Advancements and Attack Strategy Comparison in the CMDragons 2014 SSL Team

Juan Pablo Mendoza, Joydeep Biswas, Danny Zhu, Richard Wang,
Philip Cooksey, Steven Klee, and Manuela Veloso

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

Abstract. The CMDragons team reached the finals of the Small Size League of RoboCup 2014. In this paper, we present the team's recent work on the offensive and defensive tactics, low level skills, and state estimation. Among the offensive tactics, we introduce the concept of contingency options for teams of robots in the presence of uncertainty via zone-based, and support attack. We increase the robustness of the defense by creating specialized skills for handling loose balls near the defense area as well concerted multi-robot shot-blocking. We present a robot ball-manipulation skill for dribbling the ball while turning quickly. Finally, we describe improvements to the ball state estimation algorithms by accounting for collisions with dynamic robots.

1 Introduction

The CMDragons 2015 team from Carnegie Mellon University (Figure 1) builds upon the ongoing research from previous years (1997–2010, 2013-2014 [1]). This paper focuses on the technical contributions of our team over the past year. Our team website[1] provides an overall description of our team.

In the following sections, we present our work related to the offense and defense tactics, ball-dribbling skills, and state estimation. Section 2 introduces the concept of contingency options in offense, and shows two applications of this concept: Zone-based Team Coordination, and a Support Attacker tactic. Section 3 presents our innovations in defense, which consist of plays and skills to defend against corner kicks and to block incoming shots towards the goal. Section 4 describes a robust method for dribbling the ball while turning quickly. We discuss improvements to ball tracker in Section 5. Section 7 summarizes the contributions and discusses future work.

2 Offensive Contributions: Contingency options

We have previously introduced robust skills for fast interception of moving balls [1], and an attacker tactic that chooses the appropriate skills to intercept

[1] http://www.cs.cmu.edu/~robosoccer/small/

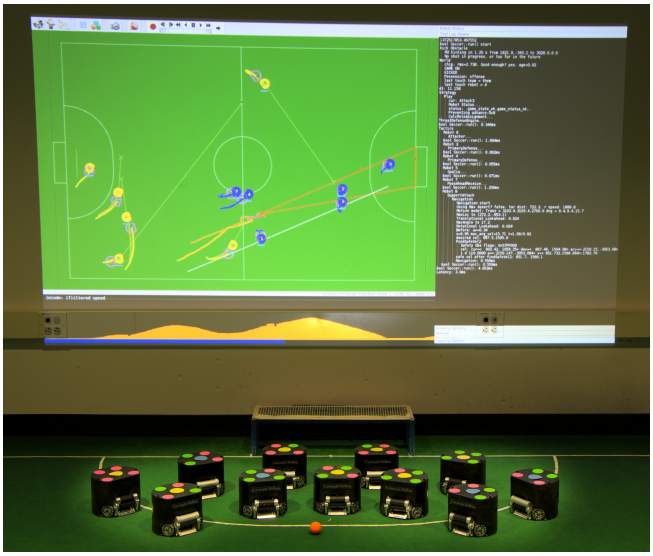


Fig. 1: CMDragons team of soccer robots. In the background, our layered disclosure viewing and debugging tool [2].

and shoot the ball in the minimum possible time. However, in an adversarial domain, there are multiple possible future outcomes that depend on the exact actions taken by opponent robots. For example, if a moving ball is unlikely to be intercepted by an opponent robot, the attacker tactic could continue to use its time-optimal ball interception skill to receive the ball, while if an opponent robot is likely to intercept the ball, the attacker tactic would drive up to the interception point. The attacker tactic always chooses to act on the *most likely* outcome, based on the current perceived world state, and the models of the opponents' capabilities. To handle the other possible outcomes, we therefore introduce *contingency options* for our offense: robot roles that enable offense to be robust in scenarios that are not the most likely ones. We implement this concept through Zone-based Team Coordination, and a Support Attacker tactic. The purpose of these contingency options is not to *replace* the primary attacker, but to *complement* it. While the primary attacker acts on the most likely future ball trajectory, Zone-based Team Coordination and Support Attacker enable offense robots to pursue other likely future ball trajectories.

2.1 Zone-based Team Coordination

Our team has introduced various algorithms to coordinate offense robots. We have introduced plays as predefined team policies based on individual robot skills and tactics [3]. In this system, the plays were organized in a playbook, with weights that could be adjusted with experience [4]. In our CMDragons 2013 team, we departed from the playbook approach for planning during free

kicks. Instead we introduced a coerce-and-attack planning strategy [5], an algorithm that searched for possible plans taking the opponent predicted moves into account. The core aspect of the algorithm consisted on the generation of two possible plans, one of which was revealed to induce the opponents to take positions that would open opportunities for a second viable plan. During the game, teamwork was still achieved through a playbook.

Since last year, our research has focused on teamwork while the game is on, rather than stopped for a free kick. We have introduced a Zone-based Team Coordination in which we divide the field in zones to be assigned to offense robots. We investigate several issues related to the definition of the zones, including (i) their dimensions and number; (ii) their degree of overlap; and (iii) their dynamic resetting during the game, as a function of the score and time left.

The Zone-based Team Coordination algorithm must assign robots to zones, and the behavior for each robot within its zone. Given a set of robots R assigned to offense, we partition the field F into $|R|$ zones, such that there exists a one-to-one mapping from robots to zones. Each robot $r_i \in R$ thus gets assigned to a corresponding zone $Z_i \subseteq F$, where $\bigcup_{i=1}^{|R|} Z_i = F$. Figure 2 shows an example of such an assignment, with three offense robots and their respective zones.

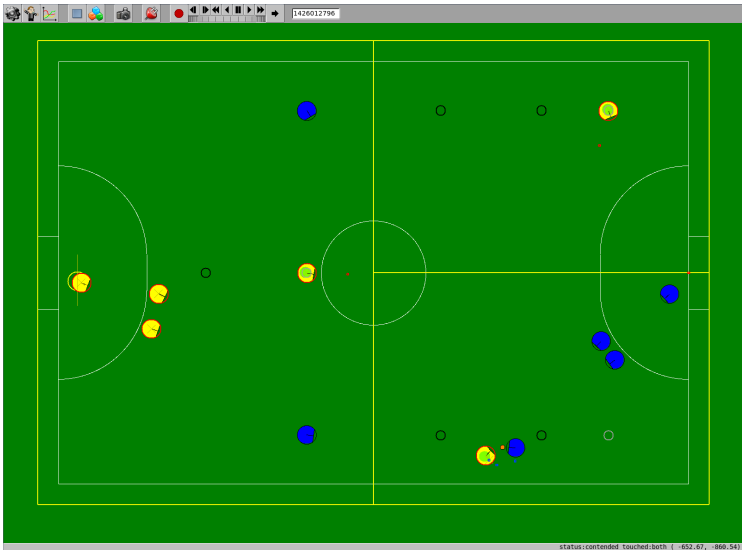


Fig. 2: Offense robots of the yellow team (three rightmost yellow robots) are each assigned a zone, delineated by yellow lines. Black and gray circles show the set of possible and assigned guard positions, respectively.

Given an assignment of zones to robots, our algorithm determines the behavior of each robot r_i in its zone Z_i. Let $\boldsymbol{X}_i^g$ be a set of guard positions in each zone Z_i. If r_i computes that the ball will enter z_i, then r_i moves to intercept the

ball in its zone at the optimal location $\boldsymbol{x}_a(r_i)$; otherwise r_i moves to one of the guard positions $\boldsymbol{x}_g(r_i) \in \boldsymbol{X}_i^g$. The target location $\boldsymbol{x}_t(r_i)$ for r_i is thus given by:

$$\boldsymbol{x}_t(r_i) = \begin{cases} \boldsymbol{x}_a(r_i) \text{ if } \boldsymbol{x}_a(r_i) \in Z_i \\ \boldsymbol{x}_g(r_i) \qquad \text{otherwise} \end{cases} \tag{1}$$

Figure 2 shows one robot intercepting the ball at its optimal location $\boldsymbol{x}_a$, and two robots placed in their assigned guard positions. We note that, when multiple robots predict the ball to be heading toward their zone, they all move to intercept it within their own zone, thus creating multiple contingency options.

We continue to investigate algorithms to set guard positions, as a function of the joint attack, the ball positioning, and the opponent positioning. Our plan is to analyze the impact of multiple alternative choices for the zone definition and robot policy assignment in extensive tests in our simulation environments.

2.2 Support Attacker

To further create contingency options in offense, we present the concept of a *Support Attacker*, which complements Zone-based Team Coordination: while Zone-based Team Coordination ensures good offense coverage of the field, Support Attacker provides a contingency option specifically near the ball.

The Support Attacker robot r_{SA} always stays at a fixed distance d_{SA} from the ball, in a direction $\boldsymbol{u}_{SA}$ from which it is likely to be recover a loose ball:

$$\boldsymbol{x}_t(r_{\text{SA}}) = \boldsymbol{x}_b + d_{SA}\boldsymbol{u}_{SA}, \tag{2}$$

Our ongoing work involves investigating algorithms to determine d_{SA} and $\boldsymbol{u}_{SA}$. A simple but effective assignment for d_{SA} is a constant value that is large enough to not interfere with the Primary Attacker, but small enough to provide support in case the ball follows a trajectory other than the predicted most likely one. It is also possible to adapt d_{SA} as a function of the current action of the Primary Attacker, or other features of the world.

Our algorithm determines the direction $\boldsymbol{u}_{SA}$ as a function of the distance from the ball at which opponents are located. If there is an opponent nearby, $\boldsymbol{u}_{SA}$ is chosen to block a potential shot on our goal; if there is no opponent nearby, $\boldsymbol{u}_{SA}$ is chosen to be aligned with the opponent's goal, ready to shoot. Figure 3 shows an example of a robot acting as a support attacker. We note that, even though the support attacker always tries to maintain a distance d_{SA} from the ball, dynamic role assignment [3] enables a Support Attacker to become the Primary Attacker when appropriate.

3 Defense

Our defense relies on a threat-based evaluator [5], which we briefly summarize here for context to our innovations. The evaluator considers the positions of the ball and the opponent robots to compute the *first-level threat* and several

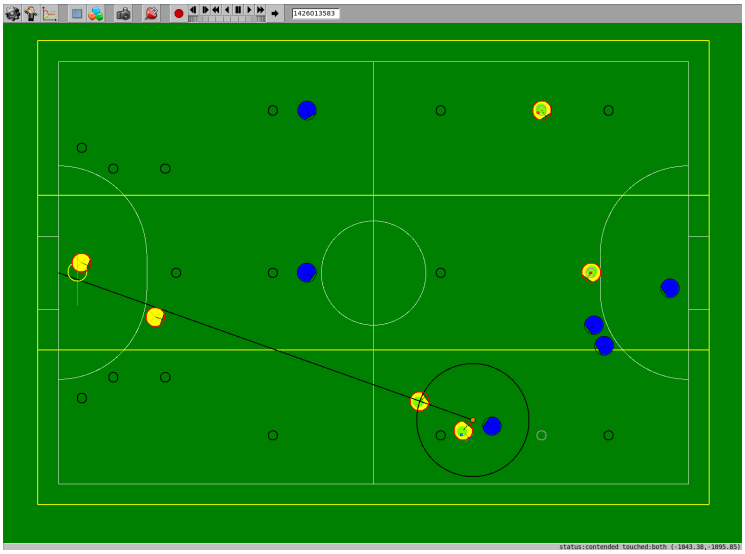


Fig. 3: Offensive play of the yellow team with three robots attacking by zone and one being a support attacker. The support attacker locates itself at the intersection between the black circle (d_{SA}) and the black line (direction of $\boldsymbol{u}_{SA}$).

second-level threats, which are positions on the field. The first-level threat represents the most immediate means for the opponents to score a goal. When an opponent is about to receive the ball, the first-level threat is the position of that opponent; otherwise, it is the position of the ball. The second-level threats represent possible indirect attacks on our goal; they are given by the locations of all opponents except one which is most likely to able to receive the ball first.

The available defenders are then positioned based on the locations of the threats. *Primary defenders*, of which there are usually two, move around the edge of the defense area, acting as the last line of defense before the goalie; *secondary defenders* move further out on the field, intercepting passes and shots by the opponent earlier on. The primary defenders typically defend against the first-level threat, staying between the ball and the goal; if there are two, but only one is needed to do so, the other will move elsewhere on the defense area to guard some second-level threat. The secondary defenders guard against the second-level threats; each one positions itself on a line either from a second-level threat to the goal (to block an indirect shot) or from the first-level threat to a second-level threat (to block a pass).

The computation is described in terms of *tasks*: each threat generates one or more tasks, each consisting of one or two positions, a priority, and other auxiliary information; the tasks are then assigned in decreasing order of priority to the available defenders. Second-level threats to block shots include two positions: one near the defense circle, in case the task is assigned to a primary defender, and one further out, for a secondary defender.

We have extended this evaluator to account for various special game states.

3.1 Three or more primary defenders

Corner kicks taken by the opponent team present a challenging problem for our defense, since opponents tend to attack with more robots than during the rest of the game, and they have an opportunity to build a play from a static ball. Assigning the appropriate defensive roles in which robots do not interfere with each other's navigation is particularly challenging when opponents quickly change formations. Figure 4 shows our solution to this, which involves assigning the role of primary defender to all our defensive robots.

This reduces the problem of positioning the defenders from a two-dimensional problem to a one-dimensional one, ensuring that the evaluator can assign positions to them in a manner which prevents collisions and interference. In this mode, the first-level threat and associated defender position or positions are computed as normal. For second-level threats, only shot-blocking tasks and only their positions near the defense are considered. All the positions are sorted by their linear position around the defense area. Now, in order to avoid collisions while allowing all robots to reach their target positions, the positions must be moved away from each other. The first-level position(s) are treated as fixed, since they represent the most important positions to block; the second-level positions are moved away from the first-level positions if necessary so that there is enough space between them for robots to reach the positions without colliding.

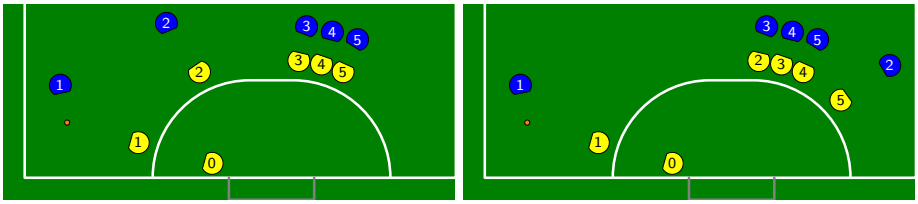


Fig. 4: Examples of our defense's response to opponent positions while all defenders are primary defenders. If opponent 2 crosses behind the other three opponents, the defenders smoothly shift to follow the movement, staying in order without crossing past each other.

3.2 Blocking incoming shots

When two primary defenders are positioned to block a potential shot, it is often necessary to leave a small gap between them to fully block the angle to the goal. While the goalie is positioned to block direct shots that pass between the

defenders, bounces from the sides of the defenders emerge from the gap at unpredictable angles. We have solved this problem by forcing the primary defenders to quickly come together when there is an incoming shot aimed between them, reducing the likelihood that this can happen. Additionally, to prevent collisions, our algorithms evaluate which is better positioned to intercept and clear the ball. That defender then moves forward to do so, while the other stays behind.

4 Fast Dribbling and Turning

Many teams in the SSL, including CMDragons, currently dribble the ball by imparting backspin on it. This method helps the robot drive into the ball while maintaining contact with it. However, imparting backspin on the ball while turning is much more difficult than doing so without turning; in this section, we describe our ongoing effort to achieve reliable dribbling while turning quickly.

First, we describe a few intuitive approaches that fail to be robust enough while turning quickly. The first approach one could take is simply to drive around the center of the ball to turn to the desired direction. This approach works well when the ball has no spin on it, as no forces are applied on it while the robot drives around. However, when the robot has imparted backspin on the ball, the ball will roll and be lost as the robot turns around it.

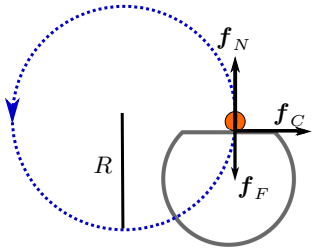
Figure 5a shows a different intuitive but ultimately insufficient approach, in which the robot drives forward with speed s while gradually changing its orientation with speed ω, forming a circle of radius R. These quantities are constrained by $s = \omega R$, and which two parameters are free depend on the use of the skill. We note that the robot turning in place is a special case of this in which R is the distance between the robot's center and the dribbler. As Figure 5a shows, there is no force to balance the centrifugal force experienced by the ball, and therefore the ball escapes the robot's dribbler.

Figure 5b shows our proposed approach for Fast Dribbling and Turning (FDaT), in which the robot turns while pushing the ball, but facing in a direction ϕ that provides the necessary centripetal force to maintain the ball on the dribbler of the robot: facing slightly inwards while turning provides a component of the normal force from the robot that always points towards the center of the circumference. The constraints $s = \omega R$ hold in this case as well. The necessary angle offset ϕ can be obtained analytically by noticing that all the forces in Figure 5b need to cancel out in the rotating reference frame. Therefore, we obtain the pair of equations:

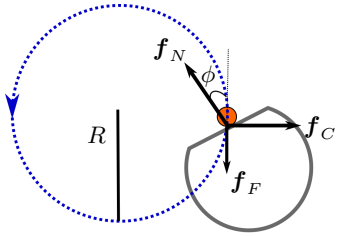
$$\begin{aligned} |\boldsymbol{f}_N| \cos\phi &= |\boldsymbol{f}_C| \\ |\boldsymbol{f}_N| \sin\phi &= |\boldsymbol{f}_F|. \end{aligned} \tag{3}$$

Then, given the acceleration of gravity g, the coefficient of friction of the carpet μ and the mass of the ball m (which cancels out in the end), we obtain:

$$\begin{aligned} |\boldsymbol{f}_N| \cos\phi &= m\omega^2 R \\ |\boldsymbol{f}_N| \sin\phi &= \mu m g. \end{aligned} \tag{4}$$



(a) Robot dribbling ball while driving forward and turning. No force can balance the centrifugal force f_C.



(b) Robot dribbling ball while facing slightly inwards. There exists an angle ϕ for which the forces are balanced.

Fig. 5: Two approaches to dribbling while turning. The gray shape represents the dribbling robot, the orange circle the ball and the dotted blue circle the desired trajectory. Black arrows show the forces acting on the ball, in the rotating reference frame.

Solving these equations for ϕ gives the result for the desired heading:

$$\phi = \tan^{-1}\left(\mu g \omega^2 R\right) \tag{5}$$

Creating a more sophisticated model, accounting for the spin of the ball, is ongoing work. However, this model showed promising results: Figure 6 shows this algorithm in action in RoboCup 2014, where it enabled multiple goals. Furthermore, Section 6 shows significant improvement in our team using this method.

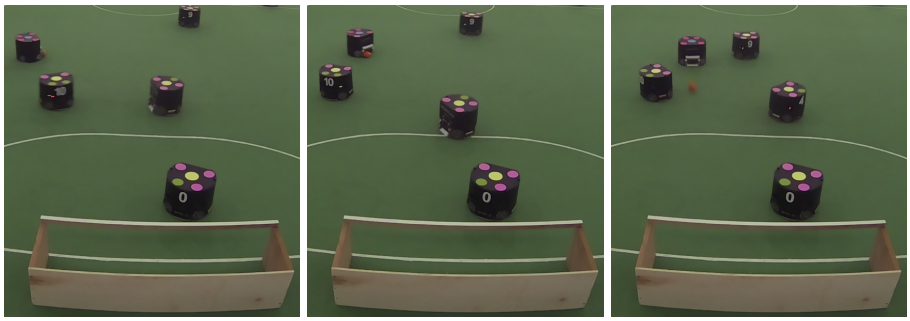


Fig. 6: Our robot using FDaT to shoot onto the opponent's goal immediately after intercepting it.

5 Ball tracking accounting for collisions

To track the state of the ball moving on the ground, we use an Extended Kalman Filter (EKF). While the EKF provides us with a good prediction model for the

ball while it is moving freely on the ground, its predictions are less useful when the ball is about to collide with something else. This is because the EKF relies on a linearization about the estimate of the state of the ball, but collisions are highly nonlinear in nature.

Algorithm 1 Function to predict ball location accounting for collisions against robots. Input: current ball state $\boldsymbol{b}_t = (\boldsymbol{x}_t^{\boldsymbol{b}}, \boldsymbol{v}_t^{\boldsymbol{b}})$ estimate and covariance $\Sigma_t^{\boldsymbol{b}}$; time Δt in the future to predict ball state. Output: ball state $\boldsymbol{b}_{t+\Delta t}$ estimate and covariance $\Sigma_t^{\boldsymbol{b}}$ at time $t + \Delta t$.

```
1:  function PredictCollisions(b_t, v_t^b, Δt)
2:      (b_{t+Δt}, Σ^b_{t+Δt}) ← PredictLinear(b_t, Σ_t^b)
3:      b_s ← b_t                          ▷ record source state of current path segment
4:      repeat                             ▷ update b_{t+Δt} while there are collisions
5:          r_c ← CollidingRobot(b_s, b_{t+Δt})
6:          if r_c ≠ ∅ then
7:              (b_s, b_{t+Δt}) ← Reflect(b_s, t + Δt)
8:          end if
9:      until r_c = ∅
10:     return (b_{t+Δt}, Σ^b_{t+Δt})
11: end function
```

To mitigate this limitation of the EKF, we have augmented the prediction used to update it to explicitly account for the ball colliding against robots on the field. Algorithm 1 describes this process at a high level. Starting from the linear prediction of the ball trajectory (line 2), the algorithm repeatedly updates the trajectory while it finds collisions on the way (lines 4-9). For this, we find whether the trajectory intersects any robots (line 5) assuming there is at most one, and then reflect the trajectory about the point of collision (line 7). Figure 7 illustrates this process, which takes into account the geometry of the typical SSL robot (flat in the front, circular around the body) as well as typical reflection coefficients for its surfaces.

6 Simulation Experiments

To test the impact of our improvements on the game, we ran extensive experiments in our PhysX-based simulator, in which robots are modeled at the component level. The form of the inputs and outputs between the AI and the simulator are identical to those between the AI and the real robots.

The experiments involved running multiple automated games between a standard CMDragons team, which was used during the SSL Finals of 2014, and variations of this team in which one element of the AI was modified. We tested each condition on 50 games of 20 minutes each – the length of a standard SSL game.

It was only possible to conduct extensive experiments with little supervision due to our automated referee [6]. The automated referee agent implements most

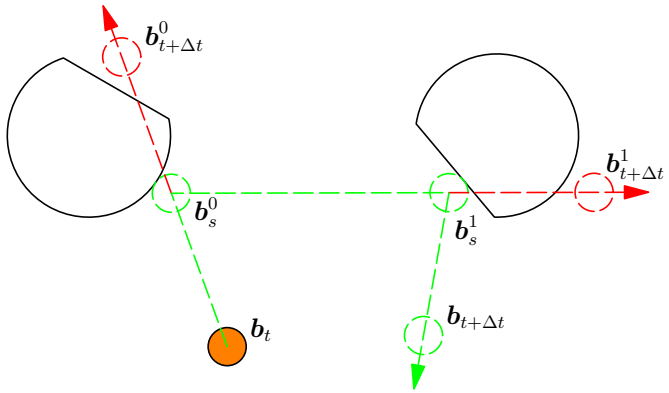


Fig. 7: Application of Algorithm 1. The ball initial estimate is orange, and robots are black. Red and green paths indicate detected collisions and the corrected trajectory, respectively. Superscripts show the iteration of the loop in lines $4-9$.

of the rules of the SSL and thus enables the games to be fully automated. Among the rules that the automated referee enforces are the beginning and end of games, appropriate restart of game once the ball goes out of bounds, detection of goals scored, neutral restart if the game fails to progress for more than 10 seconds, and detection of robot location-based infringements.

The standard AI of the 2014 Finals contained some of the innovations described in this paper. In particular, it had Zone-based team coordination of Section 2.1, the defense improvements of Section 3, and the tracker improvements of Section 5, but not the Support Attacker of Section 2.2 or the dribbling innovations of Section 4.

For the experiments below, we gathered the following statistics for each team in an automated fashion, in roughly non-increasing order of importance:

Winning Percentage The percentage of games won.
Average Goals The average number of goals scored during a game.
Average Shots on Goal The average number of goal-directed shots that were blocked by the opponent's goalie or went into the goal.
Average Blocked Shots The average number of goal-directed shots that were blocked by a robot that was not the opponent's goalie.
Offense Percentage The percentage of active game time in which the ball was on the opponent's half of the field.

Below, we describe the variations that showed the most significant improvement in our team. However, we have tried other variations and plan to use such extensive simulation process in the future to select our standard team for 2015.

6.1 Standard team

First, we tested our standard team against itself. This experiment provided us with initial results about our team's performance, allowing us to test how vari-

ations affect the performance of the standard team –for example, one variation may be particularly effective at blocking the offense of the standard team, while another may be effective at penetrating its defense. Table 1 shows the results of the standard team against itself.

Team	Win %	Avg. Goals	Avg. Shots on Goal	Avg. Blocked Shots	Offense %
Standard (a)	22%	0.34	0.96	42.34	49.55%
Standard (b)	16%	0.32	1.10	41.86	50.45%

Table 1: Results of matching our standard team against itself. As expected, the statistics are approximately equal for (a) and (b). These statistics serve as a standard behavior for our team.

6.2 Fast Dribbling and Turning (FDaT) experiments

To test the dribbling while turning innovations to our team, we only modified the Primary Attackers – i.e, the robots that try to gain and maintain control of the ball. Every time the ball is stationary, the Primary Attackers use the dribbling while turning algorithm to align itself with its intended passing or shooting target. Once the robot is aligned, it briefly drives forward until the ball's velocity along the axis perpendicular to the desired shooting direction is dissipated, and proceeds to shoot. Table 2 summarizes the results for this test. FDaT shows a significant improvement in the overall game development, showing better results than the standard in all measures. In particular, we note that FDaT shows a positive effect both offensively and defensively: not only does FDaT show higher offensive performance than the standard team in Table 1, but the standard team shows worse offensive performance than in the previous condition.

Team	Win %	Avg. Goals	Avg. Shots on Goal	Avg. Blocked Shots	Offense %
Standard	08%	0.22	0.66	31.82	43.90%
FDaT	50%	0.84	1.76	38.80	56.10%

Table 2: Results of testing our dribbling while turning technique versus the standard code of the 2014 finals. The code that uses our dribbling method outperforms the standard significantly along every metric.

6.3 Zone-based coordination experiments

To test the Zone-based coordination, we compared the base team, which already used zone-based coordination, to a similar team in which the teams involving zone-based coordination were replaced with a play not involving zones. Instead, this play had one Primary Attacker in charge of handling the ball, two robots positioning to receive passes according to an evaluation function [5], two defending robots and a goalie. Table 3 summarizes the results for this test. Zone-based

coordination significantly outperforms the team with no Zone-based coordination along the most important dimensions: Winning percentage, Goals and Shots on Goal. We note that the average number of Goals and Shots on Goal for the standard team are higher than those of Table 1, showing that the Zone-based coordination also presents defensive benefits, as the standard team could defend better against itself than the no-zone team could.

Team	Win %	Avg. Goals	Avg. Shots on Goal	Avg. Blocked Shots	Offense %
No zones	04%	0.12	0.7	42.22	50.05%
Standard	38%	0.48	1.62	36.36	49.95%

Table 3: Results of testing Zone-based coordination versus a similar team that did not rely on zones to attack. Zone-based coordination outperforms the team with no zones on all metrics except for Blocked Shot.

7 Conclusion

This paper presents the recent innovations of the CMDragons team of the RoboCup SSL. We focus on making the team more robust, by introducing the notion of contingency options on offense, expanding the abilities of defense, making our predictions of the state of the world more accurate and our skills more reliable. These innovations have allowed us to stay competitive in the RoboCup Small Size League, reaching the finals in the 2014 tournament. Furthermore, extensive simulation tests, enabled by our automated referee agent and a physics-based simulator, have allowed us to test various innovations individually, and will guide us in preparation of our 2015 team.

References

1. J. Biswas, J.P. Mendoza, D. Zhu, S. Klee, and M. Veloso. CMDragons 2014 Extended Team Description Paper. In *RoboCup 2014*.
2. Patrick Riley, Peter Stone, and Manuela Veloso. Layered disclosure: Revealing agents' internals. In *ATAL-2000*, July 2000.
3. B. Browning, J. Bruce, M. Bowling, and M. Veloso. STP: Skills, tactics, and plays for multi-robot control in adversarial environments. *Proceedings of the IME, Part I: Journal of Systems and Control Engineering*, 219(1):33–52, 2005.
4. M. Bowling, B. Browning, A. Chang, and M. Veloso. Plays as team plans for coordination and adaptation. In *RoboCup 2003 Symposium*, pages 686–693.
5. 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*.
6. Danny Zhu, Joydeep Biswas, and Manuela Veloso. Autoref: Towards real-robot soccer complete automated refereeing. In *Proceedings of the 18th RoboCup International Symposium*, July 2014.