OxBlue2008(3D) Team Description

Jie Ma and Stephen Cameron

Oxford University Computing Laboratory,
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
{jie.ma@comlab.ox.ac.uk
http://www.comlab.ox.ac.uk

Abstract. OxBlue2008(3D) is a robot football team for RoboCup 3D simulation. In this paper, the decision structure of our team will be presented, and a walking pattern problem that encountered by most 3D teams will be reviewed. We propose a novel method called PSP (Policy Search Planning), and discuss how PSP can be used to solve the walking pattern problem. We regard joints and parts from a single robot as multi-agents; a walking pattern can be defined as a cooperative plan. Policy Search is used to find an optimal policy for selecting different walking patterns from a *plan pool*; it extends an existing gradient-search method (GPOMDP) to a MAS domain.

1 Introduction

OxBlue2008(3D) is a robot football team for RoboCup 3D simulation and it is currently under development at Oxford University. The main purpose of our development of OxBlue2008 is to verify and promote our research on Multi-agent Systems (MAS). In particular, as cooperative skills essentially differentiate MASs from single-agent intelligence, we are interested in applying Machine Learning methods to yield cooperative behaviours in MAS scenarios. In RoboCup 3D simulation, multi-agents are two-fold: multiple robots from two teams and multiple joints and parts from a single robots. Since currently a team only consists of two robots, cooperation amongst agents is obscure; thus we explore the problem of planning walking patterns using a novel MAS method that we proposed called PSP (Policy Search Planning).

In §2, a layered decision architecture that is used in OxBlue2008(3D) is presented. A walking pattern problem that encountered by most teams is be reviewed in §3. A brief introduction of a novel method PSP that we proposed is given in §4 (and we submitted a separated paper that includes the complete details to the RoboCup symposium 2008); and its application in planning walking patterns is discussed in §5. It is followed by the conclusion and future work in §6.

2 Decision Architecture

In order to reduce the learning space of cooperative skills, most of today's MASs tend to adopt *vertical layered architectures* [1, 2]. This architecture is also adopted in OxBlue2008 (both 2D and 3D). In RoboCup soccer, pure reaction is required on some occasions, such as getting up when a robot falls, where the agent doesn't need to make complicated decisions but tries to stand up again. In other words, before world models are fully generated, actions will be directly sent. In more sophisticated decisions, however, such as stopping the ball from losing it, then although world models have been created, emergency actions will be directly sent to the executor without comparing different skills in the arbitrator. This is in-between decisions. In most cases, decisions are pure deliberation — local issues such as walking to the ball can be solved in the individual skill module, while global problems (for the future simulator) including formation and team strategies can be dealt with by advanced methods such as planning. Our decision structure is given in Figure 1.

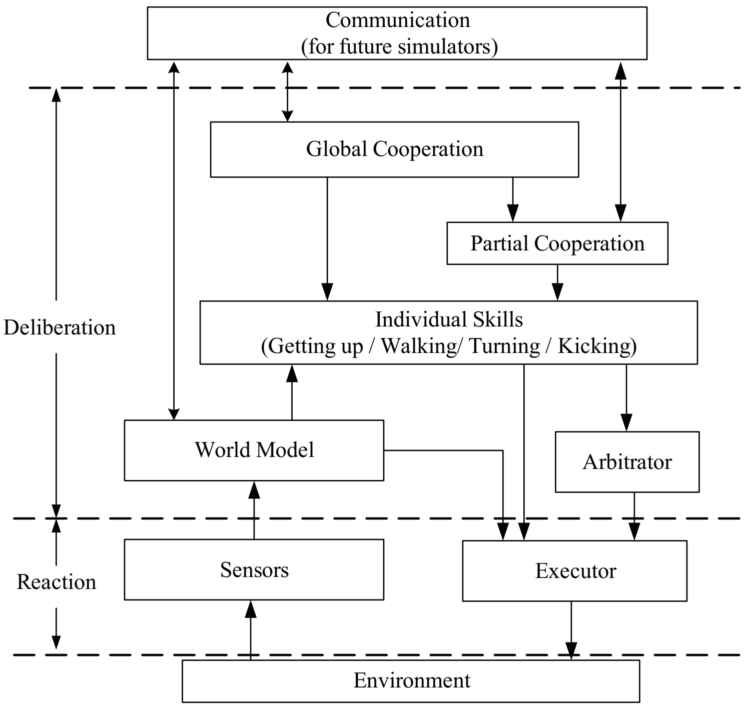

Fig. 1. Decision Architecture of OxBlue2008(3D)

3 Planning Walking Patterns

The robots in 3D simulation are biped; and walking has long been regarded as one of the most challenging problem for a biped robot. In a robot football domain, walking skill is especially indispensable, because it is a significant infrastructure for high-level skills such as running, positioning and dribbling.

Previous studies mainly focused on three methods: recording human kinematic data [3], interaction of gravity and inertia and zero moment point (ZMP). Preceding research suggested that if the contact area is planar, ZMP yields more stable walking behaviours compared with the other two methods. Basically, the ZMP algorithm is to find a set of movements, whose total inertia force equals to 0 at the contact of the foot with the ground.

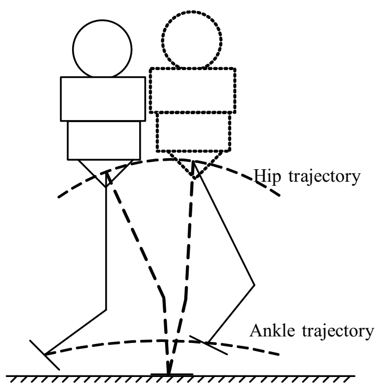

Fig. 2. A Walking Pattern in RoboCup soccer 3D simulation

Since the ground is planar in 3D simulation, we employ ZMP method to generate walking patterns. We use a parameterised walking pattern generator that proposed by Huang [4] to produce a walking pattern. In this method, a walking pattern is determined by the trajectories of the hip and the ankle. An example of these two trajectories in RoboCup 3D simulation is given in Figure 2.

A walking pattern is essentially controlled by a gait phase tuple $\langle T_c, T_d \rangle$, which controls the time of a whole step and the time of the moving leg in the air; a height tuple $\langle H_{h\min}, H_{h\max}, H_{ao} \rangle$, which defines the height limitation of the hip and the ankles; a gait length parameter D_s and two tweak parameter x_{sd} and x_{ed} to adjust the walking pattern. In order to simplify the computation, we assume the feet are always parallel to the field, and the position of the hip and the moving ankle in the step k is given as Equations 1–4.

$$x_a(t) = \begin{cases} kD_s, & t = kT_c \\ kD_s, & t = kT_c + T_d \\ kD_s + L_{ao}, & t = kT_c + T_m \\ (k+2)D_s, & t = (k+1)T_c \\ (k+2)D_s, & t = (k+1)T_c + T_d \end{cases} \tag{1}$$

$$z_a(t) = \begin{cases} h_{gs}(k) + l_{an}, & t = kT_c \\ h_{gs}(k), & t = kT_c + T_d \\ H_{ao}, & t = kT_c + T_m \\ h_{ge}(k), & t = (k+1)T_c \\ h_{ge}(k) + l_{an}, & t = (k+1)T_c + T_d \end{cases} \tag{2}$$

$$x_h(t) = \begin{cases} kD_s + x_{sd}, & t = kT_c \\ (k+1)D_s - x_{sd}, & t = kT_c + T_d \\ (k+1)D_s + x_{ed}, & t = (k+1)T_c \end{cases} \tag{3}$$

$$z_h(t) = \begin{cases} H_{h\,\min}, & t = kT_c + 0.5T_d \\ H_{h\,\max}, & t = kT_c + 0.5(T_c - T_d) \\ H_{h\,\min}, & t = (k+1)T_c + 0.5T_d \end{cases} \tag{4}$$

The positions in the equations are discrete, and the complete trajectory functions can be calculated using third-order spline interpolation, which can arguably produce smooth walking trajectories.

In practical, when we regard joints and parts from a single robot as multi-agents, a walking pattern i can be defined as a parameterised MAS plan $P_i = \langle T_c, T_d, H_{h\,\min}, H_{h\,\max}, H_{ao}, D_s, x_{sd}, x_{ed} \rangle$, and the parameters for a plan can be found by experiments or machine learning methods. However, under the same speed, the optimal walking pattern is still obscure; so we can define multiple plans in a *planpool* and use Policy Search Planning (PSP) to balance speed, stability and energy consumption (for future simulator). The details of PSP have been demonstrated in a separated paper for RoboCup symposium 2008. In this paper, a brief introduction of PSP is given, and how it can be used in planning walking pattern is illustrated.

4 Policy Search Planning (PSP)

In complex MASs, particularly in a system with hybrid individual architectures, planning plays a different role compared with that in traditional domains. In a simplified single-agent system, planning is used to directly find a goal. In dynamic MASs, however, the goal is usually difficult to achieve, or sometimes it is difficult to describe the goal. In addition, the traditional action effects will lose their original meaning: environmental state can also be changed by other agents at the same time, or sometimes it continually varies even without any actions. For example, consider a walking scenario from RoboCup 3D. A traditional planner might construct a plan in which a robot is walking to a point A at the speed of $1.5m/s$ and it takes $\frac{distance}{D_s}$ walking cycles (steps) to accomplish this task. However, if we use a smaller gait length D_s, the same goal can be achieved at the same time. Even from a human's perspective it is difficult to say which plan is better.

We propose a novel method called Policy Search Planning (PSP) for POMDPs, which is essentially a centralised planner for distributed actions. PSP can try to find the most appropriate policy for selecting a plan. Specifically, it can represent a number of walking patterns in the form of plans, and policy search is used to find the optimal policy in choosing these plans. As a plan is not designed to find the goal directly but to define cooperative knowledge, the style of it is not very critical. One possible presentation, a PDDL-like planner, is shown in Figure 3.

Compared with original PDDL, *:goal* will not be included as PSP aims not to achieve it directly, and *:effect* is not needed unless it is used for parameters in policy search. The concept of *stage* is introduced, which makes complex cooperation possible, whereby if and only if the success condition of the current stage is met a planner moves to the next stage; and *role mapping* formulae are introduced to find the most appropriate agents to implement actions

```
(define (PLAN_NAME)
   (:plan_precondition CONDITION_FORMULA)
   ((:agentnumber INTEGER(N))
    (ROLE_MAPING_FORMULA(1))
    (ROLE_MAPING_FORMULA(2))
     ...
    (ROLE_MAPING_FORMULA(N)))
   ((:stagenumber INTEGER(M))
    ((:stage_1_precondition   CONDITION_FORMULA)
     (:stage_1_success        CONDITION_FORMULA)
     (:stage_1_failure        CONDITION_FORMULA)
     (:stage_1_else           CONDITION_FORMULA)
     (:action1                ACTION_FORMULA)
     (:action2                ACTION_FORMULA)...)
    ((:stage_2_precondition   CONDITION_FORMULA)
     ...)   ...
    ((:stage_M_precondition   CONDITION_FORMULA)
     ...))
  [(:effect EFFECT_FORMULA)])
```

Fig. 3. A PDDL-like Plan Structure in PSP

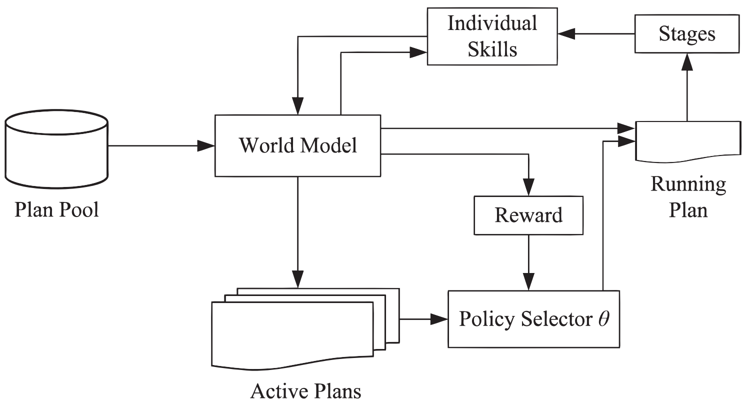

Fig. 4. Learning Process in PSP algorithm

In the PSP algorithm, a plan is actually a cooperative strategy. We can define plenty of offline plans to establish a *plan pool*, which is essentially an expert knowledge database. If the external state satisfies the precondition of a plan, the plan will be called an *active plan*. At time t, if there is only one active plan, it will be marked as the *running plan* and actions will be executed stage by stage. However, along with the growth of the plan pool, multiple active plans may appear at the same time.

Previous solutions [5, 6] chose a plan randomly, which is clearly a decision without intelligence. Q-learning is apparently a wiser approach, but unfortunately Q-learning is difficult to adopt in generalised decision architectures because all the plans cannot guarantee activation.

In this paper we employ another reinforcement learning method, policy search, to overcome this difficulty. The learning framework is illustrated in Figure 4. Due to the space, we are unable to extend the details of PSP method.

5 Application in Planning Walking Patterns

We are currently still working on the training environment in *Simspark*. Recently, Apollo3D team proposed a method to integrate an agent directly into the server, thus we intend to employ this method to undertake learning in the next stage of our research.

Planning walking patterns in RoboCup 3D is a suitable application for PSP algorithm because of the following three reasons. First, in a walking problem, multi-agents are using a pure planning decision structure and the global rewards only come from policy selector. Second, intelligent cooperation amongst joints and parts is urged. Third, how to balance speed, stability and energy consumption in a walking pattern problem is one of the most challenging issues in biped robot domain.

In a plan $P_i = \langle T_c, T_d, H_{h\,\min}, H_{h\,\max}, H_{ao}, D_s, x_{sd}, x_{ed} \rangle$, tweak parameters $\langle x_{sd}, x_{ed} \rangle$ are used to polish a plan, and the optimal tuple can be found when given $\langle T_c, T_d, H_{h\,\min}, H_{h\,\max}, H_{ao}, D_s \rangle$. Essentially, PSP is not used to define parameterised plans, but to find the policy to select the optimal walking pattern amongst a number of walking patterns based on $\langle T_c, T_d, H_{h\,\min}, H_{h\,\max}, H_{ao}, D_s \rangle$. For example, we can calculate how much a walking plan close to ZMP and how much energy a step requires, and PSP can find the optimal policy to balance the stability and energy consumption.

6 Conclusion and Future Work

In §2, a layered decision architecture that is used in OxBlue2008(3D) is presented. A walking pattern problem that encountered by most teams is reviewed and a parameterised gait generater is used in our team.

We proposed a novel method called PSP in a generalised POMDP scenario, in which a large selection of cooperative skills can be presented in a plan pool; and policy search is used to find the optimal policy to select among these plans. We briefly demonstrated why and how PSP can be used in RoboCup 3D Simulation.

PSP is our first attempt to learn the optimal cooperation pattern amongst multiple agents. Our future directions for RoboCup 3D are three-fold. First, we are going to undertake learning in *Simspark*, and explore the most efficient and stable walking pattern at different speeds. Second, we are planning to design an self-adaptive walking pattern selector in a dynamic environment as ODE (Open Dynamics Engine) generates action noises. Third, we will explore the running and dribbling skills in humanoid robot football.

References

[1] Perraju, T.S.: Multi agent architectures for high assurance systems. In: American Control Conference. Volume 5., San Diego, CA, USA (1999) 3154–3157

[2] Stone, P., Veloso, M.: Layered learning and flexible teamwork in robocup simulation agents. In: RoboCup-99: Robot Soccer World Cup III. (2000) 65–72

[3] Ha, S., Han, Y., Hahn, H.: Adaptive gait pattern generation of biped robot based on human's gait pattern analysis. International Journal of Mechanical Systems Science and Engineering **1** (2008)

[4] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., Tanie, K.: Planning walking patterns for a biped robot (2001)

[5] Obst, O.: Using a planner for coordination of multiagent team behavior. Programming Multi-Agent Systems **3862/2006** (2006) 90–100

[6] Obst, O., Boedecker, J.: Flexible coordination of multiagent team behavior using htn planning. In: RoboCup 2005: Robot Soccer World Cup IX. (2006) 521–528