A Quadratic Programming Formulation of a Moving Ball Interception and Shooting Behaviour, and its Application to Neural Network Control

Frederic Maire and Doug Taylor

Queensland University of Technology,
School of Computing Science, Faculty of Information Technology,
2 George Street, GPO Box 2434
Brisbane Q 4001, Australia
{f.maire,di.taylor}@qut.edu.au,
WWW home page: http://www.fit.qut.edu.au/{~maire,~taylord}

Abstract. A desirable elementary behaviour for a robot soccer player is the *moving ball interception and shooting behaviour*, but generating smooth, fast motion for a mobile robot in a changing environment is a difficult problem. We address this problem by formulating the specifications of this behaviour as a quadratic programming optimisation problem, and by training a neural network controller on the exact solution computed off-line by a quadratic programming problem optimiser. We present experimental results showing the validity of the approach and discuss potential applications of this approach in the context of reinforcement learning.

1 Introduction

Robotic soccer is a challenging research domain which involves teams of agents that need to collaborate in an adversarial environment. The fast paced nature of robotic soccer necessitates real time sensing coupled with quick behaving and decision-making, and makes robotic soccer an excellent test-bed for innovative and novel techniques in robot control. For example, the behaviours and decision making processes can range from the most simple reactive behaviours, such as moving directly towards the ball, to arbitrarily complex reasoning procedures that take into account the actions and perceived strategies of teammates and opponents. In order to be able to successfully collaborate, agents require robust basic skills. These skills include the ability to go to a given place on the field, the ability to avoid obstacles and the ability to direct the ball.

This paper concentrates on a neural network approach for the design of a robust navigation system. We show how to compute off-line an optimal trajectory using quadratic programming and how to train a neural network in order to generate a fundamental elementary behaviour for a soccer player; the *moving ball interception and shooting behaviour*. The objective of this elementary reactive

P. Stone, T. Balch, and G. Kraetzschmar (Eds.): RoboCup 2000, LNAI 2019, pp. 327-332, 2001.

behaviour is to get the robot to intercept a moving ball and kick it in the direction of the goal at a certain time and using as little energy as possible.

1.1 Paper Outline

The paper is organised as follows. Section 2 explains the behaviour control approach and describes our system. The experimental results are presented in section 3. Section 4 puts this work in the perspective of reinforcement learning.

2 Behaviour Control

2.1 Moving Ball Interception and Shooting Behaviour

Most of the successful robot soccer teams have adopted a bottom up hierarchical approach to agent behaviours [2–5]. Some of them (see chapter 5 of [2] for a review) have chosen neural networks to implement interception and shooting behaviours, and train their control networks by reinforcement learning (whereas our off-line computation of optimal command sequences allows us to use a supervised training method).

If a robot is to accurately direct the ball towards a target position, it must be able to approach the ball from a specified direction. The task of the *moving ball interception and shooting behaviour* module is to do this economically, with the temporal constraint that the trip to the ball should take n time-steps.

The input to this behaviour module is a state vector containing the following information. The position and velocity vectors of the ball and the player, and the number of time steps (denoted by n) before the desired interception. The output of the module is an acceleration vector for the next time step. The goal is assumed to be at position $(0, 0)$. Note that the goal in this context is simply a location, that can correspond either to the structure into which the ball is driven to score, or to a location next to a team-mate to whom the ball should be passed. The assumption that the goal is at position $(0, 0)$ is not really a restriction as the general case reduces to that particular case by considering the relative positions with respect to the goal position.

In simulation, we have to make sure that the velocity and acceleration vectors of the player stay bounded. In our mathematical formulation, inequality constraints on the norms of the acceleration and velocity vectors are used to guarantee a realistic behaviour. By minimising the sum of the norms of the robot acceleration vectors, we obtain the most economical (energy-efficient) sequence of acceleration vectors to complete the interception and shooting task. An added benefit of this approach is that the robot trajectory is very smooth.

2.2 Mathematical modelling

The *state* of the playing field at time t is determined by the position, velocity and acceleration vectors of the robot and the ball at time t. We will use the following

convention for these variables; the subscript R refers to a robot variable, the subscript B refers to a ball variable, time is indicated as a superscript. According to Newtonian mechanics, the above vectors are related by $P_R^t = P_R^{t-1} + V_R^{t-1}$ and $V_R^t = V_R^{t-1} + A_R^{t-1}$. From these relations, it is easy to derive that $V_R^i = V_R^0 + \sum_{j=0}^{i-1} A_R^j$ and $P_R^t = P_R^0 + t \times V_R^0 + \sum_{i=0}^{t-2} (t-1-i) \times A_R^i$. These equations show that the position and velocity vectors depend linearly on the acceleration vector. We are looking for a sequence of bounded acceleration vectors $A_R^0, A_R^1, \ldots, A_R^{n-1}$ such that the corresponding sequence of velocity vectors is also bounded by some constants $V_{\max}$ and $A_{\max}$. We are using the norm sup in order to have to deal only with linear constraints. The most economical sequence of accelerations is the solution to the following quadratic programming minimisation problem;

$$\min \sum_{i=0}^{n-1} \left\| A_R^i \right\|_2^2 \tag{1}$$

$$\text{s.t.} \begin{cases} \forall i, \ \left\| A_R^i \right\|_\infty \leq A_{\max} \\ \forall i, \ \left\| V_R^i \right\|_\infty \leq V_{\max} \\ V_R^n = \frac{V_{\max}}{\left\| G - P_B^n \right\|_2} (G - P_B^n) \\ P_R^n = P_B^n \end{cases} \tag{2}$$

The constraint $P_R^n = P_B^n$ expresses the requirement that the robot hit the ball at time $t = n$. The constraint $V_R^n = \frac{V_{\max}}{\left\| G - P_B^n \right\|_2} (G - P_B^n)$ expresses the requirement that the velocity vector of the robot at time $t = n$ be in alignment with the goal. The system that we have derived is easily solved with standard mathematical packages like Matlab optimisation toolbox (see [6] for an introduction to optimisation theory).

2.3 The Time Oracle

Given an initial state, we are faced with the problem of choosing a suitable value for n. To make our system as general as possible, we made the following design decision. An auxiliary neural network, that we call *the time oracle*, is trained to learn $t_{\min}$ the minimum value of n such that there exists a feasible solution to the system of inequalities (2). The input of the time oracle is the same state vector as for the control neural network without the entry n. The control neural network is fed with the n computed by the time oracle neural network. The time oracle acts as a critic who predicts how long it will take the robot to complete the interception task. In fact, the oracle learn $1/t_{\min}$, so that it outputs $0 = 1/\infty$ when the robot will fail to intercept the ball. The oracle is not superfluous, as the robot should not waste its energy trying to complete tasks beyond its physical capabilities.

3 Experimental Results

The architecture of the control NN was determined by trial and error. The 2-hidden layer networks seem to perform better than the 1-hidden layer networks. The control network used to produce the figures below had 30 neurons in each hidden layer (the time oracle was the same size).

In all figures of this paper, the solid line represents the trajectory of the robot. The circle on the solid line represents the initial position of the robot. The dotted line represents the trajectory of the ball. The circle on the dotted line represents the initial position of the ball. The star at position $(0,0)$ represents the (punctual) goal. The sequence of velocity vectors of the robots can be seen on the right hand side subplots of each figure.

In figure 1, the initial velocity of the robot is South, whereas the ball is going North. To arrive at a shooting position, the robot has first to make a U-turn, then overtake the ball, then make a second U-turn and align itself in the direction of the goal. The left-top diagram was obtained using directly the QP optimiser. The left-bottom was obtained with the control NN. In figure 2, the initial velocity of the robot is North-East, whereas the ball is going West. In figure 3, the initial velocity of the robot is South-East, whereas the ball is going North-East.

All these figures correspond to non training data initial conditions. We observe that the QP optimiser and the NN control trajectories are very similar. This demonstrates that the neural network has successfully learnt the mapping from state-space to action-space that produces the optimal trajectories and generalises well.

4 Discussion and Future Work

Although our neural controller works well in simulation, we expect that when used on a real robot, the shooting performance will have room for improvement. We plan to use recently developed reinforcement learning techniques for continuous action spaces [7, 8] to ameliorate the shooting skills (accuracy) of the robot. In reinforcement learning, the outcome of the training of a neural network controller is very sensitive to the initial weights. The agent is more likely to improve its policy if the neural network implementing the policy is initialised with a reasonably good behaviour.

Extending control systems to include learning capabilities is becoming an increasingly more important issue in autonomous agent control. Our proposed system is a step in this direction.

References

1. Brooks, R.: Cambrian Intelligence. MIT Press, 1999. ISBN: 0262024683
2. Stone, P.: Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer. MIT Press, 2000. ISBN: 0262194384

3. Wyeth G.F. , Browning B. and Tews A. UQ RoboRoos: Preliminary Design of a Robot Soccer Team. Lecture Notes in AI: RoboCup '98, 1604, (1999)
4. Robocup 1999 Team Description: Middle Robot League http://www.ida.liu.se/ext/robocup/middle/intro/index.html
5. RoboCup-98: Robot Soccer: World Cup II Lecture Notes in AI: RoboCup '98, 1604, (1999)
6. Fletcher, R.: Practical Methods of Optimisation, 2nd Edition. Wiley, 1987. ISBN 0471915475
7. Maire, F.: Bicephal Reinforcement Learning. QUT FIT Technical Report, FIT-TR-00-01.
8. Gaskett, C., Wettergreen, D., and Zelinsky, A.: Q-Learning in Continuous State and Action Spaces. in Proceedings of 12th Australian Joint Conference on Artificial Intelligence, Springer-Verlag, Sydney, Australia, (1999).

5 Figures

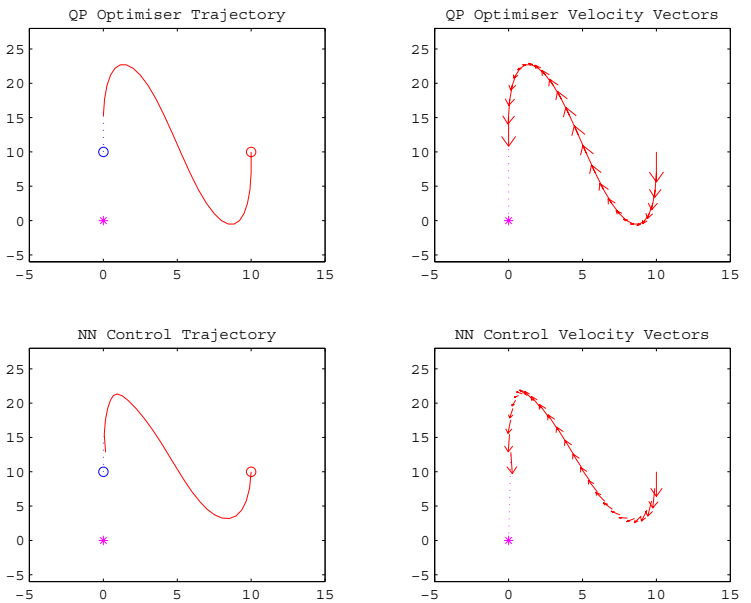

Fig. 1. Initial velocity of the robot is South, initial ball velocity is North

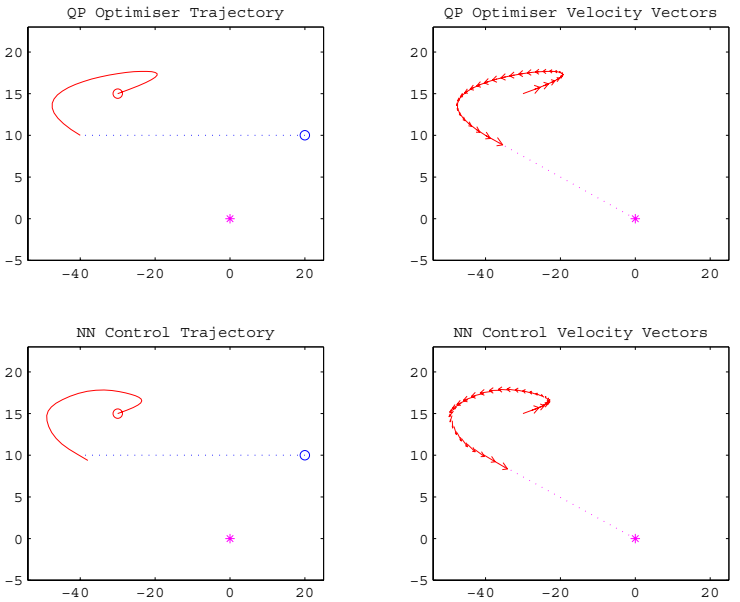

Fig. 2. Initial velocity of the robot is North-East, initial ball velocity is West

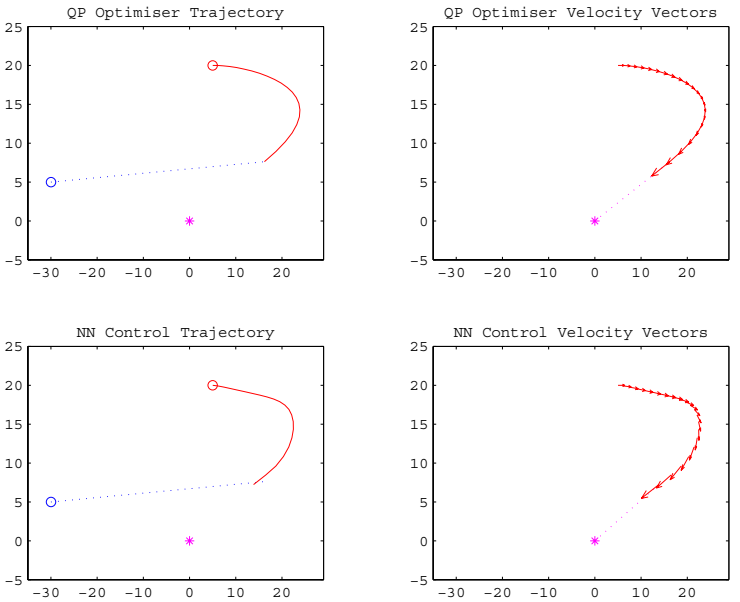

Fig. 3. Initial velocity of the robot is South-East, initial ball velocity is North-East