Robust Supporting Role in Coordinated Two-Robot Soccer Attack

Mike Phillips and Manuela Veloso
mlphilli@andrew.cmu.edu, veloso@cmu.edu

Computer Science Department, Carnegie Mellon University

Abstract. In spite of the great success of the fully autonomous distributed AIBO robot soccer league, a standing challenge is the creation of an effective planned, rather than emergent, coordinated two-robot attack, where one robot is the main attacker and goes to the ball and the other robot "supports the attack." While the main attacker has its navigation conceptually driven by following the ball and aiming at scoring, the supporter objectives are not as clear. In this work, we investigate this distributed, limited perception, two-robot soccer attack with emphasis on the overlooked supporting robot role. We contribute a region-based positioning of the supporting robot for a possible pass or for the recovery of a lost ball. The algorithm includes a safe path navigation that does not endanger the possible scoring of the teammate attacker by crossing in between it and the goal. We then further present how the supporter enables pass evaluation, under the concept that it is in a better position to visually assess a pass than the attacker, which is focused on the ball and surrounded by the opponent defense. We show extensive statistically significant lab experiments, using our AIBO robots, which show the effectiveness of the positioning algorithm compared both to a previous supporter algorithm and to a single attacker. Additional experimental results provide solid evidence of the effectiveness of our passing evaluation algorithm. The algorithms are ready to incorporate in different RoboCup standard platform robot teams.

1 Introduction

We conduct research on creating teams of robots that work cooperatively on tasks in dynamic environments. In this paper, we focus on our work within the domain of the RoboCup [4] 4-Legged robot soccer using the Sony AIBO robots. The robots are fully autonomous with onboard control, actuators, and in particular, limited visual perception. They can wirelessly communicate with each other. These features are general to other robot platforms and are the basis for our work for on the needed real-time coordination in robot soccer. While the RoboCup 4-Legged league has been very successful, we have not seen coordinated passes, for example, such as in the RoboCup small-size robot league. In this paper, we investigate a two-robot attack with distributed, limited perception equipped AIBO robots. We present cooperation algorithms for positioning and coordination and show successful passing lab results.

More specifically, one robot has the role of *attacker* whose job is to go to the ball, move it upfield, and score goals. This robot's role is very clear since it is ball and goal oriented. This paper is concerned with the interesting problem that arises when a second attacker is added to the team, known as the *supporter*. The supporting role, whose task is not quite as clear, when used effectively should shorten the time it takes to score goals. This goal can be achieved through proper positioning, to recover a lost ball more quickly, and to receive passes. At the same time, it is critical for the supporter to not hinder the attacker's performance by crowding the ball, blocking the attacker's shots on the goal, or causing hesitation. With these objectives in mind, we present sets of algorithms. We first provide an approach for selecting a supporting position while dealing with supporting constraints, such as not crossing the attacker's shot on goal with domain constraints, like a bounded field. We present details as to how a supporter gets to its support point quickly while accounting for opponent movement.

Secondly, we give insight into how a supporter can be used to evaluate the option of passing. The attacker is defined to be the robot in possession of the ball, making all decisions regarding how the ball is moved. We argue that this generally accepted concept for deciding how the ball should be moved is not ideal in the presence of limited perception and a highly dynamic environment, such as robot soccer. We introduce an approach *to outsource* some of the decision making to the supporting robot, which we note has more information about the game state than the attacker does. The attacker has less information about the ball's surroundings because the attacker's perception is completely focused on the very close ball. Furthermore the ball's attractive nature draws opponents which obstruct the camera's already limited view of the attacker.

Teamwork and the positioning of a supporter have been addressed with a few approaches. The supporter robot has been positioned using potential [1]. However, this potential-field based algorithm does not take into consideration the constraint of not crossing over the attacker's shot on goal, does not take opponents into account, and does not include planned passing. Another related algorithm tries to position the supporter to be in the place where it has the highest probability of being of use, applied in simulation and in the centralized controlled small-size robot soccer [2] with centralized perception. Our problem is inspired by these efforts, but made more difficult by the robots' distributed control and local view from a small and directional onboard camera. Furthermore we address the dynamic positioning for passes in the presence of teammates and opponents, in addition to static passing [7].

First, we briefly overview the RoboCup AIBO league, and our team's role framework. We then discuss region-based positioning and the pass determination of the supporter robot. Each part includes experimental results.

The RoboCup 4-Legged league has teams of four Sony AIBO robots competing against each other. The Sony AIBO (Figure 1) is a 4-legged robot whose primary sensor is a low resolution camera, with a small field of view (less than 60 degrees). This camera is mounted in the robot's head (at the tip of its nose) and can pan 90 degrees in both directions as well as having two joints for tilting.

Fig. 1. A typical attacker situation: an AIBO robot surrounded by other robots

Our team uses a team control approach with *roles* being used to divide the tasks of robot soccer amongst the four players. The four roles that we use are goalie, defender, attacker, and offensive supporter. In CMDash'07, a specific robot took the defender role and kept it throughout the game unless the other two team robots would get penalized and out of the game. The goalie and defender are used for defense with the former being in the goal box while the latter defends from outside. The other two roles, which are the ones of interest for this paper, are the attacker and supporter. The attacker's role is well defined, being the robot that acquires the ball, moves it up the field, and finally scores goals. The supporter's role is to assist the attacker. Our play-based framework allows the attacker and supporter to swap roles when the supporter is evaluated to be more likely to attain the ball [8]. Therefore, the supporter's objectives involve trying to prevent the loss of possession of the ball to opponents, and to recover lost balls as quickly as possible, while not interfering with the attacker's objectives.

Each robot in the distributed CMDash'07 team uses a world model that maps perceptual data into positioning of all the objects in the domain, namely ball and robots. Robots communicate to improve their assessment of the complete world. Our robots follow an individual and shared world model approach [5]. The individual version contains Gaussian locations of recently seen objects, and therefore smoothes the movement of objects, which removes spurious readings. It also allows objects to continue to be modeled based on previous perceptions and models of one robot's actions, even if they have not been seen for a few frames. The shared world model includes the shared ball and teammate locations communicated between robots on the team. The world model is updated based on the confidence of the information perceived, communicated and its recency [6]. Due to compounded localization and vision errors, teammates do not yet pass position information about the other robots they have seen.

2 Region-Based Supporter

Under our framework, the supporter only exists when the team has an attacker that is either: in possession of the ball or has claimed the ball as its own, based on the role selection algorithm (the attacker/supporter swap, mentioned earlier). The supporter's purpose is to help the team keep possession of the ball. It can do this in two key ways: strategic positioning and receiving passes.

2.1 Positioning

Proper positioning of a supporter increases the number of balls that are recovered by our team. We develop an algorithm that selects a good *support point*, while also meeting several other objectives to maximize performance:

- The supporter should be in a location where the ball is likely to roll if our attacker loses possession of the ball.
- When near the goal, the supporter should be placed to recover balls that are deflected by the goalie without getting in the way of the attacker's shots.
- In the offensive side of the field, it is important that the supporter does not obstruct a shot on goal at any time. This objective can cause the ideal supporting point to be unreachable due to the field bounds. The supporter is then *pinched* and moving getting to the desired support point would require it to go outside the boundaries. Therefore, the supporter needs to have a "next best" point to support that is reachable.
- If the ball is in the defensive half of the field, the supporter should stay on the offensive side to allow defenders to *clear* the ball upfield to it.

Some of these objectives only apply to certain parts of the field. Thus, our positioning algorithm uses three regions (offense, defense, goal), as seen in Figure 2, to better handle the cases. The region is chosen based on the position of the ball. Each region slighly overlaps with its neighbors (overlap not shown in Figure 2 in order to prevent oscillation. When the ball is in an overlapping area the robot maintains the region that the ball was in previously.

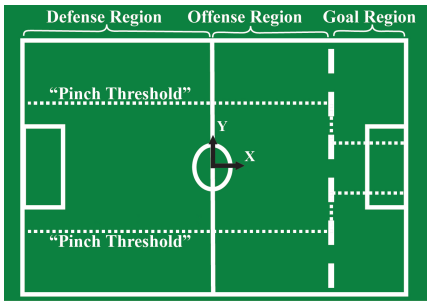

Fig. 2. Field with regions. Neighboring regions slightly overlap to prevent oscillation. Note that we assume that the positive x axis points toward the opponent's goal.

Before we present our region-based supporter positioning algorithm, we define a few functions that the algorithms uses.

The function *AligntoCloseSide* takes the ball's coordinates (b_x, b_y) and the robot's y (r_y), and a minimum value x_{min}. It returns a tuple representing a support point with coordinates (p_x, p_y), respectively set to b_x, with a lower bound, x_{min} and a constant offset from the ball, $C_{yoffset}$. p_y is on the side of b_y

that the robot is closer to; p_y is bounded at $\pm C_{ymax}$ just inside the field bounds. A similar function, $AlignToFarSide$, selects p_y by picking the side of the ball the robot is farthest from.

– $(p_x, p_y) \leftarrow$ AlignToCloseSide(b_x, b_y, r_y, x_{min})
  $p_x \leftarrow Max(b_x, x_{min})$
  $p_y \leftarrow Max(Min((b_y + C_{yoffset} * Sign(r_y - b_y)), C_{ymax}), -C_{ymax})$
  return (p_x, p_y)

The function $AlignUpField$ takes the ball's coordinates (b_x, b_y) and robot's r_y and returns a support point that is upfield of the ball, keeping its p_x bounded at the top of the field, C_{xmax}. The p_y value is set to the edge of the field that the robot is closer to than the ball.

– $(p_x, p_y) \leftarrow$ AlignUpField(b_x, b_y, r_y)
  return $(Min(b_x + C_{xoffset}, C_{xmax}), C_{ymax} * Sign(r_y - b_y))$

The function $Pinched$ returns a boolean that tells if the robot's r_y is "pinched" between the ball's b_y and the nearest field boundary. The ball's b_x is important since we introduce different pinch constants for different regions.

– $bool \leftarrow$ Pinched(b_x, b_y, r_y)
  return $((b_x > C_{goal,offense\ bound}$ **and** $abs(b_y) > C_{goal\ pinch})$ **or**
  $(b_x \leq C_{goal,offense\ bound}$ **and** $abs(b_y) > C_{offense\ pinch}))$ **and**
  $(abs(r_y) > abs(b_y)$ **and** $Sign(r_y) == Sign(b_y))$

The function $PathCollideWithAttack$ checks if the path from the robot (r_x, r_y) to the support point (p_x, p_y) intersects with the attacker. The attacker is estimated to be on the line segment straight back from the ball (b_x, b_y) at a distance $C_{robot\ length}$.

– $bool \leftarrow$ PathCollideWithAttack$(b_x, b_y, r_x, r_y, p_x, p_y)$
  return Intersects$(((r_x, r_y), (p_x, p_y)), ((b_x, b_y), (b_x - C_{robot\ length}, b_y)))$

The function $GoBehindAttacker$ returns a support point that is behind the attacker. If the robot is not already behind the attacker, then it moves backward. Otherwise the supporter moves horizontally toward the desired support point's y coordinate, p_y.

– $(p_x, p_y) \leftarrow$ GoBehindAttacker$(b_x, b_y, r_x, r_y, p_y)$
  if $r_x > b_x - C_{robot\ length}$ **and** $abs(b_y - r_y) < C_{robot\ length}$
  return $(r_x - 2 * C_{robot\ length}, r_y)$
  else if $r_x \leq b_x - C_{robot\ length}$
  return (r_x, p_y)

The function $CloseGoalCorner$ returns a support point with fixed x and y coordinates that line up on the corner of the goal box. It chooses from the two corners by selecting the corner that is on the same side of the ball as the robot. A similar function, $FarGoalCorner$ (not shown) chooses the corner on the opposite side of the ball.

– $(p_x, p_y) \leftarrow$ CloseGoalCorner(b_y, r_y)
  return $(C_{goal\ box\ bottom}, C_{goal\ box\ side} * Sign(r_y - b_y))$

Supporter Positioning in the Offense Region The supporter in the offense region (Figure 3 (a)) keeps the same x coordinate as the ball while maintaining a constant distance offset from the ball's y position. If the supporter becomes *pinched* and is behind the ball (in terms of x), it safely and effectively goes around behind the ball and the attacker, resuming then its usual positioning. A more difficult case arises when the supporter is ahead of the ball. Going around the ball now is not only too time consuming, but it may actually be impossible to do without walking out of bounds. Therefore, to remain in bounds and to not block a shot on goal, the supporter moves upfield to allow for an upfield pass. This position is not as desirable and therefore, as soon as the supporter becomes un-pinched or gets the chance to go behind the attacker, it goes back to its normal behavior. All motion is capped by the field boundaries.

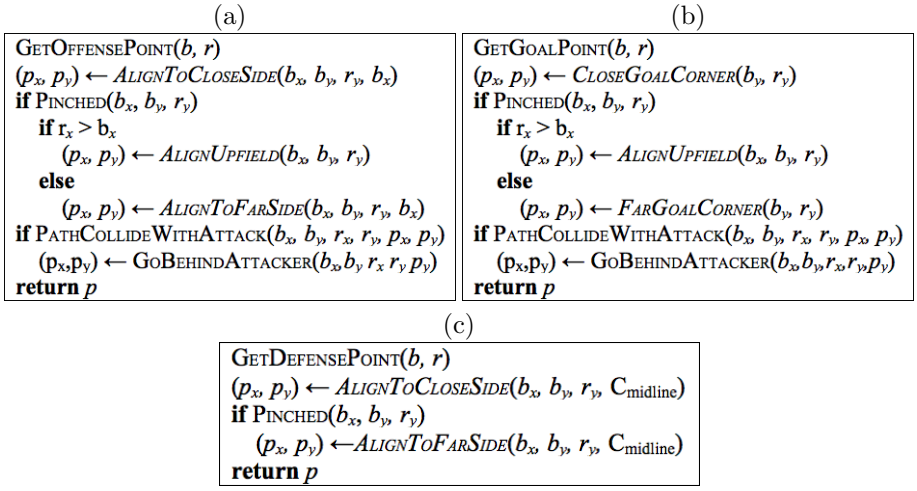

Fig.3. Region-based support point for (a) offense, (b) goal, (c) defense, regions

Supporter Positioning in the Goal Region In the goal region the focus is on recovering the ball after the opposing goalie deflects it. Therefore, the supporter chooses to sit on the goal box corner (Figure 3 (b)). The supporter still handles being "pinched" the same way, except that the thresholds are moved in more, so the ball never gets too close to the supporter. If the attacker switches sides, the supporter moves to the other goal box corner.

Supporter Positioning in the Defense Region Finally, the supporter's behavior (Figure 3 (c)) when the ball is in the defense region is very similar to that of the offensive region. However the supporter never goes below the midline, so that when a defensive robot clears the ball, there is already a robot in the offensive region. There is also no concern with interrupting a shot on goal since the defense region is far away from the opponent's goal. As a result, when the

supporter gets "pinched," it always freely crosses in front of the ball's shot on goal to get the better supporting position.

2.2 Navigation to the Support Point

The region-based positioning algorithm shows how the optimal support point is selected. However, this does not address how the robot actually moves to the selected point quickly and safely while getting enough information from the field to not hinder the support point selection algorithm. Therefore, we have some key objectives that need to be addressed.

- The robot should move to the point quickly by prioritizing its walking directions. The AIBO walks fastest forward, then backward, then sideways.
- When the supporter is going around opponents, it should keep itself between them and the ball in order to get to the support point safely.
- The robot must maintain a good view of the ball in order to choose good supporting points. This is difficult because the supporter is not moving to the ball like the attacker.
- In the event that the supporter loses sight of the ball for more than a moment, it needs some way to continue to make acceptable positioning decisions.

Our algorithm NavToPointSeeBall (pseudo code not shown due to limited space) first determines whether it has seen the ball recently. If it hasn't, then it chooses the support point based on the attacker's estimated ball location.

The algorithm prioritizes the primary walking directions so that the robot can choose the fastest walking direction that still allows it to see the ball at all times. With the proper constants, our algorithm makes minimal use of the slower sideways walk, while choosing a direction that allows the ball to remain centralized in the AIBO's field of view. In our actual implementation, we overlap the FOV regions slightly to prevent oscillation between walking directions.

Our algorithm extends the NavToPoint behavior, developed previously to be used with object detection [3]. The algorithm in that paper describes how to detect opponent robots. This behavior was written to make use of this opponent detection when walking forward to a target point. We extended this algorithm to use all four primary walking directions. This is still most effective when walking forward as the robot is most likely to see opponents. Another key change was to always go towards the ball when going around an opponent, putting the supporter between the opponent and the ball.

3 Positioning Experimental Results

In order to test the effectiveness of our algorithms, we performed three sets of experiments. All three sets involve a defensive team of a goalie and defender protecting the goal. The goalie always begins in the goal box while the defender is started a short distance in front of it. The ball is started just past the midline on the defensive side. All three sets have the same attacker which starts a small distance behind the ball. Figure 4 shows the initial field state for each trial.

Fig.4. The starting configuration for our experiments.

The three sets of experiments only differ in the supporter of the offensive team. The first set uses our new positioning algorithms. The second set uses a previously developed algorithm, where both offensive robots are attackers until one gets close to the ball. When this occurs, the farther robot cannot get closer than a constant distance to the ball, to prevent crowding. The third set is a control group with no supporter and only one robot as attacker. All robots on the field were given time to localize before each test began. We timed the time to go from the starting configuration to scoring a goal. We ran the trials following standard RoboCup rules for the 4-Legged League. After a goal was scored, the time was recorded and the field was reset, to make each of the goals independent of each other. We ran 30 trials of each approach. Table 1 shows the resulting analysis of the data.

	Region Supporter	Previous Supporter	No Supporter
Mean time to score (seconds)	75.2	133.5	164.8
Standard Deviation	67.2	113.5	133.0
Goals with supporter assistance	26/30	25/30	0/30

Table 1. Supporter positioning results

We performed unpaired t-tests using the data we collected. When comparing our supporter positioning algorithms with the previous supporter algorithm, the t-test gives us a two-tailed P value of 0.0188, which is statistically significant at the standard 0.05 level. The comparison between our algorithms and the control group with no supporter achieved a two-tailed P value of 0.0017, even more significant. These results provide solid evidence of the value of our algorithms. The algorithms described in this paper were used at RoboCup 2007 and were instrumental in our team's achievement of 3^{rd} place in our division.

4 Pass Determination

This section focuses on the problem of the specific coordination between the attacker and supporter robots towards effective passing.

4.1 Distributed Decision Making

As stated previously, in robot soccer, the ball acts as an attractor for robots on both teams. The field around the ball is therefore dense with robots, leading us to two important realizations:

- The attacker with the ball is more likely to be in an opponent-rich environment than any teammate.
- The increased number of opponents trying to get close to the ball can easily cause the attacker's limited camera to become obstructed.

We consider that the attacker only has a few ways to move the ball: shoot on goal (if close enough), pass, dodge, or dribble (if too far to shoot on goal). A pass to a teammate seems like an obvious choice to escape the situation where the attacker has opponents closing in on it, since it quickly moves the ball from a point of high opponent density to one that is much lower. Also, the attacker has very limited information about whether a pass is possible or not. This is due to many factors. First, the general view of the attacker may be obstructed by opponents which could leave other opponents unnoticed or leave the attacker mislocalized. Also, due to the attacker's closeness to the ball, the attacker may know very little about the ball's surroundings. Most importantly, the attacker cannot afford to let go of the ball and look around to determine if passes are possible. Our idea is to have the decision making be distributed to players who are better informed to make a particular decision. This allows the attacker to make better decisions since they are being sent to it from more informed teammates. This *outsourcing* of decision-making also has the added benefit of being efficient since it reduces the amount of computation the attacker must perform. The attacker now merely has to look up the most recent message. Although in our situation we deal with the decision of whether to pass or not, our concept of outsourcing decisions away from the focal player to teammates can be generalized. Specifically, we show how outsourcing can greatly improve accuracy when determining whether a pass to another player (in this case an offensive supporter) is possible.

We use an offensive supporter as the pass receiver and as the robot to which the pass evaluation algorithm is outsourced. The supporter robot is a good outsourcing choice as it can be well positioned to maintain an offset distance from the attacker robot and near the goal as described in the previous section.

4.2 Pass Evaluation Algorithm

The algorithm we present assumes that opponent and teammate robots are detected and represented in the world model, as in [3]. Our pass evaluation algorithm, assumes that it runs on a supporter that is well positioned to keep the ball in view (we used our earlier described positioning algorithm). The following algorithm uses the positions of robots in the world model and the location of the ball to determine whether a pass is likely to be successful, meaning it is not likely to be intercepted by another robot. Figure 5 shows the pseudo code for our

algorithm, EvaluatePass, running on the supporter robot for distributed decision making. It makes use of IsOpen and ApplySmoothing, two main functions of the algorithm. They evaluate the pass on a single vision frame and apply smoothing across a sequence of noisy vision frames, respectively.

```
EvaluatePass(posn, ball, robots)
open ← IsOpen(posn, ball, robots)
open ← ApplySmoothing(open)
if open ≠ +1
   ResetOpenTimer()
if OpenTime() > OPEN_THRESH
   SendMessage(MSG_OPEN)
else if open = –1
   SendMessage(MSG_NOT_OPEN)
```

Fig.5. Complete pass evaluation algorithm

EvaluatePass sends a MSG_OPEN message to its teammate (SendMessage) if a pass is possible from the ball to the supporter. A pass is considered possible if open has been +1 for an amount of time OPEN_THRESH. If open is -1, the message MSG_NOT_OPEN is sent to its teammate, stating that a pass is not possible. Lastly, if open is 0, no messages are sent since the supporter is currently unsure, which means the attacker uses the last sent +1/-1 message. The function IsOpen determines whether the supporter is open to receive a pass based only on the current vision frame.

The IsOpen function determines if the supporter can receive a pass based on the current vision frame. If the robot has not seen the ball recently, then it certainly cannot receive a pass. If the ball was seen recently, then the function uses vector math to determine if a robot is in the passing lane. It does this by "drawing" a rectangle from the supporter to the ball, with a predefined width and checks if a robot is contained within that rectangle. It allows for up to one teammate to be in the rectangle since that is likely the attacker. However, if there are opponents, or more than one teammate in the rectangle, then a -1 is returned. If the passing lane is clear though, then +1 is returned.

The returned decisions of IsOpen are fed into ApplySmoothing, a simple smoothing function. ApplySmoothing returns the decision +1/-1 if it has been seen for a predefined number of consecutive frames, otherwise it returns 0.

5 Passing Experimental Results

In order to test our algorithm, we show that it performs well in assorted scenarios that make use of all of the attacker's ball movement options. The attacker is in the shooting region so it can choose to shoot on goal, pass, or dodge. In all cases we have the attacker, supporter, and ball start at the same locations. Figure 6-A shows the set up for when the attacker should simply shoot on goal since there is no opponent stopping it. Figure 6-B now has a static opponent placed in such a way that the shot on goal is blocked and it is advantageous to pass to the open teammate. Finally, Figure 6-C displays the setup for the case when the shot on

goal is blocked as well as the pass to our teammate meaning that the attacker should resort to a dodge to attempt to not lose the ball.

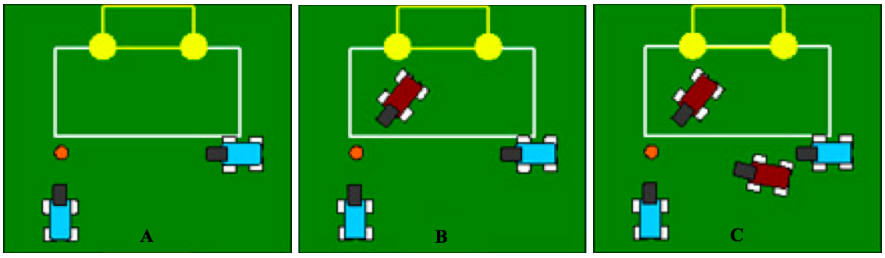

Fig.6. Several scenarios for the attacker-supporter distributed decision making
A - open shot, open pass; B - blocked shot, open pass; C - blocked shot, blocked pass

We ran 10 trials for each case for both algorithms. Successes were based on whether the attacker performed the best ball movement action, given the scenario. The two algorithms we are using are identical except one of them has the attacker make all decisions from its world model, while the other lets the supporter determine if a pass is possible.

Scenario (expected action)	Pass Determination	
	Attacker	Supporter
Goal open (shoot on goal)	100%	100%
Goal blocked, Pass open (pass to teammate)	80%	100%
Goal blocked, Pass blocked (dodge opponent)	10%	100%

Table 2. Pass success results

Both algorithms perform well on an open goal which is a good check since the two are identical up to that point. If there is not an opponent in front, the priority goes to shooting on goal. The next case is when the goal is blocked but the pass is open, making a pass ideal because the ball would be movef from a place of high opponent density to a lower one. Here our algorithm performs well again. The supporter saw an open passing lane from itself to the ball and told the attacker that it was open. When the attacker grabbed the ball and saw opponent in front of it, it then had to decide whether to pass or not. It retrieved the answer that the supporter had sent and then executed the pass.

The other algorithm is based only on what the attacker has in its world model. The data shows that 80% of the time, the attacker's world model had no opponent in the passing lane and therefore it passed, which is good since there was not an opponent in the passing lane. The 20% in which the attacker instead chose to dodge is interesting because as the attacker went to the ball, it must have seen an opponent on the edge of its field of view and added it to its world model. This makes sense because the edges of the camera's field of view are not as accurate and head movement tends to blur the image further. It is

then possible that the attacker got this false reading randomly or by seeing its own teammate. The supporter algorithm on the other hand does not run into this problem since the passing lane is in the center of its field of view and its head is relatively stationary.

The last case is when the goal and the passing lane are blocked. In this case, the supporter algorithm once again performs well as it detects an opponent in the passing lane. It then sends messages to the attacker so that if the attacker chooses not to shoot on goal, then it also chooses not to pass, instead defaulting to dodging. Not surprisingly, the algorithm where the attacker determines if a pass is open or not performs poorly. It incorrectly chooses to pass 90% of the time, giving the ball to the opponent, as it does not detect the opponent in the passing lane.

6 Conclusion

In this paper, we contributed a positioning and pass evaluation algorithms for the challenging supporter role in distributed robot soccer. We specifically introduced a region-based positioning to capture multiple supporting strategies as a function of the game challenges. We discussed the problems of having a single focal robot making all the decisions, especially when the hardware limits the amount of information the robot has. We introduced the outsourcing of passing decision-making to the more informed supporter teammate. We have fully implemented and experimentally analyzed the supporter algorithms. We showed statistically significant lab results of their effectiveness.

References

1. D. Vail and M. Veloso. Dynamic Multi-Robot Coordination. In *Multi-Robot Systems*, Kluwer, 2003.
2. M. Veloso, P. Stone, and M. Bowling. Anticipation as a Key for Collaboration in a Team of Agents: A Case Study in Robotic Soccer. In *Proceedings of SPIE Sensor Fusion and Decentralized Control II*, 1999.
3. J. Fasola and M. Veloso. Real-Time Object Detection using Segmented and Grayscale Images. *Proceedings of ICRA'2006*, 2006.
4. H. Kitano, Y. Kuyinoshi, I. Noda, M. Asada, H. Matsubara, and E. Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1), 1997.
5. C. Marling, M. Tomko, M. Gillen, D. Alexander, and D. Chelber. Case-based reasoning for planning and world modeling in the RoboCup small-size league. In Proceedings of the IJCAI Workshop on Issues in Designing Physical Agents for Dynamic Real-Time Environments, 2003.
6. P. Rybski and M. Veloso. Handling diverse information sources: Prioritized multi-hypothesis world modeling. Technical Report CMU-CS-06-182, 2006.
7. P.F. Palamara et al. A Robotic Soccer Passing Task Using Petri Net Plans. Proceedings of AAMAS, 2008.
8. C. McMillen and M. Veloso. Distributed, Play-Based Role Assignment for Robot Teams in Dynamic Environments. *Proceedings of DARS*, 2006.