Aspects on path planning for mobile robots

Prof. Gh. Lazea; As. E. Lupu
Technical University of Cluj-Napoca
Automation Department

I. Introduction

In the last twenty years mobile robots have become a subject of significant interest because they rise a multitude of challenging problems on one hand and because they are open to a broad range of possible applications on the other hand. The research devoted to mobile robots started in three different domains. The first was the need for more flexible automated guided vehicles (AGVs) in manufacturing plants. Today's customers expect products in a variety of options and also fast delivery. The rising cost of keeping large inventories has stimulated the development of just-in-time manufacturing. Mobile robots can be a solution to the transport problems in such environments where parts and sub-assemblies must be quickly transferred between manufacturing cells. A second domain to stimulate research in mobile robots was that of planetary rovers. These vehicles are used to perform exploratory actions in places which are unreachable or dangerous for humans and the ideas developed in spatial projects have also been applied to terrestrial maintenance robots used in nuclear power plants. The third domain was the research into 'intelligent systems' also generally called 'artificial intelligence'. This is due to the fact that mobile robots were found to be an appealing testbed for various artificial intelligence techniques. In these projects the emphasis was more on the decision making or motion planning and not so much on motion control.

A definition of a mobile robot can be derived from that of the classical robots:

"A robot vehicle capable of self propulsion and (re)programmed locomotion under automatic control in order to perform a certain task".

The keywords are that a robot is programmable to execute a variety of tasks and that it does so under automatic control (as opposed to human control).
There is a large diversity of mobile robots. Figure 1 illustrates a division based on vehicle characteristics [23]. A first important distinction is made between guided and non-guided vehicles. A guided vehicle is restricted to a set of predefined paths in its working area. These paths can be indicated by tracks, magnetic or optical lines, inductive guidepaths or a sequence of movements stored in memory. The vehicle may leave this path in no circumstance. The AGVs form this class. Non guided vehicles are not restricted to any predefined guidepath.

The largest category in the division in figure 1 is by far that of land mobile robots equipped with wheels. The sequel of this paper deals with path planning methods for such robots, and from here on the term 'mobile robot' will be used to refer only this category of wheeled free-ranging vehicles.

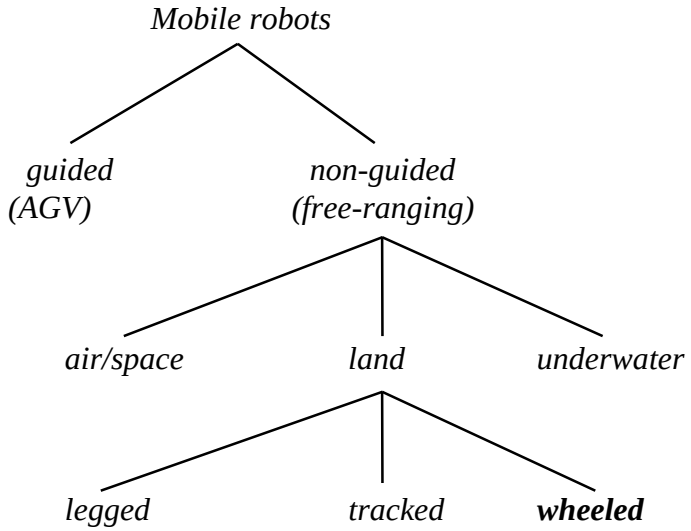


Figure 1.

As already mentioned mobile robots cover a broad range of applications:

- Material handling and transport applications in industry, harbours, airports, warehouses, offices, hospitals, libraries, etc.
- Inspection, service and maintenance applications.
- Exploration and reconnaissance actions in military or extraterrestrial environments.
- Construction robots, mining robots.
- Lawn mowers, vacuum cleaning robots.
- Autonomous wheelchairs for the disabled.
- Guard and surveillance robots.

The main subjects of research in the field of mobile robots today are:

- Path planning and trajectory planning.
- World modelling (environment mapping).
- Position and course estimation (localization).
- Obstacle avoidance.
- Sensor fusion.
- Distributed real-time control.

The basic problem of a mobile robot is that of navigation: moving from one place to another by a coordination of planning, sensing and control. In any navigation scheme the desire is to reach a destination without getting lost or crashing into anything. Put simply the navigation problem is to find a path from start (S) to target (T) and traverse it without collision. Navigation may be decomposed into three sub-tasks: mapping and modelling the environment; path planning and selection; and path following and collision avoidance. The relationship between these tasks is shown in figure 2.

A higher level process (in current applications - a human) specifies the destination and any constraints on the course, such as time (these form the 'Task'). If the robot does have a map it searches for possible paths, chooses an appropriate path

and traverses that path. Otherwise the map must firstly be constructed by exploring the environment. While traversing the path the robot must check it with sensors for uncharted obstacles. If such obstacles are found evasive actions must be taken.

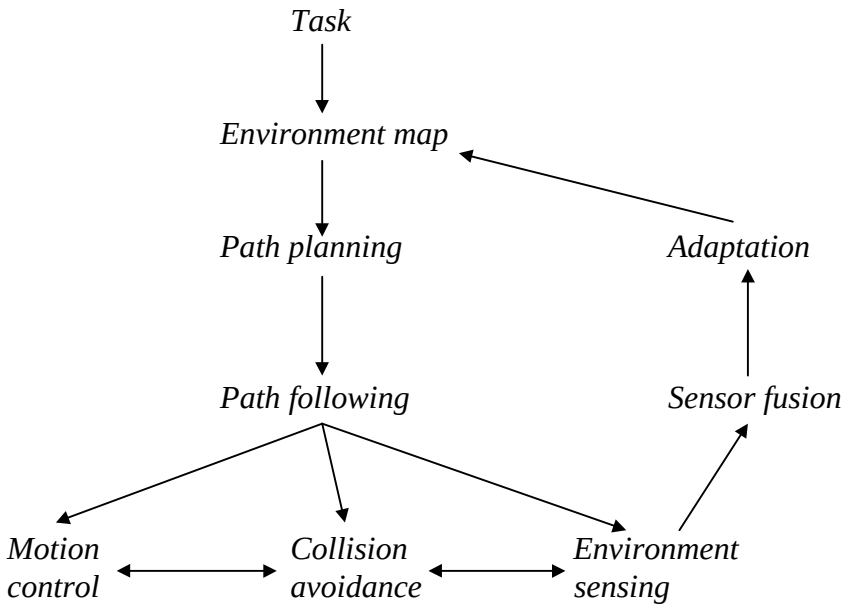


Figure 2.

In the sequel we will present some aspects (including original results) concerning mobile robots path planning.

A path planning algorithm for a mobile robot has as input data two coordinates (the actual robot position **start** and the desired robot position **target**) and must yield as output a possible path between these two points. Related to the apriori knowledge about the environment in which the robot is supposed to travel, one can distinguish between two different cases:

1. The environment is completely known;
2. The environment is completely unknown;

The path planning algorithms for the former case are called *path planning with complete information, find-path* or *piano-mover problem* algorithms, while for the latter, the problem of finding a path between a start location and a target location is called *path planning with incomplete information* or *path planning with uncertainty*. The information about the environment (either apriori known either acquired by sensing the environment) is represented by the means of a map. There are known very many ways to represent the spatial information using very diverse data structures. It has been proven that the efficiency of the different kinds of path planning algorithms developped till present is closely related to the type of data structures used to store the map. In table 1 are presented some of the well known data structures used in path planning and the corresponding algorithms:

	Data structure	Algorithm
1.	Array of pixels	Distance transforms; Potential field methods
2.	Vertex list	Search on visibility graphs

3. Hierarchic (ex: quadtree) Graph search

Table 1.

It is possible of-course to use any algorithm with any of the data structures shown but the efficiency reflected in the processing time will not be the best one.

The problem of path planning with complete information has received much attention from researchers until present. The use of the vertex list data structure associated with search on visibility graphs and the concept of "Configuration Space" were described in [4], and [11]. More recently an improved visibility graph termed "tangent graph" has been proposed [12]. One of the first papers to describe the use of bitmaps and distance transforms in path planning was [7]. More recently the same approach (also developed by us) was used to deal with changing environments [5]. The use of bitmap data structures and artificial potential field based algorithms is described in [1]. Time varying artificial potential field algorithms for moving obstacles are described in [9]. The use of quadtree data structures in path planning has been reported by S. Kambhampati [8] and more recently by S. Ghoshray [6]. The problem of planning with incomplete information has been tackled among others by V. Lumelsky [13], [14], B. H. Krogh and D. Feng [10], S.S. Iyengar et. al. [18], [19] and more recently by A. Bemporad [2] and Y.S. Nam et. al. [17]. A complete navigation system (used to drove the Navlab II robot) which combines both global and reactive planning is described in [21]. In [16] is given a good survey about path planning algorithms and data structures and in [20] are given algorithms for transformations between different kinds of data structures.

II. Path planning with complete information

The algorithms to solve this problem can be used only in very well structured environments such as laboratories or CIM factories with a perfectly known configuration and where exist the certainty that unknown obstacles cannot appear.

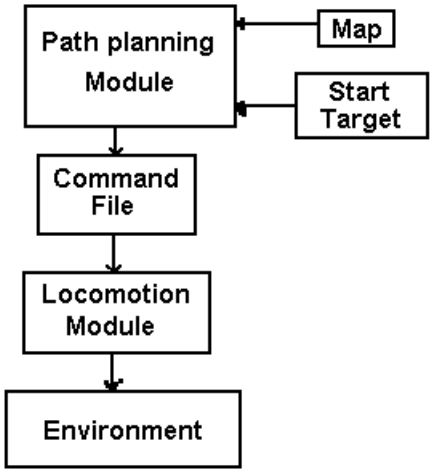


Figure 3.

In figure 3 is presented a control architecture for a mobile robot in such an environment. It is a simplified architecture in which certain modules (for example the localization module) were not shown.

In the present paper the data structure adopted is the pixels array and the algorithm is one based on distance transforms. These will be largely described in the sequel.

Representing the spatial information by the means of a pixel array is quite straightforward: the bidimensional space is divided in equal squares each square being represented by a pixel. One pixel has a well defined colour, that is it represents either free or occupied space, and is represented in the computer memory by a number, namely the number associated with its colour. This number is an element in a matrix with dimensions equal to that of the image and his position within the matrix reflects the position of the square represented by the pixel within the image. In our work, for simplicity reasons we used monochrome bitmaps in which obstacles are denoted by black colour with associated value 0 and the free space is represented by the white colour with associated value 1. This way one pixel occupies one bit in the computer memory. The reason for choosing raster image as data structure is related to the fact that most sensors (image sensors and ultrasonic and laser range finding sensors) yield such data. Also, at the time present a very large number of software packages for bitmap image processing exist and this way the construction of a computer bitmap image from existing drafts, maps and experimental measurements is not a big problem for an operator. In figure 4 is presented such a raster image (resolution 200x200) which we have used to test the path planning algorithms.

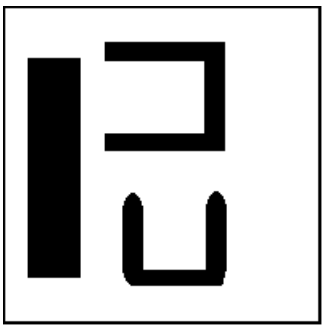


Figure 4.

The use of bitmap files as data structures has also disadvantages related on one hand to the large computer memory they require and on the other hand to the fact that the representation is not exact because of the digitization.

The path planning algorithm we have implemented is based on distance transforms [24], [25], techniques used especially in digital image processing. The distance transforms were first introduced by A. Rosenfeld almost 20 years ago. They define for each point of an image a distance to a reference point or to a set of reference points. The distances are defined by metrics. In the usual continuous 2D space the Euclidean metric is used. In digitized spaces this metric is approximated by the so called chamfer metrics introduced by Borgefors [3].

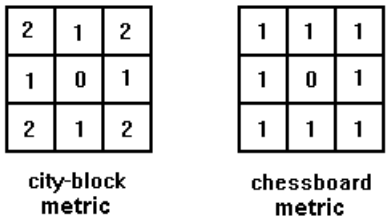


Figure 5.

In figure 5 are defined two of the most frequently used chamfer metrics: the city-block metric and the chessboard metric, and in figure 6 is represented a circle using these two metrics and the Euclidean metric.

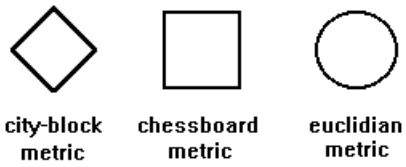


Figure 6.

In [25] it is shown that the maximal error introduced by approximating the Euclidean metric by any of these two chamfer metrics is 17,6 %. The algorithm we have implemented uses the city-block metric. The length of a path between two points in a raster image is found as the sum of the elementary metrics between the two points. In figure 7 is presented a raster image containing both occupied space (obstacle) pixels and free space pixels. For the latter the distance (the path length) to the arbitrarily chosen point of coordinates (2,2) in terms of city block metric is shown. An 8 by 8 image was considered with the coordinates origin arbitrarily chosen in the NW corner (upper left).

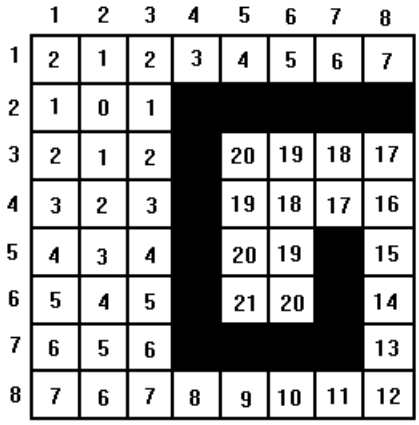


Figure 7.

As it can be seen the obstacle pixels are not considered -are avoided- this being the reason why some authors [25] call this transform "constrained distance transform". By using this technique the path planning algorithm can be described as it follows:

1. Build the distance transform map having **start** as reference point until one of the following occurs: a) the **target** is reached.
b) the whole reachable free space has been mapped.
2. If b) the algorithm stops with the mention that a continuous path between **start** and **target** cannot be found.
3. If a) process the matrix associated to the distance transform from **target** to **start** in the gradient direction (backtracking) saving the coordinates of the processed points.

A unidimensional matrix containing the coordinates of the pixels which form a path between **start** and **target** is this way obtained.

We have implemented the algorithm using the Microsoft C language on a PC machine. The program has as input data a monochrome raster image file in Windows Bitmap format (.bmp) representing the environment and the coordinates of the start and target points with the coordinates origin assumed to be in the upper left corner. The program yields as output also a monochrome bitmap file in which the path found by the algorithm is overexposed on the initial map. Besides this the program furnishes a command file (cmd.txt) which can be used by the locomotion module of a mobile robot to actually move the robot along the path. The path produced by the algorithm is represented by a string of points. This way, it is assumed that the robot is a point or more precisely his dimensions can fit in a square represented by a pixel in the bitmap file. Also, because of this, the problem if the robot is a holonomic or nonholonomic vehicle is not taken into account. The restriction implied by this is that the program would be suited only for environments in which both the obstacles and the free space are large compared to the robot dimensions (yet such a case is frequent in CIM plants). This is the case if the algorithm is used on a map which describes the actual environment in which the robot travels. But if an imaginary map is built -called in [11] the configuration space map- in which the obstacles are "grown" with the dimensions of the robot, the algorithm can be used also in situations were the robot dimensions should be considered, for example in cluttered environments. The path yielded by the algorithm in this case is that of a holonomic motion without rotation. For a differential mobile robot such a path must be preprocessed in an additional module in order to be executed.

From the results furnished by the algorithm we have seen that it tends to yield paths tangent to obstacles. This is in fact one of the disadvantages of this algorithm: it does not use efficient the free space, furnishing paths too close to obstacles (i.e. it is not robust). In a practical implementation due to the inherent errors of the locomotion and the localization modules this can yield to frequent collisions. In order to diminish this behaviour the program we have realised uses not the real map but a processed copy of it in which all the obstacles are grown with one pixel. This preprocessing is done with a function ("grow") and has the meaning to force a minimum space of one pixel between the generated path and any obstacle. By calling repeatedly the function "grow" this "ensurance space" to obstacles can be enlarged at will but under the reserve that certain narrow free space corridors can be missed.

The construction of the matrix associated to the distance transform is done using an original iterative algorithm, using as temporary variable an array for storing all the points equidistant to the start point. The dimension of the matrix associated with the distance transform (coded "asociat") equals the image dimension but it is a matrix of double words, not of bits. For example, for a 600x400 pixels image, "asociat" stretches over approximately 480 Ko. The allocation of this matrix in the computer memory is done dynamically in the conventional memory space using the function huge allocator (halloc). The standard bound of 640 Ko of the PC's conventional memory limits the maximum dimension for the images which can be processed by the program. The solution to the memory allocation problem can be further improved by using the extended memory beyond the 1 Mo bound.

The backtracking algorithm is based also on the city-block metric with eight neighbours. Because of this, the path planned in free space is composed of straight line segments oriented according to one of the following directions: S-N, E-W, NW-SE, NE-SW. This is certainly a limitation but it simplifies the command file generation. By using more advanced chamfer metrics (with more than eight neighbours for a pixel) both when constructing the "asociat" matrix and for the backtracking algorithm, it will be possible to improve the path (i.e. to obtain shorter paths). The fact that the path is made up from segments has dictated the way the command file is generated. As one would naturally expect this file enumerates these segments and their orientation, according to the following structure:

direction distance xcoord ycoord

Here **direction** is a number representing the absolute direction (with respect to the map):

5	6	7
4	x	0
3	2	1

or equivalently:

E	SE	S	SW	W	NW	N	NE
0	1	2	3	4	5	6	7

distance is the segment length in pixels. The distance the robot will actually travel is :

k·distance for the N-S and E-W directions

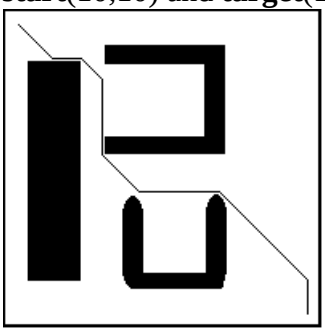
√2k·distance for the NW-SE and NE-SW directions

where k is a scaling factor equal to the edge of the square represented in the map by one pixel.

xcoord and **ycoord** are the absolute coordinates of the end of the segment.

Although the first two columns of the table (**direction** and **distance**) should suffice for instructing the robot to follow the path, due to the error accumulation this would be unreliable. This is why the absolute coordinates were added to the file. Based on them a localization system (based on beacons for example) will be able to correct such inherent errors.

The time needed to compute a path depends only by the image size and the percentage of free space (time maximal for 100% free space), and it increases exponentially with the image dimension. For a 600x400 bitmap file the computation time on a 486 PC at 80Mhz is about 5 seconds including the I/O. In figure 8 is presented the path found by the algorithm for the test image considered between **start**(10,10) and **target**(190,190) and also the command file generated.



0	0	10	10
1	21	31	31
0	18	49	31
1	13	62	44
2	47	62	91
1	23	85	114
0	50	135	114
1	55	190	169
2	21	190	190

Figure 8.

III. Path planning with incomplete information

When there is no apriori information about the environment in which the mobile robot is supposed to travel the problem of finding a path between a start point and a target point is called *path planning with incomplete information* or path planning with uncertainty. The initial data are in this case only the coordinates of the two points or equivalently the direction and distance to the target. The problem can be solved only if the robot is equipped with some sensor system which can detect the obstacles around. Examples of such sensors are: touch sensors, range finding sensors (based on triangulation or time of flight techniques), video cameras. The problem of planning a path in these conditions becomes an on-line problem in which the planning takes place as the robot moves in the unknown environment. The use of a sensor means in fact a feed-back control with all the advantages which come from such an approach. The major problem which must be solved in this kind of planning is no more to minimise the time to find a path, as it is in the path planning with complete information approach where the process takes place only once off-line and requests large memory and lenghty computation times, but to ensure the convergence. This means that if a path between **start** and **target** exists, the algorithm must find it within a limited period of time and if not, the algorithm must furnish a message that a path between **start** and **target** does not exist also within a finite period of time.

A control architecture for a mobile robot guided exclusively by this kind of planning -also known as reactive planning due to the use of the negative reaction from the environment through the sensor- is presented in figure 9.

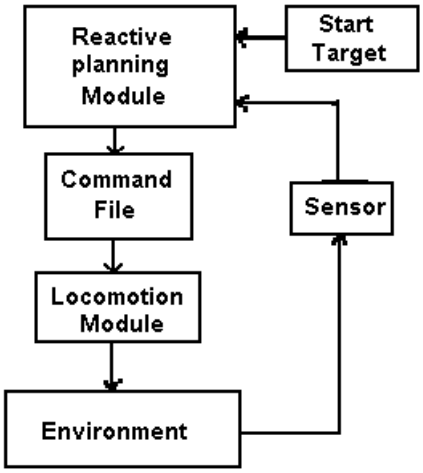


Figure 9.

In [13] V.Lumelsky proposed for the first time an algorithm to solve the path planning with uncertainty problem for which he demonstrated mathematically the convergence. The algorithm -named by its author BUG2- considers the robot as a point automaton equipped with a tactile sensor operating in an environment with unknown obstacles of arbitrary shape. This means that the robot detects the obstacles only when it hits them. The algorithm is valid also for other kind of sensors -for example the ultrasonic sensor- but it cannot use the richer information provided by such sensors. Later, in [14] the same author proposed an improved variant of BUG2 for robots equipped with range finding sensors.

We have realised a program which simulates the behavior of robots guided according to the BUG2 algorithm. The implementation was done using Microsoft C and we have used the simulator to test the paths generated by the algorithm for different environments.

The algorithm BUG2 is relatively simple and it consists mainly in the advance towards the target along the segment which connects the start with the target. If an obstacle is being hit this is avoided always on a previously decided direction (left or right) until the target or the segment is found. The exact algorithm described in pseudocode [13] consists of the following steps:

j=1; L_0=**start**;

1. From point L_{j-1} move along the straight line segment **start**-**target** until one of the following occurs: a) **target** is reached. The procedure stops.

b) An obstacle is encountered and a hit point, H_j, is defined. Go to Step 2.

2. Using the accepted local direction, follow the obstacle boundary until one of the following occurs: a) **target** is reached. The procedure stops.

b) The line segment **start**-**target** is met at a point Q such that the distance from Q to **target** is less than that from H_j to **target**, and the line segment Q-**target** does not cross the current obstacle at the point Q. Define the leave point L_j=Q. Set j=j+1. Go to Step 1.

c) The robot returns to H_j and thus completes a closed curve (the obstacle boundary) without having defined the next hit point, H_{j+1}. The target cannot be reached. The procedure stops.

The program we have produced determines the path a mobile robot guided according to the BUG2 algorithm would find in an environment described by a bitmap file using the same conventions we have established for the program solving the planning with complete information problem. The change we have made to the algorithm consists in the pre-growing of the objects with one pixel which simulates a robot equipped not with a tactile sensor but with a range finding sensor with the maximum range-over distance of one pixel. This processing is done using the same function ("grow") as in the previous program. The program uses an original algorithm for obstacle contour following in which a look-up table is employed to improve the computation speed. The **start-target** segment is computed iteratively using the Bresenham line drawing algorithm described for example in [22].

Because of the iterative way in which the path is computed, the program does not rise memory allocation problems, theoretically maps of any dimension being possible to be processed. In figure 10 is presented the trajectory found by the program BUG2 for the test map in figure 4.

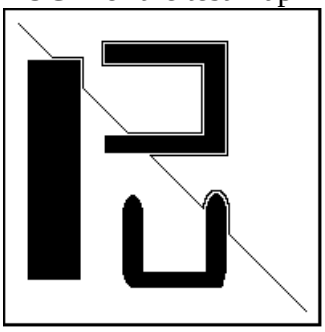


Figure 10.

One can easily remark that the trajectory obtained is quite far from an optimal one. This is due to the algorithm itself which is local by its nature. Even with the improvements proposed in [14] the BUG2 type algorithms can determine at best locally optimal trajectories which are not always globally optimal as well. A comparison with the trajectory yielded by the program which implements the path planning with complete information algorithm is meaningless, the two problems being different (more precisely complementary). This suggests that a hybrid approach which can compound the advantages of apriori information based path planning with reactive planning is the best solution for controlling mobile robots. Two architectures to implement such an approach are presented in figures 11 and 12.

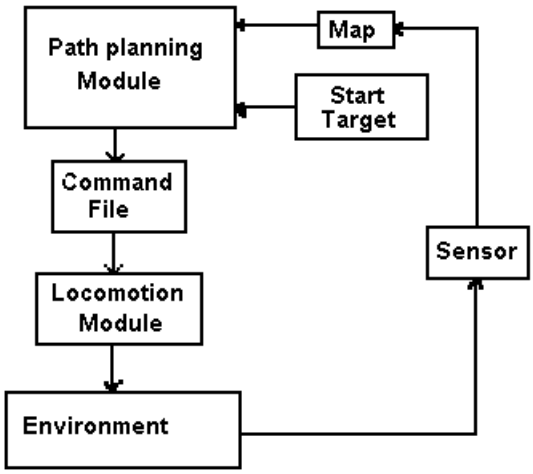


Figure 11.

The architecture depicted in figure 11 is a variant of that presented in figure 3 and it includes the on-line sensing of the environment. The controlling algorithm is the one presented in the first part of the paper or an equivalent one but it is repeatedly run in a sensing-planning-moving cycle in which the map of the environment is permanently updated. The limitation implied by this architecture is that the sensor must supply a global view of the environment (difficult to achieve). A solution might be the use of several usual sensors and the integration of their output in a global map. Usually such sensors are static, they are not mounted on the mobile robot, and this rises additional transmission problems.

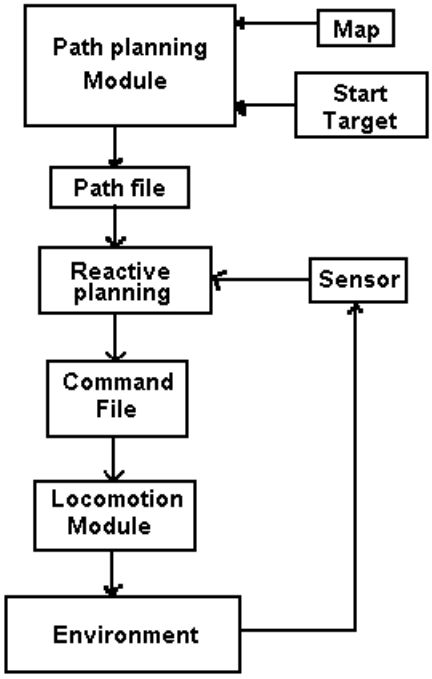


Figure 12.

The architecture presented in figure 12 is the one we support. It is a hierarchic architecture in which coexist both a module for planning with complete information and a module for local reactive planning. The basic path is decided by the former module, the latter being responsible only for the uncharted obstacles avoidance. It is

possible this way to integrate the two basic path planning strategies. The sensor used is an on-board mounted sensor and this facilitates the goal of building a true autonomous mobile robot. The architecture combines the advantages of the two types of algorithms and opens the opportunity of using a multiprocessor hardware.

Future research can be conducted to improve the global planning by using more advanced distance transforms and incorporating true range sensing functions, to improve the computation time and minimize the memory usage by using multiresolution data structures and to post-process the path according to actual geometric and dynamic constraints of the mobile robot.

References

[1.] J. Barraquand, J.C. Latombe, "Robot motion planning: a distributed representation approach" in Int. Journal of Robotics Research, vol. 10, no. 6/1991.

[2.] A. Bemporad, A. De Luca, G. Oriolo, "Local incremental planning for a car-like robot navigating among obstacles" in Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, USA, 1996.

[3.] G. Borgefors, "Distance transformations in arbitrary dimensions" in Computer Vision, Graphics and Image Processing 27/1984.

[4.] R. A. Brooks, T. Lozano-Perez, "A Subdivision algorithm in configuration space for findpath with rotation" in IEEE Trans. on Systems, Man and Cybernetics, no. 2/1985.

[5.] G. Conte, S. Longhi, R. Zulli, "Motion planning and simulation of articulated and mobile robots" in Proc. of the 3rd Int. Workshop on Robotics in Alpe Adria Region, Bled, 1994.

[6.] S. Ghoshray, K.K. Yen, "A comprehensive robot collision avoidance scheme by two-dimensional geometric modelling" in Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, USA, 1996.

[7.] G. Honderd, W. Jongkind, C.H. van Aalst, "Sensor and navigation system for a mobile robot" in Proc. of Intell. Autonomous Systems Conf., Amsterdam, 1986.

[8.] S. Kambhampati, L.S. Davis, "Multiresolution path planning for mobile robots" in IEEE Journal of Robotics and Automation, no. 3/1986.

[9.] O. Kathib, "Real time obstacle avoidance for manipulators and mobile robots" in Int. Journal of Robotics Research, vol. 5, no. 1/1986.

[10.] B.H. Krogh, D. Feng, "Dynamic generation of subgoals for autonomous mobile robots using local feedback information" in IEEE Trans. on Automatic Control, vol. 34, no. 5/1989.

[11.] T. Lozano-Perez, "Spatial planning: a configuration space approach" in IEEE Trans. on Computers vol. C-32, no. 2/1983.

[12.] Y.H. Lui, S. Arimoto, "Path planning using a tangent graph for mobile robots among polygonal and curved obstacles" in Int. Journal of Robotics Research, vol. 11, no. 4/1992.

[13.] V.J. Lumelsky, A.A. Stepanov, "Path Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape" in I.J.Cox, G.T. Wilfong - "Autonomous robot vehicles", Springer Verlag, 1990.

[14.] V. Lumelsky, T. Skewis, "Incorporating Range Sensing in the Robot Navigation Function" in IEEE Trans. on Systems, Man and Cybernetics, vol. 20, no. 5/1990.

[15.] P.J. McKerrow, "Introduction to Robotics", Addison Wesley, 1995.

[16.] J.S.B. Mitchell, "An algorithmic approach to some problems in terrain navigation" in Artificial Intelligence 37/1988.

[17.] Y.S. Nam, B.H. Lee, M.S. Kim, "View time based moving obstacle avoidance using stochastic prediction of obstacle motion" in Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, USA, 1996.

[18.] B.J. Oommen, S.S. Iyengar, N.S.V. Rao, R.L. Kashyap, "Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case" in IEEE Journal of Robotics and Automation, vol. RA-3, no. 6/1987.

[19.] N.S.V. Rao, S.S. Iyengar, "Autonomous robot navigation in unknown terrains: incidental learning and environmental exploration" in IEEE Trans. on Systems, Man and Cybernetics, vol. 20, no. 6/1990.

[20.] H. Samet, "The Design and Analysis of Spatial Data Structures", Addison-Wesley, 1990.

[21.] A. Stentz, M. Hebert, "A complete navigation system for goal acquisition in unknown environments" in Proc. of IROS'95, 1995.

[22.] R.T. Stevens, "Graphics programming in C", Addison-Wesley, 1989.

[23.] P. van Turennout, "Autonomous Motion on Wheels", Ph.D. Thesis, Technische Universiteit Delft, 1994.

[24.] B.J.H.Verweer, P.W.Verbeeck, S.T.Dekker, "An efficient uniform cost algorithm applied to distance transforms" in IEEE Trans. on Pattern Analysis and Machine Intell., 11/1989.

[25.] B.J.H.Verweer, "Local distances for distance transformations in two and three dimensions" in "Distance transforms" Ph.D. Thesis, Delft University of Technology, 1991.