AQTR 2004 (THETA 14)
2004 IEEE-TTTC - International Conference on Automation, Quality and Testing, Robotics
May 13 – 15, 2004, Cluj-Napoca, Romania

HYBRID GOAL ACQUISITION SYSTEM FOR PIONEER 2 MOBILE ROBOT

Radu Robotin, Gheorghe Lazea and Sorin Herle

Department of Automation, Technical University of Cluj-Napoca,
C. Daicoviciu str. 15, 3400 Cluj-Napoca, ROMANIA
E-mail: {Radu.Robotin,Gheorghe.Lazea, Sorin.Herle}@aut.utcluj.ro

Abstract: This paper presents our hybrid approach using the D* algorithm for generating focal points on an optimal path for a robot operating with a map of the environment. These points are used by potential field based navigator. The robot's sensor is able to measure arc costs in the vicinity of the robot, and the known and estimated arc values comprise the map. The paper describes the algorithm, illustrates its operation, presents our approach for implementation, and then concludes with a proof of operation.

Key words: mobile robot, hybrid path planning, potential field navigation, environment map.

1. INTRODUCTION

This paper describes a navigation system which integrates D* algorithm with a potential field navigation system. The advantage of such an approach is that D* (see [5] for a detailed description of the algorithm and the experimental results) works with a low resolution map of the environment to reduce the storage and processing requirements. It computes points on the optimal path toward the goal, points that are passed to a local navigator based on a potential field navigation algorithm [1]. This algorithm uses a high resolution local map, mainly to perform local obstacle avoidance and course following. The main reason is that it can be efficient especially in cluttered environments where the computational requirements for navigation only with D* can be extremely high [3]. Our goal was to extend the existing software architecture [4] for Pioneer 2 mobile robot to cope with requirements for dynamic environment path planning.

1.1 D Algorithm for optimal path planning*

D* uses an OPEN list to propagate information about changes to the arc cost function and to calculate path costs to states in the space. Every state X has an associated *tag t(X)*, such that if X has never been on the list *t(X)=NEW*, if X is currently on the list *t(X)=OPEN*, and if X is no longer on the list *t(X)=CLOSED*. For each visited state X, D*

maintains an estimate of the sum of the arc costs from X to G given by the path cost function *h(X)*.

Given the proper conditions, this estimate is equivalent to the minimal cost from state X to G. For each state X on the OPEN list (i.e., *t(X)=OPEN*), the key function, *k(X)*, is defined to be equal to the minimum of *h(X)* before modification and all values assumed by *h(X)* since X was placed on the OPEN list. The key function classifies a state X on the list into one of two types: a RAISE state if *k(X)<h(X)*, and a LOWER state if *k(X)=h(X)*. D* uses RAISE states on the OPEN list to propagate information about path cost increases and LOWER states to propagate information about path cost reductions. The propagation takes place through the repeated removal of states from the list. Each time a state is removed from the OPEN list, it is *expanded* to pass cost changes to its neighbors. These neighbors are in turn placed on the OPEN list to continue the process.

States are sorted on the OPEN list by a *biased f()* value, given by $f_B(X,R_i)$, where X is the state on the OPEN list and R_i is the robot's state at the time X was inserted or adjusted on the OPEN list. Let $\{R_0, R_1, \ldots, R_N\}$ be the sequence of states occupied by the robot when states were added to the OPEN list. The value of $f_B()$ is given by $f_B(X, R_i) = f(X, R_i) + d(R_i, R_0)$ where $f()$ is the estimated robot path cost given by $f(X, R_i) = h(X) + g(R_i, R_{i-1})$ and *d()* is the *accrued bias* function.

The function *g(X,Y)* is the focussing heuristic, representing the estimated path cost from Y to X. The list states are sorted by increasing $f_B()$ value, with ties in $f_B()$ ordered by increasing *f()*, and ties in *f()* ordered by increasing *k()*. Ties in *k()* are ordered arbitrarily. Thus, a vector of values $\langle f_B(\circ), f(\circ), k(\circ) \rangle$ is stored with each state on the list. Whenever a state is removed from the OPEN list, its *f()* value is examined to see if it was computed using the most recent focal point. If not, its *f()* and $f_B()$ values are recalculated using the new focal point and accrued bias, respectively, and the state is placed back on the list. Processing the $f_B()$ values in ascending order ensures that the first encountered *f()* value using the current focal point is the minimum such value, denoted by f_{min}.

1.2 Potential field local navigator

Figure 1 shows a high-level description of the navigation system, which consists of the global navigator D* and a potential field based local navigator. The whole system is integrated with existent robot software using Saphira functions. The client – server architecture already implemented on the robots provides the necessary background for implementing a high level navigation system. This background comprises functions for robot motion control and a local perceptual space which stores the available sonar readings and current robot configuration in space. The global navigator maintains a coarse-resolution map of the environment, consisting of a Cartesian lattice of grid cells. Each grid cell is labeled untraversable, high-cost, or traversable. The global navigator initially plans a trajectory to the goal and sends focal points on this trajectory to the local navigator which is responsible to steering commands to the robot controller.

The local navigator treats the robot represented as a point in configuration space as a particle under the influence of an artificial potential field (U) whose local variations are

expect to reflect the structure of the free space [2]. U is constructed as a sum of two more elementary potential functions:

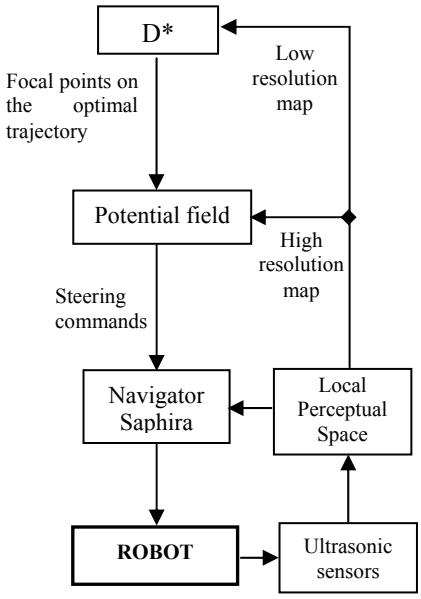

Figure 1. Navigation system diagram.

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{1}$$

where $U_{att}(q)$ is the attractive potential associated with each focal point on the optimal trajectory toward goal configuration (q_{goal}) and $U_{rep}(q)$ is the repulsive potential associated with the obstacle region. The attractive potential is defined as a parabolic well:

$$U_{att}(q) = \frac{1}{2} \cdot \zeta \cdot \rho^2_{goal}(q) \tag{2}$$

with ζ a positive scaling factor and $\rho_{goal}(q)$ denoting the Euclidean distance $\|q - q_{goal}\|$.

Thus the robot is attracted toward each of the focal points (q_{goal}) on the path with a force:

$$\vec{F}_{att}(q) = -\vec{\nabla} U_{att}(q) = -\zeta(q - q_{goal}) \tag{3}$$

The repulsive potential is defined as follows:

$$U_{rep}(q) = \begin{cases} \frac{1}{2} \cdot \eta \cdot \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & if\ \rho(q) < \rho_0 \\ 0 & if\ \rho(q) > \rho_0 \end{cases} \tag{4}$$

where η is a positive scaling factor, $\rho(q)$ denotes the distance from q to the C-obstacle region $C\beta$:

$$\rho(q) = \min_{q' \in C\beta} \|q - q'\| \tag{5}$$

and ρ_0 is a positive constant, the *distance of influence* of the C-obstacles. The function U_{rep} is positive or null, tends to infinity as q gets closer to the C-obstacle region and is null when the distance of the robot's configuration to the C-obstacle region is grater than ρ_0.

If $C\beta$ is a convex region with a piecewise differentiable boundary, $\rho(q)$ is differentiable everywhere in the free space C_{free}. The artificial repulsive force deriving from U_{rep} is:

$$\vec{F}_{rep}(q) = -\vec{\nabla} U_{rep}(q) =$$

$$\begin{cases} \eta\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\vec{\nabla}\rho(q) & \text{if } \rho(q) \le \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases} \tag{6}$$

If q_c is the unique configuration in $C\beta$ that is closest to q, i.e. that achieves $\|q - q_c\| = \rho(q)$, the gradient $\vec{\nabla}\rho(q)$ is a unit vector pointing away from $C\beta$ and supported by the line passing through q_c and q.

Using the potential field approach, at every configuration q, the artificial force $\vec{F}(q)$ determines the acceleration, and by knowing the dynamics of the mobile robot it is possible to compute the forces / torques that should be delivered by the actuators at each instant, so that the robot behaves as a particle in a force field.

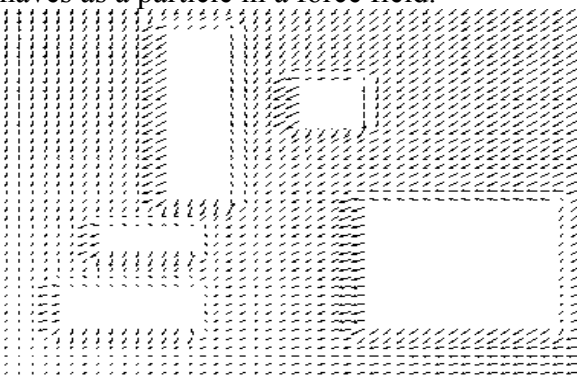

Figure 2. Gradient vector of the combined artificial potential field $U_{att} + U_{rep}$.

2. IMPLEMENTATION & RESULTS

The approach presented in this paper is to construct the path as a product of successive path segments, starting at the initial configuration q_{init}, and the goal configuration is the focal point delivered by the D* algorithm, which lies on the optimal

path toward the final configuration. Each segment is orientated along the negated gradient of the potential function computed at the configuration attained by the previous segment.

Let q_i and q_{i+1} be the origin and the extremities of the i^{th} segment in the path. Let $x_j(q_i),\ j=1,\ldots,3$ be the coordinates of q_i. The coordinates of the configuration q_{i+1} are:

$$x_j(q_{i+1}) = x_j(q_i) + \delta_i \cdot t_j(q_i) \tag{7}$$

with δ_i denoting the length of the i^{th} increment (measured with the Euclidian metric of $\boldsymbol{R^2}$) and the unit vector:

$$\vec{t}(q_i) = \frac{\vec{F}(q_i)}{\left\|\vec{F}(q_i)\right\|} \tag{8}$$

For a planar object:

$(x_1, x_2, x_3) = (x, y, \theta) \in \Re^2 \times [0, 2\pi)$

with x and y being the coordinates of the robot's reference point at q, and θ being the angle (modulo 2π) between the $\boldsymbol{x}$-axes of the reference frame and the frame attached to the robot. Then:

$$\begin{cases} x(q_{i+1}) = x(q_i) + \delta_i \dfrac{\partial U}{\partial x}(x, y, \theta) \\ y(q_{i+1}) = y(q_i) + \delta_i \dfrac{\partial U}{\partial x}(x, y, \theta) \\ \theta(q_{i+1}) = \theta(q_i) + \delta_i \dfrac{\partial U}{\partial x}(x, y, \theta) \mod 2\pi \end{cases} \tag{9}$$

δ_i should be taken smaller than the Euclidian distance between current and goal configuration, usually taken equal to some small predefined constant.

A number of modifications were made to D* to adapt it to an actual robot system. As the robot drives toward the goal, its sensors scan the terrain in front of the robot. The local navigator processes this sensor data to find obstacles and sends the (x,y) locations of detected obstacles (untraversable cells) to D* at regular intervals. Additionally, local navigator sends (x,y) locations of cells known to be devoid of obstacles (traversable cells). Since the D* map is used to represent a global area, its grid cells are of coarser resolution than local navigator's (i.e., 1 meter versus 0.2 meter). If an obstacle is added to a previously empty map cell or all of the obstacles are deleted from a previously obstructed cell, then this constitutes a significant event within D* since a traversable cell becomes an untraversable cell or an untraversable cell becomes a traversable cell, respectively. The C-space expansion provides some buffering to keep the robot away from obstacles. Additional buffering in the form of a high-cost field leads to better performance. The idea is to penalize the robot cost-wise for passing too close to an obstacle, causing the robot to approach obstacles only when open space is unavailable. We have conducted our experiments in both simulation and real-life indoor environment. For D* algorithm the Euclidian metric was used as focusing heuristic. The focal points for the obstacle avoidance routine (potential field) were selected to be equidistant on the optimal path generated by D*. The system uses all available prior map data to plan a route to the goal, and then begins to follow that route using its ultrasonic rangefinder to examine the terrain in front of the

robot for obstacles. If a discrepancy is discovered between the sensor data and the map, the map is updated and a new, optimal path is planned to the goal.

Figure 3 shows a run of D* algorithm over a cluttered environment, represented as a rectangular grid. (Black cells – untraversable; Light gray – low cost; Dark gray – high cost).

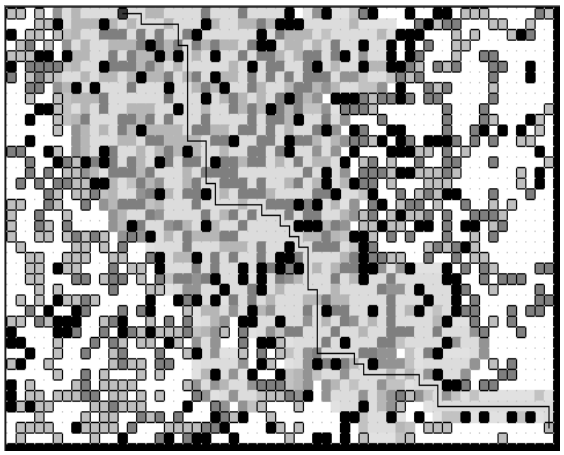

Figure 3. Results of D* algorithm: optimal path and cell expansion.

3. CONCLUSIONS AND FUTURE WORK

This paper describes a navigation system for goal acquisition in partially unknown environments. The system uses all available prior map data to plan a route to the goal, and then begins to follow that route using its ultrasonic rangefinder to examine the terrain in front of the robot for obstacles. If a discrepancy is discovered between the sensor data and the map, the map is updated and a new, optimal path is planned to the goal. In the near-term, a number of improvements will be made to minimize unnecessary processing and increase overall system speed. We plan to improve the performance of local navigator in terms of speed and maximum range in order to support higher robot speed.

4. REFERENCES

[1] Fujimura, K. (1991). Motion Planning in Dynamic Environments, Springer-Verlang, ISBN 4-431-70083-8, Tokyo, Japan.

[2] Latombe, J.-C. (1991). Robot Motion Planning, Kluwer Academic Publishers, ISBN 0-7923-9206-X, Boston MA

[3] Nilsson, N. J. (1980). Principles of Artificial Intelligence, Tioga Publishing Company, ISBN 3-540-11340-1, Palo Alto CA

[4] Robotin, R.; Lazea, G.; Herle S. (2002). Hybrid Navigation System for Pioneer 2 Mobile Robot, *Proceedings of 13th DAAAM Symposium*, Katalinic, B. (Ed), pp. 471-472, ISBN 3-901509-29-1, 23-26th October 2002, Vienna, Austria, EU.

[5] Stentz, A. (1994). Optimal path planning for partially known environments, In *Proceedings of the 11th IEEE International Conference on Robotics and Automation*, San Diego, California