Applied Soft Computing 11 (2011) 4667–4676

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm

Qingbao Zhu[a,b,*], Jun Hu[a,b], Wenbin Cai[a,b], Larry Henschen[c]

[a] School of Computer Science, Nanjing Normal University, Nanjing 210097, PR China
[b] Jiangsu Research Center of Information Security & Confidential Engineering, Nanjing 210097, PR China
[c] Northwestern University, Evanston, IL 60208, USA

ARTICLE INFO

Article history:
Received 9 January 2010
Received in revised form 22 March 2011
Accepted 24 July 2011
Available online 12 August 2011

Keywords:
Unknown dynamic environment
Robot navigation
Scout ant
Path planning

ABSTRACT

A difficult issue in robot navigation or path planning in an unknown environment with static or dynamic obstacles is to find a globally optimal path from the start to the target point and at the same time avoid collisions. We present a novel and effective robot navigation algorithm for dynamic unknown environments based on an improved ant-based algorithm. In our approach two bidirectional groups of scout ants cooperate with each other to find a local optimal static navigation path within the visual domain of the robot. The robot then predicts the collision points with moving objects, and the scout ants find a new collision-free local navigation path. This process is carried out dynamically after each step of the robot, thereby allowing the robot to adjust its path as new obstacles come into view or existing obstacles move in new directions. Therefore, the robot can not only avoid collision but also make its paths globally optimal or near-optimal by making a series of dynamic adjustments to locally optimal paths. The simulation results have shown that the algorithm has good effect, high real-time performance, and is very suitable for real-time navigation in complex and dynamic environments.

1. Introduction

Mobile robot path planning or navigation is an important application for robot control systems and has attracted remarkable attention from many researchers. As a result, many interesting research results have been obtained [1–21]. Two important features that distinguish these algorithms are whether the environment is known or unknown and whether it is static or dynamic. This paper presents a new approach for robot navigation in unknown, dynamic environments.

Known environments are those in which the robot knows the coordinates of the obstacles and the target point is clearly defined. Examples of successful algorithms for path planning or navigation in this kind of environment include the traditional artificial potential field algorithm, the A* and D* algorithms [22], and many others. In unknown environments, the robot does not have any previous knowledge about the environment, which means that it does not know if there are obstacles, where the obstacles are, and what is the accurate location of the target point. Therefore, the robot must use sensors to detect the environmental information and can only sense information within the range of those sensors. The range of the sensors is called the robot's visual domain (VD). The robot can compute positions of obstacles in its VD but must move and explore in order to find other obstacles. So, it cannot plan a global path in a single attempt. In recent years, many researchers have achieved interesting research results on robot path planning or navigation in such environments. For instance, Zhu et al. proposed a real-time navigation algorithm based on fuzzy logic combined with sensing and a state memory strategy [1]. This algorithm effectively solved the so called 'dead cycle' problem perfectly; Yang et al. presented a layered goal-oriented motion planning strategy based on fuzzy logic [2]. There are also other navigation approaches based on Genetic Algorithms [3], the Q-learning Algorithm [4], neural networks [5], the RRT (Rapidly-exploring Random Tree) Algorithm and other kinds of orientation and navigation methods [6–10]. In addition, some scholars have researched robot navigation algorithms [11–15] based on the Ant Colony Optimization (ACO) [24,25] or the improved ACO algorithm. These works have made some innovative achievements. However, the precondition of the majority of these navigation algorithms is that the robot must be able to detect the target point by its sensor. In other words, the robot's VD should be able to cover the whole working field. Therefore, these algorithms are better suited for robots working in a small working field or a field with few obstacles [1–4]. The time complexity of these algorithms will increase greatly when the environment becomes

* Corresponding author at: School of Computer Science, Nanjing Normal University, Nanjing 210097, PR China.
E-mail addresses: zhuqingbao@njnu.edu.cn, jin_1234@126.com (Q. Zhu), hujun_njnu@163.com (J. Hu), caiwenbin20081224@163.com (W. Cai), henschen@eecs.northwestern.edu (L. Henschen).

1568-4946/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2011.07.016

larger and more complex. For example, the path planning algorithm based on the Genetic Algorithm may produce many invalid paths and may fail when the number of obstacles increases [26]. Furthermore, deadlock and oscillation happen easily in the rolling window method, and stagnation is a general problem for the ACO algorithm [27].

There are two conditions that make environments dynamic. The first condition is that the target moves continuously during the navigation. Many robot navigation algorithms have been proposed for such environments [16–19]. A characteristic of this category of navigation algorithms is that the target point must always be in the robot's VD. The second condition is that there are moving (dynamic) obstacles appearing randomly in the known or unknown portions of the environment. Navigation algorithms in such an environment are often based on the combination of a navigation algorithm for static environments and a collision-avoidance strategy [20,21,23]. In this paper we focus mainly on the second kind of environment.

In situations where the workplace of the robot is larger than its VD and contains unknown static and dynamic obstacles, the robot navigation has to solve three problems: (1) How to carry out local path planning quickly enough to solve the real-time requirement of the global dynamic planning algorithm. (2) How to guarantee the path walked by robot is the shortest in the global environment. (3) How to avoid collision safely in dynamic environments and also solve the problem of deadlock and oscillation. These three problems still have not been solved satisfactorily before now. For example, robot navigation with the ACO algorithm can only perform path planning within the robot's VD [15]. The reason is that the convergence speed of the ACO is far from satisfying the real-time requirement of global dynamic planning.

In order to solve these three issues, we propose a novel robot navigation ant (RNA) algorithm, which borrows ideas from some principles of scout ants during the food searching process. In this algorithm, first the global target position g_{end} is mapped into the closest node outside the robot's VD (visual domain) according to the approximate orientation of g_{end}, and this new point is taken as a local navigation sub-goal. (Of course, the robot cannot see g_{end} but may know where it is by remembering past explorations or being told explicitly. Similarly, the robot cannot see the local sub-goal but can still compute its coordinates.) Then, the robot uses its sensors to detect information about obstacles and build an environmental model within its VD. On the basis of this, the Multi-Scout Ants Cooperation (MSAC) algorithm we propose in Section 4 is used by the robot to plan a static local navigation path within VD, in which only static obstacles are considered. After that, the robot checks if any of the obstacles are moving, analyzes their trajectories, and predicts if the initial planned path contains any point of collision with those obstacles. Finally, if there is an unavoidable collision, the predicted collision point is temporarily set as a static obstacle and the MSAC algorithm is used again to plan a new local navigation path that avoids collision. The robot moves forward one step along the planned path and then repeats the entire process. The process is repeated until that robot finds the global target in its VD. Thus, the RNA algorithm is a dynamic global navigation algorithm. The RNA algorithm uses MSAC as its local static path-planning algorithm and again if collision avoidance is needed. Because MSAC may be used twice for each step of the robot it must obviously be very fast.

Computer simulation experiments show that even in environments with complex obstacles the robot can move along a globally optimal or near-optimal path and arrive at the destination without collision by using this algorithm.

The remainder of the paper is organized as follows. In Section 2, the robot's working environment is described and the correlative definitions are given. In Section 3, the global navigation algorithm of RNA is presented. In Section 4, the MSAC algorithm is described. In Section 5, computer simulations and comparisons with other algorithms are given. Finally, Section 6 is the conclusion.

2. Description of environment

For convenience, we call the robot *Rob*. *Rob* moves in a finite, two-dimensional field, called *AS*, in which a finite number of static obstacles $Sb_1, Sb_2, \ldots, Sb_n$ and dynamic obstacles $Db_1, Db_2, \ldots, Db_q$ are distributed at locations not known initially to *Rob*. Let $OS = \{O_1, O_2, \ldots, O_m\}$ $(m \geq n)$ be the set of grids occupied by static obstacles. Every Sb_i occupies one or more grid cells in *OS*, hence $m \geq n$. Suppose *Rob*'s motion diameter, that is, the length of one step, is δ and the range of *Rob*'s sensor is r.

Establish a Cartesian coordinate system $\sum_0$ in *AS* with *Rob*'s start point g_{begin} as the origin point and δ as the unit of measurement in both axes. This partitions *AS* into a grid, with cells of size $\delta \times \delta$. The set of all grid cells in *AS* is called A.

Definition 1. Let $P_i(x, y) \in AS$ be an arbitrary position in *AS*. Its coordinates are taken to be the same as the grid cell g_i in which it occurs, and P_i is regarded as equivalent to g_i, denoted by $P_i \sim g_i$.

Definition 2. The distance between any two grid cells g_i and g_h is the length of the line between the center points of the two grids and is denoted by $d(g_i, g_h)$ or $d(P_i, P_h)$.

Definition 3. If g is a grid cell, the set $BR(g) = \{h | h \in A, d(h, g) \leq E\}$ is called the neighborhood of g. E is either 1 or $\sqrt{2}$. When *Rob* can only move horizontally or vertically, E is chosen as 1; otherwise $E = \sqrt{2}$.

At any time t, the position of *Rob* is denoted as $P_R(x(t), y(t))$ or abbreviated as $P_R(t)$, $P_R(x, y)$.

Definition 4. $VD(P_R(t)) = \{P | P \in AS,\ d(P,\ P_R(t)) \leq r\}$ is called the visual domain of *Rob* at position $P_R(t)$ and time t.

We make the following conventions and assumptions in the remainder of this paper. Although VD is a circle, in the presentation below and in our experiments we used a square for VD, which facilitates depiction using the grid method. We assume that the distance between any two dynamic obstacles is more than $2r$, which means that during any planning stage at most one dynamic object will be in VD and involved in the planning of the local path. (This assumption can be relaxed, but the algorithm will then be somewhat more complex.)

Rob's velocity is denoted by V_R and is assumed to remain constant within VD. The velocity of Db_i $(i = 1, 2, \ldots, q)$ at time t is denoted by $v_d^i(t)$. The paths and velocities of the Db_i are unknown to *Rob*, but in computing one step for *Rob* we assume that each obstacle moves in a linear path with constant velocity. If some Db_i moves in a nonlinear path or with changing velocity, the method will correct for this in later robot steps because the path is recomputed after each step.

We assume that *Rob*'s start point is g_{begin} and target position is g_{end}, g_{begin} and $g_{end} \notin OS$.

3. The global navigation algorithm of RNA

The global navigation algorithm consists of several parts. First, when the goal lies outside VD, *Rob* cannot see g_{end}. Therefore, it is necessary that g_{end} is mapped into a grid cell near VD that can be used as a local navigation sub-goal. Section 3.1 describes how this local goal is determined. Next, when there are dynamic objects, *Rob* has to predict if there might be a collision and determine how to avoid it. Section 3.2 describes our method for this. Then, Section 3.3 gives the global algorithm, and Section 3.4 illustrates the method with some examples. It is important to remember that this global navigation algorithm is repeated after every robot step.