Colony intelligence for autonomous wheeled robot path planning

1st Antouan Anguelov
Faculty of Computer Systems and Technologies (FCST)
Technical University of Sofia
Sofia, Bulgaria
antouan@tu-sofia.bg

2nd Roumen Trifonov
FCST
Technical University of Sofia
Sofia, Bulgaria
r_trifonov@tu-sofia.bg

3rd Ognian Nakov
FCST
Technical University of Sofia
Sofia, Bulgaria
nakov@tu-sofia.bg

***Abstract*—Mobile robot path planning in dynamic environments answers the question of how to find the shortest path from the initial position to its final destination by avoiding any obstacle. This paper is trying to improve known probabilistic sampling-based algorithms for the road map robot planning introducing a hybrid between wave-front planner cell technique, tangent bug algorithm, and ant colony intelligence strategies, thus minimize the heuristic logic dropping ineffective paths to the target. The proposed colony intelligence tangent bug algorithm (CITBA) determines the shortest path taking into account available historical sensor data for the dynamic surroundings inside the landscape and collected from all autonomous robots while travailing.**

***Index Terms*—Tangent Bug algorithm(TBA), Colony intelligence, colony intelligence tangent bug algorithm (CITBA), ant colony optimization algorithm (ACO), path problem, swarm intelligence, obstacle avoidance, QR NAV, autonomous mobile robots (AMR), mobile wheeled robot (MWR)**

I. Introduction

One of the main problems in mobile robot operation lies in understanding the world surrounding it. The robot needs to be able to handle dangers and to avoid obstacles and operate in unexpected situations saving energy while operating in battery mode, finding a continuous trajectory leading from the initial position of the automaton to its target position. We may classify the robot path planning into two categories as path planning with complete information (the piano movers problem), where detailed information about the obstacles is assumed. In the second model, called path planning with incomplete information, an element of uncertainty about the environment is present[6].

With ACO algorithms, the shortest path in a graph, between two points A and B in uncertainty environment, is built from a combination of several paths solution represented by an ant moving though the search space nodes using historical path data from the others in the colony. Optimization algorithms that simulate such behavior of ants were proposed at the begging of the 90s. The first article on ant algorithms was published in an international journal in 1996 [1], and new research was needed in a few years (Swarm Intelligence and Ant Algorithms). As usual, an ACO algorithm consists of three steps: model selection, pheromone update, and the iteration (Blum, 2005) [5]. In the first stage, the model selection represents the series of ant's position changes that are determined by the stochastic-mechanism based pheromone. In the ACO algorithm, the process of pheromone update is very important. Values are updated by all ants that pass the node. Due to the amount of pheromone accumulated in the node may confuse the other ants choosing a non-optimized path. As ants are marking the path to prove the learning of landmark stability in an indoor environment the cell structure can be marked with fluorescent pigment as well. Such orientation techniques [3] are used often in robotic indoor mapping. Under normal lighting conditions, a QR code would be invisible for humans but becomes visible when near infra-red light is passed over it. This process, known as up-conversion, involves the absorption of photons by the UV/IR ink at a certain wavelength and the subsequent emission of photons at a shorter wavelength. Once illuminated by the near infra-red light, the specific QR code reveals the correct quadrant can be read by an IR camera in a conventional manner [2]. The QR code will guarantee that at some point and at a time AWR will be synchronized and will know the exact position.

The indoor robot landscape often is divided into structures called cells or pixels for easy calculations and representations in the real search space or landscape. The wave-front planner [7] affords the simplest solution to the local minima problem, but can only be implemented in spaces that are represented as grids. The goal pixel is labeled with a two. In the first step, all zero-valued pixels neighboring the goal are labeled with a three. Next, all zero-valued pixels adjacent to threes are labeled with four. This procedure essentially grows a wavefront from the goal where at each iteration, all pixels on the wavefront have the same path length, measured with respect to the grid, to the goal. This procedure terminates when the wavefront reaches the pixel that contains the robot's start location. Unfortunately, the planner has to search the entire space for a path.

In section II we will be using a part of so-called bug algorithms [6] which include a few sample steps such as: 1. Have the shortest ray's direction which implies the nearest distance head toward goal having the goal's center pixel coordinates following on the "m-line" pixels (the line from the starting point to the goal the m-line); 2. if an obstacle is encountered, circumnavigate it until encounter the "m-line" pixel again; 3. leave the obstacle and continue toward the goal on "m-line" pixels;

II. Setup

A. Limitations And Terms

We are implementing a few limitation in our scenario such as:

- MWRs are operating in an indoor environment (the scene) in which the robot travels is defined in a two-dimensional plane;
- the target position changes over time;
- the starting point can be any location in the search space;
- all MWR are having the same target (T) goal;
- the environment includes moving obstacles as well as static ones. The scene may be filled with unknown movable (not fix) obstacles of arbitrary shape and size. The information about the obstacles comes from the robot sensors whose capability is limited to detecting an obstacle only while travailing. In other words, the robot learns about the presence of an obstacle just before hit its surface. The only information MWR is provided with by its sensors is its current coordinates and the distance to an obstacle;
- a few center pixels of the quadrant are marked/printed with UV or IR ink QR code reveals a specific quadrant solving exact point location during the path planning process;

B. Landscape Configuration - The Scene

Landscape contains the following specific:

- QR code positions are known withing the search space;
- the landscape is a two-dimensional landscape divided into the parts - pixels. Structure of 8-connected pixels where neighbors to every pixel touch one of their edges or corners is called quadrant or cell. These pixels inside the quadrant are connected horizontally, vertically, and diagonally and each pixel with coordinates (X+-1, Y+-1) is connected to the pixel at (X, Y). Quadrants also are connected in the same manner as the pixels (x,y) in the Earth axis direction. Each quadrant is marked with QR readable code (pixel size) using IR or UV ink [2] at the center pixel. The size of the robot projection at the ground is equal to the size of a quadrant;
- each obstacle or end/start point is localized within the quadrant coordinates;
- each MWR knows the quadrant locations of its start and end goal. A scene is a plane with a set of obstacles, the robot Start (S) points, and one Target (T) in it.

C. Robot Stack Configuration

The Bug strategy is assuming known direction to goal with local sensing walls/obstacles and MWR encoders for step positions. The only information MWR is provided with by its sensors is its current coordinates and the distance to an obstacle in front. MWR is also given the coordinates of the target. Thus, it can always calculate its direction toward and its distance from the target. The memory available for storing data and intermediate results are not limited but also stored in NoSQL data storage over the wireless communication channel [4].

D. SWARM data-lake for landscape, obstacles and historical data streamed by MWR's while traveling to the goal

TABLE I
SAMPLE DATABASE STRUCTURE

MRS	TSD	SXYC	XYCOi	data	data	RT	n-data
1							
..	..	..	..	..	..	...	...
n							

TSD - total steps to destination, SXYC - Start XY Coordinates, XYCOi - XY Coordinates of Oi, RT - rial timer recordings, etc.
A similar data structure is used for other objects like sensors, landscape pheromone, obstacles where detailed data is kept.

Each MWR contains the following modules:

- Processor and command module (PCM)
- Robot real-time controller (RTC) with a real-time communication system for exchanging messages between robots with special wireless transceiver [4];
- real-time vision with ML capabilities (GPU) module;
- No-SQL streaming database back-end for collecting all sensor data in real-time;
- differential steering wheels for locomotion with PID control based on BLDC motors and PWM controller;
- Sonar Sensors - MWR contains onboard eight ultrasonic traducers for fast and near (¡ 1m.) obstacle detection such as Long-Range Ultrasonic Time-of-Flight Range Sensor capable to sens long obstacle boundaries;
- 9-DOF Inertial Measurement Unit with 3-axis accelerometer, 3-axis magnetometer, and 3-axis gyroscope;
- LiDAR and RGB Camera with for precise volumetric measurements of objects;
- LED IR light source with IR (NoIR) camera at the bottom;

III. Overview Of The Proposed Strategy

Our pathfinder strategy adds a few steps taken from the colony intelligence techniques on top of mentioned in section I. Definitions of the terms and equations that are used in the algorithm are as follows:

$\mathcal{X}$: is the current position of the robot.

$\mathcal{L}$ angle of interest - direction to the goal and can be set grammatically

$\mathcal{Q}$goal: is the goal position.

$\mathcal{Q}$tmp: is the temporary goal position for the advance avoiding of a recognized obstacle.

$\mathcal{O}$: is the i-th discontinuity point as it is shown in figure 1. The discontinuity points are the locations where the sensor information suggest that a continuity interval has ended and another interval has started

$\mathcal{D}$: is the estimated distance from the current position to the goal through the local tangent point with the heuristic function

$$D(x, Qgoal) = D(x, Oi) + D(Oi, Qgoal) \tag{1}$$

$\mathcal{D}$reach : is the output of the heuristic function taking into consideration the best local tangent point.

$\mathcal{D}$followed : is the minimum heuristic value calculated while following the boundary.

$\mathcal{R}$: radius of the sensor range.

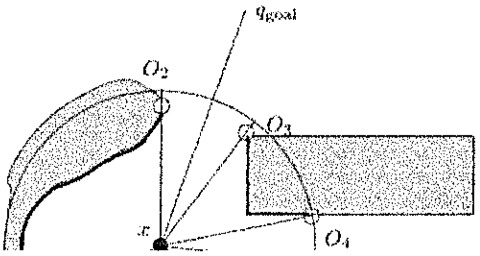

Fig. 1. Discontinuity points of the robot distance sensor reading as radar.

Here is how our proposed modified TBA works shortly: The motion capabilities of the wheeled robot include three possible actions - move toward by one-pixel step in a straight line (up - North pole/bottom - South pole/left - East pole/right side - West Pole) or diagonally (NW, NE, SW, SE), turn/rotate at +-45 degree or stop.

2 motion behaviors (GB and BFB) are adapted by the robot among the two different A and B scenarios:

A. When there is no available historical data for the obstacle nor pheromone at the landscape

Act as usual TBA [8] assuming that robot is still alone in the landscape or no data is shared yet. It is recording/sending all data in regards to data structure mentioned in Table I. Each pixel reached by the MWR is marked with the pheromone number of the robot during its travel in the landscape. Also, all parameters are recorded such as radar sensing points, Oi, etc.

a) goal behavior (GB) : This behavior is activated either when there is no obstacle sensed on the direction from robot to the goal or the robot can find a decreasing heuristic distance from a followed obstacle to the goal. This procedure is a simple gradient-descent in which the robot moves in the direction so that its distance to the goal becomes less and less.

- Follow the goal via "motion-to-goal behavior" while the robot "thinks" (no obstacle in the sensor range) can get to the goal. Having the goal's center pixel coordinates robot follows on the "m-line" pixels - the direct line from the starting point (x) to the goal by sending all data in regards to data structure mentioned in Table I;

b) Boundary Following Behaviour (BFB) : Such behavior is activated when the robot senses an obstacle on the direction to the goal by its range sensor within the radius $\mathcal{R}$.

- If the first obstacle is observed via a range sensor, the robot starts to decrease the speed of steps and switch from "GB to BFB and start approaching the sensed obstacles. In this behavior, the robot detects points of discontinuity on the sensed obstacle. A discontinuity point that separates a continuity interval of an obstacle that is located away from the direction to the goal ($\mathcal{L}$ angle) is not a point of interest.
- The robot starts to calculate a heuristic distance to the goal computed as in equation 1
- The robot will try to find a decreased heuristic distance. When the robot finds that point, the robot starts to move towards that point, therefore in the belief of decreasing the heuristic distance. However, at some stage (can be set programmatic as a distance of radius), this heuristic distance cannot be decreased any more. Here the robot starts compute two different distance parameters such as: $\mathcal{D}$reach and $\mathcal{D}$followed. In general $\mathcal{D}$followed is the shortest distance between the point on the followed boundary (among all of the sensed points since the beginning of the boundary following for this particular obstacle) of the obstacle and the goal. $\mathcal{D}$reach is the shortest distance between the points on the range of the MWR (this range includes both the points that are located on the followed obstacle and the points on the range of the sensor in the free space).
- When the boundary following is initiated, the robot starts to move to the direction of the last Oi selected from the motion to goal behavior. With the direction selected, the robot moves on the tangent line. This tangent line is perpendicular to the line that connects the robot and the point closest to the robot on the followed obstacle. The MWR continues to execute this mode while the following statement is valid:

$$Dfollowed >= Dreach \tag{2}$$

Whenever $\mathcal{D}$reach falls below $\mathcal{D}$followed., the robot switches back to the GB mode.
- However, if robots encounter a point that it has followed before, stops the execution and exits with the no solution at the moment signal. As more data come to data-lake and this data is valuable then the MWR starts following a new direction to the goal.

If the robot sens existing sensor information in the back-end data-lake pushed by others its switches to the 2nd scenario where colony intelligence tangent bug algorithm (CITBA) is activated:

B. When there are reach historical data for obstacles and pheromone left at the landscape

a) goal behavior (GB) : this behavior is the same as it is in the first scenario.

b) Skip Following Boundary (SFB) : Unlike the A scenario, paragraph b, MWR tries to recognize the obstacle sensed having the data from other MWR sensors as well as pheromone left in the landscape. It sends the LIDAR/Sonars data in real-time gets back from the back-end computed coordinated (if there is a much at all) for the $\mathcal{D}$followed for the expected boundary of the obstacle in front as in figure

2, tries to shortcut boundary following. Now the MWR has a new temporary goal to follow $Qtmp$ while the main goal coordinates are backed up and stored in the pathfinder FIFO stack. In case the obstacles are recognized but with angle change (rotation), the back-end calculates and approximate the Qtm having mind timestamps of the sensor historical measurements and the pheromone left as blue line at Fig 2.

The heuristic function now changes to:

$$Dn(x, Qtemo) = Dn(x, Oi) + Dn(Oi, Qtemp) \quad (3)$$

Where $\mathcal{N}$ is $\mathcal{N}$-obstacle sensed by the sonars located in MWR.

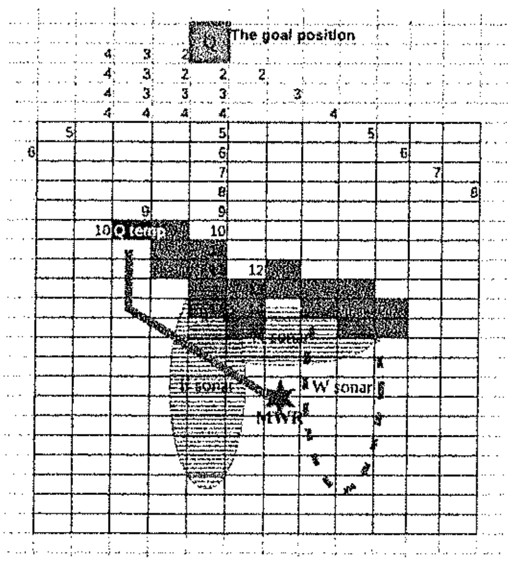

Fig. 2. Sonar data is recognizing an object by sensing the wave planner pixels: 13, 14 and 15.

We expect to have situation where particular obstacle could not be recognized. In this case algorithm continue with scenario A again.

IV. Conclusions

For the algorithm described in the chapter, it should be noted that only the main steps are given and that implementation details are missing. It this paper we proved an abstract model that will be used in further Ph.D. thesis. That is why, further researches are needed, involving researchers for innovative SWARM approaches for robot pathfinding in indoor environments.

Acknowledgment

This work was encouraged by the Faculty of Computer Systems and Technologies at the Technical University of Sofia. We would like to thank our colleagues from the faculty for the valuable comments, suggestions, and encourage us to work on colony intelligence for pathfinding.

References

[1] Dorigo, M., Maniezzo, V., and Colomi, A., The Ant Sys- tem: Optimization by a Colony of Cooperating Agents, IEEE Trans. Systems, Man Cybernetics, Part B, 1996, vol. 26, no. 1, pp. 29-41.

[2] Jeevan M Meruga1, William M Cross1, P Stanley May2, QuocAnh Luu2, Grant A Crawford1 and Jon J Kellar1, Security printing of covert quick response codes using upconverting nanoparticle inks, Published 11 September 2012 • 2012 IOP Publishing Ltd.

[3] Thrun S (2003) Robotic mapping: a survey. In: Lakemeyer G, Nebel B, editors. pp. 1–35. Exploring artificial intelligence in the new millennium: Morgan Kaufmann.

[4] Antouan Anguelov, Roumen Trifonov and Ognian Nakov. 2019. Emerging and secured mobile ad-hoc wireless network (MANET) for swarm applications. In Proceedings of 9th Balkan Conference on Informatics (BCI'19). September 26–28, 2019, Sofia, Bulgaria, 4 pages. https://doi.org/10.1145/3351556.3351557

[5] Christian Blum, Ant colony optimization: Introduction and recent trends, ALBCOM, 11 October 2005

[6] Vladimir J. Lumelsky and Alexander A. Stepanov, Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape, Algorithmica (1987) 2: 403–430

[7] Howie Choset et al., Principles of Robot Motion—Theory, Algorithms, and Implementation, The MIT Press, Cambridge, Massachusetts, London, England, © 2005 Massachusetts Institute of Technology

[8] Kadir Firat Uyanik, A study on Tangent Bug Algorithm, KOVAN Research Lab. Dept. of Computer Eng. Middle East Technical Univ. Ankara, Turkey