Neural Network-Based Coordinated Motion Planning of Multiple Mobile Robots

Nirmal Baran Hui

Abstract

The present paper deals with the coordination issues of multiple wheeled robots working in a common dynamic environment, in a decentralized manner. Two different motion planners, one based on Neural Network and other using the potential field method have been developed to plan the motion of the robots. A strategic approach has been proposed to develop the decision making support. Performance of the developed approaches have been tested through computer simulations. Proposed strategy has been found to solve the conflicts and induced coordination among the agents.

Keywords: Mobile Robots, Coordination, Robot Motion Planning, Neural Network, Potential Field Method.

1 Introduction

Multiple mobile robots working in a common work-space will have to negotiate their motion. Main aim here is to find collision-free paths of all the robots while they are moving from their respective starting points to the destinations. The path for each robot is constrained by its current position, the goal point and the movements of the other robots. Therefore, it is a complicated task and there must be an intelligent and adaptive motion planner to solve the same. Motion planner can be designed in two ways. Firstly through centralized manner, in which there will be a master robot who will dictate the motion plan to other robots and other robots obey the master. However, it suffers from several drawbacks. Therefore, most of the researchers are preferring the other option, which is known as decentralized motion planning. In case of decentralized system, each robot carries out tasks cooperatively. This sort of system offers more freedom to the robots and allows each robot to take the decision independently/selfishly. However, to build a full-proof decentralized system, it should have the following design characteristics: **Coordination, Communication**, and **Cooperation** [1]. A decentralized system has been applied in many places, such as, sharing of files, transportation of goods, maintaining the signaling system of airplanes, medical diagnosis, Industry automation, etc.

Quite a few researchers [2] have considered soccer playing could be one example of decentralized motion planning of multiple agents. They have set some long term goal. Most of the researchers are presently trying to fulfill the long term goal set by Alan Macworth [3]. Several algorithms / techniques are available in the literature. Out of which, mixed integer non-linear programming method [4], reinforcement

Nirmal Baran Hui
Department of Mechanical Engineering, National Institute of Technology Durgapur,
West Bengal - 713209, India. E-mail: nirmal.hui@me.nitdgp.ac.in.

learning [5], probabilistic road map [6], potential field method [7], cell-to-cell mapping [8] are important to mention. Soft computing-based motion planning schemes have also been proposed by some researchers. Both fuzzy logic [9] as well as neural network [10] have also been proposed by the investigators for solving the similar kind of problems. However, all such methods (both algorithmic and soft computing-based) have some common drawbacks, such as: (a) competition among the robot is not considered, (b) role of a agent is kept fixed, thus the robustness and adaptability of a agent is very low and (c) a particular agent is allowed to navigate in a fixed zone.

Therefore, the coordination among the agents is still a challenging research issue in robotics. In the present study an attempt was made to solve the motion planning problem of multiple mobile robots moving in a common dynamic environment. The rest of the paper is structured as follows: In Section 2, coordination of multiple mobile robots have been studied. Developed navigation schemes and the preferred coordination strategy are discussed in Section 3. Results are presented and discussed in Section 4. Finally, some concluding remarks are made and the scopes for future work are indicated in Section 5.

2 Coordination of Multiple Mobile Robots

In a 2-D space, multiple robots are moving starting from an initial position with different speed and in different direction. Starting and final positions of all the robots are defined a-priori and those of one robot are different from the other. The total path (starting from a pre-defined position to a fixed goal) of any robot is assumed to be a collection of some small segments (either a straight one or a combination of straight and curved paths), each of which is traversed during a fixed time ΔT. If a robot finds any other robot to be critical robot (which may collide with the planning robot if it moves along the previous direction and by maintaing the same speed), the motion planner is activated. Otherwise, the robot moves toward the goal in a straight path with a maximum possible velocity. The task of the motion planner is to determine the *acceleration* (a) and *deviation* (θ_1) of the robot based on the *distance* and *angle* inputs, to avoid collision with it. Since distance is one of the major factor based on which critical robot is identified. Thus, there is a chance that a robot critical to the planning robot may also consider the planning robot to be critical during its own motion planning. As a result of which, both will get deviated from their previous direction of motion and their speed will also be hampered. It will then increase the traveling time to be taken to reach the goal by the robots. In order to avoid the same a strategic decision tool is adopted to predict which robot will cooperate with the other in a particular time step. This process of motion planning will continue, until all the robot reaches their individual destination and total traveling time for each robot (T) is then calculated by adding all intermediate time steps taken by the robot to reach it. It is important to mention that the last time step (T_{rem}) may not be a complete one and it depends on the distance left uncovered (d_{goal}) by the robot. If it (i.e., the goal distance d_{goal}) comes out to be less than or equal to a predefined minimum distance (d_{min}), it starts decelerating and stops at the goal. Again, sometimes the robot's motion as provided by the motion planner may violate its kinematic and/or dynamic constraints. In such a situation, the robot is stopped at the present position itself. Our aim is to design a suitable adaptive and cooperative motion planner, so that all the robots will be able to reach their destination with the lowest possible traveling time by avoiding collision among themselves. Therefore, the present problem can be treated as a constrained traveling time (T) optimization problem as indicated below.

$$Minimize\, T = \sum_{i}^{n} (U^i \times \Delta T + T^i{}_{rem}), \tag{1}$$

where U^i indicates the number of complete time steps for robot i and n denotes the total number of robots present in the environment.
subject to: (a) the path is collision-free, and (b) both the kinematic and dynamic constraints of the robots are satisfied.

3 Developed Navigation Schemes & Coordination Strategy

Several methods had been tried by various investigators to solve similar kind of problems. The authors have developed two motion planning approaches along with a novel coordination strategy. Neural network has the capability of solving different complex real-world problems and it may also provide a feasible solution to the present problem. Therefore, an attempt has been made to develop an NN-based motion planner in Approach 1. On the other hand, potential field-based motion planner is the widely used traditional motion planner. Thus, performance of Approach 1 is compared with a potential field-based motion planner, i.e., Approach 2. Both these approaches have been discussed in subsequent sections, respectively.

3.1 Approach 1: neural network-based motion planner

Figure 1 shows the architectural graph of a three-layered feed forward neural network with a single hidden layer. In the first layer, two neurons representing the two inputs of the controller, such as *distance* of the robot from its most critical obstacle and their included *angle* with reference to the goal are considered, in the present work. There are two neurons at the output layer expressing two different outputs of the controller, namely *deviation* and *acceleration* of the robot required to avoid collisions with the moving obstacles and to reach the destination in minimum traveling time. The number of hidden layer neurons are varied in a reasonable range to get the best result from the controller. For ease of use, we have assumed a fixed bias to each neuron of the architecture and a tangent hyperbolic function is utilized in the present study. Realizing the fact that it is difficult to develop a neural controller through explicit design, researchers working in this field started thinking whether it can be evolved by using an evolutionary technique. Simultaneous optimization of weights and the architecture of a neural network is addressed in this section. To select proper magnitudes of the constant of activation functions and to optimize the weights of the network, we need to deal with a few continuous variables, whereas tuning of the architecture involves the problem dealing with discrete variables. Thus, the present problem can be treated as

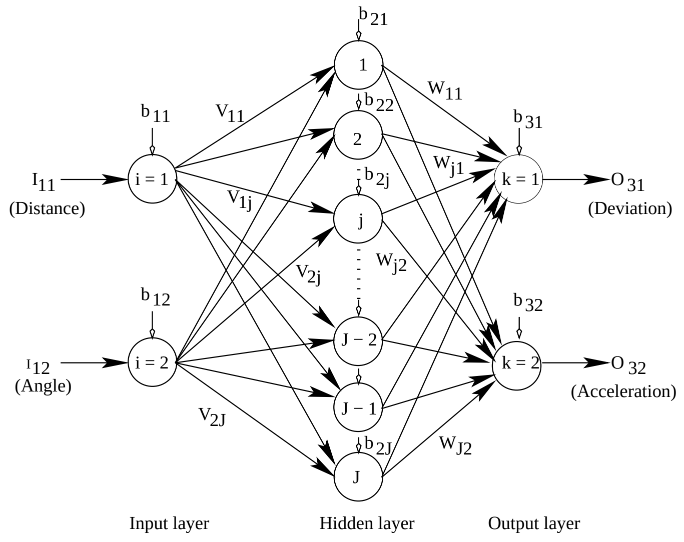

Figure 1: A schematic diagram of the neural network structure.

a mixed-integer optimization problem, involving both the integer as well as real variables. A binary-coded GA with 850-bits long string is used for this purpose. The first 30 bits will carry information of three continuous variables (10 bits for each variable), representing the constants of hyperbolic functions at three different layers. Out of the remaining 820 bits, every 41 bits (starting from 31st bit location of 850-bits long string) are used to indicate the existence of a hidden neuron (1 for presence and 0 for absence) and its corresponding four synaptic weights, (10 bits for each weight). Therefore, a GA string will look as follows (in which 41-bits have been shown to indicate the presence of j^{th} neuron and its connecting weights, such as $v_{1j}, v_{2j}, w_{j1}, w_{j2}$):

$$\underbrace{1\cdots 1}_{C_1}\ \underbrace{0\cdots 1}_{C_2}\ \underbrace{1\cdots 0}_{C_3}\ \underbrace{\cdots\cdots \quad \underbrace{1}_{j^{\text{th}}\ hidden\ neuron}\quad \underbrace{1\cdots 1}_{v_{1j}}\ \underbrace{0\cdots 1}_{v_{2j}}\ \underbrace{1\cdots 0}_{w_{j1}}\ \underbrace{0\cdots 0}_{w_{j2}}\cdots\cdots}_{Architecture\ of\ NN}$$

It is important to mention that we have restricted our search up to a maximum of twenty neurons lying in the hidden layer. During optimization, the constants of activation function for three layers are varied in a range of (0.1 to 15.0) and the weights are allowed to vary from 0.0 to 1.0. The ranges of variation of different variables are selected after a careful study.

The GA begins its search by randomly creating a number of solutions (equals to the population size) represented by the binary strings and each string indicates a typical neural network-based controller. A particular NN-controller differs from other, in terms of the number of hidden neurons, connecting weights and constants of activation function at different layers. Each solution in the population is then evaluated, to assign a fitness value. After the fitness is assigned to each solution in the population, they are modified by using three operators – reproduction, uniform crossover and bit-wise mutation. One iteration involving these three operators followed by the fitness evaluation, is called a generation. Generations proceed until a termination criterion is satisfied. In this approach, the GA is allowed to run for a pre-specified number of generations. The fitness of a GA-string (say, n) is evaluated using the expression mentioned below.

$$Fitness = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{S_m}\sum_{s=1}^{S_m}\sum_{k=1}^{2}\left|T_{3k}^{ms} - O_{3k}^{ms}(n)\right|, \tag{2}$$

It is important to mention that the absolute value of error is taken for the fitness determination. Moreover, if the output of the controller in the predicted distance step is such, that the robot may collide with the most critical obstacle during its movement from the present position to the predicted position, a fixed penalty equal to 200 (selected at random) is added to the fitness. Again, sometimes the robot's motion as suggested by NN-based controller, is not possible to implement due to its kinematic and/or dynamic constraints. In such a situation, the robot is stopped for that particular time step and a fixed penalty equal to 2000 (selected at random) is added to the fitness, to avoid such incidences. As there are two inputs of the NN-architecture and there is some coupling effect among them, topology of the NN having less than three hidden neurons may affect the generalizing capability of it. To avoid such a situation, another fixed penalty (equal to 2000, which is selected at random) is added to the fitness of the GA.

3.2 Approach 2: potential field method as proposed in [11]

According to this approach, any robot moves due to the combined action of attractive potential generated by its goal and repulsive potential created by opponent robots moving in the same environment. Different potential functions have been proposed in the literature. The most commonly used attractive potential takes the form.

$$U_{att}(q) = \frac{1}{2}\xi\rho^2(q, q_{goal}) \tag{3}$$

where ξ is a positive scaling factor, $\rho(q, q_{goal})$ is the distance between the robot q and the goal q_{goal}.

There are several drawbacks associated with the traditional PFM. Mostly it is due to the potential function. Therefore, Ge and Cui have proposed a new repulsive potential function [11] as mentioned below.

$$U_{rep}(q) = \begin{cases} U_{rep}, & if\rho(q, q_{obs}) \le \rho_0 \\ 0 & if\rho(q, q_{obs}) > \rho_0 \end{cases} \tag{4}$$

where $U_{rep} = \frac{1}{2}\eta\left(\frac{1}{\rho(q,q_{obs})} - \frac{1}{\rho_0}\right)^2 \rho^2(q, q_{goal})$ and all other terms have been defined earlier. Also, η is a positive scaling factor, $\rho(q, q_{obs})$ denotes the minimal distance from the robot q to the obstacle, q_{obs} denotes the point on the obstacle and ρ_0 is a positive constant denoting the distance of influence of the obstacle.

The robot is then allowed to move due to the combined action of attractive and repulsive forces derived by differentiating the corresponding potentials. Total force is computed by summing the above two force vectors. In the present study, *acceleration* of the robot is made proportional to the resultant force and future direction of movement is made along the direction of the resultant forces.

3.3 Proposed coordination strategy

In the present study, two different strategies have been used to resolve the conflicting situations, i.e., when two robots in a time step mutually treat each other to be critical.
Strategy 1: Both the robot will adopt zero coordination strategy, i.e., they will move along the direction and with acceleration as planned by the motion planner.
Strategy 2: Here one robot will adopt the zero coordination and the other one will adopt partial coordination, i.e., it will move with the acceleration as suggested by the motion planner and will altercate its future direction of movement using the collision avoidance scheme as suggested by Hui et al. [12]. The robot whose planned deviation is less will be allowed to follow zero coordination and the other will follow the partial coordination.

4 Results and Discussions

The performances of the developed approaches have been compared among themselves for solving navigation problems of multiple robots. Eight robots are moving in a grid of $19.95 \times 19.95m^2$ in a 2-D space. Each of these robots will start from its starting point and reaches its destination without colliding each other. The minimum distance between the planning robot and its most critical neighbor (d_{min}) is kept fixed to $3.2m$, to avoid collision between them. The time interval (ΔT) is taken to be equal to sixteen seconds and robot is assumed to have a maximum and minimum acceleration of 0.05m/s2 and 0.005m/s2, respectively. In the developed soft computing-based approach, the NN is trained off-line, with the help of a GA, as explained earlier. For tuning of the NN, a set of 200 training data is created at random, in which initial position, final position and direction of movement of the robots have been varied. With all such randomly-generated training data, the robots start moving towards their goal. It is to be noted that a batch mode of training has been adopted in the present study. As the performance of a GA depends on its parameter setting, experiments are carried out with different sets of parameters to find the most suitable set. In these experiments, only one parameter is varied at a time, keeping the other parameters fixed and the fitness values are recorded. The best results are obtained with the following GA-parameters: Uniform Crossover probability (p_c) = 0.5, mutation probability (p_m) = 0.003, population size ($popsize$) = 120, maximum number of generation ($Maxgen$) = 170.

After the tuning of the NN is over, their performances have been compared among the Approaches 1 and 2, in terms of traveling time and their CPU times, for a set of five randomly-generated test scenarios. The traveling time taken by the robot by following both the approaches, while moving among eight robots is shown in Table 1 using Strategy II. The results of both the approaches are found to be similar. However, time taken by the robots using Approach 1 is found to be less in most of the cases. It may be due to the absence of any in-built optimization module in potential field method.

For a particular scenario (say third of Table 1) involving eight cooperating robots, the paths planned by all the robots are shown in Figure 2 as obtained using Strategy-II.

Table 1: Traveling time taken by different robots (seconds).

Robot	*Scenario 1*		*Scenario 2*		*Scenario 3*		*Scenario 4*	
No.	*App. 1*	*App. 2*	*App. 1*	*App. 2*	*App. 1*	*App. 2*	*App. 1*	*App. 2*
1	75.69	75.69	29.91	48.56	60.44	67.57	79.45	79.45
2	30.34	30.34	51.96	51.96	35.77	35.77	33.19	33.19
3	59.33	64.63	52.92	52.92	47.42	47.42	88.81	88.81
4	57.69	57.69	49.03	64.49	45.43	45.43	26.52	29.37
5	64.11	64.11	91.63	97.8	60.16	60.16	77.25	74.13
6	70.59	63.3	111.38	111.38	58.72	76.62	81.25	97.61
7	84.8	99.61	91.18	104.25	70.21	71.54	127.64	141.32
8	24.23	24.23	56.76	56.76	81.53	79.4	62.69	76.19

During simulations, Approach 1 is found to outperform the other and Strategy-II has solved the conflict in a more better way than Strategy-I. Therefore, NN-based motion planner along with the proposed strategy might be useful in designing autonomous and intelligent multi-agent system. To understand the feasibility in implementations of the developed approaches on real experiments, CPU times have been compared. This test was carried out in a P-IV PC. The code is written with the help of a C-programming language and compiled in Linux environment (Fedora 10). CPU times of the developed computer program was tested through time command. CPU times of the Approaches 1 and 2 are found to be equal to 0.026 and 0.011 seconds, respectively. This clearly indicates that both the approaches are suitable for real experiments.

5 Concluding Remarks

Motion planning problem of multiple robots is solved, in the present work. Two different approaches have been developed for this purpose. In Approach 1, a GA-tuned NN has been considered. On the other hand, a modified potential field method as proposed by Ge and Cui [11] has been used. The effectiveness of both the approaches are tested through computer simulations for five different scenarios (randomly generated and are different from the training scenarios). Approach 1 is found to outperform for most of the scenarios in compared to the other. CPU times of both the approaches are found to be low, thus making them suitable for on-line implementations. Therefore, both NN as well as PFM-based motion planner might be useful in designing a controller for each agent of a multi-agent system.

All the robots here are assumed to have the same velocity profile. But, in practice, they may have different velocity profiles. The velocities of the robots can be determined using soft computing techniques also. Moreover, adaptability, robustness, cooperativeness and communicativeness of the developed motion planners might be tested through some mathematical formulations. Presently the author is working with these issues.

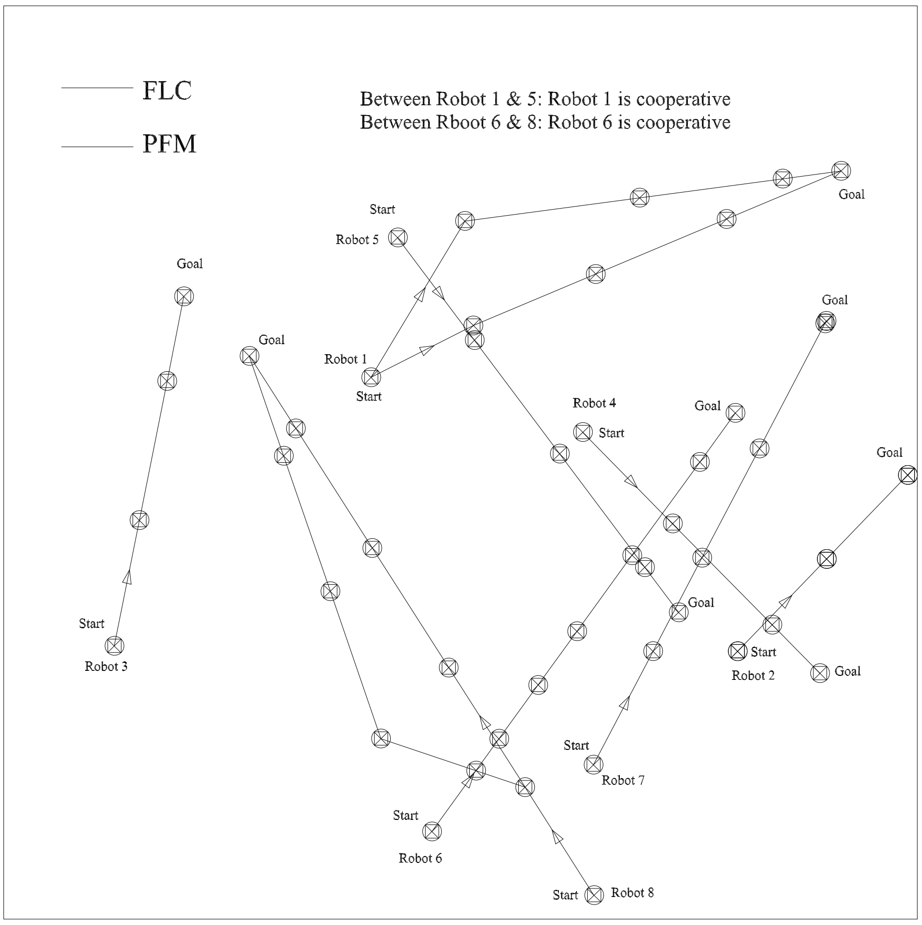

Figure 2: Movements of eight different robots in 2D space.

Acknowledgment

This work is supported by the Department of Science and Technology, Govt. of India, New Delhi (Sanction No. SR/FT/ET-018/2009 dt. 19.06.2009).

References

[1] J.-H. Kim, D.-H. Kim, Y.-J. Kim, and K.-T. Seow, *Soccer Robotics*. Berlin, Germany: Springer-Verlag, 2004.

[2] M. Asada, H. Kitano, I. Noda, and M. Veloso, "Robocup: today and tomorrow–what we have and learned," *Artificial Intelligence*, vol. 110, pp. 193–214, 1999.

[3] A. Mackworth, "On seeing robots," *computer vision systems, theory and applications*, pp. 1–13, 1993.

[4] J. Peng and S. Akella, "Coordinating the motion of multiple robots with kinodynamic constraints," in *Proc. of IEEE Intl. Conference on Robotics and Automation*, (Taipei, Taiwan), pp. 4066–4073, 2003.

[5] T. Nakamura, "Development of self-learning vision-based mobile robots for acquiring soccer robot behaviors," *Lecture Notes in Computer Science*, vol. 1395/1998, pp. 257–276, 2006.

[6] P. Sevstka and M. H. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic road maps," in *Proc. of IEEE Intl. Conference on Robotics and Automation*, pp. 1631–1636, 1995.

[7] D. Vail and M. Veloso, "Multi-robot dynamic role assignment and coordination through shared potential fields," *Multi-Robot Systems*, pp. 87–98, 2003.

[8] F. Y. Wang and B. Pu, "Planning time-optimal trajectory for coordinated robot arms," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, vol. 1, pp. 245–250, 1993.

[9] J. H. Kim and S. G. Kong, "Fuzzy systems for group intelligence of autonomous mobile robots," in *Proc. of IEEE Intl. Fuzzy Systems Conference*, (Seoul, Korea), 1999.

[10] Y. Yang, M. M. Polycarpou, and A. A. Minai, "Opportunistically cooperative neural learning in mobile agents," in *Proc. of IEEE Intl. Conference on Neural Networks*, vol. 3, pp. 2638–2643, 2002.

[11] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[12] N. B. Hui, V. Mahendar, and D. K. Pratihar, "Time-optimal, collision-free navigation of a car-like mobile robot using neuro-fuzzy approaches," *Fuzzy Sets and Systems*, vol. 157, no. 16, pp. 2171–2204, 2006.