Autonomous vacuum cleaner

Iwan R. Ulrich Francesco Mondada J.-D. Nicoud

Laboratory of Microcomputing (LAMI)
Swiss Federal Institute of Technology
Lausanne, Switzerland
`iwan@umich.edu mondada@di.epfl.ch nicoud@di.epfl.ch`

Abstract

This paper presents the results of the development of an autonomous mobile robot, designed according to some new concepts established in this field during the last decade. These principles can be found in other work based on the constructivist approach, artificial life, subsumption architecture and other bottom-up methodologies. These ideas have been applied to the complete robot design, spanning from the shape of the robot to the sensors, from the electronics to the software control structure. By this way we have developed what we call an "Application Specific Mobile Robot" (ASMR). The target application is a domestic autonomous vacuum cleaner. Despite the actual limitations of the final robot, this work shows how this methodology can bring many interesting results.

1 Introduction

The domestic autonomous vacuum cleaner that cleans your home while you are outdoors doing some shopping, is an old wish and it may stay so for some time. Many companies and research institutions have tried to develop such a robot, with only few results. The human operator can not be replaced easily by "artificial intelligence" [Schraft et al., 1994, Schofield and Grünke, 1994]. The lack of results is common to a large range of applications in autonomous mobile robotics, and it is due to the complexity of the task and the limitations of the actual technology. "Autonomous", in this case, does not simply mean that the robot has batteries, good computational power and good behavioral rules, but much more. As Steels explains very well [Steels, 1995], autonomy is not simply composed by a set of smart rules, but is the ability to create its own rules. To move around, finding a path or doing a job like cleaning seems very simple for us and for many animals. The difference between robots and animals is that animals use their brain to solve a problem, and we start to understand the complexity of these "simple" tasks when we analyze the complexity of the brain.

In the last decade, new approaches have tried to bring "artificial intelligence" (AI) and the computer science community back to the real world to get better results in the field of autonomous mobile robotics [Brooks, 1990]. This field needs smart systems that are well adapted to their environment through their shapes, sensors, actuators and behaviors. The embodiment [Brooks and Lynn, 1993] of the system is important and brings a new and complex dimension to the problem of artificial intelligence.

In this paper, we present the development of a domestic autonomous vacuum cleaner based on some basic principles, like bottom-up approach, simplicity and coherence between the different parts of the system. Domestic environments are mostly unstructured and often very complex, much more than the industrial environment where some cleaning robots are already being used [Schofield, 1995]. Moreover, this application sets some important constrains for the size, the self-sufficiency, the efficiency and the autonomy of the robot.

To get a satisfying cleaning job, we define that the robot has to cover at least 90% of the open surface. The robot has to accomplish this task without any artificial landmarks, as the client would probably not like to have beacons installed in his apartment. Furthermore, the robot has no initial map of the environment as it would be too expensive to program a map for each customer. Also the possibility that the customer himself programs the map must be excluded.

2 "Application Specific Mobile Robot" (ASMR)

2.1 Basic principles

Despite a demanding market [Pellerin, 1995], only few cleaning robots have been developed until now [Schofield, 1995] and many companies went out of business trying to develop new products for this domain [T. Gomi, personal communication 1995].

A new approach must be followed to design an autonomous vacuum cleaner working in an unknown and unstructured environment. It is not sufficient to use a standard mobile robot platform, to surround it with rings of general purpose distance sensors and then to emphasize the work on the "intelligence".

As it is necessary to develop the behavior of the robot, it is also necessary to optimize its shape and its sensor system. All these three criteria (shape, sensors, software) must not only be designed individually but also as a whole. Only with a total integration of hardware and software can such an application become feasible. This integrated design is what we call an "Application Specific Mobile Robot" (ASMR).

Developing the proper shape is essential for the success of such an application, even though its importance is often neglected. If the robot's body is well adapted to the application, it will be more compatible with the sensor system and facilitate the programming of the robot's behavior.

The sensor system is an important part of the robot as it provides information about the environment. The better adapted and the more reliable this system is, the easier the programming of the robot will be. Furthermore, the robot's behavior can only be as good as its sensor system.

Finally the control system is responsible for the "intelligent behavior" of the robot. Benefiting from a well-designed robot, the program will be shorter, faster, less complicated and more reliable.

2.2 The basic mobile platform and the Koala robot

Two motion structures are mainly used on wheeled mobile robots: the "synchro-drive" and the simple and classical structure based on two lateral wheels or tracks. These two structures can deal with complex situations using simple control strategies. For our cleaning robot we have chosen the second structure which is simpler to build. To deal with small obstacles as doorsteps, but to avoid expensive tracks, the robot has been equipped with six wheels, actuated by two DC motors, each controlling three wheels on each side. Due to the fact that we already had such a mobile robot in our laboratory (the robot called "Koala"), this platform was chosen as the basic element for this project. In addition to its mechanical structure, this robot has the advantage of a modular structure that is easy to modify and extend. Moreover, the CPU and the BIOS of this robot are compatible with the "Khepera" miniature mobile robot [Mondada et al., 1994], used by more than 100 universities and for which a lot of software has been written.

The Koala has a square shape and measures about 30 cm by 30 cm. With its dimensions, the Koala fits the application of an autonomous vacuum cleaner. A bigger robot would complicate the navigation between obstacles and would not be easily stored after use.

The Koala main processor is a Motorola 68331, with 1MBytes of RAM and 128KBytes of ROM. A specialized processor helps the main processor in all time-specific tasks. The motion is based on two DC motors with incremental encoders. The rechargeable Ni-Cd batteries have a capacity of 2.4Ah, providing the robot with more than three hours of autonomy.

The new version of the Koala robot is equipped with sixteen IR proximity sensors disposed all around the robot, with a higher density at its front. These sensors were not available yet during this project but could be used as described in section 6.

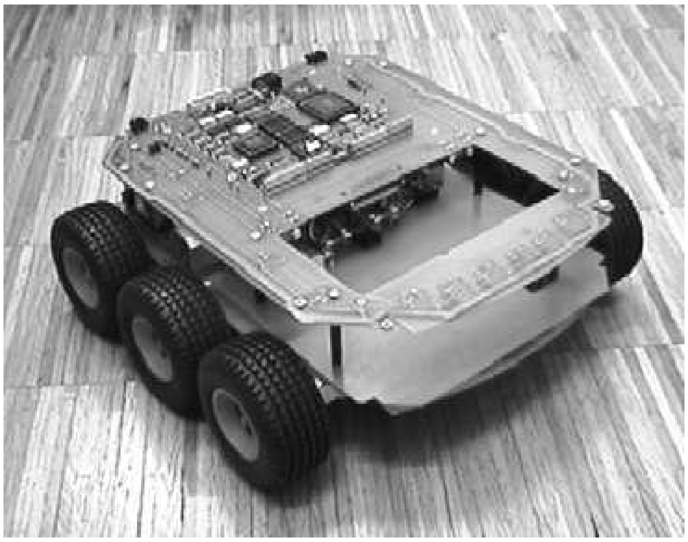


Figure 1: The Koala robot in its commercial version.

2.3 The shape of the robot

The robot's shape has an important influence on the facility for covering a surface, which is the main task of an autonomous vacuum cleaner. Theoretically, many shapes allow the robot to cover most of the surface, but only a few allow this without requiring a very complex behavior. The ease of the surface coverage is a very important aspect as it largely determines the complexity of the robot's algorithm and also the power consumption due to its movements. Furthermore, the less the robot has to move, the better the accuracy of its odometry.

2.3.1 The possible shapes

To find the best shape for our application we compared many different possibilities shown in figure 2, where the cleaning area is colored black. In general, the possible solutions can be classified into three categories: simple shapes, round robots and robots with an arm.

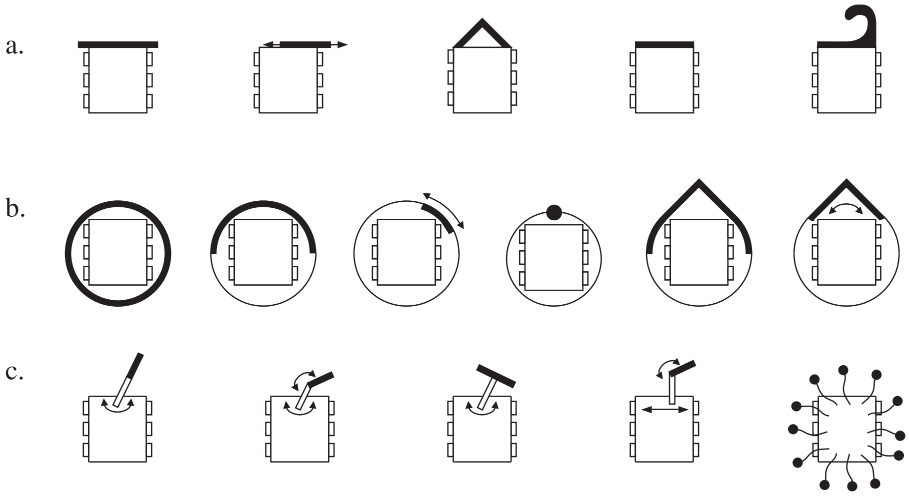


Figure 2: a. Simple shapes. b. Round robots. c. Robots with an arm.

All the simple shapes are easy to build. But all of them either require a complex trajectory to clean along walls or can't clean in corners and around legs in an efficient way. They all need a very complex algorithm for these tasks and are therefore far away from an ASMR.

The round robots have the advantage of facilitating the navigation around obstacles. But round robots can not clean in corners which is a big disadvantage for potential markets. They also require a complex trajectory to clean around small obstacles like legs. Another problem is the cleaning along walls due to the non-compliant contact between the massive robot body and the wall. In addition, many of these solutions require a too large cleaning area. The power available on the Koala limits the cleaning area to less than 40 cm^2 to guarantee an acceptable suction.

2.3.2 Mobile robot with 2-dof arm

Mobile robots equipped with an arm offer many interesting possibilities. This lead us to choose the solution of an arm with two active degrees of freedom. The arm consists of revolute joints (RR) and an additional passive degree of freedom allowing vertical movements to cope with floor irregularities. As the cleaning head becomes mobile, it is possible to reduce its size to obtain a better suction. It also allows us to increase the density of sensors by concentrating them on the small cleaning head.

The major advantage of the arm is that it allows the decoupling of the cleaning movements from the robot displacements. The Koala is responsible for the global displacements while the arm acts locally. While the arm is cleaning in corners, around legs or other obstacles, the Koala can stay stationary and therefore keeps its accuracy of the odometry. In the case of concave corners the robot can turn easily, even though it has a rectangular shape, for the platform's center of rotation is far enough from the walls. The arm also helps to clean along walls, as its joints introduce a compliance which reduces the friction between the robot and the wall.

The same advantage of the decoupling occurs during the exploration of an unknown obstacle. As the Koala can stay immobile it represents a good reference. With the information of the arm, the robot can construct a local map of the obstacle relative to a fix point, the position of the Koala.

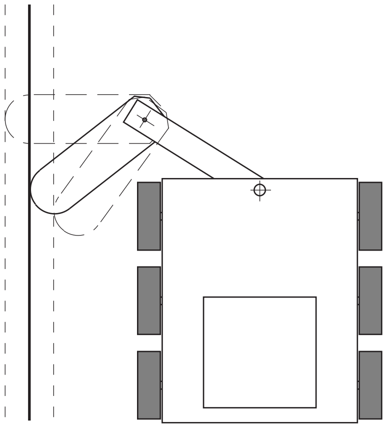


Figure 3: Wall following.

The major drawback of the additional arm is its construction cost and the need for an additional control system. But this cost is worth it as it really facilitates the further development of the robot and makes the application feasible.

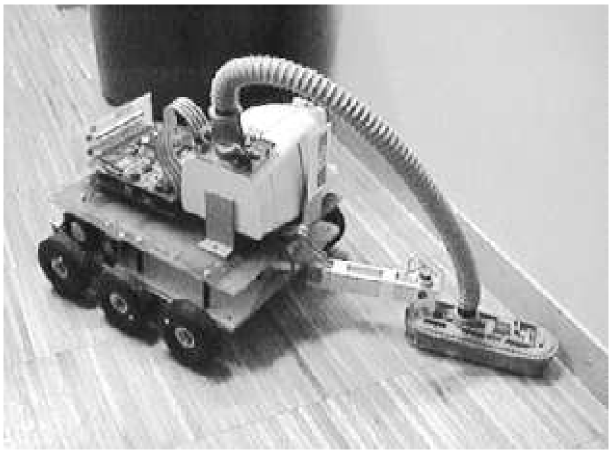


Figure 4: Koala robot modified and equipped with an arm.

2.4 The sensor system

In addition to the robot's shape we have to define a sensor system which is best adapted to its tasks and shape. For a mobile robot navigating in an unknown, cluttered environment, the priority of its sensor system must be its reliability. The sensor system must be capable of detecting as many types of obstacles as possible. The accuracy is less important.

In the case of an autonomous vacuum cleaner, the sensor system must especially be able to detect legs of chairs and tables which often have small diameters but are typical obstacles in an apartment. In addition, the sensor system must be cheap.

2.4.1 The tactile sensor system

Tactile sensors are rarely used in mobile robots as most of the applications need a robot that avoids obstacles without touching them. Some mobile robots are surrounded by a bumper that uniquely turns on an emergency stop in case the obstacle avoidance algorithm failed.

In the case of the autonomous vacuum cleaner, it is desired that the cleaning head touches the obstacle to guarantee a good cleaning. A tactile sensor system on the cleaning head can reliably detect a contact with an obstacle. As the contour of the cleaning head is much smaller than the contour of the robot it is also easier to get a high density of sensors.

In addition, tactile sensors are low cost, easy to implement and especially very reliable. Especially this last point, the reliability, is a big advantage in unstructured and unknown environments [Yamamoto, 1993][Rucci and Dario, 1994].

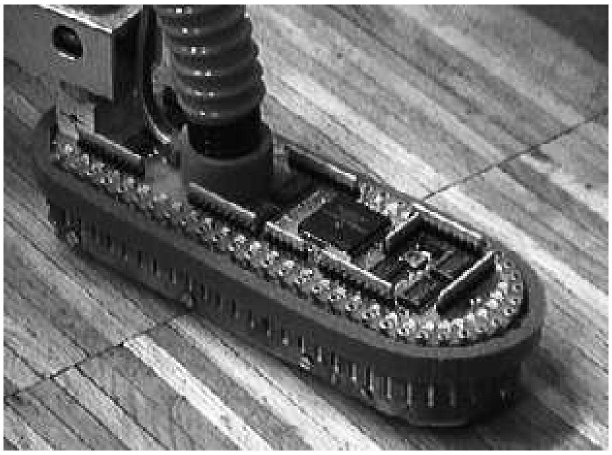


Figure 5: Cleaning head equipped with the tactile sensor.

We designed a low cost tactile sensor system consisting of plug pins and a metal plate in the shape of a comb (figure 6). The plate is fixed vertically around the cleaning head. To protect the plate and to reduce shocks during a contact, a flexible band is put around the top of it. In addition, the band distributes an exterior force on the local plug pins which allows it to obtain information about the intensity of the force: The larger the force, the more contacts will be made, as illustrated in figure 7.

The actual system consists of 54 contact points with spacings of 6 mm. This allows the detection of most of the obstacles. The necessary force to detect a contact is less than 1 Newton for the worst case of several contacts in parallel. The information of the 54 contacts is preprocessed by a MC68HC11 microcontroller.

2.4.2 The compass

The robot is also equipped with a compass. Unfortunately, the magnetic flux is not constant in an apartment but is highly influenced by metallic objects and electronic equipment. Nevertheless, using the compass in an appropriate and careful way can give precious information to facilitate the determination of the robot's position.

An implemented application is the classification of walls. When the robot follows a wall it memorizes the magnetic flux along it. The magnetic flux stays constant for some walls, but

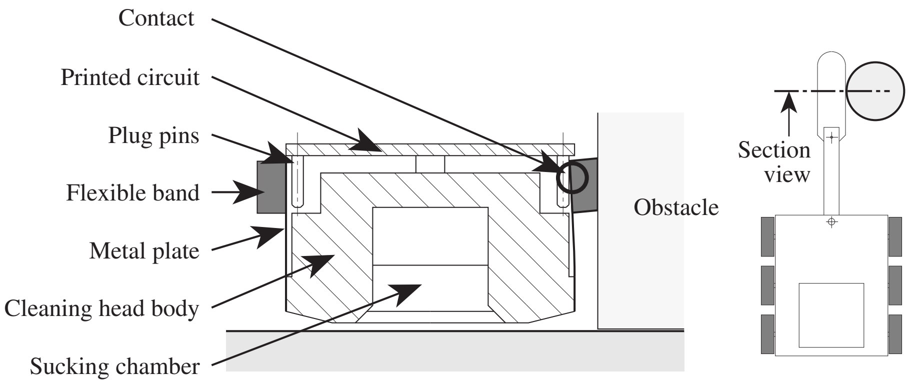


Figure 6: Section view of the tactile sensor.

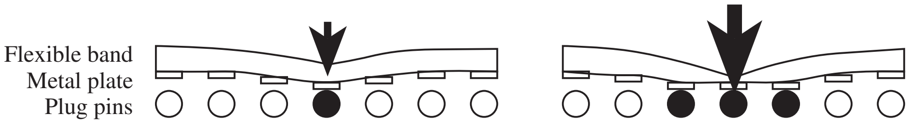


Figure 7: Effect of different forces on the tactile sensor.

changes greatly for others. To reduce this problem, the robot memorizes the maximum and the minimum values. The memorization of this flux helps later to determine the robot's position in case it gets lost (section 3.3.2).

2.4.3 The dust sensor

It is also important to adapt the cleaning speed to the accumulation of dust. This is something we do automatically when we use a vacuum cleaner, looking at the floor and insisting where more dust is. To implement this behavior, the robot is equipped with a dust sensor that measures the quantity of dust sucked up. This sensor is composed of an optical barrier placed in a narrow section of the sucking tube. This device is similar to others that can be found in commercial vacuum cleaners.

2.5 The vacuum cleaner

The energy source needed to create the vacuum effect is a big problem. Normal domestic vacuum cleaners consume between 600 and 1500 W taken from an electrical outlet.

Unfortunately, it is not possible to have batteries with such power on the Koala. The only possibility would be to add a cable connected to an electrical outlet. But such a cable would be very annoying for the robot. It is easy to imagine the result of a trajectory between several legs and other obstacles. The user would probably need more time to undo the knots than to clean the apartment himself.

Hand vacuum cleaners consume less power and often have their own batteries. For this reason, we chose a model that is able to clean effectively on an area of about 8 x 1 cm^2 for about 15 minutes with its own batteries. By using the batteries of the Koala this time could be extended to about 45 minutes.

2.6 The cleaning robot structure

The robot consists of four microprocessors working in parallel (see figure 8). The main boards "Master" and "Slave" each contain a MC68331 processor. The other two boards contain a MC68HC11 microcontroller and are used for the preprocessing of the tactile and IR proximity sensor data. The serial line to a Sun workstation is used only for downloading and debugging. Most of the communication between the boards is done by bi-directional FIFO memories. The communication between the MC68HC11 of the tactile sensor system and the "Slave" CPU is done by a serial line.

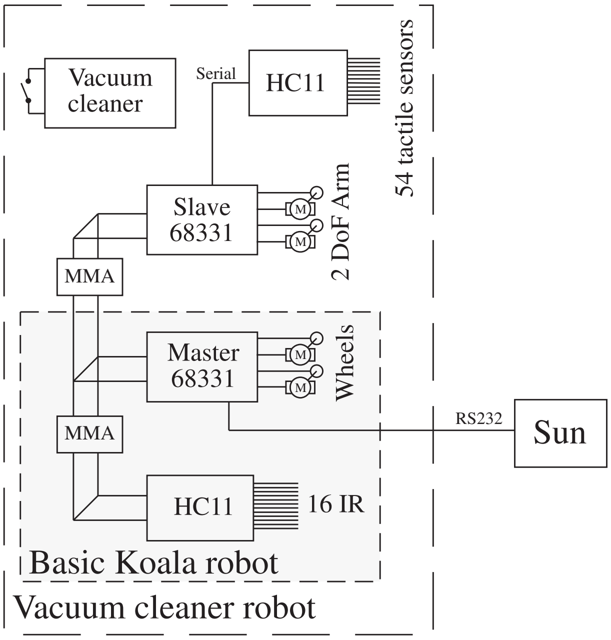


Figure 8: The global hardware structure of the vacuum cleaner robot.

3 Navigation

3.1 Identification of an obstacle

When the robot detects an obstacle it tries to roughly classify it. The classification of obstacles allows the robot to create a more useful representation than a map containing just the information "empty" or "occupied". Moreover, the identification allows the robot to use these obstacles as natural landmarks. The confidence in such a landmark is higher for supposedly immobile obstacles like walls than for obstacles that could be moved around as for example chairs. The robot is then able to determine its position by using the detected and identified obstacles it came across during the exploration of the previously unknown environment.

In this project, we limited the identification to the following classes that are typical obstacles in domestic apartments: legs, walls, convex corners, concave corners and unidentifiable obstacles:

- "Legs" are all obstacles with a round shape whose diameter is less than 15 cm. Whether the obstacle is a chair, a table or any other piece of furniture has no importance to the robot.

- “Walls” are all obstacles with a straight edge. Hence this class can also contain obstacles like a cupboard or a bookcase.
- “Corners” are the junctions between two obstacles classified as walls. Corners can be either convex or concave.
- “Unidentifiable obstacles” are all obstacles that do not fit into these categories. In future works, it will be possible to identify some of these obstacles into new and more specific classes.

This identification of obstacles can be done relatively easily by using the arm and its tactile sensor system. With a simple algorithm based on where the cleaning head successively touches the obstacle, the robot is able to conduct a first, often correct identification.

Afterwards, the result is verified by contouring the obstacle with the cleaning head. This process allows the robot to create a local bitmap representation of the obstacle by using the forward kinematics and the tactile sensor data. This local map not only verifies the identification but also gives some other interesting information to the robot. In the case of a wall, it gives the distance and the angle between the robot and the wall. In the case of a leg, it is easy to extract its diameter. Furthermore, the robot simultaneously cleans closely around the obstacle. During this process, the robot benefits from the fact that its platform can stay immobile. If the robot itself had to move around, not only the odometry would be worse but also the quality of the local map due to the loss of the reference point.

An interesting analogy exists between this mobile robot and a blind person. If a blind person enters an unknown room, he uses his arms or his cane but not his body to explore the environment. The use of his arms allows him to keep his body as a point of reference during the exploration. Moreover, his hands have a large density of tactile sensors which allow him to identify objects.

3.2 Using hypotheses

Knowing the identity of an obstacle, the robot can make some hypotheses which considerably help to create a coherent map. To avoid problems due to an incorrect identification, the hypotheses are verified during their use.

When the robot identifies an obstacle as a wall it can make the hypothesis that the wall is straight. Following the wall, the robot can measure its length accurately by using its odometry which is very precise and reliable for distance measurements. Knowing that the obstacle is a wall and knowing its length, it is easy to draw a straight line into the bitmap representation. It is even possible to recalibrate the odometry at its end.

As most walls are straight, this hypothesis is correct in most cases. But what happens if the wall is not straight but has a small curvature? In this case the robot will identify it as a wall in a first step, as it seems to be locally straight. However, even though this obstacle is a wall it should not be classified as a “wall”. Hence the identification is wrong and the robot will make a wrong hypothesis. But after having followed the “wall” for a certain distance, the robot will detect that its odometry changed too much for an expected straight trajectory and it will reject the hypothesis. The obstacle will then be treated as an unidentifiable obstacle.

Another important hypothesis is that most of the corners have an angle of 90°. When the robot arrives at a corner, it measures the angle with its arm. But this measurement contains an error of several degrees. If the measured angle is between 80° and 100° the robot will assume that the angle is 90°.

Even if the corner had in reality an angle of 92°, it would still be better to assume that it was 90° than to use the real value. Unless the wall is very long, the error due to the hypothesis will not be very important. Corners with an angle of 90° are much more convenient and facilitate the generation of a coherent map.

3.3 Map creation

The only way to guarantee a complete cleaning is by using a map in which the robot can mark cleaned areas. The creation of a map is a well-studied subject in mobile robotics. Many interesting methods are described by [Holenstein, 1992] and [Knieriemen, 1991].

The major problem of creating a map is to keep it coherent. Primarily, it must contain all obstacles, the exact position is secondary. It is more important that the relative positions of the obstacles are right.

The easiest way to create a coherent map is to contour the room first and then to explore its interior. The map of the contour will be a reference for further obstacles being detected inside the room. With a map of the border, the robot will always be able to determine its position even when it gets completely lost. Hence, the major problem of a mobile robot, getting lost, will be solved.

Therefore, the problem of creating a coherent map reduces to the problem of creating one of the room's border. Hence, the major problem for the robot is to detect when it comes back to the starting point. Otherwise it risks to run around the room until its batteries get empty.

3.3.1 Bitmap representation

The easiest method to memorize cleaned areas is certainly a bitmap approach. Used for unknown environments, bitmaps are also more likely to yield results in real-time [Hoppen]. In our implementation, each pixel has a value that characterizes the environment in the following way:

Value	Class
0	unknown (not cleaned)
1	empty and cleaned
2	empty but not cleaned yet
3	wall
4	concave corner
5	convex corner
6	leg
7	unidentifiable obstacle

A bitmap representation also allows us to memorize the shape of unidentifiable obstacles by putting the local image, created by contouring the obstacle with the arm, into the global map. This procedure is much simpler and faster than to memorize a complex obstacle with a polygonal representation. In addition, the bitmap approach also permits some interesting algorithms for the navigation [Jarvis, 1993] and especially the cleaning of the open surface (section 4).

The implemented map uses pixels of 2 cm by 2 cm. Using 8 values for a pixel requires only about 245 kilobytes of memory for a surface of 100 m^2. In case of a lack of memory or a much bigger surface, it is possible to reduce the amount of memory by combining areas of surfaces without obstacles. A common method is called “Quadtree” [Hoppen, 1992].

3.3.2 List representation

To facilitate some of the implemented algorithms, the robot also creates a list for each class of obstacles. In our implementation, the robot uses four lists, each containing specific attributes.

The list of walls is very similar to a polygonal representation but with the difference that it contains only obstacles with straight edges and not all obstacles. It contains all obstacles classified as walls with the following attributes:

- global position x-y of the starting point
- global position x-y of the ending point
- length
- orientation
- compass values

Some of the attributes are redundant, for example, the length and the two end points. This redundancy is desired as it makes possible repeated instant access to any attribute, without having to recalculate it. This speed-up requires a higher amount of memory. But the required memory for the lists is small compared to the amount needed for the bitmap representation.

The list of corners contains the global positions x-y and the corner angles. Similarly the list of legs contains the global positions x-y and the leg diameter.

These lists are very useful when the robot gets lost. The robot must try to move towards the border of the room. If the robot arrives at a wall, it tries to identify it by following it until the next corner. Comparing the values of the compass and the minimum length to the attributes stored in the list of walls, the robot can already eliminate some of the potential walls. If there is more than one wall left, the robot measures the angle of the corner, compares it to the list of corners and can then maybe again eliminate some of the possible walls. If the identification is still not unique, the robot follows the next wall until its end. Using again the values of the compass and the length of the wall, the identification of the wall should be unique in most cases. Otherwise, the robot goes on with this procedure until the identification is unique.

Once the wall(s) and corner(s) are identified by using the lists, the robot knows where it is relative to its global map and can hence reinitialize its odometry with the help of the last corner. To relocate itself by using the bitmap representation instead would be much more complicated and more time intensive.

The list of corners is especially useful to recalibrate the odometry of the robot. When the robot needs to recalibrate its odometry, this list allows it to easily find the nearest corner which is all that the robot needs. To find the nearest corner could also be done by a bitmap approach, but again, this would be more complicated and slower.

3.3.3 Correction of the wall lengths

Due to the hypotheses of the straight walls and the perpendicular corners, it is very easy for the robot to detect when it is back to the starting point. Nevertheless, the measurements of the lengths of the walls are not perfect. Also only few corners are exactly perpendicular in reality. Hence, the coordinates of the starting point and the ending point are not exactly the same. The typical errors in our experiences were about 5 cm for a perimeter of the cleaning area measuring 6 meters.

Now the robot can benefit from the fact that these two points must be the same, or in other terms that the vector sum of all walls must yield zero. Actually, during the contouring of the room, the robot measures all the distances twice. For example, if the robot first goes south for some distance, it will have to go north later for the same distance in order to come back to the starting point. Hence, the robot can average all the x-projections and y-projections to calculate the new corrected lengths of the walls. These new lengths are more accurate as they are obtained by an average of two measurements.

Doing this correction algorithm we get two very useful results. First the lengths of the walls are more accurate and second the ending point exactly matches the starting point. Hence we get a closed image of the room's border which will be convenient for the exploration and cleaning of the interior.

3.3.4 Result of map creation

The combination of the bitmap and the list representations allows the optimization of the different tasks by using the best suited representation. It also allows us to apply and to combine existing algorithms for the bitmap approaches and the polygonal models.

Due to the hypotheses based on the identification, the detection of the starting point is very reliable and the robot is able to generate a coherent map of the environment's border. This map is memorized in two different forms, bitmap and list representation, both containing the exact same information if there were no unidentifiable obstacles.

The good quality of the generated map insures that the robot will not get lost and greatly facilitates the next step, the cleaning of the interior.

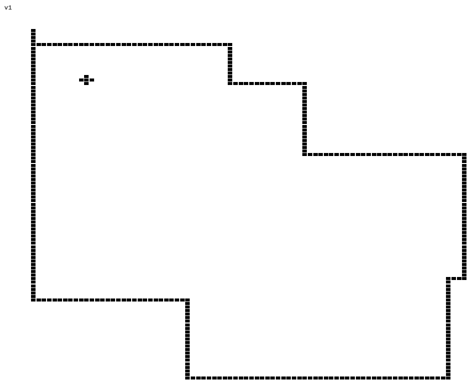
An example of a bitmap representation created by the robot is shown in figure 9. The cross at the upper left side is the position of the robot after having contoured the room. The robot used only its odometry and its tactile sensor system to create this accurate map. The difference between the starting and ending point in this example is only 6 cm and 1 cm in the two main directions for a contour of about 6 m. The errors on the individual wall lengths is less than 2 cm, the size of a pixel. During the whole exploration, the robot never needed more than 5 ms to choose the next action.

4 The cleaning operation

There are many different ways to clean a surface, or simply cover it.

One way would be to use the created map of the border and to find the shortest path to cover the whole surface by using well-known AI techniques. But this approach would not work well, as the interior of the area is not known. There will likely be some obstacles in the middle of the room that have not been detected during the exploration of the border. Hence this so called optimal path would have to be recalculated each time the robot comes across a new obstacle. In addition, as the first generated path is based on incomplete information, the final path will be far from optimal in many cases.

In this work, we used another approach which is easy to implement and furthermore allows the robot to often recalibrate its odometry. To simplify the navigation of the robot, it is preferred


v1