

SENSOR BASED PATH PLANNING OF MINI-ROBOT “KHEPERA” USING PHYSICAL A* (PHA*) ALGORITHM

Mohamed S. Abdel-Wahab¹, Sayed Fadel¹, Sara Yousef Serry³

¹Professor in Department of Scientific Computing Ain Shams University,
³Lecturer in Department of Information Technology King Abdel-Aziz University
(EGYPT, KINGDOM OF SAUDI ARABIA (KSA))
E-mails: wahabms@fcisonline.net, sayed_fadel@yahoo.com,
syserry@hotmail.com

ABSTRACT

This paper is concerned with the issue of enabling robots (mechanical devices equipped with actuators and sensors under the control of a computing system) to decide their own motion and to find the shortest path between two points in the unknown environments using the Physical A* algorithm (PHA*). Due to the physical nature of the problem, the complexity of the PHA* algorithm is measured by the traveling effort of the moving robot and not by the number of generated nodes. PHA* is presented as a two-level algorithm, such that its high level, A*, chooses the next node to be expanded and its low level directs the robot to that node in order to explore it. We then applied this algorithm in the control protocol of mini robot called “Khepera” developed to study robotics technology for different applications. The results show the ability of the algorithm to let “khepera” maneuver successfully among obstacles founded in its environment without colliding them and to compute the shortest path to the goal configuration. Thus, it is recommended to apply PHA* algorithm on a group of robots to help them moving from one place to another to carry out some complex tasks. This study has been concluded with the hardware implementation of the mentioned algorithm and demonstration of the implemented systems.

Key words: obstacle avoidance; degrees of freedom; sensor path planning; PHA*; “khepera”

1. INTRODUCTION

1.1. Motivation

The focus of this paper is *motion planning* and *obstacle avoidance*, which constitute a fundamental problem in robotics [1]. The goal is to develop algorithm to find a path between two points which minimizes both the *cost and the path* between these two points in unknown or partially known environments. Then applying this algorithm to a mini-robot called “khepera”; which has originally been designed as a research and teaching tool in the framework of a Swiss Research Priority Program.

1.2. Problem Definition

The objective of motion planning is to find a sequence of states that carries the robot from the start state to the goal state [2]. In [3], the planning problem incorporates the generation of a motion control sequence that achieves the control objective (goal attainment), while complying with constraints, optimizing performance, and restraining the effects of disturbances. Many researchers from different fields of robotics and computer science have contributed to the solution of this problem.

Any robot has a number of directions in which its motion can be controlled known as its *Degrees of freedom (DOF)* [4]. Free body in space has 6 DOFs, three for position, and three for orientation, if there is an actuator for every degree of freedom, then all degrees of freedom are controllable, in which a robot *can* move instantaneously in any direction and

then it is called *holonomic robots*. However, most robots are *non-holonomic*, in which; they *cannot* move to change their pose instantaneously in all directions [5].

In many autonomous *holonomic robots*, motion planning is performed on two levels [2]: (i) *model based*; and (ii) *sensor based*.

At the model level, planning is typically based on an *a priori* map of the environment; a search is performed for free paths between the given start and goal states. In many of these approaches, the set of possible robot states is mapped to a *non-directed graph* using a method of environment representation (e.g. [6]).

However, **Sensor path planners** [7] deal with unexpected or un-modeled objects in the environment and are usually most efficient for obstacle avoidance using robot's sensors. Accordingly, the input to the sensor path planner consists of the initial and goal configurations. The other obstacle configurations are assumed not to be known in advance.

Sensor based path planning is important because [7]: (a) the robot often has no a priori knowledge of the world; (b) the robot may have only a coarse knowledge of the world because of limited memory; (c) the world model is bound to contain inaccuracies which can be overcome with sensor based planning strategies; and (d) the world is subject to unexpected occurrences or rapidly changing situations.

There are two famous sensor path planning approaches for navigating an unknown environments which are Generalized Voronoi Graph (GVG) [8] and Physical A* (PHA*) [9] approaches. The problem in GVG is there are no considerations for limited range sensor robots. Therefore, in this paper, *the Physical A** is presented as a sensor path planner in which the robot is assumed to be located at the initial point. The task is **to find the shortest path in the (unknown) graph between the initial point and the goal point** for future usage. In order to accomplish that, the robot is required to traverse the graph and explore its relevant parts leading to the desired solution. The robot is allowed to visit nodes and travel from one point to another via existing edges.

This paper is structured as follows: following section describes in details the PHA* approach. Section 3 give a complete overview on the hardware design of robot "khepera" and provides the features and characteristics of it. Then section 4 describes the experimental results of simulation to verify how PHA* can be applied to "khepera", followed by a conclusion in section 5.

2. PHYSICAL A* (PHA*) SENSOR PATH PLANNER APPROACH

2.1. A* Algorithm for finding the shortest path

The A* algorithm [10] is the common methods for finding the shortest paths in large graphs. A* keeps an open list of nodes that have been generated but not yet expanded, and chooses from it the best nodes for expansion. When a node is expanded it is moved from the open list to the closed list, and its neighbors are generated and put in the open list. The search terminates when a goal node is chosen for expansion or when the open list is empty. The cost function of A* is $f(n) = g(n) + h(n)$. Where $g(n)$ is the distance traveled from the initial state to n , and $h(n)$ is a heuristic estimate of the cost from node n to the goal. With such a heuristic $h(n)$, A* is guaranteed to always return the shortest path. The complexity of A* can also be measured in terms of the number of generated nodes.

2.2. Physical A* Algorithm

As we said before, in this paper we deal with finding the shortest path in graphs in unknown environment; many of the nodes and edges of this graph are not known in advance. Therefore, to expand a node that is not known in advance, a mobile robot must first travel to that node in order to explore it and learn about its neighbors. The cost of the search

in this case is proportional to the distance traveled by the robot. An efficient algorithm would therefore minimize the distance traveled by the robot until the optimal path is found [10].

In this paper we will present the Physical-A* algorithm (PHA*) [9] for solving the motion planning problem in unknown environments. PHA* expands all the mandatory nodes that A* would expand and returns the shortest path between the two points. However, the complexity of the algorithm is measured by the traveling effort of the moving robot. PHA* is designed to minimize the traveling effort of the robot by intelligently choosing the next move of the moving robot. PHA* is presented as a two-level algorithm [8], such that its **high level**, chooses the next node to be expanded and its **low level** moves the robot to that node in order to explore it as we will discuss in the following sections.

2.2.1.High level algorithm

The high level algorithm acts like A* search algorithm [10]. It chooses at each cycle a **node from the open list for expansion**. The heuristic function **h (n)** used here is the Euclidean distance between **n** and the goal node.

If the node chosen by the high level has not been explored (physically visited and learned about its location and location of its neighbors) by the robot, the low level, which is a navigation algorithm, is activated to move the robot to that node and explore it. The pseudo-code for the high level is given in table 1.

Table 1. The high level algorithm

<p>ALGORITHM High-level (open-list) INPUTS: The configurations q_{init}, q_{goal}. OUTPUTS: An expanded node</p> <ol style="list-style-type: none"> 1. While (open-list is not empty) 2. Target = best node from open-list; 3. If target is unexplored then 4. Explore (target) by the low level; 5. End while 6. Expand (target);
--

2.2.2.Low level algorithm

In the high-level algorithm, if the chosen node has not been visited by the robot, the low level instructs the robot to visit that node which called target node. In order to reach this target node, we must use an efficient navigation algorithm.

This paper suggests the use of a A* DFS [A* Depth First Search] navigation algorithm that was proposed by [11] for the low level. A* DFS is an intelligent navigation approach for finding a path to the target that can pass also through unexplored nodes. As shown in Table 2 the low level algorithm proceeds as follow,

(1)The robot moves to a neighboring node that has not been visited in which, it chooses the neighbor **w** that minimizes the sum of the distances from the current node **v** to **w** and from **w** to the target node **t**. Note that this cost function is used here locally to find a path from the current node to the target node.

(2)The algorithm backtracks upon reaching a dead-end and the search continues until it reaches the target.

(3) If there is more than one neighbor, we use a heuristic to evaluate which neighbor is more likely to lead faster to the target, and visit that node first. At each step the robot chooses the neighbor **w** that minimizes the sum of the distances from the current node **v** to **w** and from **w** to the target node **t**. Note that this cost function is used here locally to find a path from the current node to the target node.

Table 2. The low level algorithm

<p>ALGORITHM low-level A*DFS(v) INPUTS: Unexplored node OUTPUTS: Explored node to be expanded in high level</p> <ol style="list-style-type: none"> 1. Mark v as visited; 2. While (there is an explored vertex w adjacent to v) 3. Add (v,w) to list; 4. A*DFS (w) 5. End (while)


The benefit of A*DFS algorithm [9] is that they explore new nodes during the navigation, and they will not revisit such nodes, should the high-level procedure expand them at a later stage.

In the following section we will give some details about mini robot “khepera” which will be controlled by PHA* algorithm to maneuver successfully among obstacles not to be known in advance.

3. MINI-ROBOT “KHEPERA” HARDWARE CONFIGURATION

“Khepera” has originally been designed as a research and teaching tool in the framework of a Swiss Research Priority Program. It allows confrontation to the real world of algorithms developed in simulation for trajectory execution, obstacle avoidance, preprocessing of sensory information. Table 3 shows some of the features and advantages of “Khepera”

Table 3. The Features and Advantages of “Khepera”

Features	Advantages
Size: 30mm × 55mm Ø Weight: 70g Processor: Motorola 68331,25 MHz RAM/Flash: 512 kB / 512 kB Motors: 2 DC motors Sensors: 8 infrared sensors 8 ambient light Energy: 4 NiMh accumulators Life time: 60 min 	Compact, Easy to use. Easy to carry with you for demonstrations. • Programmable microcontroller. • A lot of sensor and actuator extensions. • Easy integration of new sensors and processing components. • Perform complex behaviors on a table top. • Wide range of experiments: • Mobile environment, dynamic experiments

Internal and External Components of “Khepera”

Figure 1, shows the internal and external most important components founded in “Khepera”

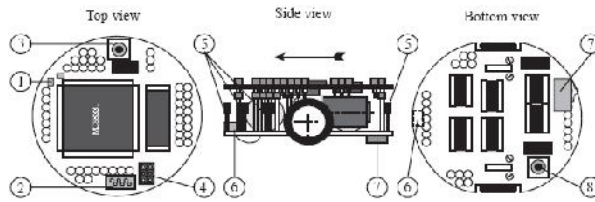


Fig. 1. Position of some parts of the robot

Note the location of the following parts [12]:

1. ON - OFF battery switch.
2. Jumpers for the running mode selection.
3. Serial line (S) connector.
4. Motors and motor control
5. Infra-Red proximity sensors.
6. Batteries and LEDs.

1- On - Off battery switch

As shown in figure 2, battery switch allows the user to switch the battery of the robot ON or OFF. When switch is ON, the robot is powered by the Ni-Cd internal batteries. In this case the robot cannot be powered by an external supply. When OFF, the batteries are disconnected and the robot can be powered by the S serial line connector.

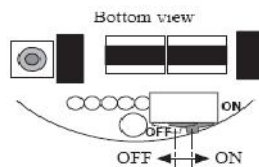


Fig. 2. Position of the Battery power supply ON - OFF switch

2- Jumpers

The ROM installed on Khepera robot has an important library of software modules for the real time control of the robot. These modules ensure the interface with the user through the serial line. The serial link set-up is always 8 bit, 1 start bit, 2 stop bit, no parity. Figure 3 demonstrates the set-up of the jumpers that can be changed at any time. If the robot is running it is necessary to reset it to make the set-up effective.

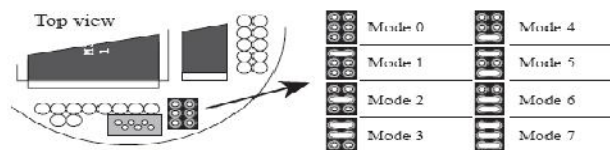


Fig. 3. Position and setting of the jumpers

3- The serial line

The S serial line is an asynchronous serial line with TTL levels (0-5V). An interface is necessary to connect this line to a standard RS232 port. This interface is included in the interface/charger module present in the package. The S serial line can power the robot.

4- Motors and motor control

Every wheel is moved by a DC motor coupled with the wheel through a 25:1 reduction gear. In figure 4, an incremental encoder is placed on the motor axis and gives 24 pulses per

revolution of the motor. This allows a resolution of 600 pulses per revolution of the wheel that corresponds to 12 pulses per millimeter of path of the robot.

The Khepera main processor has the direct control on the motor power supply and can read the pulses of the incremental encoder. An interrupt routine detects every pulse of the incremental encoder and updates a wheel position counter.

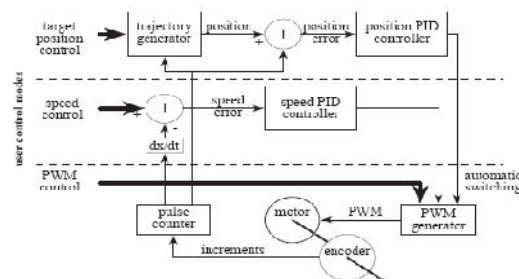


Fig. 4. Structure of the motor controllers and levels of user access

5- Infra-red proximity sensors

Eight sensors are placed around the robot and are positioned and numbered. These sensors embed an infra-red light emitter and a receiver.

6- Batteries

The robot is equipped with batteries with a capacity of 180 mAh. This capacity allows the robot autonomy of about 45 minutes in the basic configuration.

Robot-Computer Communication

The illustrated hardware configurations allow the communication between the robot and a host computer through a serial link [12]. On the host computer side the link is made by a RS232 line. The interface module converts the RS232 line into the S serial line available on the robot.

In the following section we will introduce how the hardware configuration of "khepera" will enable it to find obstacles in its environment also will show how PHA* algorithm will let "khepera" avoid collision with these obstacles, and find the shortest path in the (unknown) graph between the initial point and the goal point.

4. PERFORMANCE EVALUATION OF PHA* ALGORITHM USING MINI-ROBOT "KHEPERA"

It is of interest to assess the performance of PHA* algorithm which demonstrated in section 2 in details. Therefore, to control the functionalities of "Khepera" robot (motors, sensors etc.) the PHA* algorithm is implemented in the control protocol of khepera. Also, the communication with the "Khepera" robot is made by sending and receiving ASCII messages. In all communications the host computer plays the role of master and the "Khepera" the role of slave.

5. RESULTS AND DISCUSSIONS

Our purpose is to let khepera maneuver among obstacles safely without colliding it in an environment using PHA* source code which is implemented in the control protocol of khepera.

So the tasks of "Khepera" were to:

1- Discover the obstacles configurations in the run time by using khepera's eight infra-red sensors [11] which allow two measures:

- The normal ambient light. This measure is made using only the receiver part of the device. A new measurement is made every 20 ms. During the 20 ms, the sensors are read in

a sequential way every 2.5 ms. The value returned at a given time is the result of the last measurement made.

•The light reflected by obstacles. This measure is made emitting light using the emitter part of the device. The returned value is the difference between the measurement made emitting light and the light measured without light emission (ambient light). The value returned at a given time is the result of the last measurement made.

The output of each measurement is an analogue value converted by a 10 bit A/D converter.

2- Visit the closed nodes that expanded by A* algorithm according to the obstacles configurations calculated by “khepera”. These nodes comprise the set of nodes that are located in the free space and achieve the shortest possible path from start to goal configurations.

Figure 5, shows the simulation output which had been achieved successfully by the robot which assumed to be located in the initial point. The experiments demonstrate the ability of PHA* planner to let “khepera” maneuver among the obstacles without colliding them by let “khepera” traverse the environment and explore its relevant parts leading to the desired solution. In almost of all trials in a number of different environments, paths to goal configuration was computed within 5.25 s.

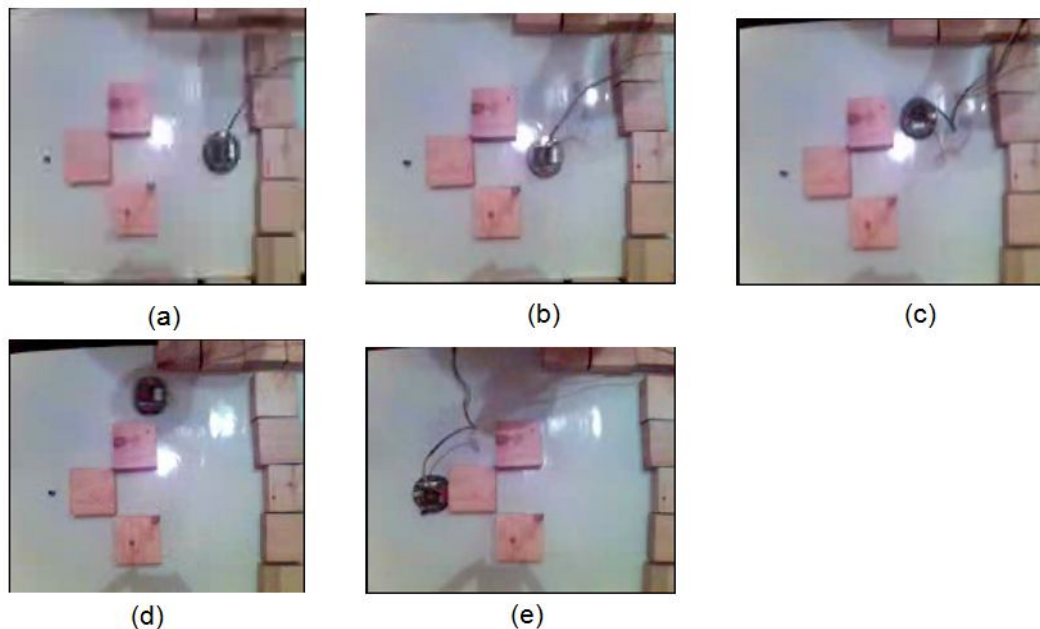


Fig. 5. Real case output for Maze Solving Achieved by Mini Robot “Khepera” Using PHA* Algorithm

6. CONCLUSIONS

We have addressed the problem of finding the shortest path to a goal node in unknown graphs that represent physical environments. We have presented the two-level algorithm, PHA*, for such environments for a single robot. We have also, experimented the algorithm in different algorithms by installing it in the control protocol of mini robot “Khepera”. The sensor information which reflecting the current state of the environment have been incorporated into a robot’s planning process. PHA* algorithm yield a successful results in the ability to let “Khepera” maneuver among obstacles without colliding them, also the ability to compute the shortest path to the goal configuration. The path was computed within average 5.25s. Thus, our future work is to apply the PHA* algorithm to a group of robots to help them moving from one place to another without colliding with each other or any obstacles in the environment.

The cooperation of robots enables them to carry out complex tasks, such as transferring objects, and in some medical applications.

REFERENCES

1. Roland Siegwart, Illah Nourbakhsh. *Introduction to Autonomous Mobile Robots*. A Bradford Book, The MIT Press, Cambridge, Massachusetts and London, England, 2004.
2. Latombe J.C. *Robot Motion Planning*, Kluwer Academic Publishers, Norwell, MA, 2002.
3. H.Choset W.Burgard S. Hutchinson G.Kantor and S.Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, April 2005.
4. Gillespie R. B., Colgate J. E., Peshkin M. A. "A general framework for robot control", IEEE Transactions on Robotics and Automation, Vol. 17, No.4, 391-401, 2001.
5. M.J. de Smith. "Distance and Path: The Development, Interpretation and Application of Distance Measurement in Mapping and Modelling" PHD THESIS, University College, 2003.
6. Micheal A.Bender, Antonio Fernandez, "The Power of Pebble: Exploring and Mapping Directed Graphs", IEEE Transactions on Information and Computation, 2002.
7. Alberts S. and Henzinger M. R., "Exploring unknown environments", ACM Symposium on the theory of Computing, 1999.
8. Keneth E.Hoff III, Tim Culver, John Keyser, "Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware", IEEE Transactions on Robotics and Automation, 2000.
9. Ariei Felner, Roni Stern, Asaph Ben-Yair, Nathan Netanyahu, "PHA*: Finding the Shortest Path with A* in Unknown Physical Environment", Journal of Artificial Intelligence Research, 2004.
10. Shmoulina L., Rimon, "Roadmap-A*: An algorithm for minimizing travel effort in sensor based mobile robot navigation", IEEE international Conference on Robotics and Automation, 1999.
11. Stephen Kweh, "Simple Depth-First Search Strategy for Exploring Unknown Graphs", 2002.
12. Khepera user manual, k-team manuals 6.0, 2000.