Proceedings of the
34th International Conference on Information Technologies (InfoTech-2020)
IEEE Conference, Rec. # 49733, 17-18 September 2020, Bulgaria

DEEP NEURAL NETWORKS IN COLLECTIVE ROBOTICS CONTROL

Vanya Dimitrova Markova and Ventseslav Kirilov Shopov

Institute of Robotics - Bulgarian Academy of Sciences
e-mail: markovavanya@yahoo.com
Bulgaria

Abstract: Mobile agent formation control is an important issue for the collective of robots . Especially when they move independently without human oversight, controlling the movement and forming a collective of agents is a critical and challenging task. In this work, we propose a method for applying the training algorithm to assist the formation of groups of agents in a leader scenario. To exercise control through supportive learning, we present the problem of formation as a Markov decision-making process. This allows us to use deep reinforcement learning to obtain the law of leadership of the successor leader. The feasibility and effectiveness of this control approach is tested in simulation.
Key words: agent formation control; collective of robots; deep reinforcement training.

1. INTRODUCTION

In recent years we have witnessed considerable interest in the research on the coordination of many robotic systems. This is largely due to many practical applications such as urban search and rescue [1], surveillance and intelligence [2], environmental monitoring [3], and mapping of unknown environments [4]. Among the various coordination tasks of many robotic systems, the problem of control of formation is always a major problem whose solution can dramatically increase the efficiency of coordination. For example, in the practice of exploring an unknown area, the well-organized formation may have wider coverage and reduce the possibility of reducing search time.

Generally, the problem with the formation of many robotic systems involves two steps: first determining the desired formation, which is beyond the scope of this study and on the second hand designing the appropriate control algorithm to reach and maintain this formation. Recently, most results with respect to the formation problem have focused on the latter problem. In addition to designing specific controllers, the connection between the robots in the team affects the controller's performance. Most

of the aforementioned formation strategies are based on traditional control methods that can hardly handle the uncertainty in the dynamics of the robot. Intelligent management approaches such as neural networks and fuzzy logic are well suited to this case. In this study, the classic control methods were replaced by the neural network, and the linear and angular velocities of the robots were also evaluated by the neural network. However, the first step of the formation problem, how to determine the desired formation, is not well understood. There are usually two ways to describe a formation. One is to use the relative distance s and angles between robots.

The other is to use shape theory, which can provide a much shorter description of formation. The theory of face figures is interested in the essential geometric connection between robots. While the two graphs are similar, they have the same shape. For example, through the theory of shapes, we can use the term equilateral triangle to describe a formation, but the orientation and size of that equilateral triangle are free to choose. Form theory can, therefore, provide a set of executable forms with the same form. How to choose a formation from this set is the part of the kernel to determine the desired formation.

To the best of our knowledge, most papers related to formation problems pay little attention to this. However, some researchers are still doing some seminar work [5, 6]. In [6] finding the desired / optimal formation becomes a standard optimization problem by the theory of form y and a numerical algorithm is proposed to solve the optimization problem. For conventional digital algorithms, the calculation time required to obtain the solution usually depends on the size of the problem [7, 8]. In other words, the efficiency of numerical algorithms decreases as the size of the problem increases. It becomes a tight spot when the size of the problem is huge.

Recently, this critical problem has been alleviated to some extent by parallel numerical algorithms, especially those that can be applied to graphics processing units [9, 10]. However, the time to calculate parallel numerical algorithms still has a positive correlation with the size of the problem. Therefore, a narrow place still exists. Also, it is noted that in [6] each robot corresponds to one specific position in the formation, but this is unnecessary if the robots are functionally identical, since in this case the positions of the robots are interchangeable. To address this difficulty, we extend the optimization problem in [6] to the combined optimization problem in this document. It turns out that the optimization problem in [6] is a special case of the proposed problem of combined optimization. Unfortunately, the extended optimization problem is a difficult NP problem, which is much more difficult to solve using traditional numerical methods.

2. METHODS AND MATERIALS

2.1. Reinforcement Learning

In general, the reinforcement learning process used is a cycle that begins with the observation of a given state st, selects an action at, waits for the action to complete, observes the received state s t + 1, and accordingly updates the evaluation of its value function. and the next cycle begins. Since updating the value estimate of a

feature is the most often expected intensive step, we have introduced the idea of an update buffer. It caches the observations of s t + 1 and the reward and delays the evaluation of the value function until the next time the robot waits for the action of its choice. This, of course, has some implications as it introduces an aspect similar to offline learning in the online learning process and also requires an additional calculation cycle after reaching a terminal state to properly evaluate the value of the function. It should be noted that these changes do not change the applicability of the algorithm, but simply have to be taken into account. Speeding up the processing also greatly improves the effectiveness of the training as there is no blind time in which the leader continues to distance himself from the follower.

2.2. Recurrent Neural Networks for Optimization Problems

The key moment for building an optimal formation is to effectively solve the optimization problem. Due to the rapid development of intelligent techniques, scientists in the computing intelligence community find that repetitive neural networks have shown great potential as a powerful weapon for solving optimization problems due to their parallel computational nature. Unlike conventional numerical methods, the efficiency of a repetitive neural network does not decrease as the size of the optimization problem increases [7, 8, 11]. In addition, the degree of convergence of the neural network can be arbitrarily increased by properly adjusting the parameters in the neural network. This article is dedicated to solving the problem of optimal formation from a repetitive neural network.

One common assumption of the aforementioned neural networks is that the optimization problem is smooth. However, the optimal formation problem discussed in this document may not be transformed into a smooth optimization problem. This is because if the desired formation is the one that has the minimum distance from the original formation of the multi-robot system, the Euclidean norm of the decision variables is included in the objective function, which leads to the problem of non-smooth optimization. Fortunately, there are several recurring neural network models in the literature on problems with smooth optimization. In [13], a generalized neural network based on the model proposed in [12] was presented for non-smooth problems with nonlinear programming. Extensions of the non-smooth convex optimization problem are made in [14].

3. EXPERIMENTS AND RESULTS

This section describes the results of the aforementioned experiments in the simulation scenario as well as on the real robot. The main objective of using a training algorithm to control leaders followers was achieved.

We have a group of robots moving together in formation. The formation is led by a leader, with the rest of the robots orienting themselves to their position in the leader formation. The problem of the leader's dismissal is solved by taking his place from another robot. All robots have an arrangement to take the leader position.

We have N robots and N positions that the robots have to position themselves. When we have heterogeneous robots, (ie, each robot has a specific position), the solution to the problem is trivial.

When there is a difference between the reference position of the robot and the current position, the robot forms an error vector. This vector actually points from the current position of the robot to the reference position. The vector coordinates as well as the rotation angle are transmitted to the controller of each robot.

In the case we are looking at, the robots in the group in which they work do not have predetermined positions. We are therefore tasked to iteratively determine how robots occupy the positions closest to them, without violating the condition of competition between those who do not have a particular position.

When we have several robots that are grouped around one position, a collision will occur - the first robot will occupy this position, and the others will have to look for the new closest free position in the formation.

For a larger number of robots in certain formation configurations, the last occupied position can be expected to be occupied much later than others. To avoid this situation, the robots should distribute their positions so that the total path to occupying these positions is minimal.

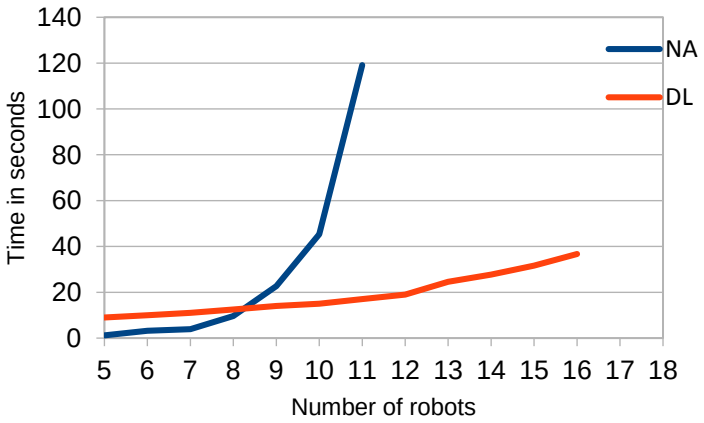

Fig 1: comparison of Naive Approach and the Deep Learning.

At the input of the neural network we feed the vector q with the starting positions of the robots and the vector s with the positions in the formation. At the output of the neural network, we will receive a vector of correspondences of the type [q_i -> s_j]. In this way there is a correspondence between which robot should stand in which position in the formation. Completing the training sample for the neural network is done by the method of the naive approach.

One possible solution is the naive approach: we create a combinatorial ordinance of all possible distributions, we calculate the total path for each distribution. Then we find the distribution with a minimum total distance. For large formations of robots, this may be unacceptable because of the combinatorial explosion. We suggest using a neural network to optimize this process.

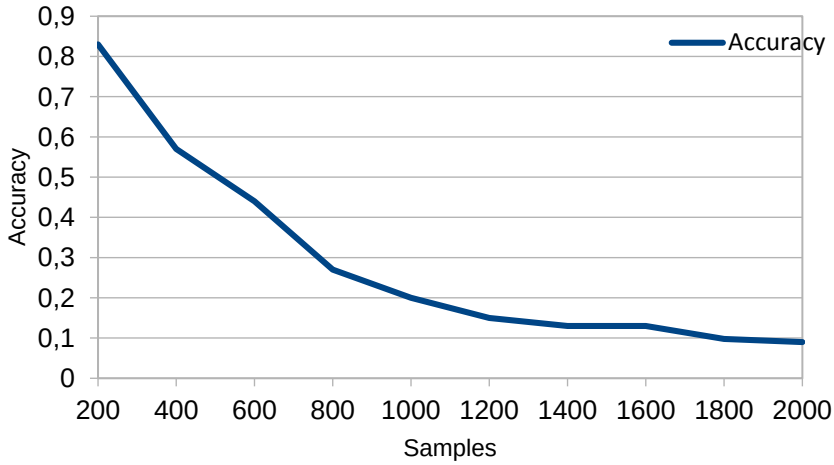

Fig 2 Examines how the volume of the training sample influences the accuracy of prediction.

4. CONCLUSION

Mobile agent formation control is an important issue for the collective of robots. Especially when they move independently without human oversight, controlling the movement and forming a collective of agents is a critical and challenging task. In this work, we propose a method for applying the training algorithm to assist the formation of groups of agents in a leader scenario. To exercise control through supportive learning, we present the problem of formation as a Markov decision-making process. This allows us to use deep reinforcement learning to obtain the law of leadership of the successor leader. The feasibility and effectiveness of this control approach is tested in simulation.

AKNOWLEDGEMENTS

This work was supported by the European Regional Development Fund within the OP “Science and Education for Smart Growth 2014 - 2020”, Project CoC “Smart Mechatronic, Eco- And Energy Saving Systems And Technologies“, № BG05M2OP001-1.002-0023

REFERENCES

[1] Casper, J., & Murphy, R. R. (**2003)**. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *33*(3), pp. 367-385.

[2] Saptharishi, M., Oliver, C. S., Diehl, C. P., Bhat, K. S., Dolan, J. M., Trebi-Ollennu, A., & Khosla, P. K. **(2002)**. Distributed surveillance and reconnaissance using multiple autonomous ATVs: CyberScout. *IEEE Transactions on Robotics and Automation*, *18*(5), pp. 826-836.

[3] Paley, D. A., Zhang, F., & Leonard, N. E. **(2008)**. Cooperative control for ocean sampling: The glider coordinated control system. *IEEE Transactions on Control Systems Technology*, *16*(4), pp. 735-744.

[4] Zhu, A., & Yang, S. X. **(2007)**. Neurofuzzy-based approach to mobile robot navigation in unknown environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *37*(4), pp. 610-621.

[5] Spletzer, J. R., & Fierro, R. **(2005, April)**. Optimal positioning strategies for shape changes in robot teams. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* pp. 742-747.

[6] Derenick, J. C., & Spletzer, J. R. **(2007).** Convex optimization strategies for coordinating large-scale robot formations. *IEEE Transactions on Robotics*, *23*(6), pp. 1252-1259.

[7] Xia, Y., & Wang, J. **(1998).** A general methodology for designing globally convergent optimization neural networks. *IEEE Transactions on Neural Networks*, *9*(6), pp. 1331-1343.

[8] Kamel, M. S., & Xia, Y. **(2009).** Cooperative recurrent modular neural networks for constrained optimization: a survey of models and applications. *Cognitive neurodynamics*, *3*(1), pp. 47-81.

[9] Fei, Y., Rong, G., Wang, B., & Wang, W. **(2014)**. Parallel L-BFGS-B algorithm on gpu. *Computers & graphics*, VOL. 40, pp. 1-9.

[10] Ploskas, N., & Samaras, N. **(2015).** Efficient GPU-based implementations of simplex type algorithms. *Applied Mathematics and Computation*, *250*, pp. 552-570.

[11] Wang, J., & Kusiak, A. **(2000).** Neural network applications in intelligent manufacturing: An updated survey. In *Computational intelligence in manufacturing handbook* pp. 45-72.

[12] Kennedy, M. P., & Chua, L. O. **(1988).** Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems*, *35*(5), pp. 554-562.

[13] Forti, M., Nistri, P., & Quincampoix, M. **(2004)**. Generalized neural network for nonsmooth nonlinear programming problems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *51*(9), pp. 1741-1754.

[14] Xue, X., & Bian, W. **(2008).** Subgradient-based neural networks for nonsmooth convex optimization problems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *55*(8), pp. 2378-2391.