Qatar Univ. Sci. J. (1995), 15 (1): 11 - 21

PATH PLANNER FOR A MOBILE SHAPE-CHANGEABLE INTELLIGENT BOBOT IN A 3-DIMENXIONAL ENVIRONMENT

By

M.G. ABOU-ALI*, I.A. AWAD**, M.N. EL-DERINI***, AND E.H. ABO EL-ELA***
* Faculty of Engineering, Alexandria University, Alexandria, Egypt.
** Computer Science & Automatic Control Department, Faculty of Engineering, Alexandria University, Alexandria Egypt.
***Currently: Department of Computer Science, Faculty of Science, University of Qatar, Doha, Qatar.

مخطط مسار منفذ آلي ذكي متنقل نحو أوضاع متغيرة في بيئة ذات ثلاثة اتجاهات

محمد جابر أبو علي و إبراهيم عبد السلام عوض و محمد نزيه الدريني
و عماد حسن أبو العلا

يتعامل هذا البحث مع مشكلة تخطيط مسار المنفذات الآلية . وتعتبر هذه المشكلة أساساً واجباً يسبق تحديده ثم يتم التحكم في المنفذ عند تنفيذ التعليمات اللازمة للوصول إلى الهدف . وترتكز هذه الدراسة على تجزئة الحل إلى ثلاثة مراحل .

فأولاً يتم تمثيل المنفذ الآلي والبيئة التي يتحرك بها ، ثم يتم الحصول على المسار الأمثل للمنفذ الآلي طبقاً لموضعه الابتدائي وموضعه النهائي . ويقصد بالمسار الأمثل المسار الذي به أقل صعوبة تواجه المنفذ الآلي خلال مساره . وأخيراً يتم تحديد الحركات المتبعة بالمنفذ الآلي خلال مساره المختار . ويعتبر العمل المقدم بهذا البحث بناءاً لنظام مكتمل لحل هذه المشكلات الفرعية . وقد تم تطوير هذا النظام أساساً باستخدام هيكل نظام خبير هدفي الاتجاه .

Key Words: Robot, Mobile robot, Knowledge-based systems, Expert systems, Intelligent systems, path Planning; Shortest route.

ABSTRACT

This study deals with the path planning problem in robotics. This problem is a basic problem to solve some pre-specified task, and then controlling the robot as it executes the commands necessary to achieve its goal. This study concentrates on three sub-problems of the path planning problem. The first one is the representation of the robot and the environment in which it moves. The environment is represented according to a given map of boundaries and obstacles. The second sub-problem is to find the optimal path for the robot according to its initial and final positions. Optimal path means the path with minimum total hardness faced by the robot through its route. The last sub-problem in this study is about determining the course of actions to be followed by the robot through the candidate route. The work reported in this paper is about building a complete system that solves the previous sub-problems. This system has been developed mainly using an object oriented expert system shell.

INTRODUCTION

The task of path planning in mapped environments is well known in robotics. Given the robot specifications and a set of obstacles located in space that represents the environment map, the problem is to find the optimal path for the robot from the initial position to any destination position in the environment. Optimal path means the path ,with minimum total hardness faced by the robot through its route.

For a shape changeable robot, the hardness of any region in the environment is measured by how much effort the robot suffers to pass by this region. Relative values can be assigned to the hardness of each regions. The minimum value of the hardness is assigned to the regions where the robot can turn around without changing its configuration. While the maximum value is assigned to the regions where the robot must take the most difficult action with a complex configuration.

Finding the optimal path is followed by producing the set of actions to be followed by the robot through the candidate path. This will increase the integrity of the system by adding the interface to the robot as well as the environment map.

Many researches have discussed in depth the path planning problem by using different approaches. Many computer algorithms have been suggested to solve each of the sub-problems of the main path planning problem. Most of these algorithms depend mainly on conventional programming. Fan [5] had divided the robot's environment into prime areas to represent it in a graph form. Searching a graph for the required path can be done by many algorithms such as A* [6, 9]. Also, DIJKSTRA algorithm [8, 10] is a famous algorithm to solve the problem of shortest route. Solving the path planning problem must discuss the problem of collision free path that had been investigated in [11 and 12].

Expert systems, which are one of the artificial intelligence applications, are appropriate approaches that can be used in solving the path planning problem. This is because of the characteristics of AI such as heuristic and interactive processing which are essential for the robot's fields. Some of the researchers such as Morad [14] and Xue [23] had used the expert system in implementing some of their systems' components.
Abo El-Ela [2] had described the used expert system to find the optimal path for the robot. The system described in this paper gives additional modules to manipulate the environment representation and the set of actions the robot must take to follow the candidate path. The following sections start by describing the proposed system and its tools. Following each component of the system will be discussed in detail.

The System Description and Tools

The proposed system is a path planner for a mobile shape-changeable robot. The assigned task to the robot is to reach any destination position in its environment from an initial one. It is required to choose the optimal collision free path between all alternative paths. Optimal path means the path with minimum total hardness faced by the robot through its route.

The block diagram in figure 1 represents the different components of the proposed system. This system starts from the environment map and ends with the set of actions to be followed by the robot to reach its destination position. The figure shows that the planner system has four main components. The first one is used to get the representation of the robot's states and the environment where the robot moves. The second and third components represent the two stages of the planning process. Finally the fourth component produces the set of actions to be followed by the robot.

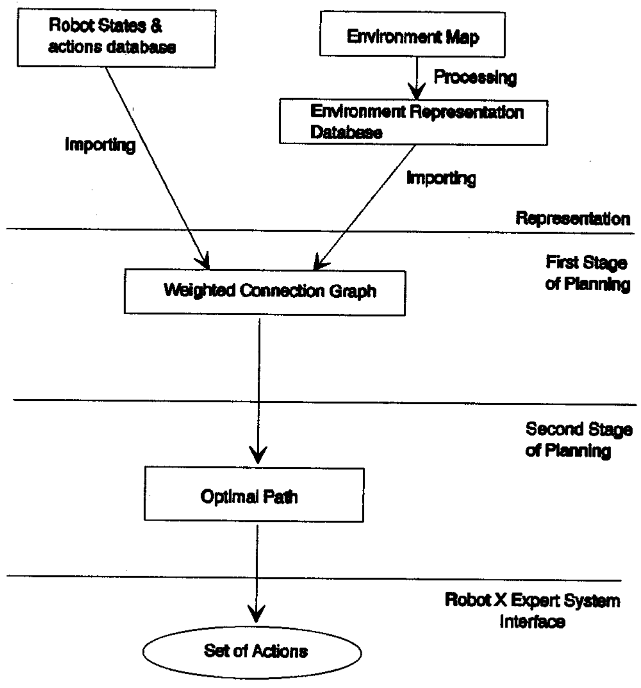

Figure 1. The proposed system block diagram

Conventional programming approach is used to implement the first component of the system. This component deals with manipulating the environment map to isolate the open and hard regions. Also, at the departure of the system, in the interface component, the conventional programming is used to re-display the environment map and allow the user to choose any destination position for the robot. According to this destination position the set of actions through the corresponding optimal route are produced by the program. Using conventional programming in this stage is useful for future real implementation of the system where it is easy to use conventional programming in hardware communications with a real robot.

The second tool that is used in the planner system is the database management system. It is used to save the data which represent the environment map, the robot specifications, and finally the results from the expert system that include the information of the optimal paths.
The expert system approach has been chosen to implement the planner system. This is related to the characteristics of expert systems that are compatible with the nature of the robot's problems especially path planning problems.

The used expert system shell is NEXPERT. This expert system shell has been designed to allow easy integration into existing computing environments. NEXPERT supports both reasoning system and powerful, object-oriented representation. Also, Nexpert supports both control strategies, the forward and backward chaining. As a powerful feature in NEXPERT, pattern matching conditions allows the user to create a subset of objects belonging to any class that satisfies the conditions. Also, NEXPERT has built-in functions for database access and supports most standard databases.

The Robot Representation

In the path planning problem, representing the robot requires only its outer configurations. Outer configurations mean the different states and actions that the robot has. The used method in representing the robot's different configurations, is to put these information in a database file. The database structure for the robot states and actions keeps track of the generality of the proposed system. This means that the contents of this database can be modified to be suitable for the used robot states. Table 1 contains the fields of this database and its meaning while figure 2 shows an example of the values that can be put in the robot's database.

Table 1
The structure of robot-specification database

Name	Meaning
STATE	The name of the robot state whose properties are included in the other fields.
ACTION	The action to be done by the robot for the corresponding state.
HEIGHT	The minimum height the robot needs for the corresponding state.
WIDTH	The minimum width the robot needs for the corresponding state.
HARDNESS	A value represents the relative hardness corresponding to the related action and state. This value is per unit of dimension.

Robot Specification

State	Action	Height	Width	Hardness
STAND	WALK	20.00	3.00	1.00
SIDE-STAND	SIDE-WALK	20.00	1.00	2.00
SQUAT	WALK-ON-KNEE	10.00	5.00	4.00
LIE-FACE-DOWN	CREEP	5.00	5.00	8.00
IMPOSSIBLE	NO-ACTION	0.00	0.00	10000.00

F1-Add F2-Toggle Delete F3-Exit

Figure 2. The screen of manipulating the robot's specifications

The Environment Representation

In general the robot's environment consists of boundaries and obstacles. Concerning the 3-dimensional environment the existence of bridges in the environment can be taken into consideration. The height of these bridges limits the actions of the robot to be taken in order to pass the bridges' regions. In the proposed system the environment's components are approximated by the minimum enclosing rectangles (see figure 3). This approximation is necessary to use the McCluskey method [5] to isolate the prime rectangular area to represent the free space as will be seen in the following sub-section. So, the environment map will contain a set of rectangles representing the obstacles.

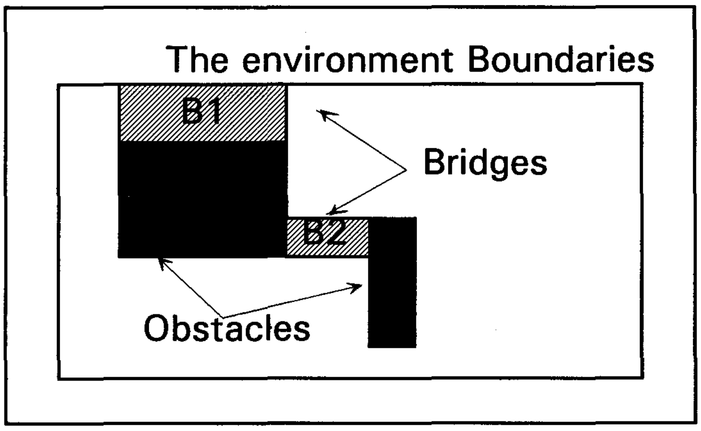

Figure 3. An environment example

1. Isolation of Free Areas

The simplified form of the environment is just a set of rectangles represent obstacles or bridges. The height of the bridges must be specified. In this sub-section the free areas in the robot's environment will be isolated. This isolation will help in dividing the environment into rooms with doors between them.

Instead of defining the different areas in the environment by their coordinates another definition will be used. This definition depends mainly on the binary representation. The binary representation helps in using the Boolean operations to find the relations between the different areas.

In this stage of isolation the bridges areas are considered to be free areas till finding what is called the prime areas. Given a map of boundaries and obstacles, the environment is partitioned by the edges of these shapes into a grid of at most 2n+1 X 2n+1 rectangles where n is the number of such rectangular areas representing obstacles. Each such rectangle is represented by a pair of binary strings 2n+1 bits long at most [5].

Before assigning values to the rectangle's pair of binary strings, the environment must be divided into horizontal and vertical strips. For the environment defined in figure 3, the horizontal strips marked from 1 to 5 and the vertical strips marked from a to e are shown in figure 4.

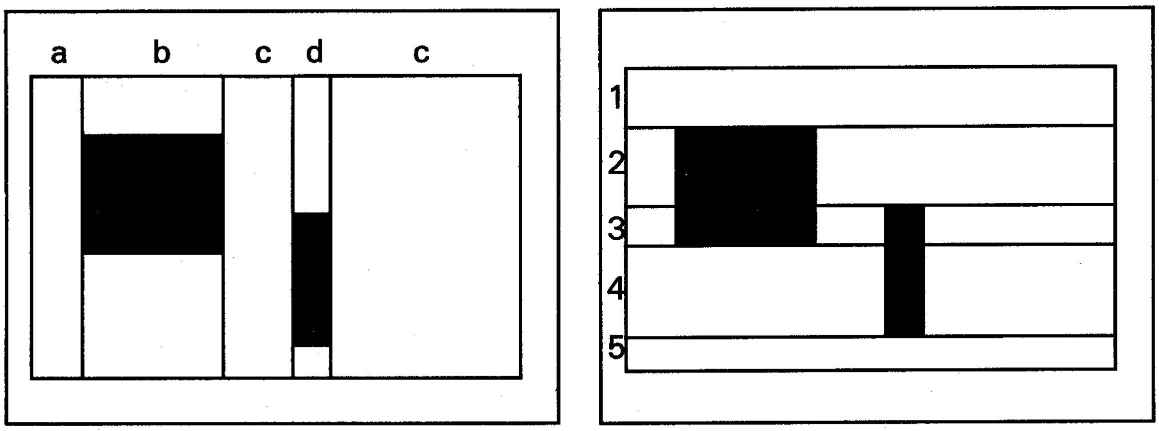

Figure 4. Horizontal and vertical strips of the given environment

The pair of binary strings, which defines any free rectangular area, consists of left and right sub-string. The right sub-string has the bits, correspond to the horizontal strips contained in the rectangle, set to one. Similarly, the left sub-string has the bits, correspond to the vertical strips contained in the rectangle, set to one.

After defining the horizontal and vertical strips, the McCluskey method [5] will be applied to isolate the prime rectangular areas. The result of applying this method to the environment of figure 3, is shown in table 2 and figure 5.

Table 2.
Binary strings of prime areas and bridges

Prime Area	String L - R
A	11111 - 10000
B	11111 - 00001
C	00111 - 11000
D	11100 - 00011
E	10000 - 11111
F	00100 - 11111
G	00001 - 11111

Bridges	String L - R
B1	01000 - 10000
B2	00100 - 00100

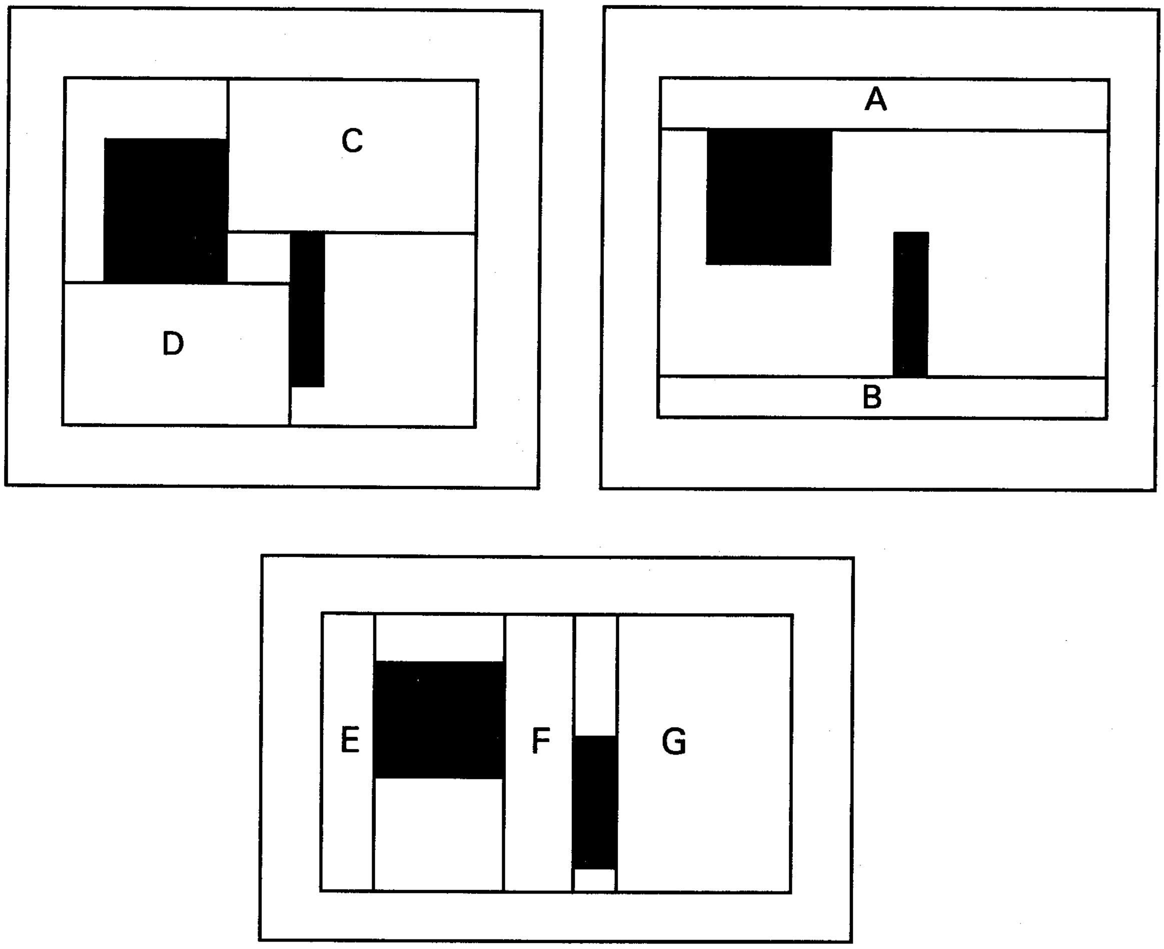

Figure 5. The produced prime areas

2. Rooms Generation

From the previous sub-section the two lists of prime areas and bridges are defined using the binary string method. In this sub-section these two lists are used to divide the free area of the environment into rooms. These rooms must follow four conditions:-
(1) Each room is rectangular in shape and has a fixed height where the robot should not change its action while moving in the same room.
(2) The number of these rooms must be as minimum as possible.
(3) There is no intersection between any two rooms.
(4) The union of these rooms must cover all the free areas in the environment.

To produce the rooms taking under consideration the previous issues, the following steps can be used.
Step 1: Form the following three lists:
* list 1 has the binary strings define the bridges with height less than the maximum height of the robot's states.
* list 2 has the binary strings define the open prime areas.
* list 3 has the binary strings define the hard prime areas.
Step 2: Let each string in list represent a room. If any prime area in list 2 or list 3 has an intersection with any of the bridges area in list 1, there will be three cases as following:

The bridge area totally encloses the prime area. The modification is to remove the prime area from its corresponding list.

The remaining of the prime area after removing the bridge area from it, can be represented by one binary string. The only required modification is to replace the prime area in its list (2 or 3) by the new binary string.

The remaining of the prime area after removing the bridge area from it, must be represented by two binary string. The modification is simply replace the prime area in its list by one of these binary strings and appending the other binary string in the same list.

Step 3: Choose the prime area in list 2 with the maximum area between the remaining prime area in the list. Let it to be a room and remove it from list 2. Solve the problem of intersection between this prime area and other areas in lists

2 or 3 as in Step 1. Repeat this step till no remaining prime areas found in list 2.

Step 4: Choose the prime area in list 3 with the maximum area between the remaining prime area in the list. Let it to be a room and remove it from list 3. Solve the problem of intersection between this prime area and other areas in lists 2 or 3 as in Step 1. Repeat this step till no remaining prime areas found in list 3.

After applying the previous steps on the lists of prime areas and bridges, another list will be produced that contains the binary string define the rooms areas. For the current environment, the generated rooms list will be as shown in table 3. The corresponding locations in the environment map are as in figure 6.

Table 3
List of the generated rooms

Rooms #	Binary String L		Binary String R
a	01000	-	10000
b	00100	-	00100
c	00001	-	11111
d	11100	-	00011
e	00110	-	11000
f	10000	-	11100
g	00010	-	00001

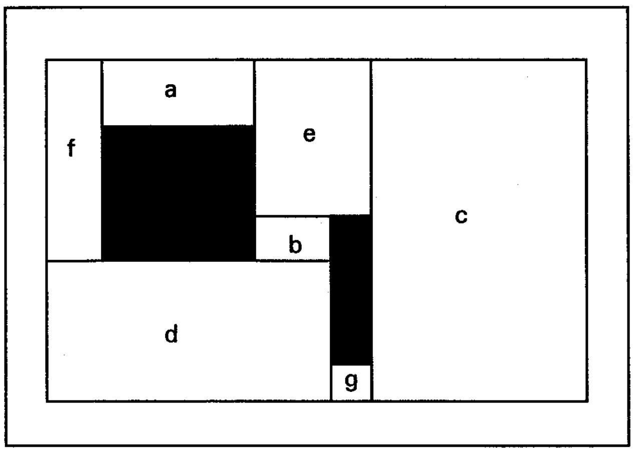

Figure 6. The environment's rooms

3. Doors Generation:
After defining the rooms in the previous sub-section, the remaining step is to produce the graph representation of the environment. The required graph is unweighted one with its vertices represent the rooms while its edges represent the doors. From the binary strings of the rooms (see table 3), the following steps are followed to define the doors between rooms:

Step 1: For any untested pair of the rooms
IF the ANDing of the two left strings <> zero THEN GOTO step 2 ELSE
IF the ANDing of the two right strings <> zero THEN GOTO step 3 ELSE GOTO step 4.
Step 2: IF the XORing of the two right strings has one block of adjacent ones THEN the two rooms have a door in common
ELSE GOTO step 4.
Step 3: IF the XORing of the two left strings has one block of adjacent ones THEN the two rooms have a door in common
ELSE GOTO step 4.
Step 4: IF all rooms are tested THEN stop ELSE GOTO step 1.

Using the previous method the list of doors will be generated and the required graph will be as shown in figure 7. Any door must be assigned some specifications from its position in the environment and the specifications of its two joined rooms. For example the door's width is calculated directly from the length of the interface between the door's two rooms. The door must be assigned the two heights and widths of the door's two rooms (if one of the room is not bridge, its height will be infinite). Other specifications, the door must have, are the distances between the door's centre and the corresponding two centres of the rooms it joins. These distances and measurements will help in weighting the graph edges as will be seen in the next section.

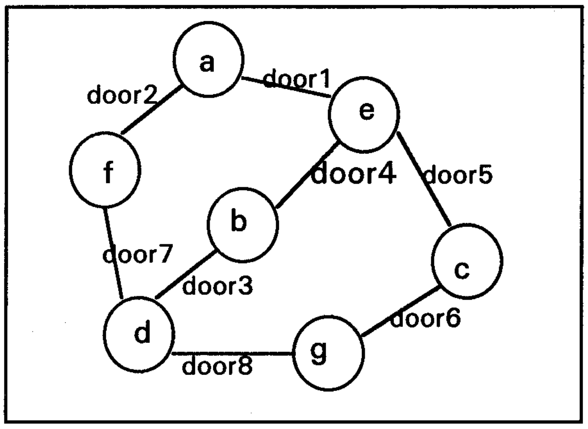

Figure 7. The graph representation of the environment

Finally the graph will be stored in a database form so that it can be imported later by the expert system to be processed. The database structure is shown in table 4. Each record in this database represents the specifications of a door in the environment.

The database that represents the environment doors' specifications as well as the number of the room that represents the initial position will be imported into the expert system to be processed for finding the optimal paths for the robot.

Table 4
The structure of environment-representation database

Name	Meaning
ROOM1	A number represents the first room that the door is connected to.
ROOM2	A number represents the second room that the door is connected to.
HEIGHT1	The height of ROOM1.
WIDTH1	The minimum width of ROOM1.
DISTANCE1	The distance between the door's centre and ROOM1's centre.
HEIGHT2	The height of ROOM2
WIDTH2	The minimum width of ROOM2
DISTANCE2	The distance between the door's centre and ROOM2's centre.
WIDTH	The width of the door.

Planning Stage

In the previous section the modules of the proposed system that are used to get the representation of the robot's states and the environment where the robot moves, have been explained. The databases contain the representation knowledge as well as the chosen initial position from which the robot starts to move, are imported into the planner module. This module with its different stages will be discussed in this section. The expert system shell NEXPERT will be used to implement this module. The results of the path planner module are the optimal paths between the initial position and any other position in the robot's environment. After assigning the destination position in the interface component, the set of actions to be followed by the robot through the candidate route will be produced.

1. Producing the Connection Graph

The unweighted connection graph that is represented by the imported data, is processed by the expert system NEXPERT to assign the proper weight for each edge in the graph. This weighted graph will be used later to search for the optimal path for the robot.

The knowledge base used in the expert system will be filled with the data stored in the two databases obtained in the representation component of the proposed system. The main structure of the knowledge base consists of classes. These classes contain objects that are common in their properties. Some of these properties are assigned directly from the corresponding fields in the databases, and the others require pre-processing. Table 5 shows the different classes in the knowledge base of the proposed system.

Table 5
The classes of the knowledge base and their properties

Class	Properties	Meaning
ROBOTS	STATE, ACTION, HEIGHT, WIDTH, HARDNESS	The same as the fields of the robot-specification database described in Table 1.
	NUMBER	The door number.
	CHOSEN	Boolean property to indicate if this door has been chosen (tested) or not yet.
	ROOM1, ROOM2	The two rooms that the door connects. These correspond to the same fields in the environment-representation database (see table 4).
DOORS	COST1, COST2	The current cost of the two rooms as a total of the hardnesses from the initial room of the robot to the corresponding room through the candidate path.
	SOURCE	A number represents ROOM1 or ROOM2 according to the current direction of the robot through this door (i.e. if the robot enters ROOM1 from ROOM2 through the current door, the source will equal to ROOM2).
	HARDNESS	The calculated hardness of this door. This property is calculated by the expert system from the door specifications and the robot's properties stored in the ROBOTS' class.
	DACTION	The action to be taken by the robot to pass by this door. This property is related to the hardness.
	ROOM-NUM	The room number.
ROOMS	COST	The cost of the room. This cost represents the hardness faced by the robot till reaching the current room through its trip in the candidate path.
	FROM	The door from which this room has been entered.
	RACTION	The action to be taken by the robot to move in this room.

After creating the objects of the knowledge base classes and assigning the different values to the objects' properties the next step is to produce the weighted connection graph. The objects of the DOORS class represent the unweighted connection graph. To convert this graph to a weighted one the HARDNESS property of the DOORS' objects must be assigned its proper value. Figure 8 shows the flowchart that is used to manipulate the objects of the knowledge base and assigning the HARDNESS to weight the connection graph.

Let DBDOOR be the name of the temporary object that is used to save the records of the doors' database. The following equation is used to assign the value of the HARDNESS for each imported DOOR:

$$\begin{gathered}\text{DBDOOR.DISTANCE1} * \textit{the first room hardness}\\ + \\ \text{DBDOOR.DISTANCE2} * \textit{the second room hardness}\\ + \\ \textit{The hardness to pass the door itself}\end{gathered}$$

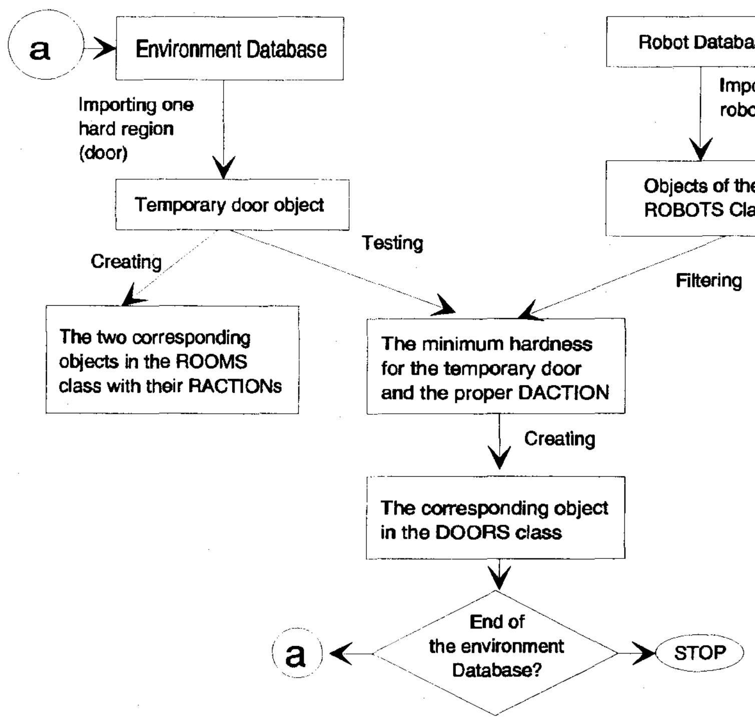
a
Environment Database
Importing one hard region (door)
Temporary door object
Creating
Testing
The two corresponding objects in the ROOMS class with their RACTIONs
Robot Database
Importing all robot's states
Objects of the ROBOTS Class
Filtering
The minimum hardness for the temporary door and the proper DACTION
Creating
The corresponding object in the DOORS class
End of the environment Database?
a
STOP

Figure 8. The flowchart of weighting the connection graph

Table 6 explains the preconditions (left hand side) and actions (right hand side) that are used in the expert system rule to assign the HARDNESS values and DACTION for each DOOR. Also, this will assign the RACTION for each ROOM.

Table 6

The preconditions and actions to prepare the weighted graph

preconditions (LHS)	Actions (RHS)
- <ROBOTS>.HEIGHT is less than DBDOOR.HEIGHT1	- The first room hardness =
- <ROBOTS>.WIDTH is less than DBDOOR.WIDTH1	MIN (<ROBOTS>.HARDNESS)
- <<ROBOTS>>.HEIGHT is less than DBDOOR.HEIGHT2	- The second room hardness =
- <<ROBOTS>>.WIDTH is less than DBDOOR.WIDTH2	MIN (<<ROBOTS>>.HARDNESS)
- <<<ROBOTS>>>.HEIGHT is less than DBDOOR.HEIGHT1	- The hardness to pass the door itself=
- <<<ROBOTS>>>.HEIGHT is less than DBDOOR.HEIGHT2	MIN (<<<ROBOTS>>>.HARDNESS)
- <<<ROBOTS>>>.WIDTH is less than DBDOOR.WIDTH	- RACTION of DBDOOR.ROOM1 =
- <<<<ROBOTS>>>>.HARDNESS is equal to	<<<<ROBOTS>>>>.ACTION
MIN (<ROBOTS>.HARDNESS)	- RACTION of DBDOOR.ROOM2 =
- <<<<<ROBOTS>>>>>.HARDNESS is equal to	<<<<<ROBOTS>>>>>.ACTION
MIN (<<ROBOTS>>.HARDNESS)	- DACTION of the door =
- <<<<<<ROBOTS>>>>>>.HARDNESS is equal to	<<<<<<ROBOTS>>>>>>.ACTION
MIN (<<<ROBOTS>>>.HARDNESS)	

Referring to the example that has been previously studied, applying the expert system rules, the resulted weighted connection graph will be as shown in figure 9.

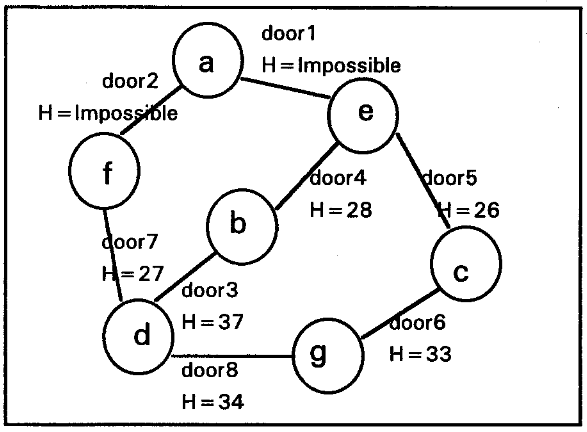

Figure 9. The weighted graph of the environment

2. Finding the Optimal Paths

The problem now is to find the optimal paths for the robot from its initial room to any other room in the environment. The flowchart shown in figure 10 explains the different steps that are used to find the optimal paths. These steps are similar to what have been explained in the second stage of planning by Abo El-Ela [2]. The exception here is that the used method find the optimal paths to every position in the environment not only to a specific position. The final results of the example in this paper can be summarized as in table 7.

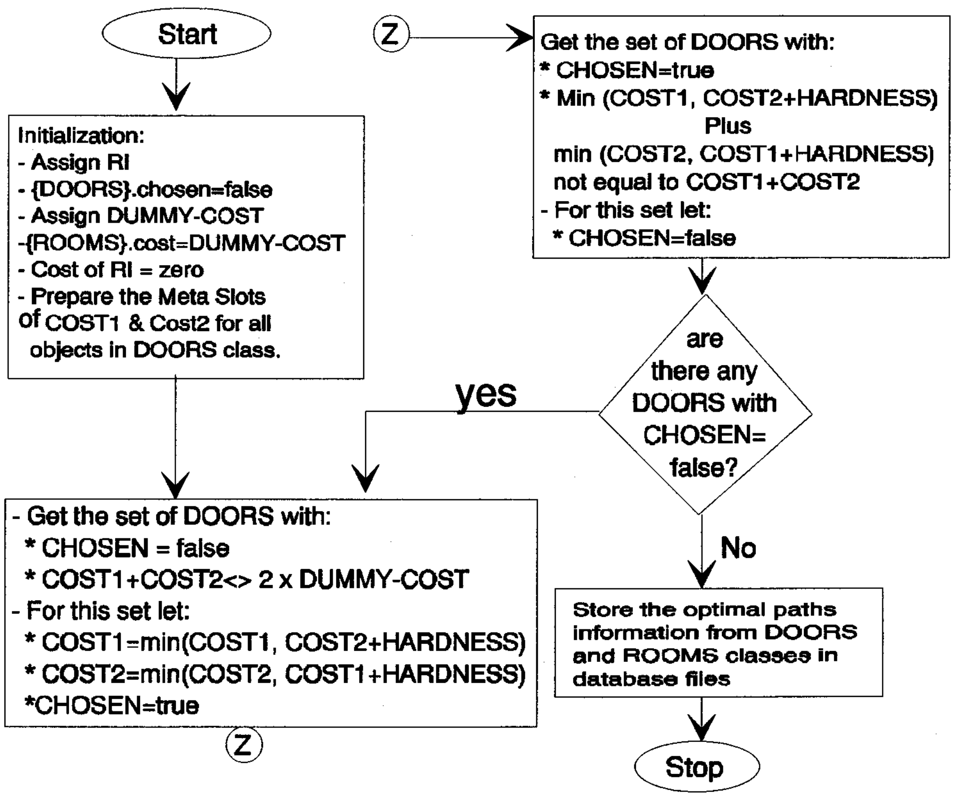
Start
Initialization:
- Assign RI
- {DOORS}.chosen=false
- Assign DUMMY-COST
-{ROOMS}.cost=DUMMY-COST
- Cost of RI = zero
- Prepare the Meta Slots of COST1 & Cost2 for all objects in DOORS class.
- Get the set of DOORS with:
* CHOSEN = false
* COST1+COST2<> 2 x DUMMY-COST
- For this set let:
* COST1=min(COST1, COST2+HARDNESS)
* COST2=min(COST2, COST1+HARDNESS)
*CHOSEN=true
Z
Z
Get the set of DOORS with:
* CHOSEN=true
* Min (COST1, COST2+HARDNESS)
Plus
min (COST2, COST1+HARDNESS)
not equal to COST1+COST2
- For this set let:
* CHOSEN=false
are there any DOORS with CHOSEN= false?
yes
No
Store the optimal paths information from DOORS and ROOMS classes in database files
Stop

Figure 10. The flowchart of finding the optimal paths.

Table 7
After three passes of processing

DOORS					ROOMS		
NUMBER	CHOSEN	COST1	COST2	SOURCE	ROOM-NUM	COST	FROM
1	true	10000	20000	a	a	10000	2
2	true	10000	0	f	b	64	3
3	true	64	27	d	c	94	6
4	true	64	92	b	d	27	7
5	true	94	92	e	e	92	4
6	true	94	61	g	f	0	0
7	true	27	0	f	g	61	8
8	true	27	61	d			

After producing the optimal paths, the expert system stores the results to be used in the interface program that is written in conventional programming. From table 7, the final result can be obtained starting from any destination room. These results are used to guide the robot through its candidate paths to reach its final position.

CONCLUSIONS

The proposed planner can be considered as a complete system. It starts from the robot specifications and the environment representation and ends with the set of actions to be followed by the robot to reach any destination position in the environment. The structure used in representing the specifications of the robot and its environment is open so that any extension can be easily done.

The software architecture utilizes several programming techniques. These techniques are used in implementing the system. Conventional programming, database storage, and the expert system (NEXPERT) are used. Each technique is used in its proper place according to its facilities and capabilities. Also, using conventional programming at the entrance and departure of the system makes it easy in the future to extend the system by hardware communications to a robot. The used rules and objects in the expert system can be manipulated in parallel on hardware that supports parallel processing.

The work represented in this paper can also be extended by importing a real image of the robot's environment to the system. This map can be processed using pattern recognition techniques instead of simplifying the map. Also, on line facility of the system can be realized. in this case the system can support the facility of error adapting. This facility can change the plan of the robot on lime according to any sudden mistake or changes faced by the robot.

REFERENCES

[1] **Abo El-Ela, E.H.,** 1993. Path Planner for a Mobil Shape-Changeable Intelligent Robot in a 3-Dimensional Environment, M. Sc. Thesis, Department of Computer Science and Automatic Control, Faculty of Engineering, Alexandria University, Alexandria, Egypt.

[2] **Abo El-Ela, E.H., I. Awad, M. N. El-Derini and M. G. Abou-Ali.** Optimal Paths for a Mobil Shape-Changeable Intelligent Robot, Proceedings Intern. AMSE Conference "Systems", London (U.K.) AMSE Press, No. 1: 155-166.

[3] **Abou-Ali, M. G.,** 1991. An Expert System for Optiised Process Planning of Fabrication and Welding, Ph. D. Thesis, Paisley College, Scotland, U.K.,

[4] **Amble, Tore,** 1987. Logic Programming and Knowledge Engineering, Addison-Wesley Publishing Company.

[5] **Fan, K. C., and P.C. Lui,** 1991. A Path Planning Algorithm for a Mobile Robot, SPIE No. 1468 Application of Artificial Intelligence IX.

[6] **Fu,L.M., A.J. Gonzalez and W.C.Y. Lee,** 1987. Robotics Control, Sening, Vision, and Intelligence, McGraw Hill Book Company.

[7] **Fukuda, T, H. Hosokai, and M. Otsuka,** 1989. Path Planning Expert System for Pipeline Inspection Robots, IEEE International Symposium on Intelligent Contgrol.

[8] **Goodman, S. E. and S. T. Hedetniemi,** 1977. Introduction to the Design and Analysis of Algorithms, McGraw-Hill..

[9] **Grevera, G. J. and A. Meystel.** 1988. Searching for an Optimal Path Through Pasadena, Third International Symposium on Intelligent Control, Arlington, VA, USA.

[10] **Horowitz, E. and S. Sahni,** 1982. Fundamentals of Data Structures, Computer Science Press Inc.

[11] **Lin, M.C. and J.F. Canny,** 1991. A fast Algorithm for Incremental Distance Calculation, Proceedings of the 1991 IEEE, International Conference on Robotics and Automation, Sacramento, California.

[12] **Luo, R. C., Y. Guo, and T. J. Pan,** 1989. Intelligent Mobile Robot Path Planning System, SME Technical Paper (Series)MS.

[13] **Luo, R. C. and T. J. Pan,** 1990. Maintaining Knowledge for an Adaptive Path Planning System, Applications of Artificial Intelligence VIII Conference, Orlando, Fl, USA.

[14] **Morad, A. A.; A. B. Cleveland; Y. J. Beliveau; V. D. Fransisco; and S. S. Dixit,** 1992. Path Finer. AI-Based Path Planning System, Journal of Computing in Civil Engineering.

[15] **Snyder, W. E,** 1985. Industrial Robots, Computer Interface and Control, Prentice/Hall International Inc.

[16] **Suh, S.H., I.K. Woo; and S.K. Noh,** 1991. Development of An Automatic Trajectory Planning System (ATPS) for Spray Painting Robots, Proceedings of the International IEEE Conference on Robotics and Automation, Sacramento, California.

[17] **Taha, H. A.** 1981. Operations Research, an Introduction, Macmillan Publishing Co. , Inc.

[18] **Tavora, J. and P. M. G. Lourtie** 1991. Parallel-Message-Passing Architecture for Path Planning, SPIE Vol. 1468 Applications of Artificial Intelligence IX.

[19] **Todd, D. J.** 1986. Fundamentals of Robot Technology, Kogan Page Ltd.

[20] **Walker T. C. and R. K. Miller,** 1990. Expert Sy tems 1990, An Assessment of Technology and Applications, Turing Institute Library, SEAI Technical Publications, USA.

[21] **Welbank, Margaret,** 1984. A Review of Knowledge Acquisition Techniques for Expert Systems, Martlesham Consultancy Services, England.

[22] **Wolfgram, D. D., T. J. Dear, and C. S. Galbraith,** 1987. Expert Systems for the Technical Professional, John Wiley & Sons.

[23] **Xue, Qing and P. C-Y Sheu,** 1990. Collision-Free Path Planning for a Shape-Changeable Mobil Robot in a 3-Dimensional Environment, Proceedings of the 2nd International IEEE Conference on Tools and Artificial Intelligence, Herndon, VA, USA.

[24] **Zhu, D, and J. C. Latombe,** 1991. Pipe Routing=Path Planning (with Many Constraints), Proceedings of the International IEEE Conference on Robotics and Automation, Sacramento, California.