ITAndroids 2D Soccer Simulation Team Description 2018

Felipe Vieira Coimbra and Lucas Alberto Bilobran Lema

Aeronautics Institute of Technology,
São José dos Campos, São Paulo, Brazil
{felipecoimbra97,skybilobran}@gmail.com
itandroids-soccer2d@googlegroups.com
http://www.itandroids.com.br/

Abstract. ITAndroids 2D Soccer Simulation team is composed by undergraduate students of Aeronautics Institute of Technology. The team is currently one of the strongest teams in Brazil, having won 1st place from 2012 to 2015, 2nd place in 2016 and 3rd place in 2017 Latin American Competition. Moreover, the team has also qualified for the last four editions of RoboCup. This paper references our previous developments, describes the newest enhancements in world modelling and defense strategy and predicts our next steps for 2018.

1 Introduction

ITAndroids is a competitive robotics team from Aeronautics Institute of Technology reestablished in 2011. The group participates in the following leagues: RoboCup 2D Soccer Simulation (Soccer 2D), RoboCup 3D Soccer Simulation, RoboCup Humanoid Kid-Size, IEEE Humanoid Robot Racing, IEEE Very Small Size and has recently started research in RoboCup Small Size league.

Our Soccer 2D's team, ITAndroids 2D, has continuously participated in Latin American Robotics Competition (LARC) and Brazilian Robotics Competition (CBR) since 2011. Moreover, ITAndroids 2D competed in RoboCup in 2013, 2015 and 2016. The team also qualified for RoboCup 2014, but unfortunately it was not able to attend to the competition.

ITAndroids 2D is considered a strong Soccer 2D team in Brazil and Latin America, winning 1st place in LARC/CBR from 2012 to 2015 and placing at 10th in 2012 and 13th from 2013 to 2016 in RoboCup. Unfortunately, due to lack of continuation and documentation of the project and the spreading of the team towards other categories, the ITAndroids 2D slowed down its improvements. As a result, we won 2nd place in LARC 2016, fell down to the 15th place at RoboCup 2017 and couldn't finish an ambitious project in the defense strategy in time, awarding 3rd place in LARC 2017.

However, the ITAndroids 2D has greatly recovered in 2017, after a complete restructuring of the project. As a result, interesting advancements were made: the creation of a state machine that takes into account uncertainty of world information to analyze ball possession and a brand new positioning model for the team's defense.

The team has made substantial advancements in previous years from developing good strategy heuristics to implementing neural networks to select between more aggressive behavior or not[1,2]. These previous accomplishments until 2016 are compiled in [3]. The current work focuses on the team's developed work during 2017, explained in details in sections 2 and 3.

2 Uncertain Possession Automaton

2.1 Basic Ball Possession Determinations and the Possession Automaton

Determining the current ball possession in an environment like the Soccer Simulation 2D may not be a trivial task, mainly because the world information received from the server is noisy and incomplete, matches are very dynamic and the players' actions close to unpredictable and the proper concept of ball possession is not clear enough.

Attempting to solve these issues, Agent2d [8] comes with an intercept estimation possession heuristic, handled by a class named InterceptTable. Simulating the process of reaching the ball from every player, it finds the one with minimum reach cycles required. This player determines a *Situation*: *OffenseSituation* when from our team and *DefenseSituation* otherwise.

However, Agent2d labels the situation where players of opposite teams have just a couple of cycles of difference between their reach cycles as *NormalSituation* and does not try to determine which team has the ball. In ITAndroids, we have further developed the intercept model by realizing we may infer actual possession status at any moment if we define it as a function of past and current *Situations* (Defense, Normal or Offense) determined by InterceptTable simulations. This prediction structure was called **Possession Automaton**.

The automaton consists of two states, OurPossession and TheirPossession, with a transition function that depends on the actual state, the last *Situation* and the current *Situation* newly calculated by the InterceptTable, as illustrated in figure 1. This way, the possession estimation is a result of a sequence of states, and not uniquely determined by a single instance of the world. This brings consistence to the possession determination, better overpassing difficulty 2.

2.2 Uncertainty Coefficient and addition of Uncertain States

Noise and incomplete information may substantially interfere in InterceptTable's role, cause fluctuations in the transition function's result and by accident give wrong assertions about the ball possession. To solve it, we add *Uncertain States* to the *Situations*. Uncertain States correspond to the equivalents of each *Situation* when the information used to derive the InterceptTable's result is not accurate enough. This way, we append a new *Situation*, called uncertain equivalent

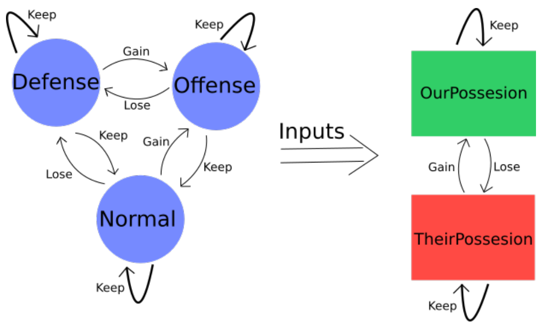

Fig. 1. Illustration of the Possession Automaton. In the left, each possible change between *Situations* (blue circles) is mapped to a corresponding transition(the neighboring names) that is then given as input to the Possession Automaton(in the right), changing the possession state.

or conjugate, on each already existing one. Thereafter, the transition function is changed in a way that every *Situation* transition throws a neutral input except for the transitions between equivalent states (see Figure 2).

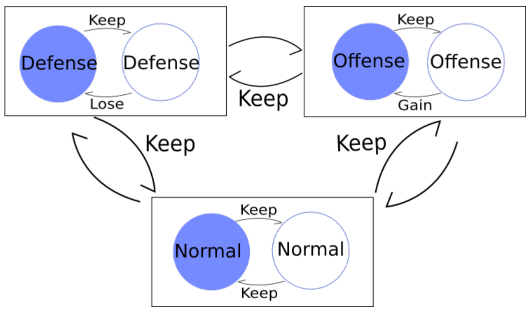

Fig. 2. Illustration of the generation of input of the Uncertain Possession Automaton. The hollow circles are the uncertain conjugates of *Situations*. The only allowed transitions to change the automaton state are the ones between conjugates.

The purpose of this manipulation is to stop players from making assumptions from too noisy information. The principle behind it is that a stabilized decision towards ball possession persists throughout cycles and we use this as a measurement of trustfulness of the decision.

This new feature requires of measuring quantitatively how accurate a received sensory information is; to solve this we elaborated the Uncertainty Coefficient γ calculated by equation 1. It is an individual property of every player and it is defined as a weighted linear combination of the range error of its seen position and the number of cycles since this information was received. It is said that a *Situation* is uncertain (the observed player is in an Uncertain State) when the fastest reachable player to the ball has its Uncertain Coefficient above some threshold ϵ.

$$\gamma = w_1 \times RangeError + w_2 \times CycleCount \tag{1}$$

Weights w_1 and w_2 may be adjusted according to any chosen heuristic. In ITAndroids we have adopted values to w_1 and w_2 that comes from our concept of how many outdated cycles and what maximum distance of observation should be tolerated by the automaton and how should be their prioritization. Notice that defining uncertainty by equation (1) makes any *Situation* naturally tend to its uncertain conjugate because of the $CycleCount$ term. As long as the consistency of the new possession status keeps the *Situation* as Offense, the Uncertainty Coefficients oscillate and certainly in a couple of cycles a non-neutral transition happens, changing the possession state.

2.3 Results

The Uncertain Possession Automaton predicted correctly the ball possession during **(98.9±0.4)%** of the match while the intercept heuristics alone was able to proper identify the ball holder team during only **79.7%** of the time with a higher standard deviation of ±**0.9**. This statistics was calculated comparing simultaneously the prospection of each method with and without any noise, using the fullstate debug mode (the fullstate debug grants perfect information) and the same InterceptTable for *Situation* estimation. The goalkeeper prediction was not taken into account.

Table 1. Average percentage of cycles with correct possession estimation by some specific group of players

Side Players	Center Players	Defense Players	Offense Players
98.7%	99.1%	98.9%	98.9%

Interesting details of the statistics raised are pointed out in Table 2.3. Side players tend to have inaccurate information, thus another good heuristics for ball possession determination would be to trust the decision to central middle players. The rest of the team could be informed of their decision by audio messages. In ITAndroids we had preferred the individual decision though.

3 A New Defense Model: Man-Marking

3.1 Problems of formation-based positioning of the defense

Delaunay Triangulation based positioning used by Agent2D[8] does not always result in good defense positioning, because it comprises a task that should heavily take into account the position of opponents too.

ITAndroids decided to change the way the players stood defensively to a more dynamic way by attacking where Delaunay Triangulation lacks: marking. Basically changing a zoned-marking strategy to a man-marking strategy.

In this new approach, the players are now responsible for individual opponents, thus making their positioning naturally dynamic and adaptive to different teams.

3.2 Marking as an assignment problem

The theoretical basis of man-marking comes from graph theory once the problem is modeled as an assignment problem. We define a bipartite player graph, where our players fill the vertices of one partition and the opponent players fill the vertices of the other partition. We are looking for a perfect matching in this graph, which is done by an assignment function that, given a world state, returns a mapping of every marker to a single marked opponent. This idea was inspired by the dynamic role assignment, developed by Austin Villa 3D.

Defining which players to create representative vertices in the graph is a challenge. We have only put players in the opponent graph partition that are inside a pre-defined region of interest and filled the remaining slots with dummy nodes. The predefined rectangle has width equal to the field width and variable length, that linearly increases from 15m to 25m as the defense line comes closer to our goal. Players assigned to dummy nodes move accordingly to Delaunay Triangulation.

The assignment problem may be solved by the well known Hungarian Algorithm [4]. If we define the weight of the edges as distances between markers and marked players, the Hungarian Algorithm returns a matching that minimizes the total sum of distances between these players. This is a good as it minimizes the time required to players to approach enemy players. More sophisticated weight functions may be elaborated, when heuristics demand players to be differently prioritized.

Some example of marking types are marking to avoid direct passes, marking to avoid long passes, marking to protect the goal from shoots and, of course, hassle marking to pressure the opponent holder to lose possession.

3.3 Difficulties found and proposed solutions

Handmade marking behaviours would need to be very complex so that they could handle every game situation in an optimal manner. Besides, they would be very non-adaptive and would have many performance disparities when used against different teams.

Behaviour generation is a class of problems that seem to achieve great results using Reinforcement Learning techniques [5]. We are currently analyzing the possibility of using Deep Neural Reinforcement Learning to create neural behaviours for marking, as well as currently implementing the MMDR (Minimal Maximal Distance Recursive) proposed by Austin Villa 3D to better seek the treatment of marking as a concurrent process.

4 Conclusions and Future Work

This paper presented the most recent efforts of team ITAndroids 2D. Our current code is based on agent2d. The category has become discontinuous in recent years mainly due to lack of documentation. In 2017, the code has been deeply restructured and automated documentation was implemented, accelerating learning, teaching and development. In addition, there were changes in the defensive heuristic of the team, moving towards more adaptive positioning.

Currently, our efforts are focused on: the dynamization of team defensiveness and offensiveness and optimized behaviour generation by reinforcement learning techniques.

5 Acknowledgements

We would like to acknowledge the RoboCup community for sharing their developments and ideas. Specially, we would like to acknowledge Hidehisa Akiyama for agent2d [8], librcsc [7], soccerwindow2 [6] and fedit2 [9] software. Current team members are also grateful to previous ones who have made great contributions to ITAndroids 2D. Finally, we thank our sponsors Altium, ITAEx, Metinjo, Micropress, Poliedro, Poupex, Rapid, and Solidworks. We also acknowledge Mathworks (MATLAB), Atlassian (Bitbucket) and JetBrains (CLion) for providing access to high quality software.

References

1. Mello, F., Ramos, L., Maximo, M., Ferreira, R., Moura, V.: ITAndroids 2D Team Description 2012 (2012)
2. Mello, F., Ramos, L., Maximo, M., Ferreira, R., Moura, V.: ITAndroids 2D Team Description 2013 (2013)
3. Coimbra, F., Marinot, G., Marcondes, M.: ITAndroids 2D Team Description 2017 (2017)
4. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2(1-2), 83–97 (1955)
5. Gabel, T., Riedmiller, M., & Trost, F. (2008). A case study on improving defense behavior in soccer simulation 2D: the NeuroHassle approach. In Iocchi, L., Matsubara, H., Weitzenfeld, A., & Zhou, C. (Eds.), LNCS. RoboCup 2008: robot soccer world cup XII, Suzhou, China. Berlin: Springer
6. soccerwindow2-5.1.0, 2011, online, avaliable at: http://pt.sourceforge.jp/projects/rctools/ downloads/519 42/soccerwindow2-5.1.0.tar.gz/, consulted on December 2017.
7. librcsc-4.1.0, 2011, online, avaliable at: http://pt.sourceforge.jp/projects/rctools/ downloads/51941/librc sc-4.1.0.tar.gz/, consulted on December 2017.
8. agent2d-3.1.1, 2012, online, avaliable at: http://pt.sourceforge.jp/projects/rctools/ downloads/51943/agen t2d-3.1.0.tar.gz/, consulted on December 2017.
9. fedit2-0.0.1, 2017, online, avaliable at: https://pt.osdn.net/projects/rctools/ downloads/68531/fedit2-0.0.1.tar.gz/, consulted on December 2017.
10. cloc, 2017, online, avalaible at: https://github.com/AlDanial/cloc