# Analysis of Multi-Robot Play Effectiveness and of Distributed Incidental Play Recognition

Colin McMillen and Manuela Veloso

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, U.S.A. {mcmillen,veloso}@cs.cmu.edu

**Summary.** Distributed play-based approaches have been proposed as an effective means of switching strategies during the course of timed, zero-sum games, such as robot soccer. In this paper, we empirically show that different plays have a significant effect on opponent performance in real robot soccer games. We also consider the problem of distributed play recognition: classifying the strategy being played by the opponent team. Play recognition in real robot soccer is a particularly challenging problem because our observations are only "incidental" — that is, the primary task of our team is to play soccer, not to explicitly observe members of the other team. Despite these challenges, we achieve high classification accuracy in the robot soccer domain.

## 1 Introduction

Many distributed robotic systems operate in complex, uncertain environments. In order to function robustly in such domains, robot teams may need to be equipped with multiple alternate approaches to solving the same tasks. For instance, a team that is traveling together through a crowded hallway may need to adopt a significantly different team-level behavior than a team that is navigating through a wide open space. In the past, we have proposed *plays* as a means for a distributed team to adapt its high-level strategy to the features found in the environment [2]. In this paper, we present an experimental study showing that different plays have a significant effect on team performance in a robot soccer domain. We also consider the problem of distributed play recognition: combining our team's observations to classify the behavior of the opponent team. In a real robot soccer game, our team members must focus their limited field of view on the most important objects in the environment, such as the ball and the localization landmarks. In contrast to a team that is fully engaged in tracking multiple moving objects, such as in [5], our team does not have the luxury of purposefully tracking opponent robots or spreading out to maximize the portion of the field that is viewed by the team. Despite these challenges, we show that our distributed team is able to effectively recognize the behavior of the opponent team.

## 2 Background

In previous work, we have presented a distributed, play-based strategy selection algorithm that allows a team of robots to autonomously change team strategies based on features such as the score of the game and the time remaining [2]. A *play* is a team plan that provides a set of roles; these roles are assigned to robots upon initiation of the play [1]. Each role specifies the top-level behavior of the associated robot. In the RoboCup domain, our robots' roles are *region-based*: each robot is assigned to play in a specific region of the field, and will only pursue the ball when the ball is within its own region.

A *play selector* runs continuously on one robot that is arbitrarily chosen to be the leader. As input, the play selector receives the state of the world (including the score of the game and the time remaining) and a *playbook* — the list of all strategies available to the team. The play selector finds the best play from the playbook and broadcasts the results to all teammates. In the past, we hand-coded a set of weights — one for each play in the playbook — and the play selector would choose the highest-weight play that was applicable in the current state. This allowed us to manually specify team behaviors such as "if there are fewer than two minutes remaining in the game and we are losing, play aggressively." This algorithm was originally used in the RoboCup 2005 competition and was the first demonstrated instance of a distributed RoboCup team changing strategies in response to the score and time of the soccer game.

One weakness of our previous work is that the play weights were hand-coded by humans, rather than selected optimally. We therefore introduced *thresholded-rewards MDPs*, which allow an agent to act optimally in domains in which the goal is to achieve a specified reward level within a given time deadline [3, 4]. In RoboCup, we treat each of our goals as $+1$ reward and each opponent goal as $-1$ reward; then the goal is to maximize the probability of achieving a positive cumulative reward by the end of the game. The solution to a thresholded-rewards MDP gives us the policy that optimizes our chances of winning in such a situation. In previous work, we have shown that switching strategies based on score and time allows us to win more often than we lose, compared to an opponent that does not switch strategies — even if the opponent has strictly better low-level skills (i.e. probabilities of scoring at each time step) than our team does. However, this work only addressed a simple MDP-based model of the RoboCup domain, and was not actually implemented on real robots.

In this paper, we address the problem of strategy switching in a real distributed robot team. We address two separate aspects of this problem: measuring the effectiveness of switching our own plays, and successfully recognizing the strategy played by the opponent team. All our results are based on extensive experiments using real robot teams in a realistic robot soccer domain.

Section 3 describes our experimental domain. In Section 4, we present empirical results showing that switching strategies against a given opponent can produce a significantly different distribution of goals scored. In combination with our previous results, this shows that we should be able to win more often than lose when playing against opponents that do not switch strategies, even if the teams are otherwise equally matched. In Section 1, we show how our team's distributed perception of the game state and opponent positions can be used to classify the strategy being used by the other team with high accuracy. In a real game scenario, these classification results could be used to change our own strategy in response to the opponents' behavior. We present our conclusions in Section 6.

## 3 Experimental Domain

All the experiments presented in this paper were carried out in a domain based on the official RoboCup Four-Legged League rules for 2008 [7]. In each experiment, two teams of three robots played a 10-minute-long game of soccer. The software used on both teams was the code for Carnegie Mellon's CMDash'08 entry in the 2008 RoboCup US Open. Therefore, the underlying behaviors and skills of each team were identical; the only difference between experimental trials was the selection of plays for each team. Each robot on each team was a field player; there were no goalkeepers. Due to this, the Illegal Defender rule (which prevents the defending team's field players from entering the goal box) was not enforced. All other penalties, such as player pushing, leaving the field, and ball holding, were enforced. In case of a penalty, the offending player(s) were picked up and moved immediately to the halfway line (a "0-second standard removal penalty") rather than remaining out of play for 30 seconds. This change was made such that a single human could successfully referee a game.

In this paper, we focus on two separate plays: RoboCup and SuperDefense. The RoboCup play is the default play we have generally used in previous competitions. This play assigns three roles: an *attacker* robot that chases the ball over the entire field, a *defender* robot that protects the defensive area of the field, and a *supporter* robot that stays in the offensive region of the field. The RoboCup play is a balanced strategy that allows our team to score effectively and also provides a reasonable defense. In contrast, the SuperDefense play assigns all three field players to defensive roles. The *front defender* covers approximately the same region as a normal defender robot, but stands much further forward when the ball is not in the defensive half of the field. The *middle defender* covers a slightly smaller region and usually stands about a meter behind the front defender; the *rear defender* only covers the area closest to the goal and usually stands about a meter behind the middle defender. Typically, when playing SuperDefense, the front defender will immediately engage the ball when it enters the defensive half of the field. If the front defender fails to clear the ball, the ball will eventually enter the middle defender's region; the middle defender will then join the front defender in trying to clear the ball. If the attacking team is still able to advance the ball forward near the goal, the rear defender joins in the defense and also attempts to clear the ball. By employing this strategy of defense-in-depth, the SuperDefense play is intended to make it very difficult for an opponent to successfully score a goal. However, the SuperDefense play comes with a significant drawback; namely, it is very unlikely that this play will ever score a goal. In a real game situation, we would never run the SuperDefense play for an entire game; rather, this is a strategy we would like to employ when our team is ahead near the end of the game and we wish to prevent the opponents from scoring an equalizer goal.

FIXME: show photos of the two different plays + the regions of the roles

## 4 Multi-Robot Play Effectiveness

In this section, we aim to answer the question: do different team strategies produce significantly different outcomes when run against a real opponent team? To test this, we played a series of games against a static team configured to use the RoboCup strategy. For the control case, our team also played the RoboCup strategy. For the experimental case, our team played the SuperDefense strategy. We are interested in seeing if the SuperDefense play leads to fewer opponent goals than RoboCup, and also if the time between opponent goals is significantly higher for SuperDefense. Each case was tested in 12 independent 10-minute games, for a total of 4 hours' worth of game time. To ensure fairness between the trials, the colors of the robots' uniforms and the sides of the field were swapped between each trial. The final score of each game and the time of each goal was recorded.

When playing RoboCup against RoboCup, our team scored 35 goals in total; this is a mean of 2.93 goals per game. The opponent team scored 42 goals in total, a mean of 3.50 goals per game. When playing SuperDefense against RoboCup, our team scored no goals; the opponent team scored a total of 22 goals, a mean of 1.83 goals per game. Figure 1 shows a histogram of the number of goals scored by our opponents in each of the games.

We tested the statistical significance of the difference between the number of opponent goals scored per game using Student's two-tailed $t$-test (assuming unequal variances) and a one-way analysis of variance (ANOVA). Both tests indicate that the difference in distributions between our two conditions are statistically significant ($p = 0.0107$ for the $t$-test; $p = 0.0104$ for the ANOVA).

In addition to the number of goals scored by our opponents, we are also interested in how long it takes for the opponents to score. We therefore calculated the time between consecutive opponent goals in each of the games. At

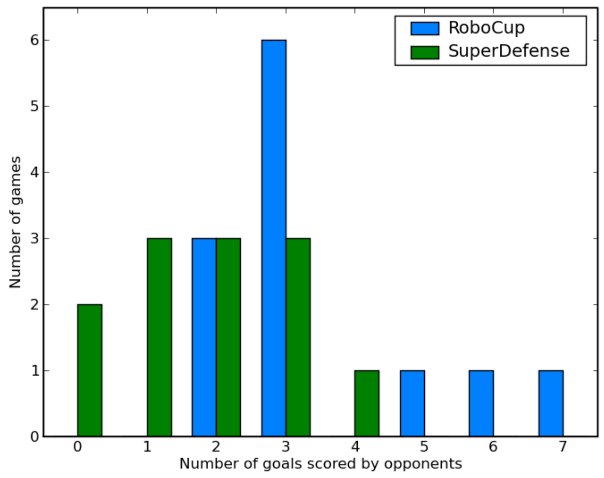

**Fig. 1.** Histogram of the number of goals scored per game for the two conditions. When our team uses the SuperDefense play, the opponents score significantly fewer goals.

the end of the game, we pessimistically assume that the opponents score immediately after the game is over, since we don't know how long it would have taken for the opponents to score their next goal (unless the game time remaining at the time of the last opponent goal was less than 29 seconds, which was the fastest time between goals in any trial). This leads to a mean of 133.3 seconds between goals for the RoboCup condition, and a mean of 223.4 seconds between opponent goals for the SuperDefense condition. Figure 2 shows a histogram of the time between opponent goals for the two conditions.

We again tested the statistical significance of the difference between the distributions using Student's two-tailed $t$-test (assuming unequal variances) and a one-way analysis of variance (ANOVA). Both tests indicate that the difference in distributions between our two conditions are statistically significant ($p = 0.0077$ for the $t$-test; $p = 0.0015$ for the ANOVA).

## 5 Distributed Incidental Play Recognition

In the previous section, we considered the effect of switching our own play in order to prevent the opponent team from scoring goals. In this section, we consider the problem of *play recognition* — recognizing the behavior of the opponent team from our own team's observations. This is an interesting distributed perception problem because our team's observations of the environment and other robots are *incidental* rather than *purposeful*. By this, we

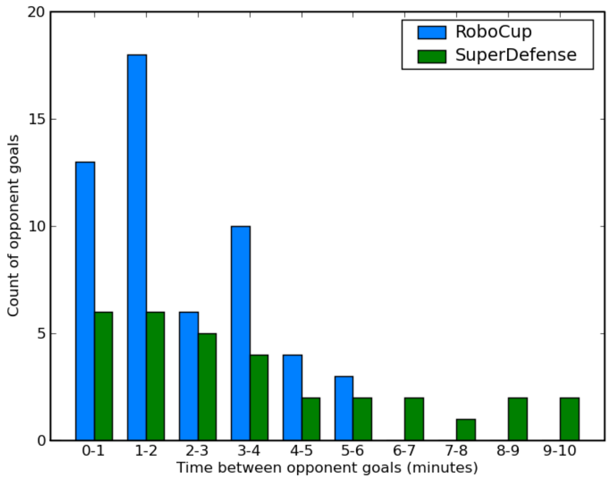

**Fig. 2.** Distribution of time between opponent goals. When our team uses the SuperDefense play, the opponents take significantly longer to score goals.

mean that the primary task of our team is not to observe the environment and classify the behavior of the other team; rather, our task is to play soccer well, and any observations of the environment or opponent robots happen by "accident" while we play soccer. Since the AIBO camera's field of view is very narrow — under 60 degrees — each robot has a very limited focus of attention, which is usually focused on the ball since the ball is the most important object in the environment. In particular, our team does not explicitly track the movements of the opponent robots nor attempt to maximize the portion of the field that is viewed.

In this paper, we use hidden Markov models (HMMs) to model the behavior of the other team [6]. To enable distributed play recognition, each of our robots communicates the following data once per second:

- The robot's best estimate of its own position.
- The robot's best estimate of the position of the ball.
- The position of the last opponent robot seen by the robot.

All positions are communicated in global coordinates, so the team has a common frame of reference. For each of these features, each robot also broadcasts a boolean feature indicating whether the observation is considered "valid". For example, the robot's estimate of its own position is valid iff the robot's estimate of localization error is relatively low. For purposes of this experiment, the data broadcast by each robot was also sent to an offboard computer for later processing; in a real game situation the robots would only broadcast to each other and use the results to classify the opponent behavior online. Fig-

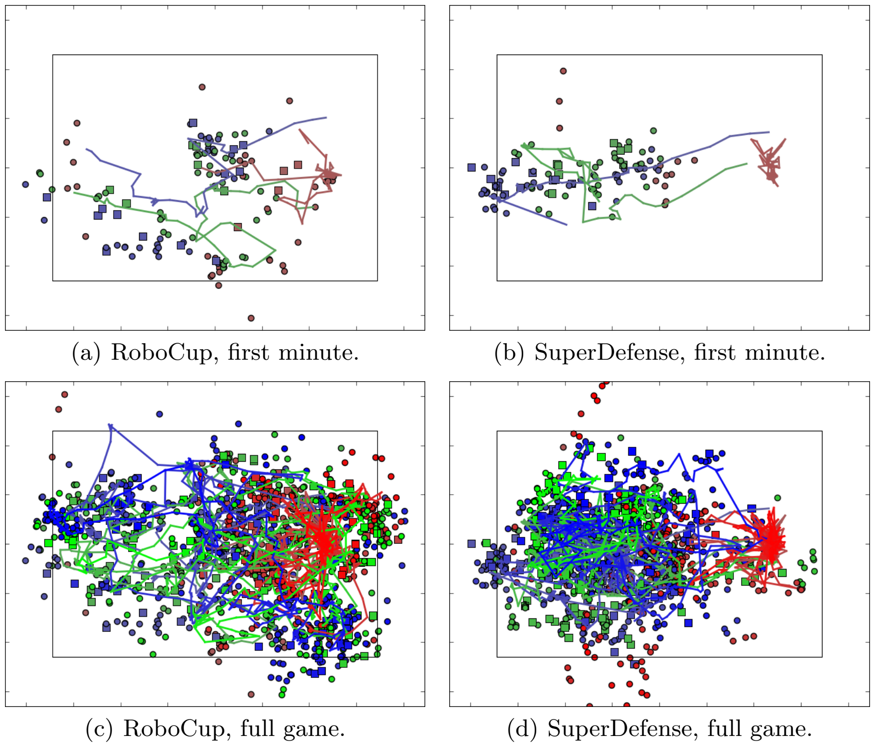
(a) RoboCup, first minute.
(b) SuperDefense, first minute.
(c) RoboCup, full game.
(d) SuperDefense, full game.

**Fig. 3.** Sample log data from two games. Lines indicate the positions of our own robots, circles indicate observations of the ball, and squares indicate observations of opponent robots. Our goal is located on the right side of each figure; the opponents' goal is located on the left side of each figure. Figure (a) shows data from the first minute of a game in which the opponents are playing the RoboCup strategy; Figure (b) shows data from the first minute of a game in which the opponents are playing the SuperDefense strategy. Figures (c) and (d) respectively show the data from entire RoboCup and SuperDefense games.

ure 3 shows some sample data logged by the offboard computer from a typical RoboCup game and a typical SuperDefense game.

Formally, the observation of a single robot $r$ at timestep $t$ is:

$$o_{r,t} = \langle bx_{r,t}, by_{r,t}, bv_{r,t}, px_{r,t}, py_{r,t}, pv_{r,t}, ox_{r,t}, oy_{r,t}, ov_{r,t} \rangle, \tag{1}$$

where $bx_{r,t}$ and $by_{r,t}$ give the position of the ball in global coordinates as seen by robot $r$ at time $t$ and $bv_{r,t}$ is a boolean flag saying whether robot $r$ considers the ball observation to be valid. Similarly, $px_{r,t}$, $py_{r,t}$, and $pv_{r},t$ give the position and validity of robot $r$'s own position and $ox_{r,t}$, $oy_{r,t}$, and $ov_{r,t}$ give the position and validity of the opponent most recently observed by robot $r$. We then define the joint observation $j_t$ of the team at time $t$ as the combination of the individual robots' observations at time $t$: $j_t =$

**Algorithm 1** PLAY-RECOGNITION algorithm.

```
1: Given: training sets $RC_{train}$ and $SD_{train}$, observation sequence $J$ of length $t$
2: $\lambda_{RC} \leftarrow$ BAUM-WELCH($RC_{train}$)
3: $\lambda_{SD} \leftarrow$ BAUM-WELCH($SD_{train}$)
4: for $i = 1$ to $t$ do
5:     $J_i \leftarrow \langle j_1, j_2, \ldots, j_i \rangle$
6:     $p_{RC} \leftarrow P(J_i|\lambda_{RC})$
7:     $p_{SD} \leftarrow P(J_i|\lambda_{SD})$
8:     if $p_{RC} \geq p_{SD}$ then
9:         $C[i] \leftarrow$ ROBOCUP
10:    else
11:        $C[i] \leftarrow$ SUPERDEFENSE
12:    end if
13: end for
14: return $C[i]$
```

$\langle o_{1,t}, o_{2,t}, o_{3,t} \rangle$. Over the course of an entire game that is $t$ timesteps long, the robots obtain a joint observation sequence $J = \langle j_1, j_2, \ldots, j_t \rangle$.

Algorithm 1 presents our play-recognition algorithm. This algorithm assumes we have a training set $RC_{train}$ consisting of a set $\{J_1, J_2, \ldots, J_n\}$ of observation sequences gathered while the opponent was playing the RoboCup play, and another training set $SD_{train}$ consisting of a set $\{J_1, J_2, \ldots, J_n\}$ of observation sequences gathered while the opponent was playing the SuperDefense play. On lines 2–3 of the play-recognition algorithm, we train one HMM $\lambda_{RC}$ by providing the training sequences $RC_{train}$ as input to the Baum-Welch algorithm; we also train another HMM $\lambda_{SD}$ by providing the training sequences $SD_{train}$ as input to the Baum-Welch algorithm. Lines 4–13 provide online classification at each timestep $i$. $J_i$ is the vector of all joint observations $\langle j_1, j_2, \ldots, j_t \rangle$ seen by the team in the first $i$ timesteps. On lines 6–7, we compute the likelihood of $J_i$ according to the two models $\lambda RC$ and $\lambda SD$. These likelihoods are computed using the forward algorithm. The model $\lambda$ which maximizes the likelihood of the observation sequence is chosen as the best estimate of which play the opponent team is running. The classification output is stored in the array $C$.

To collect training data, we ran twelve 10-minute games against an opponent team running the RoboCup play and twelve 10-minute games against an opponent team running the SuperDefense play. In all trials, our own team was running the RoboCup play. Let $RC$ be the set consisting of the 12 joint observation sequences gathered when playing against a RoboCup opponent; let $SD$ be the set consisting of the 12 joint observation sequences gathered when playing against a SuperDefense opponent. We perform leave-one-out cross-validation to evaluate the effectiveness of our approach. Algorithm 2 presents our cross-validation procedure. Since we have 12 observation sequences for each case, the main loop (lines 3–12) runs 12 times. In each iteration, one element $sd$ is held out of $SD$ and one element $rc$ is held out

**Algorithm 2** Cross-validation procedure.

```
1: Given: sets of observation sequences $RC$ and $SD$, each of size $k$
2: $correct \leftarrow 0$; $incorrect \leftarrow 0$
3: for $i = 1$ to $k$ do
4:     $rc \leftarrow RC_i$
5:     $sd \leftarrow SD_i$
6:     $C_{rc} \leftarrow$ PLAY-RECOGNITION$(RC - \{rc\}, SD - \{sd\}, rc)$
7:     $C_{sd} \leftarrow$ PLAY-RECOGNITION$(RC - \{rc\}, SD - \{sd\}, sd)$
8:     $correct \leftarrow correct +$ number of elements $e$ in $C_{rc}$ s.t. $e =$ ROBOCUP
9:     $incorrect \leftarrow incorrect +$ number of elements $e$ in $C_{rc}$ s.t. $e \neq$ ROBOCUP
10:    $correct \leftarrow correct +$ number of elements $e$ in $C_{sd}$ s.t. $e =$ SUPERDEFENSE
11:    $incorrect \leftarrow incorrect +$ number of elements $e$ in $C_{sd}$ s.t. $e \neq$ SUPERDEFENSE
12: end for
13: return $correct$ / ( $correct + incorrect$ )
```

of $RC$. The remaining observation sequences are used to classify $sd$ and $rc$, and the number of successful and unsuccessful classifications is recorded. The procedure returns the fraction of timesteps which were successfully classified.

In order to achieve good classification performance, we need a good domain representation. We are therefore interested in selecting an informative set of *features* from the raw joint observations. In principle, we can apply any arbitrary function $f(j)$ to map a single joint observation to a vector of features. In this paper, we only consider feature-selection functions of a limited form. Namely, $f(j)$ can only perform two operations:

1. *Filtering* out some features, removing them from the joint observation entirely. Some of the fields communicated by the robots may not be useful for play recognition, in which case the models may overfit if these fields are present. Filtering allows useless features to be ignored.
2. *Re-ordering* features in the joint observation. It may be more useful for a specific field in the feature vector to represent "the position of the robot closest to the opponent goal" rather than "the position of robot 1". Feature re-ordering allows such semantics to be an explicit part of the model.

To help understand which features are most important for successful play recognition in robot soccer, we tried a variety of feature-selection functions and computed the overall classification accuracy of each.

By itself, ball position is the most informative feature. If the observation vectors include only the ball's $x$- and $y$-coordinates plus the "ball valid" flag, and the ball observations are sorted by their $x$-coordinates, we achieve an overall accuracy of 86.98%. In our coordinate system, the center of the field is $(0, 0)$, and positive $x$ points towards our own goal, so sorting in ascending order corresponds to ordering the ball observations from "nearest the opponent goal" to "furthest from the opponent goal".

Using only the reported $x$-positions of our own robots (sorted by $x$) achieves an accuracy of 82.34%. It is interesting to note that adding our own

$y$-positions actually lowers our classification accuracy to 76.65%, due to overfitting. Adding the "own position valid" flag also hurts classification accuracy; we assume that this is because the robots' position confidence estimates were tuned for single-robot behavior, such as deciding when the robot needs to look at localization landmarks, rather than team behavior or opponent recognition. By itself, using the $x$-positions of opponent robots (sorted, plus the "opponent valid" flag) performs quite poorly, achieving an accuracy of only 69.54%. Adding the $y$-positions of opponents also decreases classification accuracy, to 63.74%, which we also attribute to overfitting.

If the feature vector incorporates the ball position plus our own positions, overall accuracy increases to 87.44%. If we also include the opponent positions, we reach our best accuracy, of 88.63%. We conclude that the position of the ball is by far the most important feature; however, including additional information can help somewhat in classification performance. This is unsurprising given the *incidental* nature of our classification task — since most of the robots on our team spend most of their time focusing their attention on the ball, the ball is the object that is seen most and has the lowest observation error. The accuracy levels achieved with different sets of features are summarized in Figure 4.

| Features used | Classification accuracy |
|---|---|
| $\langle$ ball $x$, ball $y$, ball valid $\rangle$ | 86.98% |
| $\langle$ own $x$ $\rangle$ | 82.34% |
| $\langle$ opp. $x$, opp. valid $\rangle$ | 69.54% |
| $\langle$ ball $x$, ball $y$, ball valid, own $x$ $\rangle$ | 87.44% |
| $\langle$ ball $x$, ball $y$, ball valid, own $x$, opp. $x$, opp. valid $\rangle$ | 88.63% |

**Fig. 4.** Summary of classification accuracy when using different sets of features as input to the HMM.

Through the selection of the proper set of features, we can achieve classification accuracy of 88.63%. One question remains: in which circumstances do we still tend to make errors? Figure 5 shows our classification accuracy at each time step of a typical RoboCup game and a typical SuperDefense game. We can see from the figure that all of our classification errors occur early in the game, before we have collected many observations. After about 100 seconds have elapsed, our classification accuracy for both games is 100%. In fact, this is the typical pattern seen in almost all the games: the classification accuracy is rather poor at the beginning, but improves significantly by the time 100–200 seconds have elapsed. We therefore claim that in a real robot soccer scenario, we would not want to change the behavior of our own team to adjust to the opponents until either a certain amount of time has elapsed or the likelihoods of the observation sequence given each of the models have diverged significantly.

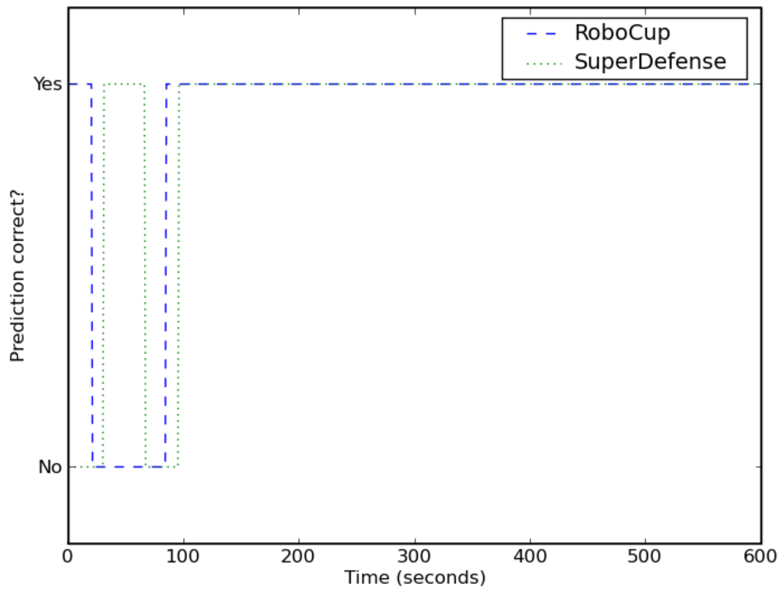

**Fig. 5.** Classification accuracy at each timestep for a typical RoboCup game and a typical SuperDefense game.

## 6 Conclusion

## Acknowledgements

This research was sponsored in part by United States Department of the Interior under Grant No. NBCH-1040007, and in part by the Boeing Corporation. The views and conclusions contained in this document are those of the authors only.

## References

1. M. Bowling, B. Browning, and M. Veloso. Plays as team plans for coordination and adaptation. In *Proc. 14th Intl. Conf. on Automated Planning and Scheduling (ICAPS-04)*, June 2004.
2. C. McMillen and M. Veloso. Distributed, play-based role assignment for robot teams in dynamic environments. In *Proceedings of Distributed Autonomous Robotic Systems*, July 2006.
3. C. McMillen and M. Veloso. Thresholded rewards: Acting optimally in timed, zero-sum games. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, July 2007.

4. C. McMillen and M. Veloso. Unknown rewards in finite-horizon domains. In *Proceedings of the Twenty-Third Conference on Artificial Intelligence (AAAI-08)*, July 2008 (to appear).
5. L. E. Parker and B. Emmons. Cooperative multi-robot observation of multiple moving targets. In *Proceedings of 1997 International Conference on Robotics and Automation*, 1997.
6. L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
7. RoboCup Technical Committee. Robocup four-legged league rule book (2008 rules, as of January 23, 2008). Available online at: https://www.tzi.de/4legged/pub/Website/Downloads/AiboRules2008.pdf, 2008.