Action Selection via Learning Behavior Patterns in Multi-Robot Domains

Can Erdogan **Manuela Veloso**
Computer Science Department, Carnegie Mellon University
Pittsburgh, PA 15213-3890, U.S.A.

Abstract

The RoboCup robot soccer Small Size League has been running since 1997 with many teams successfully competing and very effectively playing the games. Teams of five robots, with a combined autonomous centralized perception and control, and distributed actuation, move at high speeds in the field space, actuating a golf ball by passing and shooting it to aim at scoring goals. Most teams run their own pre-defined team strategies, unknown to the other teams, with flexible game-state dependent assignment of robot roles and positioning. However, in this fast-paced noisy real robot league, recognizing the opponent team strategies and accordingly adapting one's own play has proven to be a considerable challenge. In this work, we analyze logged data of real games gathered by the CMDragons team, and contribute several results in learning and responding to opponent strategies. We define episodes as segments of interest in the logged data, and introduce a representation that captures the spatial and temporal data of the multi-robot system as instances of geometrical trajectory curves. We then learn a model of the team strategies through a variant of agglomerative hierarchical clustering. Using the learned cluster model, we are able to classify a team behavior incrementally as it occurs. Finally, we define an algorithm that autonomously generates counter tactics, in a simulation based on the real logs, showing that it can recognize and respond to opponent strategies.

1 Introduction

The RoboCup robot soccer Small Size League is an interesting multi-robot system where two teams of five small wheeled robots, as built by the participants, compete to score goals. The robots move at high speeds of approximately 2m/s in a confined playing field of 6m×4m. Each team is autonomously controlled by its own centralized computer that receives input from two shared overhead cameras running at 60 Hz. The processed visual data, made equally available to both teams, consists of the location and orientation of each robot, and the location of the ball [Zickler *et al.*, 2009].

Most of the playing teams have a finite set of pre-planned strategies [Ruiz-del-Solar *et al.*, 2010]. Our own current CMDragons team also follows an STP (skills, tactics, plays) behavior architecture of pre-planned play strategies, as we introduced for our earlier teams [Browning *et al.*, 2005]. In a game, the behavior planning algorithm chooses different team strategies, plays, depending on the position of the ball and other features of the game. For example, an attacking play may include three roles, namely one robot preserving the possession of the ball while two other robots position themselves to receive a pass. The set of strategies of each team is unknown to the other teams. In this work, we investigate the learning of such strategies from data collected in real games.

Our CMDragons system includes a sophisticated logging module to record a complete temporal sequence of the state of a game. At every frame, the log includes the location and orientation of each of own and opponent robots, the location of the ball, and any referee calls, such as penalties, game stoppages, and goals. At a frame rate of 60Hz and with 20mn games, the logs result in a considerable amount of data.

We focus on learning the attacking strategies of the opponent team, and define episodes of interest, as the sections of the log where the ball possession is awarded to the opponent team and a passing motion is about to be executed. We extract a set of trajectories corresponding to such logs, where we define a trajectory as a sequence of time-stamped points in 2D space. The problem is thus reduced to finding common patterns in a set of trajectories observations.

We formalize the notion of similarity between sets of trajectories and then contribute a hierarchical clustering technique to identify similar patterns. The key idea we build upon is that trajectories can be represented as images and thus, shape matching algorithms can be utilized in the similarity metric. We show that our method can indeed extract a small set of multi-robot strategies of opponent teams in real games we played. We then show how we can use the learned behavior clusters to predict the selected strategy from partial observations incrementally over time. Finally, we present how we can autonomously generate counter tactics, as we recognize the opponent pattern before it reaches an end. We present experiments simulating three opponent teams from games in RoboCup'2010 playing against CMDragons with the new counter tactics, showing their effectiveness, under the assumption the opponent team maintains its strategy.

2 Related Work

Opponent modeling has been widely studied in a variety of games. In particular, in the RoboCup simulation soccer, the Coach competition led to many successful efforts towards the use of the learned opponent models as coaching advice before, and occasionally during the games. For example, given a predefined set of possible opponent models extracted from log data, classifiers are used to recognize a model, and the team effectively adapts to adversarial strategies [Riley and Veloso, 2006]. Support vector machines have been used to model defensive strategies in a simulated football game [Laviers *et al.*, 2009]. Models of opponents have been represented and learned as play formations, with combined events such as passes and shots [Kuhlmann *et al.*, 2006]. In the small-size robot league, fewer efforts model the opponent robots, and most have been sparsely applied in real games, if at all. Online adaptation was achieved by adjusting the weights of team plays as a function of the result of their application in a game, without the use of an explicit opponent model. Planning follows an expert approach that adapts to the opponent by probabilistically selecting plays based on their weights [Bowling *et al.*, 2004]. Offline learning from log data has been used to learn conditional random fields to capture multi-robot tactics, such as marking and defensive robot walls [Vail and Veloso, 2008]. In our work, we model the spatial and temporal behavior data of an opponent play as instances of geometrical trajectory curves, and learn the underlying patterns that govern executions of different strategies. We then find counter tactics to the trajectories recognized.

In the area of trajectory analysis, the Longest Common Subsequence model is shown to be robust to noise in multidimensional trajectories, with efficient approximations [Vlachos *et al.*, 2002]. Similarly, Dynamic Time Warping is a widely used algorithm that can robustly measure the similarity of trajectories with different temporal properties. In this paper, we compare trajectories using the Hausdorff distance metric [Rote, 1991]. Modifications for the Hausdorff metric have also been proposed to address different data updating policies and sampling granularities [Shao *et al.*, 2010].

For the clustering of the trajectories, we adopted an agglomerative hierarchical technique, where we provide an algorithm to create partitions from the hierarchical tree representation, as a modification of previous similar approaches [Kumar *et al.*, 2002], starting with each cluster as an element and building up to partitions by a series of merges.

3 From Logs to Multi-Robot Behaviors

In the RoboCup robot soccer small-size league, the vision processing by the overhead cameras produces the visual input to the central control computer at 60Hz, i.e., at 60 frames per second. For each frame f, the input contains: (i) time t_f, (ii) location, orientation and team of each robot r, <$x_{r,f}$, $y_{r,f}$, $\theta_{r,f}$, team$_r$>, (iii) location of the ball <$x_{b,f}$,$y_{b,f}$>, and (iv) the referee command r_f, which is a stop, a free kick, indirect kick, penalty, or kickoff (for a particular team), namely ‘S’,‘F’,‘I’,‘P’,‘K’, respectively. To change the state of the game, the referee issues a stop command, waits for the robots to relocate at least 500 mm from the ball and then rewards the ball possession to a team. CMDragons records the input in log files for further analysis [Zickler *et al.*, 2010].

A typical log file is composed of more than 70,000 entries for a 20mn game. We define an **episode** E = [t_0,t_n] as a time frame during which we analyze the behavior of an attacker team to deduce its offensive strategies. An episode starts at time t_0 when the ball possession is given to a team with a free kick or an indirect kick; next,, the offensive team has to make a pass or shoot at the defensive team’s goal according to the game rules. An episode ends at time t_n when the actuated ball goes out of bounds or is intercepted by the defensive team. Figure 1 depicts the trajectories of two offensive robots that move to receive a pass.

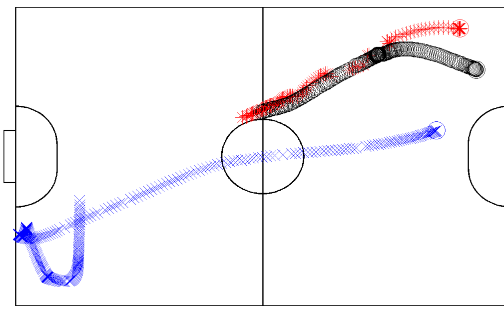

Figure 1: The trajectories of two robots that position for a pass and the ball (in red) is passed to one depicted in black.

We formalize the definition of an episode where team T is rewarded the ball possession. An episode E = [t_i,t_k] *begins* at time t_i if r_{i-1} is the stop command ‘S’ and r_i is a free or indirect kick, i.e. $r_i \in \{$‘F(T)’, ‘I(T)’$\}$.

To detect the *actuation* of the ball, we check if the distance between a robot and the ball is less than $(R+R_b+\epsilon)$ for some small value ϵ where R is the radius of a robot and R_b is the radius of the ball. So, at some time t_j, such that $t_j \in [t_i, t_k]$, and for some robot A, the following conditions must hold: (i) atan2$(y_{b,j} - y_{A,j}, x_{b,j} - x_{A,j}) = \theta_{A,j}$ (robot faces the ball), (ii) $((y_{b,j} - y_{A,j})^2 + (x_{b,j} - x_{A,j})^2) < (R + R_b + \epsilon)^2$ (robot actuates the ball) and (iii) $team_A = T$.

An episode *ends* at time t_k if one of the following conditions are fulfilled: (i) $((y_{b,j} - y_{B,j})^2 + (x_{b,j} - x_{B,j})^2) < (R+R_b+\epsilon)^2$ and $team_B \neq T$ (ball interception by defense) or (ii) $|x_{b,t}| > 3m$ or $|y_{b,t}| > 2m$ where the field coordinates are within [-3,-2] by [3,2] in meters (ball is out of bounds).

This definition excludes candidate cases where although a kick command is given, one of the following complications arises: (i) the actuator robot moves the ball without its kicker, (ii) a stop command is given before the end conditions are met, or (iii) neither team responds to the command.

In this work, we focus on our games with teams TeamA, TeamB and TeamC in RoboCup 2010. From the three games, we detected 136 episodes and excluded 33 candidates. Our team was defending in 30, 23, and 16 episodes against TeamA, TeamB and TeamC. The mean length of the 69 episodes when our team was in defensive state is 3.946 seconds with standard deviation 1.554 seconds.

A robot A is qualified as an **active agent** in the time frame

$[t_i, t_k]$ if, for some $t_j \in [t_i, t_k]$, one of the following conditions holds true: (i) $((y_{b,j} - y_{A,j})^2 + (x_{b,j} - x_{A,j})^2) < (R + R_b + \epsilon)^2$ (robot actuates the ball) or (ii) $((G_A^y - y_{A,j})^2 + (G_A^x - x_{A,j})^2) > (D)^2$ where $< G_A^x, G_A^y >$ is the goal location of A's team and D is a constant for distance from the goal (robot is not a defensive robot). Figure 2 depicts the trajectories of three active robots in blue and of two defensive robots in black. From this point on, the behavior of a team is only described in terms of its active agents.

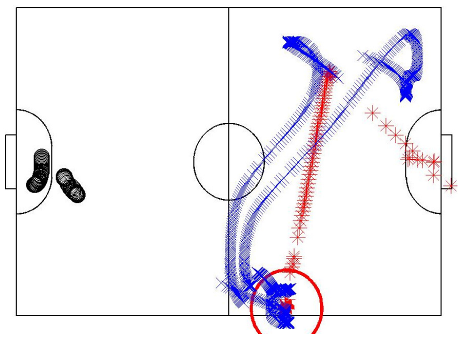

Figure 2: The trajectories of active and defensive robots in an episode. The trajectory of the ball is depicted in red.

Given a robot A and a time frame $[t_i, t_k]$, we define a **trajectory** as $T_A(t_i, t_k) = \{(x_{A,i}, y_{A,i}) \ldots (x_{A,k}, y_{A,k})\}$. The **behavior** of a team with a set of active robots $\mathbb{A}$, during $[t_i, t_k]$, is the set of the trajectories of each robot: $S(t_i, t_k) = \cup_{A \in \mathbb{A}} T_A(t_i, t_k)$. If a behavior $S(t_i, t_k)$ is observed during an episode $E_j = [t_i, t_k]$, we may also denote it as S_j.

4 Learning Behavior Patterns

Let $\mathbb{E}$ be the set of episodes and let S^{E_i} be the behavior of A during episode $E_i \in \mathbb{E}$. Let $\mathbb{S}_A(\mathbb{E}) = \cup_{E_i \in \mathbb{E}} S_A^{E_i}$ be the set of all our observations of team A. We define a **behavior pattern** of team A, $\mathbb{P}_{\mathbb{A}}$, as a cluster of similar sets of trajectories, such that $\mathbb{P}_A \subset \mathbb{S}_A(\mathbb{E})$. Intuitively, a behavior pattern $\mathbb{P}_A$ represents the set of executions of a play during different episodes. Figure 3 demonstrates the trajectories of active robots in two episodes where they perform similar behavior. Next we formalize the notion of similarity between sets of trajectories.

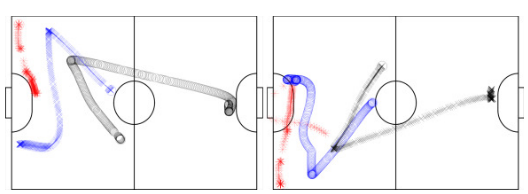

Figure 3: The similar behavior of robots in two episodes, that are mirror images of each other horizontally.

4.1 Similarity between Sets of Trajectories

A similarity function between two sets of trajectories has to be built on a similarity metric or algorithm that evaluates a pair of trajectories. In our work, we have chosen the Hausdorff metric for its generality and efficiency [Rote, 1991]. Given two trajectories T_1 and T_2 with m and n points respectively, the Hausdorff distance is defined as:

$$H(T_1, T_2) = \max\{\max_{p \in T_1} \min_{q \in T_2} d(p, q), \max_{p \in T_2} \min_{q \in T_1} d(p, q)\}$$

where *d(p,q)* is the Euclidean distance between points p and q. The computation time is O(mn).

Let S_1 and S_2 be two sets of n trajectories during episodes E_1 and E_2 respectively, and we want to compute their similarity, $Sim(S_1, S_2)$. We denote the ith trajectory in a set S as $S[i]$. Let P^n be the set of all of the permutations of $\{1 \ldots n\}$. Any permutation $P \in P^n$ represents a matching between the trajectories $S_1[i]$ and $S_2[P(i)]$ for $1 \leq i \leq n$. The similarity between S_1 and S_2 is the minimum sum of pairwise distances between the trajectories of S_1 and S_2 across all possible matchings:

$$Sim(S_1, S_2) = \min_{P \in P^n} \sum_{i=1}^{n} H(S_1[i], S_2[P(i)]).$$

We additionally compute the similarity between S_1 and symmetries of S_2 about the horizontal and vertical axes of the field. If we let *Flips* be the set of functions where each function returns a symmetric match of the input, the revised similarity function becomes:

$$Sim(S_1, S_2) = \min_{F \in Flips} \min_{P \in P^n} \sum_{i=1}^{n} H(F(S_1[i]), S_2[P(i)]).$$

Given a distance measure, we use a variant of agglomerative hierarchical clustering (AHC) due to two challenges imposed by our datasets. First, due to the small size of our data with 20 to 100 elements, outliers can significantly affect the results of partitional algorithms. Second, the variance in the densities of clusters we obtain from hierarchical algorithms show that such data can not be successfully clustered by density-based algorithms.

Upon executing AHC on the data set, we analyze the nodes bottom-up in the tree, and merge sets as long as their size is less than a variable *maxClusterSize* defined as a function of the dataset. If a final set size is less than another variable *minClusterSize*, then we mark it as an outlier.

Figure 4 presents the hierarchical clustering of the offensive sets of trajectories of TeamA. The figures above the tree depict the sets of trajectories of particular episodes (labeled with episode numbers). The analysis of the tree yields the clusters depicted with the red boxes around the leaves. For instance, trajectory sets 17, 22 and 24, belong to the same cluster as the movements of the offensive robots along with the ball trajectory are more similar to each other than to those in the rest of the data set.

4.2 Experimental Results

We provide experimental results that demonstrate that our learning algorithm can indeed find the behavior patterns of a team by observing its game play. We evaluate our work on real game logs, testing whether we can deduce patterns from both our opponents' and our own game play.

To quantify the quality of the clusterings obtained, we compare the results with the clusterings generated by ten people

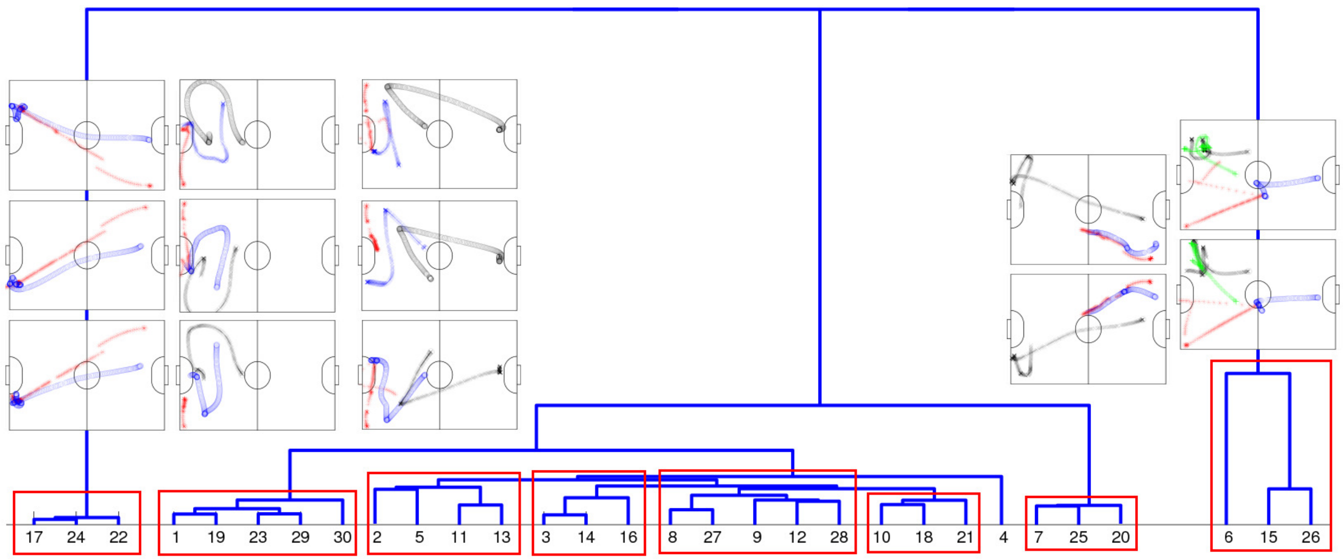

Figure 4: A sample clustering of sets of robot trajectories into behavior patterns. The red boxes represent the clusters obtained by hierarchical clustering. The trajectory sets in the same cluster are instances of the same behavior pattern.

classifying the behavior patterns during the game episodes. The comparison of clusterings is based on the Rand index, an objective criteria frequently used in clustering evaluation. For two clusterings C_i and C_j of n elements, we define two values p_{same} and p_{diff}. p_{same} is the number of elements in the same cluster and p_{diff} is the number of elements in different clusters in both C_i and C_j, with Rand index $(p_{same} + p_{diff})/\binom{n}{2}$. If the Rand index is 1.0, then the two clusterings are the same.

Table 1 summarizes the clusterings obtained in our experiments by providing the average Rand Index computed between the output of our AHC algorithm and the ten human clusterings. The *minClusterSize* is set to 2. The *maxClusterSize* is set to 1/3 of the total size of the dataset. Note that we uniformly sample the trajectories with 10:1 ratio before we compute the Hausdorff distance between them.

Table 1: Clustering Results in RoboCup Games

Team	Episodes	Clusters	Ave. Rand Index
CMDragons	100	11	0.96
TeamA	30	8	0.87
TeamB	23	4	0.91
TeamC	16	3	0.94

5 Responding to Behavior Patterns

After learning the clusters corresponding to behavior patterns, we now focus on the online recognition of an opponent behavior pattern and on a response to it.

5.1 Online Classification

Let $\mathbb{C}$ be a clustering of sets of trajectories such that $\mathbb{C} = \{c_1 \ldots c_n\}$ and let each cluster c_i be denoted as multiple sets of trajectories: $c_i = \{S_{i,1} \ldots S_{i,m_i}\}$ where m_i is the size of c_i. Let t be the current time. Let S_p be the *partial* set of trajectories that has been recorded since time t_0 when the beginning of an episode E_p was detected. Let t_n be the end of the episode, such that $t_n > t$. The goal is to determine the cluster into which $S_p(t_0, t)$ would be placed if it is observed for the entire episode duration in $[t_0, t_n]$.

The distance between a set of trajectories S_p and some cluster c_i in $\mathbb{C}$ is the mean of the similarity values between S_p each $S_{i,j}$ in c_i. To compare the partial set of trajectories S_p with a complete set of trajectories $S_{i,j}$, we limit the duration of $S_{i,j}$ to that of S_p. Formally, the similarity between S_p and any complete set of trajectory $S_{i,j}$ in any cluster c_i during an episode $E = [t'_0, t'_n]$ is computed as $Sim(S(t_0, t), S_j(t_0, min(t'_n, t'_0 + t - t_0)))$.

Figure 5 illustrates how the comparison of a partial set of trajectories S_p with two complete sets A and B would proceed in time. As the observation duration t increases, more points from the trajectories of A and B, depicted in blue, are used in the similarity function.

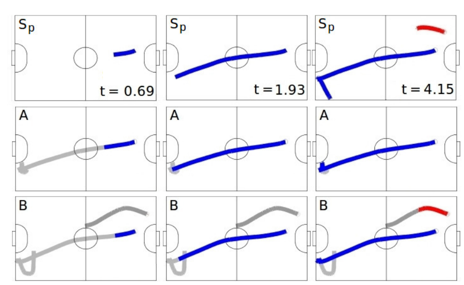

Figure 5: Incremental partial comparison of pattern S against patterns A and B. By $t = 4.15$, B is correctly identified.

To evaluate the effectiveness of our online classification, for each team and for each set of trajectories in the dataset, we first compute a clustering excluding that set; then determine the final behavior pattern it would be classified into if it is observed entirely; and last, test what percentage of the set of trajectories should be observed to classify it in its final behavior pattern. Figure 6 presents the effect of different observation percentages on the correct classification of sets of trajectories for several teams.

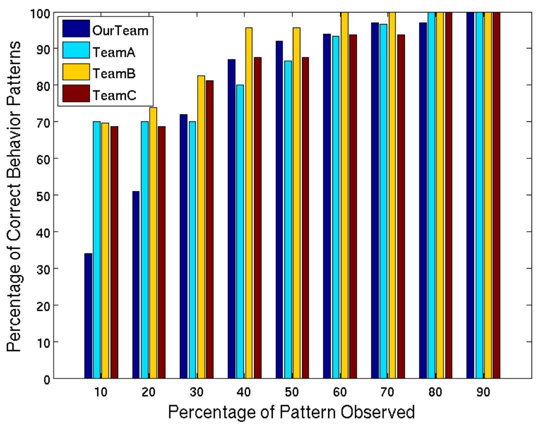

Figure 6: Graph depicting the effect of observation duration of partial patterns of a team on their correct classification.

For instance, for each team, it is sufficient to observe the beginning 30% of its episodes to identify with 70% success rate the behaviors with the correct behavior patterns.

5.2 Non-Responsiveness Attacking Assumption

Given that we can correctly recognize offensive strategies in the earlier stages of their executions, our goal is to take preemptive actions that counter the attack. As there is no feasibility at this time to experiment with the other teams, we perform experiments in simulation. We interestingly extended our simulation to be able to play an opponent team by replaying the log data captured from a past game. As we investigate how to change the behavior of our team to counter act the offensive play of the opponent team, we make a *Non-Responsiveness* attacking assumption, in the sense that the opponent teams would not change in the presence of our counter act. Our assumption is based on our extensive analysis of games that leads to the conjecture that teams are non-responsive to the behavior of their opponents in their offensive strategies because they commit to predefined plans.

As an anecdotal example of our conjecture, Figure 7 illustrates two real episodes of the same attacking pattern that does not change even when the defense changes.

Robots R1 and R2 are attacking robots that perform the same two trajectories even if the defense robot R3 executes two considerably different trajectories.

5.3 Action Selection

Given a set of offensive trajectories, the goal is to adapt the defensive behavior to intercept the ball after its first actuation.

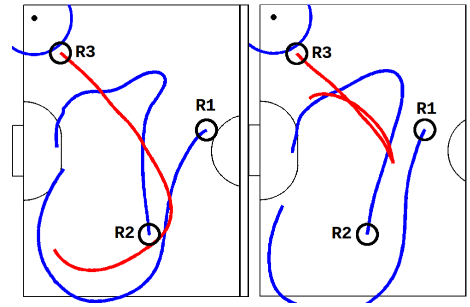

Figure 7: Example of the execution of a pre-defined attacking behavior (blue robots R1 and R2) non responsive to the trajectory of the defense (red robot R3).

To this end, we modify the behavior of a single robot with a different action. An action is defined with three fields: (i) id, the identification of the executor robot, (ii) t, the time the action will take place; (iii) loc, the location the executed robot should move to.

Let H denote the online history of states as they will be logged such that $H[0]$ is the current state. Let C be the set of learned behavior patterns. Given H and C, the **SelectAction** algorithm returns a preemptive action with the aforementioned parameters, if one exists.

Algorithm SelectAction
Input: H: History of states; C: Set of behavior patterns
Output: A preemptive action
1. $\mathbb{P}$ ←onlineClassify(H, C);
2. **if** $\mathbb{P}$ = null, **then return** null;
3. ballTraj ←*getEstBallTraj*($\mathbb{P}$);
4. minTime ←∞; bestRobot ←-1;
5. **for each** defensive robot R_i^d
6.   [tempTime, tempLoc] ←*simulate*(H, R_i^d, ballTraj);
7.   **if** tempTime $<$ minTime, **then**
8.   minTime ←tempTime;
9.   bestRobot ←R_i^d;
10.   bestLoc ←tempLoc;
11.   **end if**
12. **end for**
13. ballTime ←*getBallTravelTime*(H, bestLoc);
14. **if** ballTime $<$ minTime, **then return** null;
15. **else return** Action(bestRobot, H[0].time, bestLoc);

The function *onlineClassify(H,C)* classifies a set of offensive trajectories S observed in H, with the most similar behavior pattern $\mathbb{P} \subset C$ if their similarity, $Sim(S, \mathbb{P})$ is greater than some threshold. If the online classification fails, SelectAction returns null. Given a behavior pattern $\mathbb{P}$, that is a cluster of similar sets of trajectories, the function *getEstBallTraj* returns the ball trajectory observed in the duration of the *centroid* of $\mathbb{P}$. Given the online history H, the function *getBallTravelTime*(H, loc) returns the time the ball takes to move from its current location to loc. The function *simulate*(H, R, ballTraj) returns the time a robot R takes to reach the trajectory ballTraj from its current location.

Intuitively, the algorithm classifies an observed offensive pattern (lines 1-2); obtains an estimate ball trajectory (line

3); determines the closest robot to intercept the ball (lines 4-10) and checks whether it can reach the interception location before the ball does (lines 11-13).

5.4 Experimental Results

We ran experiments on real game data in the following manner. For each team, we first let our system process the previous games from the logged files, learning the behavior patterns. Second, we simulate a new game where we run our system as usual, with the exception that visual data of opponents is from the log files. Only the detected episodes are replayed based on the *Non-Responsiveness* assumption. For each episode, we observe if the **SelectAction** algorithm intercepts the passes.

In some episodes, the pass from the actuator does not reach a receiver robot due to an interception by the other team or to the inaccuracy of the actuator. Regardless, we ignore those cases since from the log files, we can not actually simulate the movement of the ball and create a collision by intercepting the ball. Table 2 presents, for each team, the number of episodes during which we previously could not take successful counter actions but now do.

Table 2: Intercepted Passes in Log Simulations

Team	Episodes		Success (%)
	Un-responded	Stopped	
TeamA	26	21	80.7
TeamB	13	10	76.9
TeamC	15	11	73.3

A detailed analysis of the failed interception cases reveals that there are two sources of error: (i) inability of the size of the database to capture every case and (ii) the late classification of partial trajectory sets into the right clusters. In the second case, even though the right optimal action is chosen, the robots do not have enough time to execute it.

In the real games, our team stopped 4, 10, and 1 attacking behaviors out of 30, 23, and 16 episodes against teams A, B, and C, respectively. With the new counter tactics, we stop a total of 25, 20, and 12 episodes, respectively. In summary, the results show that we can identify the strategy of an opponent and successfully take counter actions for more than 70% of the time.

6 Conclusion

In this paper, we contributed the learning of behavior patterns of real multi-robot systems from temporal and spatial data. We introduced a way of interpreting the sets of robot trajectories as geometric curves, which we compared using the Hausdorff distance. We then showed a variant of a hierarchical clustering algorithms that uses the Hausdorff-based similarity metric to successfully learn behavior patterns. Finally, we provided an algorithm that maps a new behavior pattern as it develops over time into a learned cluster, and we created preemptive actions to counter act the recognized opponent action. We used extensive log data from real games and performed experiments that showed the effectiveness of our behavior learning from logs, as well as the recognition and counter acting in simulation experiments. To the best of our knowledge, our work is the first application of trajectory matching techniques to opponent modeling in adversarial multi-robot domains. We plan to follow three directions for future work, namely to analyze opponent defensive strategies, to investigate predictive models of behavior variations for the opponent team, and to apply our behavior recognition and counter acting in real games.

Acknowledgments

We would like to thank the outstanding work of the CMDragons'2010 research team, namely Stefan Zickler and Joydeep Biswas, for all the algorithms and implementation, including vision, logging, simulation, and robot behaviors. We also thank Michael Licitra for designing and building the robots, as well as the members of earlier CMDragons teams. Without those researchers, this work would not have been possible.

References

[Bowling *et al.*, 2004] M. Bowling, B. Browning, and M. Veloso. Plays as effective multiagent plans enabling opponent-adaptive play selection. In *ICAPS*, 2004.

[Browning *et al.*, 2005] B. Browning, J. Bruce, M. Bowling, and M. Veloso. STP: Skills, tactics and plans for multi-robot control in adversarial environments. In *JSCE*, 2005.

[Kuhlmann *et al.*, 2006] G. Kuhlmann, W. Knox, and P. Stone. Know thine enemy: Champion RoboCup coach agent. In *Proceedings of AAAI*, 2006.

[Kumar *et al.*, 2002] M. Kumar, N. Patel, and J. Woo. Clustering seasonality patterns when errors. In *KDD*, 2002.

[Laviers *et al.*, 2009] K. Laviers, G. Sukthankar, M. Molineaux, and D. W. Aha. Improving offensive performance through opponent modeling. In *AIIDE*, 2009.

[Riley and Veloso, 2006] P. Riley and M. Veloso. Coach planning with opponent models. In *AAMAS*, 2006.

[Rote, 1991] G. Rote. Computing the minimum Hausdorff distance between two point sets on a line under translation. In *IPL, Volume 37*. Elsevier North-Holland Inc, 1991.

[Ruiz-del-Solar *et al.*, 2010] J. Ruiz-del-Solar, E. Chown, and P. Ploeger, editors. *RoboCup 2010*. Springer, 2010.

[Shao *et al.*, 2010] F. Shao, S. Cai, and J. Gu. A modified Hausdorff distance based algorithm for 2-dimensional spatial trajectory matching. In *ICCSE*. IEEE, 2010.

[Vail and Veloso, 2008] D. Vail and M. Veloso. Activity recognition in multi-robot domains. In *AAAI*, 2008.

[Vlachos *et al.*, 2002] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *18. ICDE*. IEEE, 2002.

[Zickler *et al.*, 2009] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso. SSL-vision: The shared vision system for RoboCup SSL. In *RoboCup*, 2009.

[Zickler *et al.*, 2010] S. Zickler, J. Biswas, and M. Veloso. CMDragons team description. In *RoboCup*, 2010.