

Taste of Research Report

Robocup: Ball Tracking and Velocity

Belinda Teh

February 2011

Abstract

A major component of any soccer game is the knowledge and control of the ball. Thus, in preparation for rUNSWift's entry into the RoboCup Standard Platform League, improvements had to be made to the existing ball model. The problem with the current model was that it only recorded the ball's position using a very simple filter. However, by using an Unscented Kalman Filter, the ball's velocity could also be filtered, with more accurate and weighted estimates being produced. This additional information was then utilised for ball tracking, in particular for goalie behaviours. New motion stances also had to be developed to accommodate for new goalie reactions to the ball. This report presents the research and development undertaken as part of this Taste of Research project over the summer of 2010-2011.

Contents

1	Introduction	3
1.1	Robocup	3
1.2	Ball Model	3
1.3	Report Outline	3
2	Background	4
2.1	Existing Work	4
2.1.1	rUNSWift	4
2.1.2	B-Human	4
2.2	Issues	5
3	Method	6
3.1	Speed = Distance / Time	6
3.2	Unscented Kalman Filter	6
3.2.1	Absolute Co-ordinates	7
3.2.2	Robot Relative Co-ordinates	7
3.2.3	Off-Nao	8
3.2.4	Refinement	8
3.3	Behaviours	9
3.3.1	Absolute Filter	9
3.3.2	Robot Relative Filter	10
3.4	Motion	11
3.4.1	Goalie Sit	11
3.4.2	Goalie Dive	11
4	Evaluation	13
4.1	Results	13
4.2	Discussion	13

5 Conclusion	14
5.1 Future Work	14

Chapter 1

Introduction

1.1 Robocup

Robocup is an international organisation aimed at advancing the field of artificial intelligence by pitting robots against each other in competitions of soccer. Specifically, the Standard Platform League in which this project is under involves hardware-standardised humanoid robots (Aldebaran's Naos) so only the programming differs between teams. The ultimate goal is to be able to have a team of robots playing against a team of humans by 2050.

1.2 Ball Model

In order to control the ball, the robot must first be able to locate its position and track its velocity. The former was already in place, however without knowledge of the ball's velocity, the robot would easily lose track of the ball. For example, even if the goalie could see the ball, it would not know which direction the ball was travelling in. As such, the robot would not know how to move to block the goal effectively. If the robot had just lost sight of the ball, velocity could once again be used to calculate where to look for. Thus, it was of key significance that the ball model be adjusted to accommodate the ball's velocity.

1.3 Report Outline

The rest of this report discusses the issues involved in implementing the new ball model and related features. Chapter 2 provides some insight into the background behind ball tracking, while Chapter 3 continues on to the methods actually used. Chapter 4 presents the results and findings of the project overall, and finally, Chapter 5 suggests how future work could expand on this project and the possibilities it creates.

Chapter 2

Background

2.1 Existing Work

The problem of tracking an object's velocity is not a particularly new one, especially considering Robocup's soccer history. There are also many similarities with the methods used in localisation, as such, the system described in this report is based on foundations the past have provided.

2.1.1 rUNSWift

Although rUNSWift has not yet modelled the ball's velocity in the Standard Platform League, this was not the case in the 4-Legged League from pre-2008. In 2004, a multi-modal Extended Kalman Filter was used to track the state of the robots with a separate filter for the ball, and by 2006 these had been combined into one extensive filter which took advantage of the correlation between robot pose and ball position and velocity.¹ Although the hardware is different in the Standard Platform League and not all the robots would be tracked in the same filter, a similar approach could be taken for the ball filter.

2.1.2 B-Human

As the current reigning champions of Robocup's Standard Platform League, it is no surprise that B-Human have quite a sophisticated ball filtering system in place. Twelve Kalman Filters alone are used to track the ball, which of note include filters modelled specifically for a stationary ball,² as this eliminates the erratic velocities resulting from the noise produced despite the ball being still.

B-Human also has an advanced system compared to ours in that their goalie can react to incoming balls instead of simply standing in the middle, presumably due to their vast knowledge of the ball position and velocity. Their motion module is similar to rUNSWift's in that they manually specify the joint angles in order for their goalie to perform dives, however there is more room for flexibility. Namely, they are able to specify the stiffness for each joint, not just the robot overall, and it is this feature that allows them to move the goalie in the least damaging way possible by breaking it's diving fall.

¹Oleg Sushkov, *Robot Localisation Using a Distributed Multi-Modal Kalman Filter, and Friends*. (Undergraduate Honours Thesis, University of New South Wales, 2006).

²Thomas Rofer, Tim Laue, Judith Miller, et al, *B-Human Team Report and Code Release 2010*. (University of Bremen, 2010)

2.2 Issues

It can be difficult to evaluate the success of a velocity tracking system considering the inaccuracies involved in a human manually measuring a ball's velocity. Though with the help of Off-Nao, rUNSWift's graphical debugging system, the movement of the ball can be visualised on the screen and then compared to its physical movement to test its accuracy. Tracking a ball's absolute coordinates proves even more unreliable as this depends on the robot's location, which is handled by the localisation module.

The ultimate test would be in how the behaviour performs, although once again, the goalie movement would also depend on the motion module. Although the workings of a goalie dive are in place, one must always be careful of damaging the robots despite needing to test them.

Chapter 3

Method

3.1 Speed = Distance / Time

In order to become familiar with the existing code base, a simple start was made. Since the existing ball filter already tracked position, the distance between each respective position was calculated and divided by the time that had passed. The next step was to display this data on Off-Nao. Unsurprisingly, considering the single frame-to-frame calculations with all the noise each frame presents, there were large inaccuracies. However this step was useful in getting used to the code base and confirming that all the raw data was there, it simply needed to be treated appropriately.

3.2 Unscented Kalman Filter

As mentioned earlier, there are many similarities to the localisation module through the way both needed to filter information. As such, it was decided that a generic Unscented Kalman Filter would be created as a base class for both localisation and ball tracking. It was chosen over a typical filter due to the addition of sigma points around the mean and covariance, which would provide for truer weighted estimates. This method would require certain calculations, such as the square root of a matrix through Cholesky Decomposition, and so the libeigen library was chosen to handle these operations.

In order to accommodate the different uses of the filter, several modifications had to be made. Firstly it was turned into a template class, in particular to handle the different dimensions of the localisation's and ball's states and observations. The time update and observation update functions also had to be generalised for each subclassing filter.

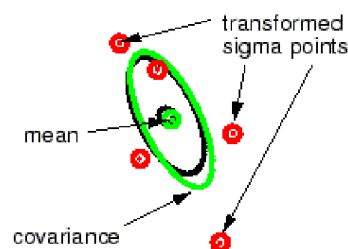


Figure 3.1: Sigma points around the mean and covariance of an Unscented Kalman Filter

3.2.1 Absolute Co-ordinates

Initial work began on filtering the more intuitive absolute cartesian co-ordinates in the form of (x, y) for position and (x', y') for velocity. The time update simply used the passed time to add the relevant amount of (x', y') to (x, y) to form the new mean, ie. using $x = x + x't$. As for the covariance, the following motion matrix was used to relate the ball's position and velocity, where 0.9 were chosen as the friction constants for updating velocity. This follows from taking the derivative like the Jacobian matrix used in Extended Kalman Filters, hence the 1s in the cross-section of position and velocity.¹

$$CovarianceUpdateMotionMatrix = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0.9 \end{pmatrix}$$

Now observations from the vision module were in the form of ball distance and heading and the filter co-ordinates were stored in x and y distance. In order for the prediction format to match the observation format for calculating the Kalman gain, they first had to be converted. This also had to be added to the pose of the robot itself. Thus the prediction was calculated as follows:

$$distance = \sqrt{(x_{robot} - x_{ball})^2 + (y_{robot} - y_{ball})^2} + variance_{distance}$$

$$heading = \frac{\pi}{2} - \arctan(x_{ball} - x_{robot}, y_{ball} - y_{robot}) - \theta_{robot} + variance_{heading}$$

where x_{ball} , y_{ball} , $variance_{distance}$ and $variance_{heading}$ are represented by the sigma points $sigma[0]$, $sigma[1]$, $sigma[4]$ and $sigma[5]$ respectively.

3.2.2 Robot Relative Co-ordinates

Since the absolute filter relied on the localisation of the robot and the observations themselves were passed in robot relative format, it was decided that a robot relative filter should also be created to further reduce inaccuracies. This system would also be more relevant for a goalie, as when viewing a ball at the goal, absolute co-ordinates would not be nearly as useful.

The time update for the robot relative filter proved identical to the absolute filter, just that instead of the mean being expressed as (x, y, x', y') , it would actually be $(ball\ distance, ball\ heading, ball\ distance', ball\ heading')$. The prediction in the observation update proved even simpler, as there was no need to incorporate the robot's pose. Thus $(ball\ distance, ball\ heading)$ was just updated as follows:

$$distance = distance_{ball} + variance_{distance}$$

$$heading = heading_{ball} + variance_{heading}$$

¹Oleg Sushkov, *Robot Localisation Using a Distributed Multi-Modal Kalman Filter, and Friends*. (Undergraduate Honours Thesis, University of New South Wales, 2006).

where $distance_{ball}$, $heading_{ball}$, $variance_{distance}$ and $variance_{heading}$ are represented by the sigma points $sigma[0]$, $sigma[1]$, $sigma[4]$ and $sigma[5]$ respectively.

3.2.3 Off-Nao

To test that the filter was actually working, its output was ported to Off-Nao to be visualised. First the values of the mean state of each filter were printed out onto the variable view tab, however, this was not particularly intuitive to the human mind. As such, drawing code was added to visually print the ball's current and expected location on the field view tab. 0.7 of the velocity was added to the current ball position to estimate the future location, with a red circle signifying the absolute filtered ball and a yellow circle signifying the robot relative filtered ball. A line was drawn between the current filtered position and the estimated position to represent the vector between them, as the longer the line, the faster the ball would be travelling.

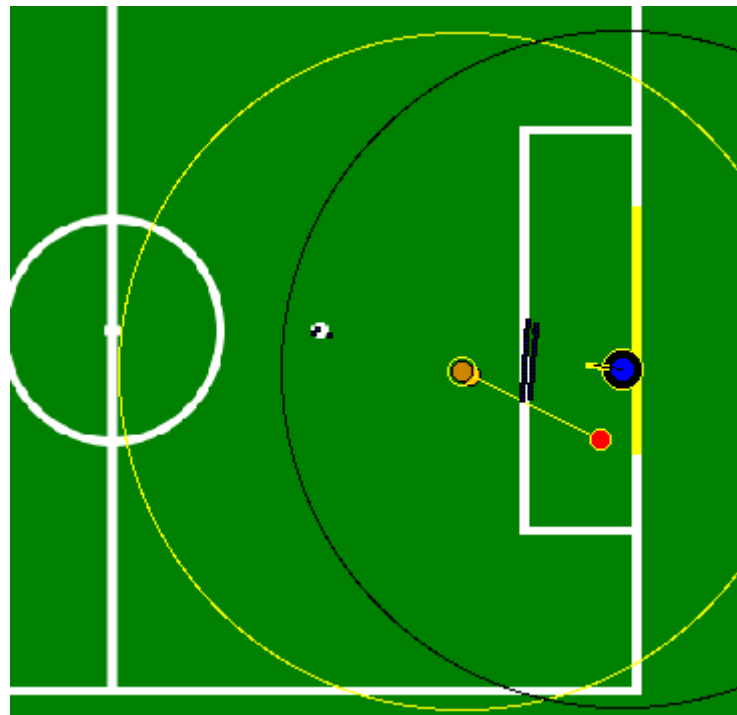


Figure 3.2: Off-Nao showing the predicted ball location from the absolute filter

3.2.4 Refinement

There were several modifications that needed to be made to improve the filters. Since the ball's mean state was always initialised to 0, when averaged with the first observation, the resulting velocity tended to be rather inaccurate. As such, the mean state would be set to the first observation, and then the filter would be applied with the observations thereafter. Now if the robot lost sight of the ball only to find it again travelling in the opposite direction, the filter would take some time to adjust and converge to the new state. Thus the filter would be reset after losing sight of the ball for more than a certain period of time.

3.3 Behaviours

The ultimate test of the filters would be through a robot's behaviour. The aim was to implement a goalie that could differentiate the ball's direction and thus react appropriately to save the goal.

3.3.1 Absolute Filter

Since the absolute filter was the first to be implemented, the initial behaviour depended on absolute values to determine the ball's location and velocity. Due to the nature of the field's centre being the origin, for ease of calculation, it was first assumed that the goalie would be at the right-hand positive side of the field.

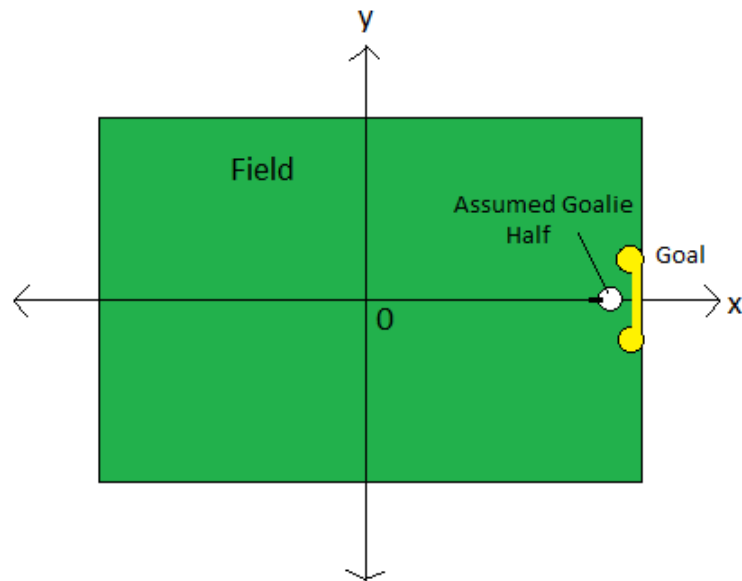


Figure 3.3: Absolute filter assumption in global cartesian co-ordinates

If $x' > 0$, then it could be assumed that the ball was travelling towards the goalie. Using the $speed = \frac{distance}{time}$ formula for x , calculations could be made to find the time taken to reach the robot, or rather the line the robot was standing on. This time t could then be subbed into a similar equation for y to find where along this line the ball would intersect.

Behaviourally, if the ball's estimated y position was greater than the robot's, then the robot should move to the right. Similarly if the ball's estimated y position was less than the robot's, then the robot should move to the left. Some adjustments were also made so that the robot would remain still if the ball was within range of the robot's centre. For further clarity on the actions of the robot, the eye LEDs were programmed to display different colours depending on what the robot chose, and its choice would also be voiced out loud.

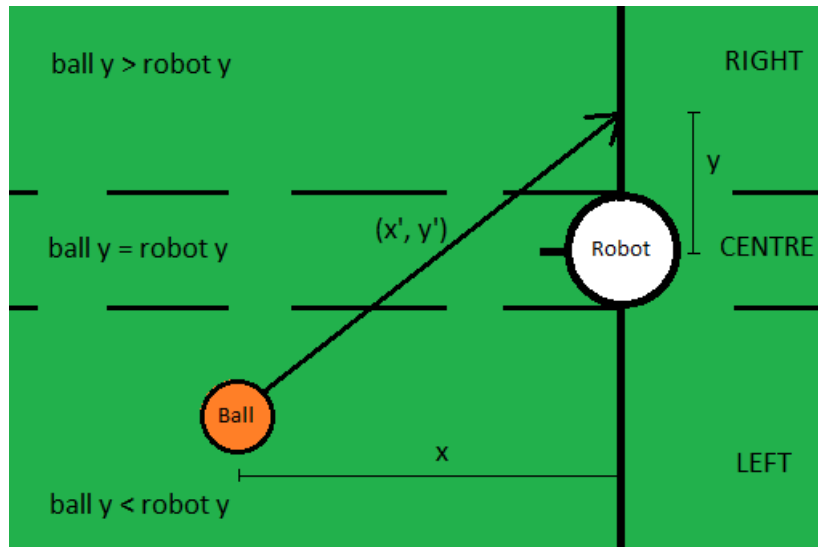


Figure 3.4: Use of absolute filter in goalie behaviours

3.3.2 Robot Relative Filter

Considering the inaccuracies of the current robot localisation module and the mathematical complications of the differing quadrants around the field, the robot relative filter was adopted in favour over the absolute filter. This time it was assumed that the line of the robot would be the x axis while the y axis would be straight through the front of the robot, ie. when the ball's heading is 0.

First, the distance and heading would be converted into cartesian co-ordinates relative to the robot. The typical use of cos for x and sin for y are switched due to the nature of the robot relative co-ordinate system and the assumption of the axes.

$$x_{p1} = distance \times \sin(heading)$$

$$y_{p1} = distance \times \cos(heading)$$

The velocities would then be added to this point, p1, to extrapolate the next point of the ball, p2. As a sanity check, the ball would only be rolling towards the robot if $y_{p2} < y_{p1}$. Once confirmed, the equation of the line of the ball could be derived from these two points. Assuming that the line of the robot is equivalent to the equation $y = 0$, the point of intersection between these two lines can be easily obtained. This results in an estimated x position along the robot's line and x axis.

$$m = \frac{(y_{p2} - y_{p1})}{(x_{p2} - x_{p1})}$$

$$y_{p1} = mx_{p1} + b$$

Subbing $y = 0$,

$$x_{estimate} = \frac{-b}{m}$$

Once again, behaviourally, if the estimated x position is less than the robot's, it should move left. If the estimated x position is greater than the robot's, it should move right. Finally, if the estimate is close to the centre of the robot, then it should remain still.

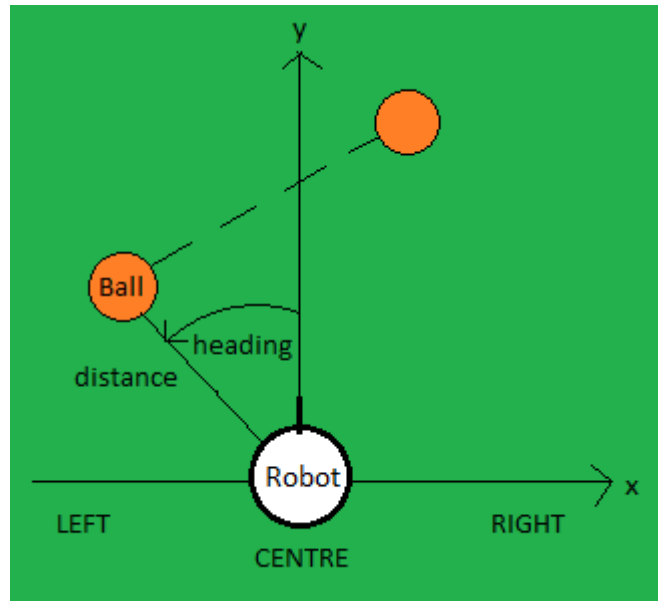


Figure 3.5: Robot relative co-ordinate system and its use in goalie behaviours

3.4 Motion

The next step of the behaviour was to have the goalie physically react to the ball in a manner that could actually save the goal. To understand how to produce a goalie dive, a case study was first performed on the motion module from B-Human. Since their joint angle files were in a different format, a perl script was written to convert them to rUNSWift format for ease and understanding. It was found that they used the flexibility of being able to set the stiffness of each joint to their advantage. Our current system was only able to specify a stiffness value for all the joints, so this was first redone with Brock White.

3.4.1 Goalie Sit

Before a dive could be developed, the lead in position had to be developed first. This became known as the goalie sit position, which involved the goalie squatting down so it could spring up again with additional power. The robot's arms would also be slightly raised in preparation, all with the ultimate aim of reducing the time taken to perform the dive overall.

3.4.2 Goalie Dive

It was found that the one of the major components to the goalie dive that was lacking was the fall of the robot. Thus through the tweaking of joint angles, the creation of a working left dive for rUNSWift's goalie had begun. B-Human would remove the stiffness of the left arm as it fell onto it, thus breaking the robot's fall. As such, a similar method was adopted, though the arm

positions were changed slightly to cover more ground once lying down. Another key tactic B-Human used was to roll forward after diving for a quick transition back to standing up. This strategy was implemented for our robot by rolling the left arm out of the way and twisting the hips towards the ground. Finally, another perl script was written to port the joint angles symmetrically over to the right dive.

Chapter 4

Evaluation

4.1 Results

The project was ultimately successful in that the robot could track the ball's velocity, and thus what direction it should react in when at the goal, whether it be dive left, right, or centre. However, the saving of the goal was not always quite perfect.

The timing of the goalie dive would sometimes be too late, meaning that the ball would get through to scoring. The robot would not always perform a perfect fall either, resulting in difficulties when standing back up.

The more serious issue would be the error margin in the data. There would be occasional and seemingly random appearance of NaNs in the filters' states. The filter would also sometimes take a long time to converge and display very large values. The problem with the noise of a stationary ball also caused problems, with a robot thinking it should dive even if the ball were still.

4.2 Discussion

The diving issue of the goalie is still a work in progress, and was more of an extension on the ball velocity in preparation for future work. This area was more of the experimental section involving motion and more work is still to be done, so it should be improved further. In particular, more care must be taken to prevent damage to the robot.

As for the NaNs, the source was pinpointed down to two sections of the code - the Cholesky Decomposition of the covariance matrix and the inverse of the prediction matrix. It would cause the robot to ignore all the behaviours related to the filters, and thus would not react to the ball at all. Once again, this was dependent on the libeigen library, though there are considerations being made to find alternatives means of square rooting a matrix, or perhaps even rewrite some of the formulas. For the convergence and covariance, methods such as incorporating the movement speed of the robot's head are being looked at to refine the update stages.

It would also have been beneficial if a more accurate method of testing the data produced had been developed. Though the filtered velocities of the ball generally looked correct from the drawing code, it was hard to tell what the margin of error was. This kind of information would certainly be more useful when introducing refinements and to measure the extent of the improvements.

Chapter 5

Conclusion

This project has demonstrated the ability of robots to improve their ball tracking ability through the use of an Unscented Kalman Filter. In particular, the addition of ball velocity to the ball model has benefited the goalie skills in that they can now differentiate how and where they must move to save a goal. It has also been shown to be possible to manipulate the robot's joints in such a way that it can dive to save that goal. These new features will certainly prove useful in the overall competition as maintaining knowledge and control of the ball is key to any soccer game.

5.1 Future Work

Though this project has achieved its goals, they are but part of the base of an infrastructure that is yet to be completed. In motion, there is a lot of work to be done to improve the goalie dive, such as reducing its fall damage and making it dive faster. There is currently no goalie position for a ball in the centre, so this must also be developed.

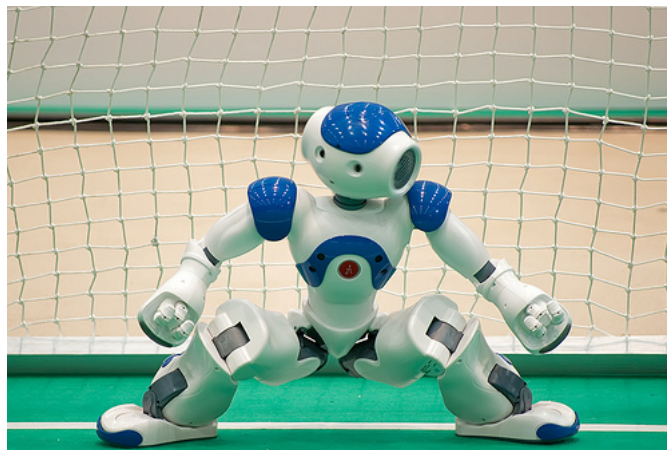


Figure 5.1: A possible centre position for a goalie with its widespread legs and low stance

On the behavioural level, ball velocity information can be applied in many other ways apart from the goalie. Firstly, the ball tracking behaviours could use the velocity of a recently lost ball to estimate where to find it again. Secondly, if the ball is being passed between two opponents, the robot could use the ball's velocity to estimate where to intercept the pass. Finally if the pass is

from a friendly team member, then the velocity could be used to predict the best reachable spot to receive the pass.

In the terms of the actual filter, there are also many improvements to be made. As mentioned earlier, incorporating the state of the robot and its head into the covariance equations could significantly improve the convergence rate of the filter, while fixing the NaNs with alternate equations would result in less behaviour crashes. One of the major refinements to be made is to make the filter multi-modal. It is particularly important that a separate filter be made to estimate the possibility of the ball being stationary, as the noise produced from a still ball can have disastrous effects on the interpreted velocity. Ideally, the observations from all teammates filters would be combined into one to improve the world model of the ball even further. This would have many benefits for ball tracking as even if a robot couldn't see the ball directly, for example if an opponent was obstructing its view, it could still use the information from its teammates to chase after the ball.