Exploration, Navigation and Self-Localization in an Autonomous Mobile Robot

Thomas Edlinger
edlinger@informatik.uni-kl.de

Gerhard Weiß
weiss@informatik.uni-kl.de

University of Kaiserslautern, Department of Computer Science
Erwin-Schrödinger-Straße, P.O. Box 3049
D-67653 Kaiserslautern, Germany
Phone: +49 631 205 2624 Fax: +49 631 205 2803

Abstract

In this paper the autonomous mobile vehicle MOBOT-IV is presented, which is capable of exploring an indoor-environment while building up an internal representation of its world. This internal model is used for the navigation of the vehicle during and after the exploration phase. In contrast to methods, which use a grid based or line based environment representation, in the approach presented in this paper, local sector maps are the basic data structure of the world model. This paper describes the method of the view-point-planning for map building, the use of this map for navigation and the method of external position estimation including the handling of an position error in a moving real-time system.

1 Introduction

Autonomous Mobile Robots (AMRs) are systems which can move and perform useful operations without any external support or human intervention. One reason, which makes it hard to use autonomous mobile robots in changing operating environments, is the fact, that the robot's representation of the operating environment needs to be adapted manually. So the ability to operate in an unknown or partially known environment is essential for an AMR to be considered fully autonomous. Fundamental components are an actuator system to perform a given task and a perception system to build up a world model of the operating environment. If no explicit world model is given to the mobile robot, it is more flexible in the case of different or changing environments. To achieve this flexibility the robot needs control software, which is able to plan exploration tours based on an incomplete description of the environment in order to get an complete and consistent world model. There are different approaches known from literature. They use idealized sensors of infinite measurement range [1] or built up only a topological representation of the environment [2], which is very critical, because a geometrical description is needed for path planning during the exploration tour. Approaches, which use a grid based method to distinguish between the known and unknown parts of the environment [3] will produce problems concerning the memory usage in case of large areas. In this paper a method is presented, which uses real rangefinders to explore the environment and to build up a geometrical and topological representation of the vehicles environment suitable for large areas. Additionally the problem of self-localization is handled. Since any dead-reckoning technique will decrease in accuracy over the covered distance, from time to time some kind of recalibration is necessary. Given a system, that starts with no knowledge of the environment, it makes no sense to rely on preinstalled landmarks for self-localization. In this case the AMR has to extract features from the sensor data, that can be used as references for position and orienta-

tion. The methods presented in this paper are implemented on the autonomous mobile robot MOBOT-IV, which is designed to operate in indoor-environments. The basic hardware and software components are described below.

2 Hardware

MOBOT-IV (see figure 1) is a vehicle with one driven and steered front wheel and two passive rear wheels. It is equipped with two laser-rangefinders. One sensor (modelling sensor) is mounted on top of the vehicle at a height of 118cm. It scans with an angular speed of two revolutions per second and produces thereby 720 range measurements. The range of the sensor

Figure 1:MOBOT-IV

data reaches from 25cm up to 8m with an accuracy of about 5‰. Using a gyroscope the angular speed of the laser scanner is held constant with respect to the environment. This has the advantage, that every scan produced by the sensor covers exactly 360° independent from the angular speed of the vehicle itself. The range data of this sensor is used for environment modelling and position referencing. The second rangefinder (collision sensor) is installed at a height of 38cm and scans the front area of the vehicle at 180° with an angular resolution of 6° (30 sectors) and an accuracy of 5cm. This sensor is used for obstacle avoidance and in addition supports the environment modelling.
The control software runs on a 680x0 based VME-bus system. A wireless radio link is used to transmit internal data to the host computer, which serves as a user interface to visualize the internal map and to interact with the vehicle.

3 Control Structure

The structure of the control software with the data flow is shown in figure 2. Every software component runs on its own computer board except the Explorer, Navigator and the Geographer, which are running on the same CPU, for efficient sharing of common data, stored in the *Global Map*. These three components and the Correlator are the main topic of this paper.
In this distributed implementation of the control structure every software component interacts

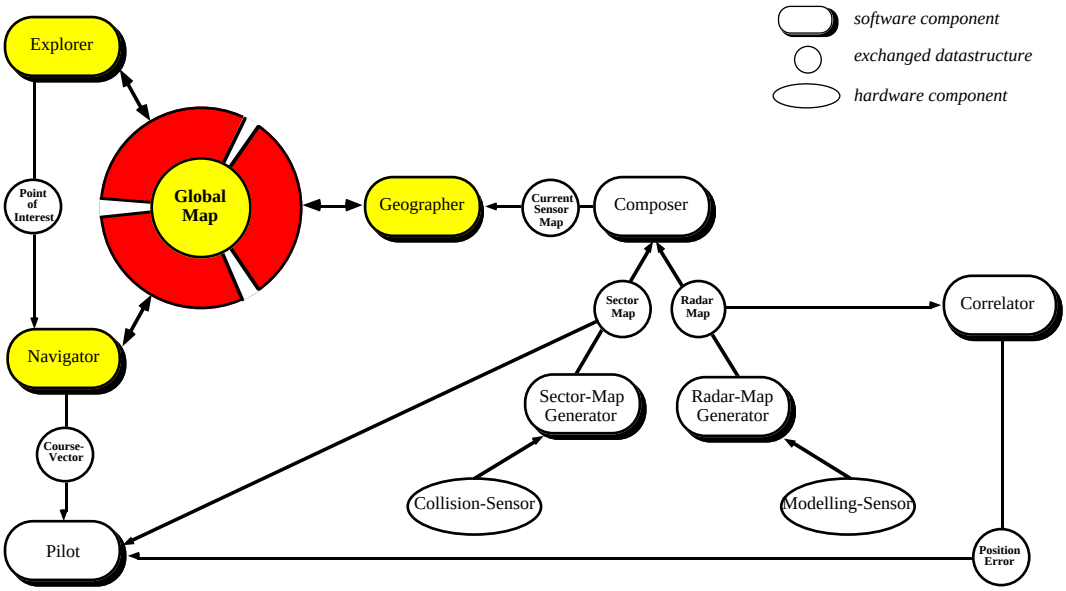


Figure 2: Control structure of MOBOT-IV

with other components by exchanging data via a central Communication Scheduler [5]. This scheduler copies the data that needs to be exchanged from the local memory of a producer to the local memory of potential consumers periodically with a fixed schedule.

The first step in the sensor data processing is an abstraction from the physical sensors and a transformation of their range data into internal data structures. This task is performed by the Sector-Map-Generator for the Collision Sensor. The data structure produced by this component is the *Sector Map*, which consists of the minimal distance to the next obstacle for each of the 60 sectors. The 30 sectors representing the area behind the vehicle will always contain infinite distances, because of the limited scanning angle (180°) of the collision sensor. The raw sensor data of the modelling sensor is processed by the Radar-Map-Generator. In addition to the transformation and filtering of the data produced by the sensor, this component has to interpolate the position of the sensor for each measurement. This is necessary because of the higher data rate of the sensor (1440 measurements per second) compared to the update rate of the position information (50ms). The generated *Radar Map* consists of 720 range measurements (full 360° scan) and the corresponding interpolated positions of the sensor. The Composer combines the Sector Map and the Radar Map to a data structure called *Current Sensor Map*. This data structure is similar to the Sector Map except that it contains a combination of the sensor data of the both sensors. The disadvantage of a slower update rate (500ms) compared to the collision sensor (40ms) and a lower resolution (60 sectors) than the modelling sensor are tolerable for the components using the Global Map.

At the end of this general description of the control software of MOBOT-IV the Pilot is described. It is placed at the end of the command hierarchy and gets its commands from the Navigator, which will be described later in detail. A command called *Course Vector* from the Navigator consists of the position and orientation where the vehicle should drive to. All Course Vectors are processed by the Pilot in the sequence in which they are sent by the Navigator. The Pilot computes a smooth path from the start to the desired goal position. The start position is given implicit by the vehicles actual position in the case that the vehicle is not moving. If the robot is still executing the last Course Vector, the desired goal position is taken for the start of the next tour. The Pilot is also responsible for collision avoidance, in case of unexpected obstacles. It therefore modifies the control commands for the driving motors to avoid the obstacles represented in the Sector Map.

4 Geographer

The Geographer only serves as a interface to the Global Map for the Explorer and the Navigator and is here mainly described by the structure of the Global Map and some of the operations defined on it. Many projects concerned with the environment modelling in mobile robots concentrate on the exact representation of the obstacle space. Practical experience has shown, that a very precise model of the environment is not necessary for tasks like navigation. In addition it is more important to have a representation of the free space for navigating the vehicle through its environment. For this reasons the Global Map in this approach consists of Current Sensor Maps with some additional information described later in this paper. The Current Sensor Maps stored in the Global Map are called Local Maps. The Local Map is a description of the local environment in polar coordinates with a centre C=(x/y) and sectors S_i (i = 0, …, n_s-1) with the minimal distance D_i to the next obstacle in each sector. A Local Map *'contains'* a position P = (x/y), if the euclidean distance between the position and the centre of the Local Map is less then the virtual sensor range and the position P is not occluded by an obstacle represented in the Local Map. That means, that a line from the centre of the Local Map to P does not intersect an obstacle represented in the Local Map. A virtual sensor range is used, because a real sensor has not a fixed measurement range but will be less reliable or more inexact at higher distances. The virtual sensor range R_v in the system described here is set to 4 meter. In order to accelerate the search for a Local Map containing a specific position P, all Local Maps are organized as a quadtree [6] sorted by the centre of them.

5 Explorer

The Explorer's task is the view-point-planning for the system in order to get a complete model of the environment. Only the parts of the environment, that are reachable for the vehicle, need to be represented in the final map. To support the navigation task, the Explorer builds up not only a geometrical description of the systems world, but will also determine the topological structure and store it in the Global Map.
The Explorer receives a *Current Sensor Map* (CSM) from the Composer every 500ms. In the CSM two areas are distinguished. The outer area is limited by a circle with the radius of the virtual sensor range R_v around the centre of the CSM. This area will be important for the creation of the exploration goals called the Points-Of-Interest (POIs). The inner part is limited by a radius $R_i < R_v$. In the actual system the radius R_i is set to 2.5 meter. The inner part of a Local Map controls, whether a new CSM will be added to the Global Map or not.
After receiving a new CSM the Explorer examines the *Global Map*, whether the centre of a new CSM is within the already known part of the environment. This is the case, if a Local Map in the Global Map exists, that contains the centre of the CSM and the euclidean distance between the centre of the CSM and the Local Map is less than the radius R_i of the inner part of a Local Map. If the CSM is not in the known part of the environment, it is added as a Local Map to the Global Map by the Geographer. In this case the Explorer computes possible passages at the limits of the inner and at the outer area of the newly added CSM. These passages have to be wide enough, that the vehicle is able to pass them. A passage at a distance D is defined as a series of consecutive sectors S_i (i=k, …, (k+n) mod n_s) in the CSM with the obstacle distance $D_i > D \; \forall$ i=k, …, (k+n) mod n_s and $D_{k-1} \leq D$ and $D_{k+n+1} \leq D$ (see figure 3). To guarantee, that the passage is wide enough, the width $W=(D_{k-1}^2+D_{k+n+1}^2-2\cdot D_{k-1}\cdot D_{k+n+1}\cdot \cos\alpha)^{1/2}$ of the passage has to be larger than the vehicle's width. In this formula α is the angle between the sectors S_{k-1} and S_{k+n+1}. The Explorer computes passages at the two distances R_i (inner passages) and R_v (outer passages). In order to construct the connectivity graph of the environment, the Explorer searches in the Global Map for Local Maps, which intersect with their inner

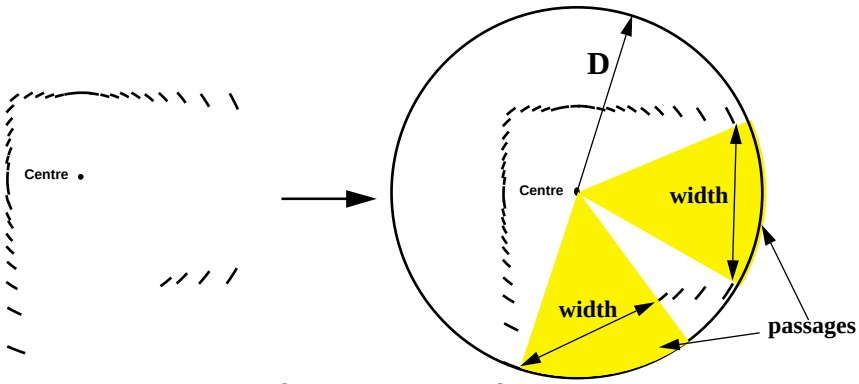


Figure 3: A Current Sensor Map and two passages at a distance D

area with the CSM. If the centre of one of these Local Maps L_i lies in the direction of an inner passage of the CSM and vice versa, a link between the two elements is stored. This link marks the two elements as neighbors. That means, that the vehicle is able to drive from the CSM to L_i and vice versa. So the local elements in the Global Map form a graph representing the topological structure of the environment. This topological information is used by the Navigator to plan a path from the current location of the vehicle to a goal given by the Explorer or a human operator.

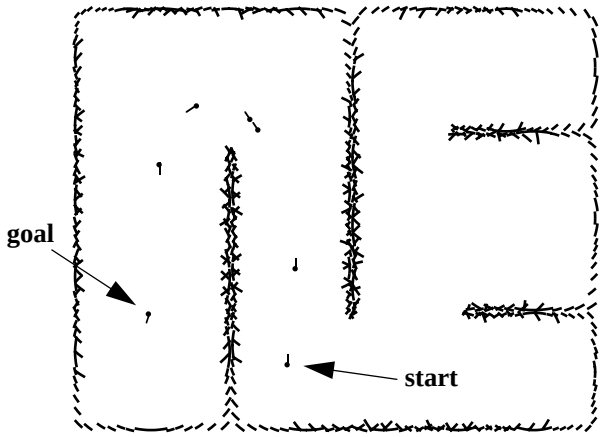
For every outer passage found in the CSM, the Explorer creates a Point-Of-Interest (POI) at a distance $R_i+\varepsilon$ from the centre of the Local Map in the direction of the corresponding passage. The newly created POIs are pushed on a stack. POIs on top of the stack, which lie in an already known area (inner area of a Local Map) are deleted by the Explorer. From the remaining POIs the POI on top of stack is chosen and given to the Navigator. When the system reaches the given POI a new Current Sensor Map is added to the Global Map and the actual POI is deleted. The exploration is finished, if the POI-stack is empty. An operator may define goals after that, where the vehicle should drive to. A Global Map of a simulated test environment after the exploration phase is shown in figure 4.

Figure 4: Global Map
(a) with inner areas of Local Maps *(b) with connectivity graph*

6 Navigator

This component plans a path to a POI given by the Explorer or to a user defined goal after the exploration phase is finished. The path computed by the Navigator needs not to be absolute obstacle free in all cases. Intermediate goals produced by this component, which lead the vehicle too near to an obstacle, will automatically be corrected by the Pilot during their execution. All paths created by the Navigator will be executed properly as long as they are correct in a topological sense. That means, that for most applications it is not necessary to plan a path in the middle of a corridor as long as this corridor will lead the vehicle to the desired goal.

The goal points for the Navigator are given in form of a position $P_G=(x/y/\alpha)$ in global coordinates. Based on the information stored in the Global Map, the Navigator creates intermediate goals for the Pilot if necessary. Therefore the Navigator searches for the Local Map S, that contains the start position P_S and the Local Map G containing the desired goal location P_G. If more than one Local Map contains either the start or the goal point, the Local Map with the shortest euclidean distance to that point is chosen. As described in the Explorer-section, two Local Maps in the Global Map are connected by a link, if the vehicle is able to drive directly from one Local Map to the other. It is easy to see that the Global Map realizes a connected undirected graph with the Local Maps as the nodes and the links as the edges. That means that for each pair of Local Maps A and B in the Global Map there always exists a path from A to B. In the first step the Navigator determines a path from S to G using the A^*-algorithm [7]. In this context a path means a sequence of Local Maps M_i (i=1, …, n), with M_1=S and M_n=G and there exists a link between M_i and M_{i+1} for i=1, …, n-1. To minimize the amount of Course Vectors created by the Navigator, Local Maps in the computed path are eliminated if their centres are nearly on a straight line. For this purpose the “Iterative-End-Point-Fits”-algorithm is applied [8]. In the first step the start and the goal point are connected by a straight line. Then the distances A_i between the centres of the Local Maps M_i and this line are computed. If none of the distances A_i is longer than a threshold T, then the path from start to goal can be described with one Course Vector at the goal point with the angle of the straight line as the goal direction. If $\max\{A_i \mid i = 2, \ldots, n-1\} = A_k > T$ then the path is split at the centre of the Local Map M_k and the algorithm is executed recursively on the two sub-paths $M_1 \ldots M_k$ and $M_k \ldots M_n$. The created Course Vectors for an example path are shown in figure 5.



igure 5:Example path

7 Correlator

While the Pilot determines the internal position of MOBOT-IV by dead-reckoning, the Correlator is responsible for a continuous reliable external position estimation of the mobile platform, regardless of slippery between the drive wheel and the floor or other inaccuracies. Since the system is working in a previously unknown environment, self-localization touches three problems: the extraction of references for position and orientation from sensor data; the construction of a map used to store new references or to calculate the displacement between actual references and those from the map, and a consistent handling of a position update in the distributed system. For this purpose the Correlator continuously reads the Radar Map and trans-

forms it into a *Reference Point*, a sequence of scan-points given in the systems global coordinate system. The Reference Points are compared to each other in order to calculate the actual error of the internal position and to establish a *Correlator Map* of the environment for further reference. The Correlator sends the calculated *Position Error* to the Pilot, which uses it to correct the internal position.

7.1 Extraction of References for Position and Orientation from Sensor Data

Given two Reference Points taken at two different, but close places, it is obvious, that a correct match of these will show the error in the position and orientation of one Reference Point, assuming that the other is correct. Therefore a matching between Reference Points must be derived. The approach was introduced in [9] and will be more formally described here. Assuming there are two sequences of scan points S_i and D_j, a matching that satisfies the following 'equation' is sought:

$$S_i \approx T \cdot D_j + v \tag{1}$$

i.e., a rotation-matrix T and a translation vector v is sought, which satisfies (1) for most pairs (i, j). If (1) is subtracted from itself, but shifted for one or more indices n, this leads to

$$S_i - S_{i+n} \approx T\,(D_j - D_{j+n}) \tag{2}$$

In order to simplify notation, $S_i - S_{i+n}$ and $D_j - D_{j+n}$ may be written as S'_i and D'_j and called differential scan. The problem of orientation estimation can now be written as

$$S'_i \approx T \cdot D'_j \tag{3}$$

without considering the position, where the scan was taken. Now a method is needed to estimate T. Since each (differential) point in the differential scan is a vector, it can be transformed into a polar coordinate system. A *Angle Histogram* is the (discrete) frequency distribution of the angles of the differential scans. This Angle Histogram is therefore a transformation from one sequence (the differential scan) to another sequence, the distribution of the discrete angles in the scan. After the Angle Histogram generation is applied to both differential scans,

$$A(S'_i)_k \approx T \bullet A(D'_j)_l \tag{4}$$

the rotation between this differential scans is now merely a index-shift in the histograms and can be found by cross-correlation of the two sequences. A problem arises here: this index-shift may be ambiguous. Therefore a rough estimation of the rotation-angle is necessary. In addition the cross-correlation delivers a quality measure after a normalization, that can be used to indicate, if both histograms have been good candidates to correlate.
An estimation for v is still sought. Since T is known now, it can be inserted into (1), so that $T\,D_j$ is replaced by D_j^*, the rotated scan:

$$S_i \approx D_j^* + v \tag{5}$$

In order to estimate v, a similar approach as for the rotation matrix is used; a distribution of scan points is computed. Since v has two dimensions, this must be done twice, once for each dimension of the coordinate system. One problem is, that sharp peaks in this distribution will occur only, if the longer lines in the original scans are orthogonal to the direction of the established distribution. Therefore the scans are rotated to the main direction, which can be found by determining the maximum in the angle histogram. This transforms (1) into

$$M \cdot S_i \approx M \cdot D_j^* + M \cdot v \tag{6}$$

where M is the rotation matrix for the main direction. The distributions for both axis of the coordinate system lead to

$$X(MS_i)_k \approx X(MD_j^*)_l + M \cdot v_x \tag{7}$$

$$Y(MS_i)_k \approx Y(MD_j^*)_l + M \cdot v_y \tag{8}$$

where the index-shift can also be determined by cross-correlation. This index-shift x/y between X_k and X_l, (respectively Y_k and Y_l) is equal to $M \cdot v$, so that v is calculable as

$$v = M^{-1} \begin{bmatrix} x \\ y \end{bmatrix} \tag{9}$$

Since there is also the problem of ambiguousness to solve, a rough estimation of v is needed. With the help of T and v, we can now transform the scan D_j to its correct origin, so that it can be used as a reference scan.

7.2 Establishing a Map for Position and Orientation Correction

If a scan of a laser-rangefinder is used as a position and orientation reference, a map of the environment that consists of such references is needed. The main problem here is, that such a map is not previously given, but must be constructed while the robot moves through its environment. The used approach is rather straight forward [10]: the first scan of the system is used as a start reference, and inserted into the grid based Correlator Map. While the system is moving on its path, the Correlator calculates its position and orientation from the actual Reference Point and the start reference. Meanwhile all newly corrected Reference Points are inserted into the appropriate grid cell of the map unless the cell is already occupied. If the start reference produces no longer a position and orientation correction of good quality, a new Reference Point is chosen from the Correlator Map as the start reference. The Correlator tests first, if there is a Reference Point entry in the map for the grid cell in the front area of the vehicle's actual position. This ensures that the Reference Points of already visited areas are considered first. If there are no Reference Points found, the system will use Reference Points from the grid cell behind its actual position. This enables map building in an unvisited region.

7.3 Consistent Position and Orientation Updates in a Distributed System

While an AMR moves, it is not only necessary to detect and calculate accumulated drift errors, but also a compensation of them is needed. The Correlator sends every 500 ms Position Errors to the Pilot, which the Pilot incorporates into its internal position. Since the Correlator does not know, if a Reference Point has taken that update into account, a representation for the position is needed, that allows to calculate the Position Error independently from already happened updates. The approach, that is used here is a simplified version of what can be found in [11]. If the Position Error P_e is defined as the transformation of a corrected position and orientation P given in a homogenous coordinate system into the uncorrected form P_u, the Position Error will be of an absolute size, regardless of how many corrections have been made. This can be expressed by the following equation:

$$P \cdot P_e = P_u \tag{10}$$

The pair of the corrected position and the Position Error is called *Position Information*. Assuming such a pair (R_p, R_{pe}) is attached to a Reference Point, for which the Correlator has calculated a new position R_{pn}, the new Position Error $R_{pe'}$ can be reckoned through

$$R_{pe'} = R_{pn}^{-1} \cdot R_p \cdot R_{pe} \tag{11}$$

This new Position Error $R_{pe'}$ can now be used to update the Position Information in the Pilot. Assuming the actual Position Information of the Pilot is (P_p, P_{pe}), then

$$P_p := P_p \cdot P_{pe} \cdot R_{pe'}^{-1} \tag{12}$$

$$\text{and } P_{pe} := R_{pe'} \tag{13}$$

leads to the new corrected Position Information. This scheme allows position updates regardless of the update history, since the position error is always expressed as an absolute value. Even if it is necessary to interpolate between different positions, e.g. during the construction of the Radar Map, this update scheme can be used, to normalize the used positions with (12) and (13) to the same Position Error.

8 Test results

Test runs have been made in the corridors of our laboratory. A global map as it was explored autonomously by MOBOT-IV can be seen in figure 6. The dimensions of the explored environment are approximately 70 by 55 meters. The dot marks the start point of the exploration tour. The vehicle needed approximately 15 minutes to explore the whole scenario at a maximum travelling speed of 40 centimeters per second. The amount of memory needed to store this map was about 40 kByte.

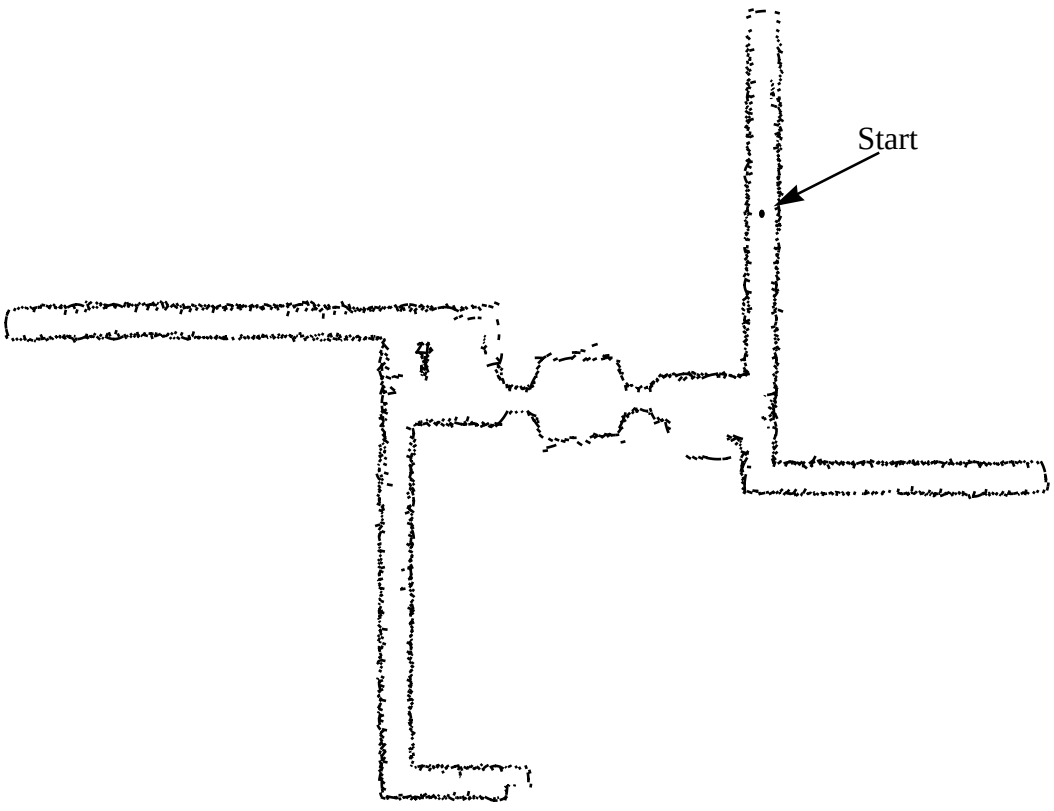


Figure 6: Global Map of corridor system

9 Conclusion

In this paper an autonomous mobile system is presented, that is able to explore its environment. No external landmarks are necessary, because of a correlation method based on features extracted from the raw sensor data. Test runs have shown, that the algorithms are able to control the mobile system in real-time. The internal map built up during the exploration phase is complete, consistent and can be used to navigate the vehicle inside the reachable area. Future work will concentrate on the problems of dynamic environments, so that walking persons could be tolerated during the exploration process or changes in the topological structure can be handled properly. In order to handle loop situations in the Correlator more general, the Correlator Map will be attached to the Global Map.

10 References

[1] N. S. V. Rao, S. S. Iyengar, B. J. Oommen, R. L. Kashyap, “On Terrain Acquisition by a Point Robot Amidst Polyhedral Obstacles”, IEEE Journal of Robotics and Automation, Vol. 4, No. 4, Tsukuba, Japan, Aug. 1988, pp. 450-455

[2] B. J. Kuipers, Y. T. Byun, “A Qualitative Approach to Robot Exploration and Map-Learning”, AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion, St. Charles, Illinois, Oct. 1987, pp. 390-404

[3] J. Iijima, S. Asaka, S. Yuta, “Searching Unknown Environment By A Mobile Robot Using Range Sensor - An Algorithm And Experiment”, IEEE/RSJ International Workshop on Intelligent Systems 89, Tsukuba, Japan, Sep.1989, pp. 46-53

[4] T. Edlinger, E. v. Puttkamer, “Exploration of an indoor-environment by an Autonomous Mobile Robot”, Intelligent Robots and Systems, Munich, Germany, Sep. 1994, pp. 1278-1284

[5] E. von Puttkamer, C. Wetzler, U. R. Zimmer, “ALBATROSS - The Communication Scheme as a Key to Fulfil Hard Real-Time Constraints”, Proc. of the Euromicro '92 Realtime Workshop, Greece, 1992

[6] S. Hanan, “The Quadtree and Related Hierarchical Data Structures”, Computing Surveys 16, No. 2, 1984, pp. 187-260

[7] P. E. Hart, N. J. Nilsson, B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, IEEE Transaction on Systems Science and Cybernetics, Vol. SCC-4, No. 2, July 1968, pp. 100-107

[8] R. O. Duda, P. E. Hart, “Pattern Classification and Scene Analysis”, John Wiley & Sons, New York, 1973

[9] G. Weiß, C. Wetzler, E. v. Puttkamer, “Keeping Track of Position and Orientation of Moving Indoor Systems by Correlation of Range-Finder Scans”, Intelligent Robots and Systems, Munich, Germany, Sep. 1994, pp. 595-601

[10] G. Weiß, E. v. Puttkamer, “A Map Based On Laserscans Without Geometric Interpretation”, Intelligent Autonomous Systems-4 (IAS-4), Karlsruhe, Germany, March 1995, pp. 403-407

[11] R. Hinkel, “Konzeption eines echtzeitfähigen autonomen, mobilen Systems”, PhD-Thesis, Dept. of Computer Science, University of Kaiserslautern, Germany, 1989, pp. 172-180