (see later) is used to compare the fundamental character of class streams and to identify probably identical landmarks, even if the contained classes are quite different.

6.1 Weighted Levenshtein Distance (WLD)

The WLD is calculated with an algorithm developed by V.I. Levenshtein [Levenshtein, 1975] to compare two strings of discrete symbols in a tolerant way (like text search). Let's call the two strings a and b with the length of i and j, so the WLD can be defined in the following recursive manner:

$$L(a_i,b_j) = min\begin{cases} L(a_{i-1},b_j) + \text{cost}_{del} \\ L(a_i,b_{j-1}) + \text{cost}_{ins} \\ L(a_{i-1},b_{j-1}) + \begin{cases} 0 & \text{if } a[i] = b[j] \\ \text{cost}_{sub} & \text{if } a[i] \neq b[j] \end{cases} \end{cases}$$

The WLD is defined as the minimum total cost required to convert a into b. There are only three editing operations:

- *Deletion* of an existing symbol (cost_{del})
- *Insertion* of a new symbol (cost_{ins})
- *Substitution* of a symbol (cost_{sub})

The recursion is terminated with the following ending, if at least one string becomes empty:

$$L(0,0) = 0$$
$$L(a_n,0) = n \ \text{cost}_{del}$$
$$L(0,b_n) = n \ \text{cost}_{ins}$$

This recursive formulation is very calculation intensive because every operation will cause three suboperations. These calculations can be drastically reduced by organizing temporary results in a table [Reuhkala, 1983].

The substitution cost (cost_{sub}) is calculated by measuring the euclidean distance of the two concerned classes in the sensor input space. The class numbers are not used in this calculation, since they are chosen arbitrarily and contain no distance information.

To make global string shifts invariant, the WLD algorithm can be changed so that deletion and insertion cost zero at the beginning and at the end of the strings.

6.2 Matching algorithm

A freshly received landmark will be compared with landmarks only containing the same trigger-class. Fig 6-1 shows the sensor signals proceeded by e.g. two different classifiers (I and II). The classifier II discovers a trigger class #2 (framed with a bold rectangle), which appears even twice by chance. Therefore, all class streams (I and II) are compared with the corresponding class streams of the stored landmarks which contain the same trigger class. Our example shows three suitable landmarks (indicated by A, B and C). The comparison of the corresponding class streams is measured by the Weighted Levenshtein Distance (see also paragraph 6.1).

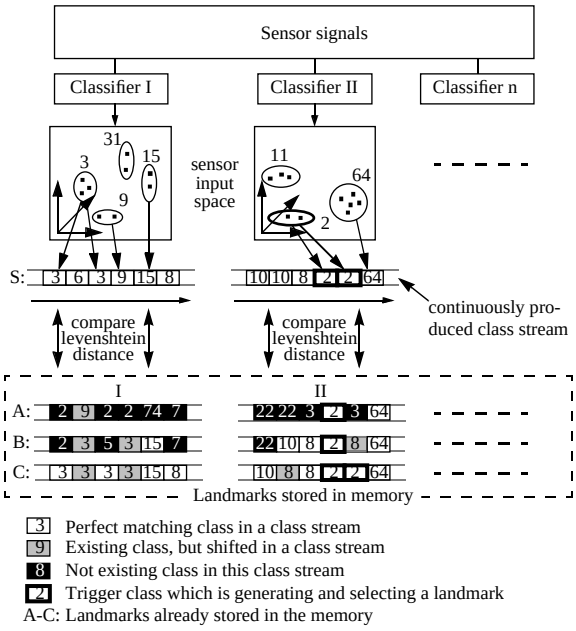

Fig. 6-1: Landmark comparison

Result of the three landmarks comparison:

- Landmark A was generated due to the same trigger class (#2). However, all class streams are very different indicating a complete dissimilar landmark.
- The landmark B is different as well (mainly because of class stream I). This can happen, if the sensors-group analyzed by classifier II are not sensitive for a specific dissimilarity. That's why several class streams are analyzed independently.
- Landmark C is determined as identical to the received landmark, because all class streams are sufficiently similar.

7 REDUCING MOVEMENT DIMENSION

The comparison algorithm explained in chapter 6 shows how an already stored landmark can be recognized. For that, the robot needs to hit the landmark. A randomly moving robot will often pass close landmarks on its way without hitting them. A navigator "hunting" landmarks can't be applied, because the needed position precision to hit a landmark is often beyond the precision of the odometry. Moreover, the needed information for data processing would break

the restriction of avoiding any prior knowledge of the sensors configuration.

An other possibility to consider this assumption is to reduce the freedom of the robot motion to some given paths. The following fitness function can be used to train a *pilot* with a NN in order to reduces the robot freedom to some lines by satisfying the following assumptions (see fig. 7-1):

A The robot has to move (cover a certain trajectory over time). This excludes stops, turning on the same place and bumping into obstacles.

B Any kind of active sensor event produces a positive feedback. This has basically two effects:
B1: The robot is attracted by walls and obstacles, through the short range distance sensors.
B2: The robot is attracted by obstacles (in general contours) located in the view of the camera.

The rules B1 and B2 are contradictory because the robot has to give up the wall (or obstacle) attraction to aim an object (or contour) in the free space. Therefore the rules for B1 and B2 can not be combined in the same fitness function and has to be used separately to train two different pilots. The pilots are applied randomly, influenced by the presence of walls and contours. The realization of these two pilots can be done for instance by a neural network. However, at this time the two pilots are directly programmed to save time for more important parts of this experiment.

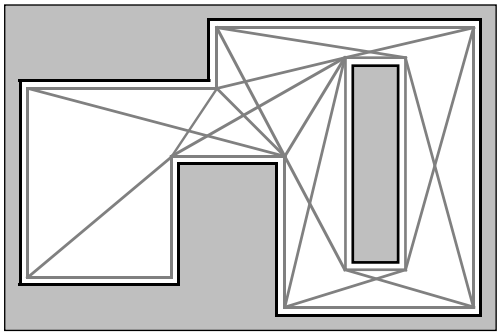

Fig. 7-1: Robot path attracted by walls and contours

The fig. 7-1 shows the possible paths (gray lines) of a robot guided by the two pilots as explained above. The robot runs along walls or steers towards corners (contours), if they appear in the view of the camera. The selection is done randomly. This pilot system trained by the mentioned fitness function considers the restriction of avoiding any prior knowledge of the sensor configuration.

8 EXPERIMENT RESULTS

The described algorithm was tested on a simulated robot environment containing static obstacles (see fig. 8-1). A pilot guides the robot along walls and towards obstacles if they appear in the view of the camera (see chapter 7). The resulting path which is constantly used is shown by shadowed lines.

During this limited movement, four classifiers extract about 300 landmarks from the camera, compass and distance sensor signals. The reason why some spots are chosen as landmarks is not always obvious. However, only twenty of the most convincing landmarks are sketched in fig. 8-1 by a cross to keep the figure clear. Some of them were generated because the robot was located in a corner which activated more distance sensors as usual. Another reason is the unexpected appearance of obstacles in the camera view.

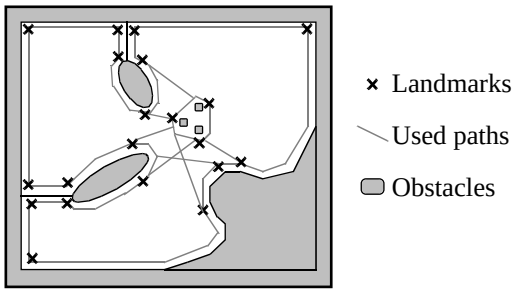

Fig. 8-1: Robot environment with created landmarks

Each landmark is stored with its estimated position which can be used to calibrate the actual robot position. Every time the robot recognizes an already known landmark on a plausible position, the current robot position is corrected to the position stored with the landmark. The result is a bounded odometry error. Refer also [Maechler, 1997] for an other method to limit odometry error).

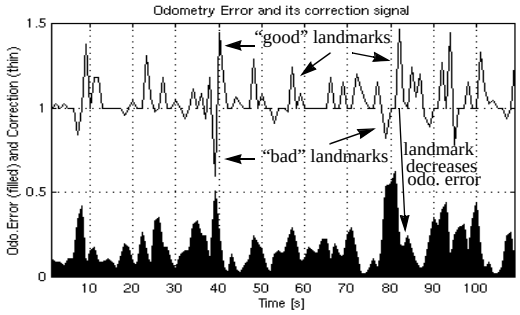

Fig. 8-2: Stabilized error due of landmark synchronization

The lower, filled graph in fig. 8-2 shows the absolute odometry error (five vertical units is one robot diameter). The upper graph shows the correction with the same scale (but shifted up by one unit). Every positive peak of the correction decrease the odometry error. Notice that there are also confusing (bad) landmark indicated by negative peaks. Such mistakes increase the odometry error, but they are quickly patched up by suitable (good) landmark corrections.

9 FUTURE WORKS

The careful reader would have probably noticed that the position assignment to new landmarks breaks the mentioned restriction of avoiding any prior knowledge. In fact, this "cheat" was done in this experience, but will be corrected soon by dividing the "learning time" into the following three phases (see fig. 9-1):

1) In the *exploration phase*, signal events are classified and most of the landmarks are created, but without assigning them to an estimated position. The robot will never quit this phase, but it will decrease the activity in the absence of new events.
2) In the *coordination phase*, near landmarks which are passed in short time are linked together with local odometry information. This behavior creates many *groups* of linked landmarks. The relative position of the landmarks inside a group is known, but no position information about the group is available.
3) In the *strategy phase*, the groups are linked together by active search. The robot will choose a big group and actively search other groups or landmarks in order to expand the original group.

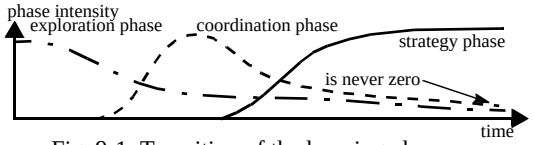

Fig. 9-1: Transition of the learning phases

Phase 2 and 3 can be explained by the “underground effect”. New residents of a big town know only the near environment of some underground stations (groups). No prior surface-link between stations exist. With time, the “station area” (groups) will expand due to explorations and other known stations will be discovered, completing and merging the topological map of the whole town.

The result is a distorted representation of the environment (see fig. 9-2) because of the inaccurate links between landmarks (and groups as well). However, landmark following is possible and completely sufficient for navigation.

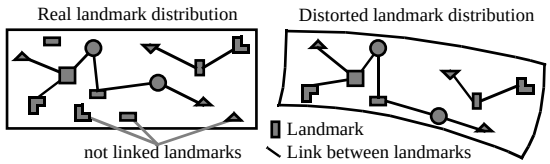

Fig. 9-2: Distorted representation of the environment

10 CONCLUSION

This paper showed that a robot can learn to stabilize its odometry error without any prior information about the environment or its sensor ability. The algorithm does not need to use human-defined concepts. It extracts and considers only adequate environment characteristics which make the algorithm adapt to new and unknown surroundings.

The three learning phases bring forth an improved interaction between the agent and its environment which is already done by the pilot. This improves the sensor signal quality and enables the robot to proceed with a useful task by actively navigating with its own concepts.

11 REFERENCES

[Carpenter and Grossberg, 1987b]
G.A. Carpenter and S. Grossberg, *ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns*, Applied Optics, vol. 26 (1987), no. 23, page 4919-4930

[Fausett, 1994]
Laurenne Fausett, *Fundamentals of Neural Networks*, Prentice-Hall, New Jersey, 1994, 440 pages, ISBN 0-13-042250-9

[Grossberg, 1988]
S. Grossberg, *The ART of adaptive pattern recognition by self-organizing neural network*, Computer, Vol. 21, Mar., 1988, page 77-88

[Hertz et al., 1991]
John Hertz, Anders Krogh, Richard G. Palmer, *Introduction to the theory of neural computation*, Addison-Wesley, Redwood City CA, 325 pages, 1991, ISBN 0-201-50395-6

[Kohonen, 1990]
T. Kohonen, *The self-organizing map*, Proc. of the IEEE, Vol. 78, No. 9, Sep. 1990, 17 pages (1464-1481)

[Kuipers and Pierce, 1997]
Benjamin Kuipers, David Pierce, *Map Learning with Uninterpreted Sensors and Effectors*, University of Texas at Austin, Austin TX 78712 USA, appeared in Artificial Intelligence Journal, 1997, 60 pages

[Levenshtein, 1975]
V.I. Levenshtein, *On the minimal redundancy of binary error - correcting codes*, Information and Control, Vol 28, Nr. 4, August 1975, 23 pages (268-291)

[Maechler, 1997]
Philip Maechler, *Robot Odometry Correction Using Grid Lines on the Floor*, M.C.P.A. Pisa, Italy, February 1997, 10 pages

[Michel, 1996]
Olivier Michel, *Khepera simulator package version 2.0*, Freeware mobile robot simulator downloadable from http://diwww.epfl.ch/lami/team/michel/khep-sim/index.html

[Pandya and Macy, 1996]
Abhijit S. Pandya, Robert B. Macy, *Pattern Recogition with Neural Networks in C++*, CRC Press, Florida, 1996, ISBN 0-8493-9462-7

[Reuhkala, 1983]
E. Reuhkala, *Recognition of strings of discrete symbols with special application to isolated word recognition*, Acta Polyt. Scand. Ma 38. Dr. Tech. dissertation, Helsinki Univ. of Tech. 1983

[Song and Choi, 1996]
Ho-Keun Song, Jong-soo Choi, *Edge detection method for range image using pseudo reflectance images*, Dep. of Electronic Engineering, Chung-Ang University, Seoul, Korea