

CMDragons 2014 Extended Team Description

Joydeep Biswas, Juan Pablo Mendoza, Danny Zhu,
Steven Klee, and Manuela Veloso

Carnegie Mellon University
{joydeepb,jpmendoza,dannyz,mmv}@cs.cmu.edu, sdklee@andrew.cmu.edu

Abstract. In this paper we present an overview of CMDragons 2014, Carnegie Mellon University’s entry for the RoboCup Small Size League. Our team builds upon the research and success of RoboCup entries from previous years.

1 Introduction

Our RoboCup Small Size League entry, CMDragons 2014, builds upon the ongoing research used to create the previous CMDragons teams (1997–2003, 2006–2010, 2013 [1]) and CMRoboDragons joint team (2004, 2005). Our team entry consists of six omni-directional robots controlled by an off-board computer. Sensing is provided by the shared vision system [2]. The software then predicts the state of the world, evaluates team coordination behaviors, and finally sends driving commands to the individual robots. This paper describes the the off-board control software required to implement a robot soccer team. We focus on the novel contributions compared to the state of the team in previous years; the overall architecture and robot hardware have remained largely unchanged since 2010, and the reader is referred to past entries [3, 1] for a hardware overview and detailed description of the software architecture. In this paper we describe a novel Coerce and Attack Planner [4] that *coerces* opponents into leaving strategic openings in the defense, and then exploits such openings in the *attack*. We further present our work on pass-ahead planning for dynamic passing between robots, and a new threat-based defense system for defending against fast passes.

2 The Robots and Basic Skills

The CMDragons team comprises 12 identical robots¹ based on the designs of the SSL robots of CMDragons from 2006. We replicated the mechanical designs and replaced the electronics main board with a newer design to create a team of 12 robots. While the actual games at RoboCup are played by teams of 6 robots, having 12 robots allowed us to test our software in the lab in full games prior to the competition. Figure 1 shows the internals of the robots, including the electronic board and driving and kicking mechanisms.

¹ Thanks to Michael Licitra for designing the mechanical designs, and for designing and fabricating the electrical designs for the robots.

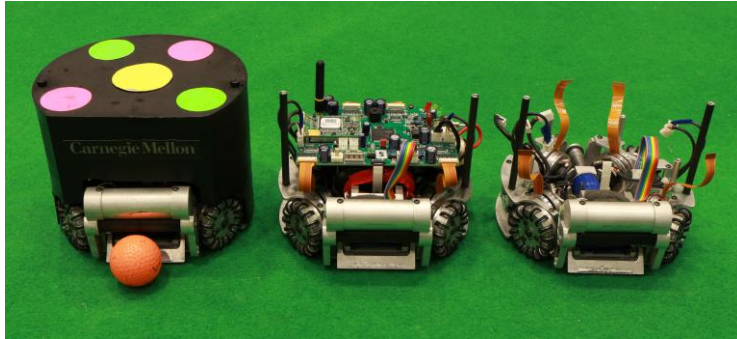


Fig. 1. The CMDragons robots, showing (left) a robot with the ball, (middle) without the cover, and (right) without the electronic main board.

One of the basic skills for robots in the SSL is the ability to drive to a target location, starting from an arbitrary start location with an arbitrary starting velocity. We use a near-time optimal trajectory planner [5], implemented as the function $\langle t^*, \mathbf{V}^* \rangle = \text{CalcMotion2D}(\mathbf{x}_s, \mathbf{v}_s, \mathbf{x}_f)$ to compute the sequence of velocity commands \mathbf{V}^* and the total time t^* required to navigate from initial location \mathbf{x}_s and initial velocity \mathbf{v}_s , to a final location \mathbf{x}_f and zero final velocity.

Using the near-time optimal trajectory planner, we can plan for intercepting moving balls. The problem of dynamic ball interception requires computation of *where* along the trajectory of the ball a robot can intercept it, and *how* to intercept it. The question of where to intercept the ball is determined by where the robot can drive to sooner than the ball can reach, and the question of how to intercept it is governed by the relative location of the intercept with respect to the kicking target location. The computation of the optimal ball intercept location is complicated because the function CalcMotion2D does not have an analytic form, so the optimal interception location can only be evaluated numerically. For a future ball location p_ball along the trajectory of the ball, we can compute

1. the ball travel time to reach p_ball based on the carpet model: $t_ball(p_ball)$,
2. the robot intercept location based on the target location: $intercept(p_ball)$,
3. the robot travel time: $t_robot(intercept(p_ball))$,
4. the slack time: $slack = t_ball - t_robot$, and hence
5. whether the intercept will be successful: $slack \geq 0$.

This sequence of computations is performed for discrete future locations of the ball along its trajectory to compute the optimal intercept location by a linear search. There are two types of ball intercept locations, the *minimum time intercept*, given by

$$\arg \min_{intercept} (t_ball) : slack \geq 0, \quad (1)$$

and the *maximum slack intercept*, given by

$$\arg \max_{\text{intercept}}(\text{slack}) : \text{slack} \geq 0. \quad (2)$$

The minimum time intercept is the location where the robot could intercept the ball fastest, whereas the maximum slack intercept is the location where interception will be most robust to execution errors due to the available slack time. Therefore, the minimum time intercept is used when the cost of failure is low, like an attacker opportunistically trying to intercept and shoot on the goal, while the maximum slack intercept is used when the cost of failure is high, like the primary defense trying to block an opponent’s shot on the goal.

3 Passing

Passing the ball between teammates is by far the most common method for creating goal opportunities in a controlled way. This section describes how CM-Dragons performs passes with coordination between the passing robot P and the receiving robot R . First, all potential receivers search for locally optimal locations $\mathbf{x}^* \in \mathbb{R}^2$ in the field to receive a pass from P , as described in Section 3.1. Then, P evaluates all potential receivers and chooses the best to perform a coordinated pass. Finally, P and the chosen receiver R coordinate their passing and receiving maneuvers to minimize the opponents’ opportunity to prevent a successful pass, as described in Section 3.2.

3.1 Pass Location Selection

When searching for the best location to receive a pass, our algorithm attempts to maximize the probability of scoring a goal if passer P were to pass to R . That is, we define \mathbf{x}^* as:

$$\mathbf{x}^* \equiv \arg \max_{\mathbf{x} \in \mathbb{R}^2} [P(\text{goal} | \mathbf{x})] \quad (3)$$

Notice that we can divide the probability on the right into two factors: the probability of R successfully receiving the ball at location \mathbf{x} , and the probability of R successfully scoring a goal from \mathbf{x} given that it has received the ball:

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathbb{R}^2} [P(\text{receive} | \mathbf{x})P(\text{goal} | \text{receive}, \mathbf{x})] \quad (4)$$

Since the SSL domain is high-dimensional, highly dynamic, and adversarial, it is unrealistic to expect to compute the two probabilities above exactly. However, the function we actually maximize attempts to approximate these probabilities in a computationally feasible way. We define a set of important conditions c_i that must be true for R to receive a pass at location \mathbf{x} and successfully score on the goal. We also assume the events to be independent to simplify computation. The approximating function is thus defined as:

$$\hat{P}(\text{receive} | \mathbf{x}) \equiv \prod_i \hat{P}(c_i | \mathbf{x}). \quad (5)$$

For R to successfully receive a pass, all c_i need to be true, and $\hat{P}(c_i|\mathbf{x})$ is an approximation to the probability that c_i will be true given \mathbf{x} . The conditions c_i we consider are:

- **c_1 : No opponent can reach \mathbf{x} faster than R can.** $\hat{P}(c_1|\mathbf{x}) \sim 0$ when an opponent can navigate to \mathbf{x} faster than R ; $\hat{P}(c_1|\mathbf{x}) \sim 1$ otherwise.
- **c_2 : No opponent intercepts the pass.** $\hat{P}(c_2|\mathbf{x}) \sim 0$ when an opponent can navigate to a point along the line between the origin \mathbf{x}_0 of the pass and its destination \mathbf{x} faster than the ball can get from \mathbf{x}_0 to that point, considering passing speed; $\hat{P}(c_2|\mathbf{x}) \sim 1$ otherwise, as visualized in Figure 2.

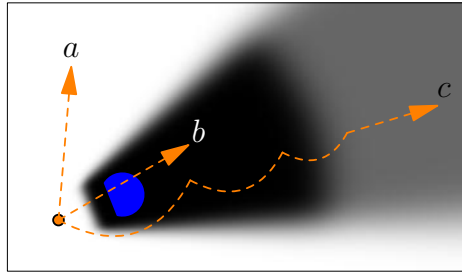


Fig. 2. The estimated probability that a pass from the ball location (orange circle) will not be intercepted. This probability is high for locations with ball trajectories that pass far from opponents, such as a , and low for those with ball trajectories that pass close, such as b . Some passes, such as c , pass close to the opponent but can still be successful using chip passes, although the prior success probability for those is lower than for regular passes, as indicated by the gray region to the right.

- **c_3 : The pass is long enough for R to react and receive the pass robustly.** $\hat{P}(c_3|\mathbf{x}) \sim 0$ when the time the ball would take to travel from \mathbf{x}_0 to \mathbf{x} is less than a minimum reaction time t_{\min} ; $\hat{P}(c_3|\mathbf{x}) \sim 1$ otherwise.
- **c_4 : The pass is short enough to be performed accurately.** $\hat{P}(c_4|\mathbf{x}) \sim 0$ when $|\mathbf{x} - \mathbf{x}_0|$ is greater than a maximum distance d_{\max} ; $\hat{P}(c_4|\mathbf{x}) \sim 1$ otherwise.
- **c_5 : Location \mathbf{x} is reliable for pass reception.** $\hat{P}(c_5|\mathbf{x}) \sim 0$ when \mathbf{x} is too close to the defense area, entrance into which is forbidden by the rules of SSL, to the boundary of the field, where passes run the risk of going out of bounds, or to other teammates, where teammates could interfere with R ; $\hat{P}(c_5|\mathbf{x}) \sim 1$ otherwise.

An analogous approximation is computed for the probability of scoring a goal from location \mathbf{x} given that the pass has been received. In this case, $\hat{P}(\text{goal} | \text{receive}, \mathbf{x})$ is a product of probabilities of the following conditions c'_i :

- **c'_1 : Shots from \mathbf{x} can reach the opposing goal faster than their goalkeeper can block them.** $\hat{P}(c'_1|\mathbf{x}) \sim 0$ if the shot time is greater than

the time t_g the opposing goalkeeper takes to block an arbitrary point on the goal; $\hat{P}(c'_1|\mathbf{x}) \sim 1$ otherwise.

- **c'_2 : There is a wide enough open angle θ_g from \mathbf{x} to the opposing goal.** $\hat{P}(c'_2|\mathbf{x}) \sim p_{g \min}$, for a constant prior $p_{g \min}$, when $\theta_g = 0$ ($p_{g \min} > 0$ because an angle may open up as robots move), and $\hat{P}(c'_2|\mathbf{x}) \rightarrow 1$ as $\theta_g \rightarrow \theta_{\max}$. When $\theta_g > \theta_{\max}$, $\hat{P}(c'_2|\mathbf{x}) = 1$, indicating that beyond a certain threshold, the value of θ_g has no influence on the probability of scoring. Figure 3 shows a visualization of $\hat{P}(c'_2|\mathbf{x})$.

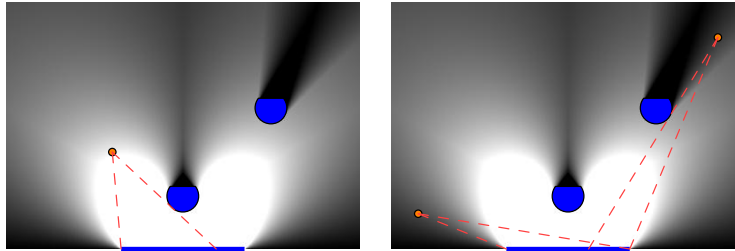


Fig. 3. The estimated probability that a given location \mathbf{x} has a wide enough open angle on the opponents' goal (blue line) to score a goal. The left image shows a location with a wide open angle, ideal for a shot. The right image shows two locations with relatively small open angles, one due to obstruction by a robot and distance from the goal, and the other because of its location near the corner of the field.

- **c'_3 : R will have enough time to take a shot before the opponents block the shot.** $\hat{P}(c'_3|\mathbf{x}) \sim 1$ for locations where R can do a one-touch shot on the goal, while $\hat{P}(c'_3|\mathbf{x}) \sim p_{turn} < 1$, for some constant prior probability p_{turn} , when the robot needs to receive the ball, turn, and then shoot (only a two-touch shot is possible).
- **c'_4 : R will have enough time to take a shot before opponents steal the ball.** $\hat{P}(c'_4|\mathbf{x}) = 1$ for locations inside the opponents' defense area, where their defenders are not allowed to enter, while $\hat{P}(c'_4|\mathbf{x}) = p_{out} < 1$, for some constant prior probability p_{out} , when \mathbf{x} is outside of the opponents' defense area.

Equation 5 and its analogue for $\hat{P}(\text{goal} | \text{receive}, \mathbf{x})$ provide a value function for all potential receiving locations; the search for \mathbf{x}^* is simply conducted by random sampling and evaluation of points. This is feasible since the space to search is a relatively small 2D space. Furthermore, at each time step, only locations close to the previous optimal are searched, to avoid big jumps in the target destination of R .

3.2 Pass-ahead Coordination

Following the receiver robot's selection for location \mathbf{x}^* , passer P and receiver R coordinate the pass so that the passed ball and R arrive at \mathbf{x}^* at approximately

the same time. P thus *passes ahead* to where R will be, rather than passing to where R is. The purpose of this coordination is to minimize the window of time in which the opponents can predict and block threats from the chosen location \mathbf{x}^* .

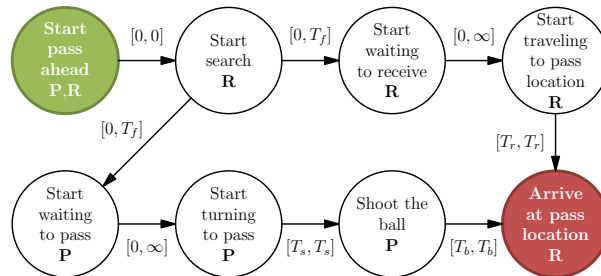


Fig. 4. A Simple Temporal Network (STN) for pass-ahead. The letters inside each node indicate whether the passing agent (P) or receiving agent (R) is involved in that event. T_f indicates a maximum allowable time to search for a pass location.

This coordination can be clearly visualized using a Simple Temporal Network (STN) [6]. Figure 4 illustrates the STN with the time constraints for passing ahead. Note that there are some constraints where a robot could potentially wait indefinitely. However, we are interested in the lowest achievable time bounded by these constraints. In pass-ahead planning, we use time constraints as part of our multi-agent plan representation [7].

Computations for pass-ahead rely on the ability to accurately estimate robot navigation time to a given location and orientation, as described in Section 2. They also require an accurate estimate of the time the ball will take to traverse a specified distance when it is imparted with a specific initial velocity. This computation is based on a two-phase (sliding then rolling) model of the ball’s trajectory. These models provided the necessary accuracy to succeed at coordinated passing ahead; such low-level coordination, combined with the higher-level planning of Section 5, led to the success of our multi-agent attacks in RoboCup 2013.

4 Human Coaching

Since RoboCup 2013, we have focused on improving our passing function with human input. Specifically, we try to use machine learning and case-based reasoning techniques to rank receiver locations. We have tried three approaches based on Gaussian naive Bayes, support vector rankings, and Model Plus Correction.

4.1 Gaussian Naive Bayes

The first approach we considered was Gaussian naive Bayes. Our 2013 passing function can easily be framed in the context of a Bayesian network. However, there was no automated process for determining a probability distribution of each event. In practice, these distributions were hand-tuned to optimize performance. So, while the robots perform well in practice, this told us little about the accuracy of our model. We wanted to see the effect of computing these values from our logs.

To do so, we looked through all of the available logs from RoboCup 2013 games. At each free-kick that lead to a goal, we recomputed the values of several of the CMDragons 2013 heuristics. After removing outliers, we computed the means and variances of the distribution of each event c_i, c'_i . Then, we assumed a Gaussian distribution.

4.2 Support Vector Machine

We contrasted the Gaussian naive Bayes approach with a support vector ranking. One advantage that this method has over Gaussian naive Bayes is that redundant features are given less weight. If strong independence was a bad assumption, we would expect this to perform much better.

We implemented two support vector rankings. In the first, we used a linear hypothesis function where our features were all of the events c, c' from before. In the second, we used a degree-2 polynomial kernel. By doing this, we can directly represent dependencies between the heuristics. In both support vector rankings we used a logistic loss function with regularization. Table 1 shows the results of both rankings on a testing set.

	Training Set	Testing Set
Linear Model	70%	74%
Quadratic Kernel Model	81%	59%

Table 1. Performance of support vector rankings on a training set of 195 examples, and a testing set of 50 examples.

As shown in Table 1, the linear model performed worse on the training set but better on the testing set. One would expect the quadratic polynomial kernel to perform as well or better than the linear model. We believe that this was caused by an insufficient amount of training data. A good rule of thumb is to have ten pieces of training data per feature. At 10 features, the degree-2 polynomial kernel would require $10 \cdot \binom{10}{2} = 450$ features. Unfortunately, we did not have enough logs of past games to generate so many examples.

4.3 Comparison

We compared the Gaussian naive Bayes approach to the linear support vector ranking in simulation. Specifically, an automated referee ran trials where a full six-player team performed free kicks against three defenders. Trials ended when the ball went out of bounds, a goal was scored, or thirty seconds elapsed. We compare the two models to the original CMDragons 2013 model by comparing number of goals scored and the average length of a trial.

Technique	Original	Naive Bayes	SVR
Number of Trials	1000	562	2896
Number of Goals	153	70	298
Avg. Goals per Trial	15.3%	12.4%	10.3%
Avg. Trial time (Sec)	n/a	9.3	9.2

Table 2. Comparison of the performance of the CMDragons 2013 model, Gaussian naive Bayes model and linear support vector ranking model.

As seen in Table 2, the difference in goal rates between these 3 techniques is very small. Notably, the difference between the linear support vector ranking and the Gaussian naive Bayes model are not even statistically significant. The average time per trial also did not change by a practically significant amount.

We expected the support vector ranking to perform noticeably better than the other two models. We believe that given enough data, the support vector ranking with a degree-2 polynomial kernel would outperform these models. However, complex models require larger data sets.

4.4 Model Plus Correction

Since we have not gathered enough data from competitions to make a sufficiently complex model, we decided to look at what work has been done to remedy this problem in other domains. Notably, we found work on a paradigm called Model Plus Correction (MPC) [8]. MPC attempts to answer the question of how to augment an imperfect model with user input.

The most basic form of MPC uses case-based planning techniques atop a machine learning model. Intuitively, case-based reasoning is used to correct the machine learning model where it underperforms. More formally, we define a similarity function to compare the current state and each case in a case library. If the state matches one of those cases, we use the candidate solution. Otherwise, we use the machine learning model. Notably, a lot of work has been done on applying case-based planning techniques to RoboCup [9]. Typically, one of the largest challenges with CBR techniques in RoboCup is the large number of cases necessary to achieve reasonable performance. However, we hopefully need fewer cases to correct a prior model.

In our implementation, we used a case-based reasoning system in conjunction with our passing function from RoboCup 2013. For free kicks, we assume the receivers have infinite time to position themselves. So, our case similarity function depends on the location of the ball and the opponent robots. If the ball and opponents have locations b, o_1, \dots, o_n and a case has locations $b_c, o_{1c}, \dots, o_{nc}$ we define the similarity function:

$$\frac{\alpha \sum_{i=1}^n (o_i - o_{ic})^2}{n} + \beta (b - b_c)^2 \quad \alpha, \beta \in \mathbb{R}^+ \quad (6)$$

Two cases are similar if this function returns a low value. α and β are constants that affect the relative importance of differences in opponent locations and the ball's location. If a case has high similarity to the current situation we rank a location's cost as the square distance from the optimal point. Figure 5 shows an example of the MPC system when a case has been matched.

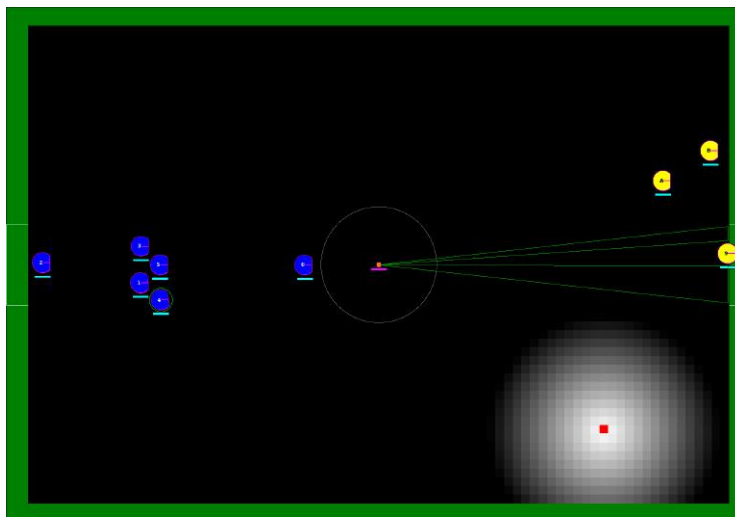


Fig. 5. A passing evaluation function is overlaid on top of the field in greyscale. Whiter values signify high-value locations. Values in black are less than 0.5% of the optimal value. The red square represents the optimal location chosen by the human coach.

This approach is not perfect. As more corrections are added, we expect to see improvement only up to a certain point. Eventually, we expect conflicting cases to be too similar to differentiate with our similarity function. For now, we have tried to strike a balance between an accurate measure of similarity and fast computation.

5 The Coerce And Attack Planner

We introduce a novel Coerce and Attack Planner (CAP) to plan attack sequences during free kicks. Since the opponents are not permitted to get closer than 50 cm to the ball until it is kicked, and since the free kick taker has 10 s to kick the ball, free kicks provide a convenient scenario for the team that is awarded the free kick to *plan* a progression of gameplay that might lead to a goal being scored. Additionally, there are certain characteristics of the defense that can be exploited to influence the plan. In general, there are two types of defending roles that the opponent robots may assume: “ball-following” roles and “robot-following” roles. The ball-following roles defend against direct shots on the goal from the ball, and hence are positioned as a function of the ball’s location. The robot-following roles, on the other hand, attempt to block passes, and hence follow the attacking robots.

The CAP relies on these characteristics to plan a coordinated attack when awarded a free kick. The CAP *coerces* robot-following opponents into positions that leave strategically advantageous openings, allowing a teammate to *attack* by moving into the opening, receiving a pass, and shooting into the opponent’s goal. The CAP interleaves planning, execution, and monitoring in the following sequence:

1. **Detect Opponent Tactics** (Monitoring): Tactic detection estimates the ball-following and robot-following tactic that each opponent robot is running.
2. **Compute Optimistic Attack** (Planning): Based on the detected tactics, the CAP computes an “optimistic attack” plan to score on the goal, considering only the ball-following opponents detected.
3. **Compute Coerce Plan** (Planning): Based on the detected tactics and the optimistic attack, the CAP computes a “coerce plan”, placing attacking robots to coerce opponents away from the optimistic attack.
4. **Execute Coerce Plan** (Execution): The coercing robots are moved into the planned positions.
5. **Verify Tactic Models** (Monitoring): The placement of the opponents in response is observed.
6. **Compute Attack Plan** (Planning): If the actual positioning of the opponents differs from the expected positions of the coerce plan, then a new “attack plan” is computed, else the previously computed optimistic attack is used as the attack plan.
7. **Execute Attack Plan** (Execution): The CAP then commands the robots to execute the attack plan.

During the free kicks, out of the team of 6 robots, one must be the goalkeeper, and one is required to take the free kicks. Hence, the CAP must reason about how many of the remaining 4 robots should be assigned to the coerce plan, and how many to the attack plan. This allocation varies based on the opponent tactics detected, and in some cases, robots may be re-used for the coerce plan as well as the final attack plan, as we now explain.

In step 3, the CAP uses the number of robot-following opponents detected from step 1, to allocate as many robots to the coerce plan. The remaining robots are allocated to the optimistic attack plan. If there are insufficient remaining robots to allocate exclusively to the attack plan, then robots allocated to the coerce plan are re-used during the attack. We illustrate the allocation of robots by the CAP in two example scenarios.

Example 1. If the CAP detects 3 robot-following opponents in Step 1, it allocates 3 robots to coerce them away during Step 4 from the optimistic attack plan. The CAP then allocates the remaining 1 robot on the team to execute the attack plan during Step 7 using pass-ahead.

Example 2. If the CAP detects 4 robot-following opponents in Step 1, it allocates all 4 robots to coerce the 4 robot-following opponents away during Step 4 from the optimistic attack plan. The CAP then reuses one of these 4 coercing robots to execute the attack plan during Step 7.

5.1 Tactic Detection

In order to determine which of the opponents are running ball-following tactics and which are running robot-following tactics, the CAP estimates the tactic controlling each opponent robot. The tactics that the CAP detects are:

- **Goalkeeper:** a ball-following tactic that stays exclusively within the defense area to block direct shots on the goal,
- **Primary Defense:** a ball-following tactic that stays on the perimeter of the defense area and always moves to cover the angle between the ball and the goal,
- **Mark:** a robot-following tactic that follows attacking robots to prevent them from receiving passes or shooting on the goal, and
- **Wall:** a robot-following tactic that stays as close as possible to the free kick taker to prevent it from passing to its teammates.

For every tactic t , given a world state W consisting of the locations of all the robots and the ball on the field, the model of the tactic M_t is used to compute the probability $P(p|M_t, W)$ that a robot positioned at location p would be running tactic t , modelled by M_t . In our work, $P(p|M_t, W)$ is computed analytically using assumptions of SSL-specific tactic behaviors and the rules of the SSL. A possible alternative would have been to estimate the probabilities numerically from logs [10, 11]. Figure 6 shows the probability distributions for the models of the tactics listed above. Let R be the set of opponent robots, and p_r denote the location of an opponent robot $r \in R$. The robots R_t running a tactic t for the current world state W are thus detected to be those robots in the set $R_t = \{r \in R | P(p_r|M_t, W) > \alpha_t\}$, where $\alpha_t \in [0, 1)$ is a threshold defined for detected tactic t . This means that the detected tactic for an opponent is the tactic that best explains its position on the field for the current world state.

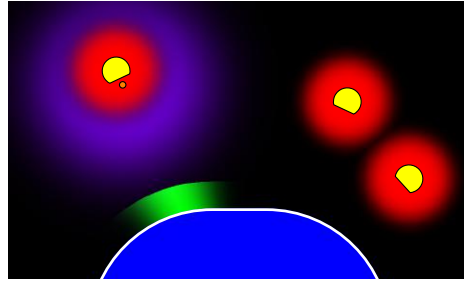


Fig. 6. The probability distributions given by the models for various tactics opponent robots might run, including Mark (red), Wall (purple), Primary Defender (green), and Goalkeeper (blue). The defense area line is shown in white, our robots in yellow, and the ball in orange.

5.2 Computing Attack Plans

There are two steps of the CAP that involve computing attack plans: step 2, when computing the optimistic attack, and step 6, when computing the final attack plan. When computing the optimistic attack plans, the only opponents taken into account are the ones detected (during step 1, opponent tactic detection) to be running ball-following tactics. When computing the final attack plan, all opponents are taken into account.

An attack plan consists of a pass from the free kick taker to a pass receiver at a specific location on the field, and a subsequent shot on the goal by the pass receiver. The possible locations of the pass receiver are evaluated by discrete sampling on a grid of size 6 cells by 4 cells spanning the entire field. While this discrete sampling could be coarser or finer, we empirically evaluated this discretization to be an acceptable tradeoff between computational complexity and the granularity of the resulting plans. Each cell on the grid is evaluated for a possible pass location as discussed in Section 3.1. The cell with the highest probability of a goal being scored from it is then chosen as the pass location for the attack plan.

5.3 Computing The Coerce Plan

Once the optimistic attack plan is computed, the coerce plan is computed on the grid to place robots to coerce robot-following opponents away from the optimistic attack plan. For the coerce plan, every cell on the grid is evaluated as follows:

1. Consider placing one attacking robot in the cell.
2. Based on the detected robot-following opponents, estimate where the robot-following opponents will drive to, in response to our robot being placed in this cell.
3. Evaluate the “interference likelihood”, defined as the likelihood of these robot-following opponents intercepting either the pass (Section 3.1) or the shot on the goal from the optimistic attack.

After these steps are performed for all possible cells, the coerce plan is then the set of those cells with the smallest values of interference likelihood. By sending attacking teammates to the cells in the coerce plan, the robot-following opponents are thus coerced into marking our robots, consequently leaving the optimistic attack plan free of interference.

6 Threat-Based Defense

The threat defense evaluator considers *threats*, which are computed based on the locations of the ball and opponent robots, and chooses locations to place defenders to defend against each of them. There are two kinds of threats: one *first-level threat* and multiple *second-level threats*.

Three distinct tactics work together to form a coordinated defense. The *goal-keeper* remains within the defense area, staying near the goal and defending against the first-level threat. *Primary defenders*, of which there are at most two at any given moment, always move along the edge of the defense area. They guard against the first-level threat if all of them are needed to do so, but one may guard against second-level threats if only one is needed for the first-level threat. *Secondary defenders* are placed away from the defense circle to guard against second-level threats.

6.1 First-level Threat

The first-level threat represents the location of the most immediate threat of a shot on our goal. It is defined to be either the location of the ball or, when the defense evaluator judges that a pass is imminent (as defined below), the location of one of the opponent robots.

A pass is defined to be imminent when the ball’s speed is above a certain threshold, its velocity is not pointed toward our goal, and the defense evaluator judges that it may be headed toward an opponent robot which might be able to receive it soon. The determination of whether an opponent is in position to receive is made using a heuristic function based on the velocity of the ball and the vector from the ball to the robot. More precisely, for each opponent, its “risk of receiving” is given by

$$-\frac{\|d\|}{\|v\|} \cdot (1 + c \cdot (1 - \cos \theta)), \quad (7)$$

where c is an adjustable parameter, v is the velocity of the ball, d is the vector from the ball’s location to the opponent’s location, and θ is the angle between v and d . This expression is greater for positions near the ball than ones far away, and for positions which are in front of the ball’s motion than for ones which are not. If the highest of any opponent’s risk of receiving is above a threshold, then the evaluator judges that the opponent is in position to receive.

When this happens, the first-level threat is the location of the opponent with highest risk of receiving. This means that when the opponent team makes

a pass, it is possible to anticipate where it will be received, and immediately defend against that location, rather than continuously following the ball as it moves, which would result in slower responses.

Once the location of the first-level threat is computed, the defense evaluator decides how to position the goalkeeper and primary defenders to block all open angles on our goal. Primary defenders guarding against the first-level threat always have their target locations along the edge of our defense area. There are three cases that need to be considered:

- If one defender can block the entire open angle by itself: one defender stands along the bisector of the open angle; the goalkeeper in front of the center of the goal.
- If one defender and the goalkeeper can block the open angle: one defender stands just inside the line from the threat location to the nearer corner of the goal; the goalkeeper stands along the bisector of the remaining open angle, as far back as possible.
- If two defenders and the goalkeeper are needed to cover the open angle: the goalkeeper stands along the bisector of the open angle, leaving two smaller open angles to either side of it; one defender stands along the bisector of each of these smaller angles. Figure 7 demonstrates these computations.

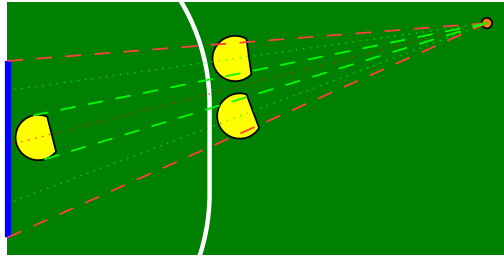


Fig. 7. Placement computation for the primary defenders when two are required. The goalkeeper is placed along the bisector (red dotted line) of the angle from the ball to the goal (red dashed lines). The defenders are placed along the bisectors (green dotted lines) of the two remaining smaller angles (green dashed lines).

6.2 Second-level Threats

The second-level threats are the opponents which might be able to receive the ball from the first-level threat. For every such opponent, two potential defense locations are computed:

- A point on the line from the opponent to the center of our goal. The point is based on latency and acceleration, chosen so that if the opponent starts accelerating, our robot can respond fast enough so that the line to the goal is always blocked. This defends against shots on our goal.

- The midpoint of the line segment from the opponent to the first-level threat. This defends against passes between opponents.

The computed positions are then ranked according to the following criteria, given in decreasing order of priority (where “opponent” refers to the opponent robot which caused a position to be generated):

- Opponents which are closer to our side of the field than a configurable threshold are ranked higher than those which are not.
- Positions which block shots (as opposed to passes) are ranked higher.
- Opponents which have a larger available open angle on the goal are ranked higher. All angles larger than a configurable threshold are treated as equal.
- Opponents which will be able to shoot on the goal sooner are ranked higher. The time to shoot is given by the passing time plus the shot time.

The highest-ranked positions are then assigned to the remaining defenders, with each one greedily assigned to the nearest defender.

An exception to the assignments is when there is a “held” task. This occurs when a defender is blocking a goal shot from a second-level threat, and then the ball is passed toward that opponent, making it the first-level threat. In this case, the defender which is blocking the goal shot continues to block that shot until the primary defenders have moved into place to guard the new first-level threat.

7 Performance

At RoboCup 2013, CMDragons played 7 games in total, and won all but the final game. We scored a total of 27 goals during regular gameplay and 7 goals from penalty kicks, while only 1 goal was scored on us during regular gameplay and 6 goals from penalty kicks. In addition to goals scored in each game, we can evaluate performance during the games using a number of metrics that evaluate the effectiveness of the defense and offense strategies of CMDragons. These metrics include:

1. **Offense Ratio:** the ratio of the game time that the ball was on the opponent’s half of the field, to the game time that the ball was on our half of the field.
2. **Attack Ratio:** the ratio of the number of times our team attempted to shoot towards their goal, to the number of times the opponent attempted to shoot towards our goal.

The offense ratio indicates how often we were on the offensive rather than the defensive, and the attack ratio indicates how often we exploited opportunities to attempt to make shots on goal, compared to our opponents. Table 3 lists the scores and performance metrics for each of the games played, including the Round Robins (RR), Quarter Finals (QF), Semi-Finals (SF) and Finals (F). The game against EMEnents during the round robins was played against an empty

Opponent	Stage	Score	Offense Ratio	Attack Ratio
RoboDragons	RR	2:0	1.38	1.82
BRocks	RR	10:0	1.48	3.14
Parsian	RR	2:1	2.57	1.9
STOx's	QF	2:0	1.78	1.55
MRL	SF	2:0	1.13	1.75
ZJUNlict	F	2(4):2(5)	1.63	1.09

Table 3. Game scores and performance for the games played by CMDragons at RoboCup 2013. Scores are in the form CMDragons:Opponent, with goals from penalty shootouts in parentheses.

field, and resulted in a winning score of 10 : 0 for CMDragons, so we do not include it in the table.

The performance metrics from the logs of the RoboCup 2013 games reveal a number of interesting features. The offense ratios for all the games were greater than 1.0, indicating that the majority of the game times was spent attacking rather than defending. The attack ratios for all the games except for the finals were significantly greater than 1, indicating that our offense was more aggressive at attempting shots on the opponent's goal than the opponents' were on ours. The strategies of the opponents varied significantly across games. RoboDragons [12] and ZJUNlict [13] had defense strategies that were very swift at responding to changes in our attack formations, particularly when transitioning from the Coerce step to the Attack step of the CAP. The BRocks [14] attack strategy included a number of opportunistic attempts on our goal, which our defense intercepted and deflected to their goal. Thanks to the new dynamic ball interception skill (Section 2) and the strategic placements of the secondary defenders (Section 6.2), we successfully intercepted many passes between opponents, some of which even resulted in goals in the games against BRocks [14], STOx's [15] and ZJUNlict [13].

8 Conclusion

This paper gave a brief overview of CMDragons 2014, covering our novel Coerce and Attack planner, and a threat-based defense. Our future work includes focusing on additional opponent model learning, and incorporating direct input from a human. We believe that the RoboCup Small Size League is and will continue to be an excellent domain to drive research on high-performance real-time autonomous robotics.

References

1. J. Biswas et al. CMDragons 2013 Extended Team Description Paper. In *Robocup 2013*.

2. S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso. SSL-vision: The shared vision system for the RoboCup Small Size League. In *RoboCup 2009 Symposium*, pages 425–436.
3. S. Zickler et al. CMDragons 2010 Extended Team Description Paper. In *Robocup 2010*.
4. J. Biswas, J. P. Mendoza, D. Zhu, B. Choi, S. Klee, and M. Veloso. Opponent-driven planning and execution for pass, attack, and defense in a multi-robot soccer team. In *AAMAS 2014*.
5. T. Kalmár-Nagy, R. D’Andrea, and P. Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46:47–64, 2004.
6. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.
7. P. Riley and M. Veloso. Planning for distributed execution through use of probabilistic opponent models. In *AIPS 2002*, pages 72–81.
8. Cetin Mericli. *Multi-Resolution Model Plus Correction Paradigm for Task and Skill Refinement on Autonomous Robots*. PhD thesis, Bogazici University, 2011.
9. Raquel Ros, Josep Lluas Arcos, Ramon Lopez de Mantaras, and Manuela Veloso. A case-based approach for coordinated action selection in robot soccer. *Artificial Intelligence*, 173(9-10):1014 – 1039, 2009.
10. K. Han and M. Veloso. Automated robot behavior recognition. In *Robotics Research: The Ninth International Symposium, 2000*, pages 249–256.
11. C. Erdogan and M. Veloso. Action selection via learning behavior patterns in multi-robot domains. In *IJCAI 2011*, pages 192–197.
12. K. Yasui et al. RoboDragons 2013 Extended Team Description Paper. In *RoboCup 2013*.
13. Y. Wu et al. ZJUNlict Extended TDP for RoboCup 2013. In *RoboCup 2013*.
14. Ö.F. Varol et al. BRocks 2013 Team Description. In *RoboCup 2013*.
15. S. Rodríguez et al. STOX’s 2013 Team Description Paper. In *RoboCup 2013*.