MRL Soccer 3D Team

Mahmood Rahmani, S.Mohammad H.S.Javadi, Ashkan Ferdowsi,
Mahdi N. Moghadasi, Siavash Aslani, , Nima Shafii

Mechatronics Research Laboratory, Qazvin Azad University, Qazvin, Iran
{rahmani_mahmood, erfanjavadi, af_programmer,
mahdinaser, siavash_a_q, wwwnima}@yahoo.com

Abstract. This paper briefly describes an effort to develop a soccer 3D team named MRL. The development concludes three main divisions: noise reduction, low-level skills, planning. Soccer Server applies a bit of a noise to sensory information and sends it to players. The noise decreases the accuracy of some sensitive low-level skills such as ball handling. MRL uses a known method, DKF (Discrete Kalman Filtering) to reduce the noise. Linear regression method was used for agents' self localization. MRL developed a specific planner by collecting ideas from existing planners and adding features to make the new planner continual and distributed.

1 Introduction

This paper briefly describes an effort to develop a soccer 3D team named MRL. The development concludes three main divisions: noise reduction, low-level skills, planning. Soccer is a dynamic environment in which situations are not completely known and actions will not always succeed. Automated planners cannot assume entire information. Execution of plans is not deterministic. MRL started to develop a specific planner that is continual and distributed. Its layered architecture has considered interleaved planning, execution, and monitoring, known as continual planning.

Section 2 describes how noise has been filtered using a common method called DKF, Discrete Kalman Filtering. Two low-level optimizations are described in section 3, self-localization and a method for controlling agent's velocity. Section 4 is about MRL planner. Section 5 concludes the paper and describes future works.

2 Noise Reduction

Soccer agents receive noisy sensory information from Soccer Server. Therefore agents see objects like ball, flags and players with noise. It is obvious that the sense could not be reliable enough and agent always carrying out its tasks inaccurately. So agent has to reduce the noise in order to achieve more reliable and accurate sense. A common method for this purpose is DKF, Discrete Kalman Filtering. The Kalman

filter estimates a process by using a form of feedback control [1]. This method is based on two equations, *state* equation (2.1) and *measurement* equation (2.2):

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \qquad (2.1)$$

$$z_k = Hx_k + v_k \qquad (2.2)$$

Besides there are two update algorithms, *time* update and *measurement* update, which use system's feedback data and update each other recursively. Consequently this method decreases the estimated error covariance to nearly zero.

As an example, suppose the ball is in distance 10 relative to an agent. Fig. 1 depicts what the agent receives (black color) and what it calibrated (magenta color).

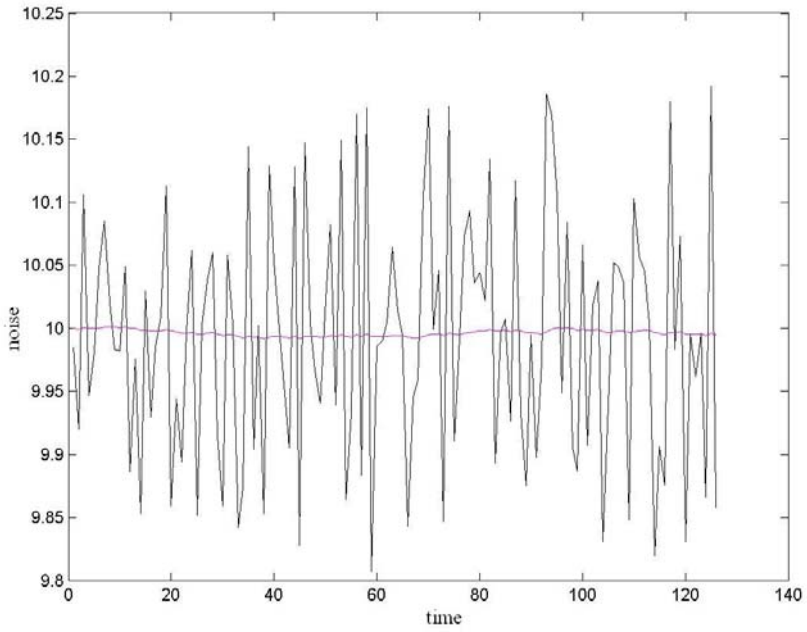

Fig. 1. Noise reduction of ball-distance using DKF, black line is the noisy data received from soccer server and the magenta line is the filtered data.

3 Low-Level Skills

Low-Level Skills are foundations for multi-layer robot control architectures. Without having reliable basic skills robot will not do anything efficiently. If a soccer robot could not estimate position of the ball trustfully, its ball handling skills will fail, and it is clear that a soccer team containing this kind of robots would not be a good team.

Because of the importance of basic computations' accuracy, MRL focused on the information that agent uses to act properly, including *self localization* and *improving agent's velocity control*.

3.1 Self Localization

Localization is the process of determining the robot's location within its environment. In soccer, localization means that agents find their local position in the field of play. MRL agent uses relative information of the eight flags that arranged around the field of play and applies 'Circle Intersections' [2] as basic estimation of its position. After applying linear regression on this estimation, agent localizes itself very well.

3.2 Efficient Velocity Control

One of major problems in agent's low-level skills is controlling agent's velocity. A soccer agent usually needs to go to a target quickly and also reach there with a desired speed. If the desired speed is less than agent's max velocity, agent has to increase and then decrease its speed along the way to the target. On the other hand, when the desired speed is unlimited agent needs only to increase its speed.

MRL agents use a mechanism to calculate the moments in which they should change their speed from ascending to descending or vice versa. The mechanism has two phases: *(I)* off-line learning that agent learns the length of traveled distance using particular velocity and power, and *(II)* on-line calculation that agent explores learned data to decide moments in which its velocity should be changed in order to arrive at a target position with a particular ultimate velocity. In *(I)* in order to reach a target position as fast as possible it is supposed that agent only uses its maximum power to change the velocity. *(I)* generates two look-up tables, one for increasing and another for decreasing velocity. During *(II)* agent explores these two tables to determine moments for changing the velocity.

An illustrative example can clear up this method. Suppose that agent is in start position (Fig. 2.a) with initial velocity V_0. It wants to go to the target position that is in distance d. In addition, agent has decided to have velocity V_t when it arrives at the target. Consider $d = 3.30$, $V_0 = 0$, and the power is 100 (max power). Fig. 2.b to 2.e shows learned data (blue curves) as well as agent's experiment data (red curves). Fig. 2.b indicates that if V_t is set to 1.6 (1.6 is max velocity of the soccer) agent should increase its velocity all the time. Fig 2.c shows that if $V_t = 1.4$ (a little less than max velocity) agent has to raise its velocity rapidly to 1.4 (and not 1.6) and continue keeping this speed. Fig. 2.d (or 1.e) denotes that agent wants to get to $V_t = 1$ (or 0.2) so it should raise its velocity then continue without any acceleration and finally reduce the velocity. This method guarantee a reliable velocity control that makes agent get to a desire position as fast as possible.

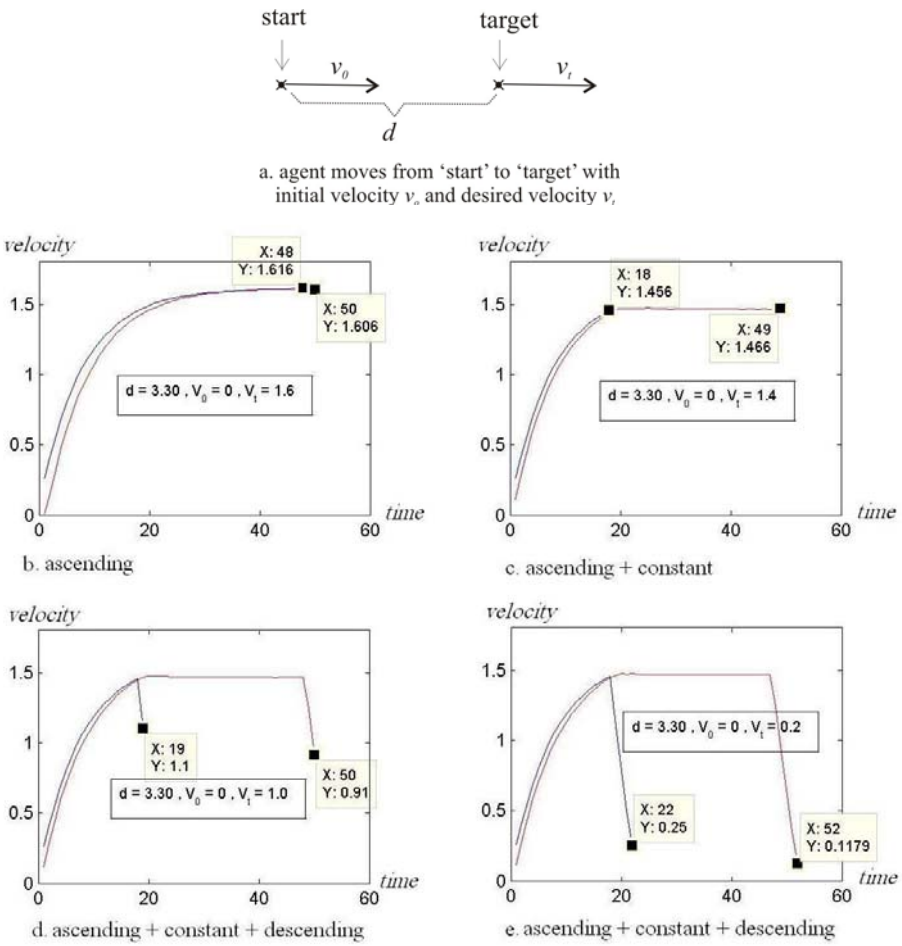

Fig. 2. red curve is learned data and blue curve is experiment data. a. Agent's velocity at target is not limited. Agent starts to move with maximum power, reaches to its max speed, and continues; b. Agent's velocity at target is less than agent's max velocity, and system offers only increasing velocity up to the ultimate velocity (not to max velocity); c. and d. The velocity is also less than max velocity and system calculated two moments for velocity changes.

4 Planning

Performance of a multi-agent system depends on how well they co-operate with each other. MRL have focused on distributed planning in order to improve co-operation between its agents. A plan in MAS is a sequence of actions which is distributed among a group of agents. Depending on the environment in which agent will execute

the plan, it needs to think about additional aspects like timing and uncertainty. Soccer is obviously a very hard and complex problem. According to Russell's classification of problems [3], soccer could be classified as a partial-observable, uncertain, sequential, dynamic, continual, and multi-agent environment. Regarding these features, a soccer agent's planner would be more complex than common planners which used for a simple robot. Such a planner should provide high-level decisions for agents as well as allows agent to behave reactively.

After testing some general planners such as Blackbox [4], Graphplan [5], PRODIGY [6] we have found out that general purpose planners could not work efficiently for some specific purposes like soccer problem. The reason is limitations that they have for being general purpose. Therefore MRL started to develop a specific planner by collecting ideas from existing planners and adding features to make the new planner continual and distributed. Fig. 3 depicts the planner's control flow diagram.

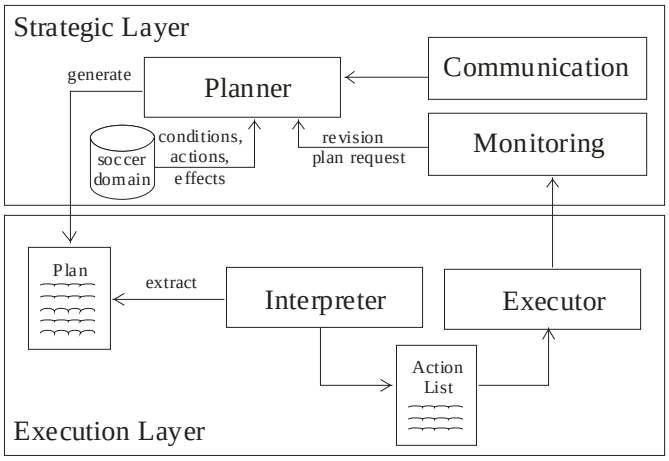

Fig. 3. Control flow diagram of MRL planner architecture

MRL's planner is a continual graph-based planner. This kind of planners generates a graph of possible sequenced actions for a given goal, and then searches the graph for the best sequence of actions that ends in a state containing goal state. Depending on the formalism, the problem of generating plan can be decidable in polynomial, or even NP-complete. Planning should not be time-consuming because it affects agent's reactive behavior. In fact, agent should be able to plan and react simultaneously. The following is a plan that a forward player generated while it possessed the ball:

```
Current facts                          Generated plan:
                                       Give-and-go(p8,p9,a1)
objects: Player(p8);Area(a1);
init:    BALLOWNER(p8)
goal:    MOVEBALL(a1)
```

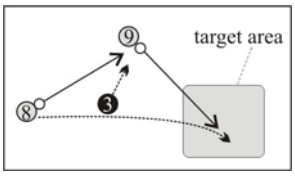

Fig. 3. A sample plan generated by MRL planner.

In this example planner generated a simple plan (give-and-go) in order to achieve the goal "moving ball to area a1". This plan should be interpreted by the *interpreter* to a sequence of actions that should be done by player 8 and player 9.

```
Agent p8:                    Agent p9:
1- CommunicatePlan(p9)
1- DirectPass(p9);           1- GetOpen2ReceivePass(p8);
2- Go2Point(20,15);          2- DiagonalPass(p8);
3- PossessBall();
```

Then the *executor* starts to execute it. And *monitoring* controls the execution and compares it with the plan. If executing does not support the plan, *monitoring* send a revise signal to the *planner*. Then *planner* modifies the old plan or generates another plan depend on the current world model.

Planning could be run as separate computational process, but can also be done by carefully coding the algorithms so they can be manually interleaved within a single computational process. MRL selected the single process approach.

5 Conclusions and Future Works

Developing a multi-agent system like robot soccer team needs to consider both deliberation and reaction aspects. MRL is improving its performance by using a planner for providing deliberation as well as using powerful learning methods for reactions.

For the planner we are going to add temporal feature. Then it would be possible to measure the performance of the execution and also *monitoring* could be more accurate.

Controlling agent's velocity is base on lookup tables and it is limited to position of the target and direction of velocities at the start and end of the path. To remove this limitation the algorithm should be extended and consequently size of the tables would increase. Afterwards searching would be time consuming and using look-up tables makes the mechanism to behave discretely. Using Neural Networks could be a solution for optimizing cost of the search and estimating non-existing entries. MRL is going to extend the mechanism of agent's velocity control in order to use it without mentions limitations.

References

1. R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems", Transaction of the ASME-Journal of Basic Engineering, pp. 35-45 (1960).
2. M. Reidmiller, A. Merke, M. Nickschas, W. Nowak, and D. Withopf. Brainstormers 2003, 2D Soccer Simulation Team Description. Proceeding of RoboCup 2003 (2003)
3. S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach" (Second Edition) (2003)
4. J. Duchi, J. Connor, and B. Lo, "An Evaluation of Blackbox Graph Planning", Stanford University, (2005)
5. A. Blum and M. Furst, "Fast planning through planning graph analysis", Artificial Intelligence, 90, 281-- 300, (1997)
6. M. Veloso, J. Carbonell, A. Perez, D. Borrajo, E. Fink, J. Blythe, "Integrating Planning and Learning: The PRODIGY Architecture ", Journal of Experimental and Theoretical Artificial Intelligence, (1995)