

A Genetic Algorithm for Robust Motion Planning

Domingo Gallardo, Otto Colomina, Francisco Flórez, Ramón Rizo
domingo,otto,florez,rizo@dtic.ua.es

Grupo i3a: Informatica Industrial e Inteligencia Artificial
Departamento de Ciencia de la Computacion e Inteligencia Artificial
Universidad de Alicante
San Vicente E-03080, Spain

Resumen

This paper proposes a solution by genetic algorithms to the problem of planning a robust and suboptimal trajectory in the velocity space of a mobile robot. Robust trajectories are obtained introducing cumulative noise in the evaluation of the fitness function and introducing modifications in the genetic algorithm to taking into account this new feature. Results are presented that show the performance of the algorithm in different environments and the influence of the noise in the planned trajectories.

Keywords: nonholonomic motion planning, robust robot motion, evolutionary computation in noisy environments.

1 Introduction

The problem of path planning for mobile robots has been addressed with many techniques, both algorithmic ([5]) as evolutionary ([7, 4, 6, 1]). Most of these approaches solve the problem in the robot configuration space (usually defined by means of its x and y position and its θ orientation). In this case, the solution is presented as a continuous sequence of space configurations (a path in the plane) that does not collide with obstacles and connects the initial and final points. A path, however, does not solve directly the problem of moving the robot, since it lacks of an associated time law (different linear and angular velocities may generate the same path).

In order to completely solve the problem a sequence of velocities must be planned, searching not the configuration space but the velocity space (in the case of synchro-drive robots, as the one used in this work, these velocities are v , the linear velocity and ω , the angular velocity). In this new space obstacles can not be defined geometrically, and many of the algorithmic techniques defined so far can not be applied. Additionally, for a great number of robots, is needed to impose constraints that locally reduce the dimensionality of its velocity space. For instance, a robot that moves like a car, as it is the case of synchro-drive robots, can not move laterally and has a limited turn angle. This problem is called trajectory planning with nonholonomic constraints, and it is a research field in the areas of control and robotics [5, 3, 2]. All the solutions proposed so far have two important problems:

1. Suboptimal solutions: most of the algorithmic techniques that solve the problem of nonholonomic motion planning find a trajectory, but they do not consider practical issues like temporal cost of the trajectory, regularity, etc.
2. Trajectory robustness: once the trajectory is planned, it has to be executed in a real robot, having to consider the uncertainty in the plan execution. This uncertainty is due to noisy execution of velocity commands and the error associated to odometry. This type of uncertainty is cumulative, so long trajectories suffer more than short ones.

For this reason, planned trajectories have to be robust, in the sense that the introduction of small changes in them do not change too much the final result. This feature of robustness is not guaranteed by any approach used so far to solve the problem.

In this work a solution to the planning of nonholonomic trajectories using genetic algorithms is proposed. This solution addresses the problems treated above. The search of the best trajectory according to some criteria will be done by introducing the criteria in the fitness function, and the robustness will be accomplished by introducing in the fitness function a term of gaussian cumulative noise.

2 Problem formulation

In this section we formulate the motion equations of a synchro-drive robot. The coding of the planned trajectories and the fitness function are based on these equations.

The state variables of the mobile robot are its position (x, y) , its orientation (θ) and its angular and linear velocities $(\omega$ and v , respectively). We consider the angular and linear accelerations as constants $(a_\omega$ y $a_v)$. They tend to be low in real

mobile robots implementations, in order to not tension the physical structure of the robot and to obtain smooth movements. Due to this, the acceleration have to be considered in order to obtain results that can be applied to real robots.

We use the variables c_v y c_ω to model the robot control commands. They take discrete values $\{-1, 0, 1\}$ and define if, for a given time, the velocities have to be decremented, not modified or incremented. The increment of the velocities will be given by the acceleration constants.

$$\omega(t_n) = \omega(t_0) + \int_{t_0}^{t_n} a_\omega c_\omega(t) dt \quad (1)$$

$$v(t_n) = v(t_0) + \int_{t_0}^{t_n} a_v c_v(t) dt \quad (2)$$

The position and angular direction of the robot at a given time are modeled by the following equations.

$$\theta(t_n) = \theta(t_0) + \int_{t_0}^{t_n} \omega(t) dt \quad (3)$$

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t) \cos \theta(t) dt \quad (4)$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t) \sin \theta(t) dt \quad (5)$$

These equations can be simplified if it is assumed that the robot has to be controlled discretely, with control intervals of time Δt , being the control variables c_v y c_ω constant for a given time interval. In this way, the former equations can be simplified

$$x(t_n) = x(t_0) + \sum_{i=0}^{n-1} \int_{t_i}^{t_i+\Delta t} (v(t_i) + \Delta_i^t v) \cos(\theta(t_i) + \Delta_i^t \theta) dt \quad (6)$$

where

$$\Delta_i^t v = a_v c_v(t)(t - t_i) \quad (7)$$

$$\Delta_i^t \omega = \omega(t_i) c_\omega(t_i)(t - t_i) + \frac{1}{2} a_\omega(t_i) c_\omega(t_i)(t - t_i)^2 \quad (8)$$

So the problem of finding a trajectory in a velocity space can be formulated as the search of a temporal sequence of values for the commands c_v y c_ω

$$\langle (c_{v_{t_0}}, c_{\omega_{t_0}}), (c_{v_{t_1}}, c_{\omega_{t_1}}), \dots, (c_{v_{t_n}}, c_{\omega_{t_n}}) \rangle$$

that makes the robot move from the initial configuration ($x = x_0, y = y_0, \theta = \theta_0, v = 0, \omega = 0, t = t_0$) to the goal position ($x = x_{\text{obj}}, y = y_{\text{obj}}, v = 0, \omega = 0, t = t_n$). Note that robot has to finish in a static configuration and that we do not impose any restriction for the final orientation.

As additional constraints, we try to minimize t_n and to obtain a robust trajectory, in the sense above stated.

3 Trajectory planning using genetic algorithms

We have chosen genetic algorithms to generate robot trajectories mainly for two reasons: firstly, they are a powerful tool for searching in high dimensional spaces, like in this case. Secondly, the method imposes few mathematical constraints in the shape of the function to optimize. In this way, this method is applicable to the generation of a wide range of behaviors (obstacle avoiding, wall following, etc.).

3.1 Solution encoding

Each individual in the population represents a trajectory starting from the initial point and trying to reach the final point. Not only a sequence of coordinates connecting the initial and the final point is needed, but also linear and angular velocities must be specified. So, a chromosome is a string of pairs v, w representing reference linear and angular velocities for the robot at each point.

$$\langle (v_{t_1}, w_{t_1}), (v_{t_2}, w_{t_2}), \dots, (v_{t_n}, w_{t_n}) \rangle$$

This sequence has to be converted into velocity commands like those in the section 2 to be executed on the robot.

3.2 Fitness function

Fitness function measures the optimality of each trajectory considering two factors: the distance between the final robot position and the goal point and the time invested.

$$f = \alpha d_{\text{obj}}(t_n, \sigma) + \beta t_n$$

where:

- $d_{\text{obj}}(t, \sigma) = \sqrt{(\chi(t, \sigma) - x_{\text{obj}})^2 + (\psi(t, \sigma) - y_{\text{obj}})^2 + v(t, \sigma)^2 + \varphi(t, \sigma)^2}$ represents the euclidean distance in the space (x, y, v, w) between the goal point and the final robot position
- t_n represents the time invested in reaching the final point
- α, β are weighting coefficients

To ensure that infeasible individuals (i.e., trajectories intersecting with obstacles) get a worse (greater) fitness than feasible ones, the fitness of the worst feasible trajectory is added to each of the infeasible trajectories.

The $v(t, \sigma)$ and $\varphi(t, \sigma)$ functions represent the linear and angular velocities, respectively, which are obtained introducing gaussian noise with standard deviation σ in the equations (1) and (2):

$$\varphi(t_n, \sigma) = \omega(t_0) + \int_{t_0}^{t_n} (a_\omega c_\omega(t) + \rho(t, \sigma)) dt \quad (9)$$

$$v(t_n, \sigma) = v(t_0) + \int_{t_0}^{t_n} (a_v c_v(t) + \rho(t, \sigma)) dt \quad (10)$$

The $\chi(t, \sigma)$ and $\psi(t, \sigma)$ functions represent the x and y values obtained using those velocities. The noise term, $\rho(t, \sigma)$, models the uncertainty in positions and velocities which would take place if the commands were executed on a real robot.

From an evaluation point of view, this uncertainty means that an individual will have not a unique fitness value. We have adopted the strategy of evaluating each individual N times and setting its fitness to the worst value obtained. This approach is addressed to promote robust trajectories. To avoid improving an individual fitness accidentally with the introduction of noise, we also evaluate each individual in absence of noise.

4 Genetic operators

- **Crossover:** this operator recombines two trajectories. A random crossover point is calculated independently for each parent, and the right part of the chromosome is exchanged between the two parents. This method produces variable length individuals, which are able to grow and eventually reach the goal point.
- **Mutation:** each mutation consists of adding a random value from a normal distribution with 0 mean and variance equal to 5 m/s to a gene value (which a variance equal to 5 degrees/s for angular velocities).

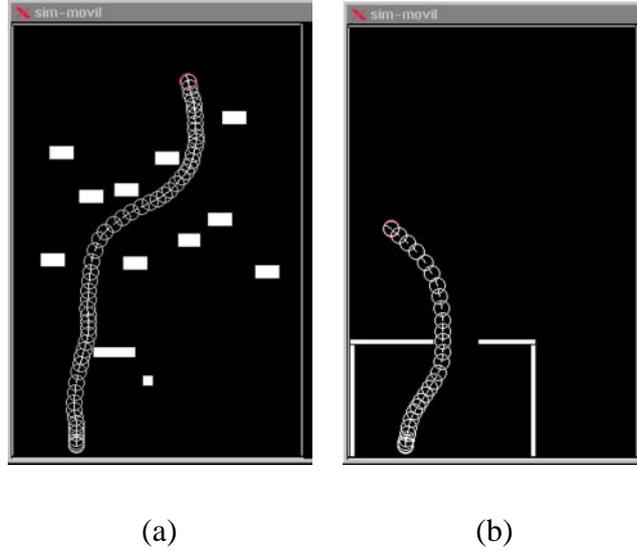


Figure 1: Results obtained in several environments without noise.

5 Results

5.1 Results obtained in several environments without noise

Figure 1 shows two trajectories which were generated using the genetic algorithm in two different environments, without noise in the fitness function. The parameter setting was the following: population size $m = 100$, generations $n = 100$, crossover probability $p_c = 0.75$ and mutation probability $p_m = 0.01$.

It is interesting to note that these trajectories usually reach high velocities, but passing too close to obstacles. A real robot following these trajectories would probably collide with the objects in the environment.

6 Results comparison with and without noise

In order to check the effects produced by noise introduction in the features of the obtained trajectories using the evolutionary algorithm, several tests have been performed in the same environment (figure 2), without changing the parameters between runs, except standard deviation σ of the gaussian noise. Table 1 reports the obtained results averaged over 20 runs with each noise level. The parameter D_{min} represents the smallest distance from the robot to the obstacles along the trajectory. The parameter V_{avg} is the average linear velocity.

As can be seen in table 1, noise forces trajectories to pass farther from obsta-

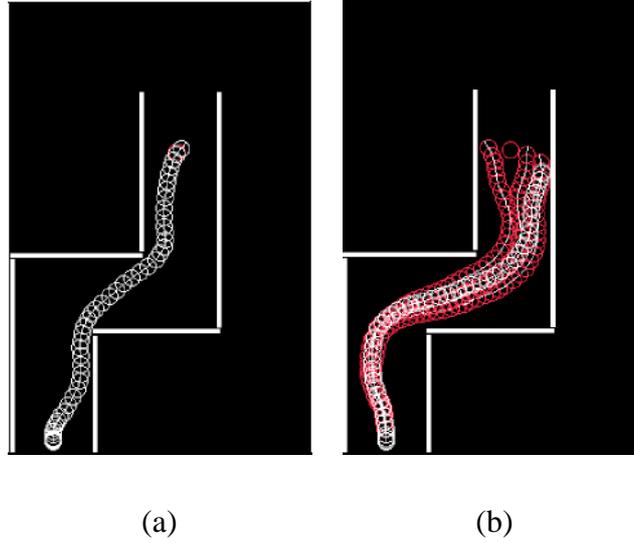


Figure 2: Results obtained for the *corridor* environment

cles. Noise-free trajectories achieve faster average velocities because they have no distance constraints (figure 2), but noisy trajectories are more robust and safer if they have to be executed on a real robot.

PARAMETER	NOISE LEVEL (σ)		
	0.0	0.04	0.1
$D_{min}(cm)$	38.2	39.8	43.8
$V_{avg}(cm/s)$	51.31	50.92	46.4

Table 1: Results obtained for the *corridor* environment.

7 Conclusions

We have presented a solution to the robust trajectory planning problem, using an evolutionary algorithm that searches in the velocities space. We have obtained satisfying results and the robustness of trajectories is improved in comparison with other methods.

A cumulative noise term has been introduced in the fitness function and the evolutionary algorithm has been adapted to find a suboptimal and robust solution to satisfy this function.

Future work includes to modify the fitness function to plan trajectories corresponding to various schemata of local behavior, as wall following or navigating throughout corridors. The resulting trajectories will be used to learn a local controller capable of guiding the robot using the sensed data. We are also exploring some alternatives in solution encoding to obtain better results in environments with strong local minima.

Referencias

- [1] J.M. Ahuactzin, E.G. Talbi, P. Bessiere, and E. Mazer. Using genetic algorithms for robot motion planning. In *European Conference on Artificial Intelligence (ECAI92)*, 1992.
- [2] R. Chatila, M. Khatib, H. Jaouni, and J-P. Laumond. Dynamic path modification for car-like non-holonomic mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1997.
- [3] A. Divelbiss and J. Wen. Nonholonomic motion planning with inequality constraints. In *Proc. IEEE Int. Conf. Robotics and Automation*, 1994.
- [4] A. Doyle. *Algorithms and computational techniques for robot path planning*. PhD thesis, School of Electronic Engineering and Computer Systems, University of Wales, 1995.
- [5] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
- [6] M.J. Rendas and W. Tetenoire. Definition of exploratory trajectories in robotics using genetic algorithms. In *Tenth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 1997.
- [7] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski. Adaptive evolutionary planner/navigator for robots. *IEEE Trans. on Evolutionary Computation*, 1, 1997.