THE UNIVERSITY OF NEW SOUTH WALES

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

rUNSWift RoboCup Challanges 2005

Andrew Million Sianty

Thesis submitted as a requirement for the degree

Bachelor of Engineering (Software Engineer)

Submitted: 22 September 2005

Supervisor: Dr. Will Uther

Assessor : Prof. Claude Sammut

Abstract

RoboCup challenges provides an important ground for research of specific techniques that enable the robots to play soccer in real environment and use those techniques to overcome key constraints in this domain. In year 2005, Robocup offers three specific challenges: (1) Pick Up Soccer; (2) Variable Lighting Challenge; and (3) Almost SLAM Challenge. Within the research, we try to extend the success rate and accuracy by implementation of different techniques based on the previous developments. We adopt various mechanisms that enhance the performance of the players. This paper describes those techniques and mechanisms, our work with the RoboCup challenge and projects the direction of future research.

Acknowledgements

I would like to thank Dr. Will Uther and Prof. Claude Sammut for guiding me in the whole process of rUNSWift 2005 team. I would also like to recognize the contributions of many colleagues, especially Alex North, Nobuyuki Morioka, Wei Ming Chen, Joshua, and others who help us in the development of this research.

Contents

List of Figures

Chapter 1

Introduction

Challenges in RoboCup four legged league has been running for years, the main reason behind it is to display advanced research related to RoboCup. Apart from finding critical development, challenges is used for testing the current ability to the best. This year (2005) challenges are similar to last year(2004). There are a few changes introduce in year 2005, one of them is the changes of the field size. This impact on localisation which also impact the main code. previous efforts at solving the challenges need to be modified due to some changes applied this year. This paper will contain 3 major sections, each consists of detail documentation of 2005 rUNSWift Challenges. In 2005, for the Open Challenge we perform an idea of Pickup Soccer in RoboCup. Chapter 2 gives an overview of Pickup Soccer, discusses the packet structure, and changes to main code for strategy and compatibility.

Chapter 3 describes the Variable Lighting Challenge, the and evaluation on different settings. able to perform well, the chapter gives report of rUNSWift 2005 vision algorithm testing on different brightness and light level.

Last challenge to be discuss in this paper is Almost SLAM Challenge. Chapter 4 discusses the various approaches to solved the problem. The last, chapter 5 concludes the overall chapters.

1.0.1 Terminology

RoboCup International joint project to promote AI, robotics, and related field.

(http://www.robocup.org)

rUNSWift Team from The University of New South Wales / National ICT Australia (UNSW/NICTA) RoboCup four legged league (http://www.cse.unsw.edu.au/ robocup)

NUbots Team from Newcastle University. (http://robots.newcastle.edu.au/robocup.html)

AIBO A robot product from Sony that is used in the competition.

UDP User Datagram Protocol.

Main code / main stream code The rUNSWift code that is used for the RoboCup soccer competition.

Visual ball An object that recognised as a ball by vision algorithm.

GPS Global Positioning System develop by rUNSWift. The information derived from localisation processing visual images such as landmarks and line.

Pickup packet An UDP packet for Pickup Challenge. Describe in section 2.2.2.

Agent A code that is run in a robot(AIBO).

Player Same as above. It is use interchangeably with agent.

Worldviewer A tool for displaying top view of the field and displaying player position by accepting incoming UDP packets.

System rUNSWift code.

CMU Carnegie Mellon University.

Beacons Landmark for RoboCup soccer.

Half-beacon Beacons consist of two different color, half-beacon is the one of them.

Walk type A type of walk for robot. rUNSWift is developes many walk type to find the fastest and better calibration.

Feature points Pixels in the image that possible to be part of an object. (Beacon, Goal, field lines)

Chapter 2

Open Challenge - Pick Up Soccer

2.1 Introduction

The purpose of this chapter is to show the design and development of rUNSWift 2005 Open Challenge. This year, rUNSWift collaborates with NUbots(Newcastle University Robotics) to demonstrate the concept of Pickup Soccer. The idea derived from human behaviour when various soccer players gather in the park to play soccer. Developing robots to play soccer is not just making a system that works in limited constrains, but to overrule those constrains. With this Pickup Soccer challenge, we try to show that rUNSWift players are able to play in a team with other players that not from the same code. In other words, answering a question of is rUNSWift capable of playing pickup soccer with any team.

The pickup soccer idea is explained in section 2.2. There is a UDP Packet that has to agree. The packet is for communicating between robots. In addition, the packet is minimal in terms of contents in information that being communicate. Later described in section 2.2.2, the packet could be used easily by any other teams. The packet contains basic common information and explicitly not customised to have behavioural information. However, to be able to play as a team, the player need to understand what other players trying to do. As an example, if a player is chasing the ball, you might want to become a defender. The packet does not contains the role or behaviour information, because if it does then it would just become a team with well-known strategy. Therefore we can only predict the behaviours based on basic knowledge. As a real example, human player does not know what other player trying to do, but, he or she

knows the location of that player and could predict the behaviour. The prediction of teammates' behaviour is discussed in section 2.4. After the behavioural information is predicted and changes are compatible to the main code, then it could use current rUNSWift game play code. For this pickup challenge, we use Striker/Defender formation that is developed by M.Nobuyuki[Mor05]. In Osaka 2005, although with no trial game before hand, we are able to show an exciting match between mix players of rUNSWift and NUbots against the German Team. In section 2.7

2.2 The idea

Eight kids with AIBOs is gathering in the park and they want to have an AIBO soccer match, the field is set up in the park so they plug'n'play pickup soccer code from rUNSWift or different code from other teams which have its unique ability to play soccer.

The next few subsections will explain in detail the principle of pickup soccer and how it is related to RoboCup, the basic information, and the rules. A summary of the challenge can be found in rUNSWift/NUbots combined 2005 Open Challenge Entry document[DWU05],also provided in appendix.

2.2.1 Pick up soccer

As mention in chapter 2.1, pick up soccer is one in which anyone is welcome to play soccer. Players is basically show up and play in a game. In the case of Robocup, the players is the corresponding code from a team that being loaded to an AIBO. The difference between specialised practised team and the pickup soccer team is that players in pickup soccer not having any agreement on strategy or position before the game. Apart from that, similar to human teams which each individual has unique ability, each team's code has its uniqueness. As RoboCup is more mature every year, new currents teams come up with major improvements in different aspects of artificial intelligent that modify the system. Although the teams might share some common base code, they tend to have different strategies, behaviour skills, visual readings, localisation, etc. However, as the code become more complex due to strategies become more specific, teams might tend to make overly specialise so the code could only works with their same code or team.

We believe that specialisation to team behaviour such as roles positioning and strategy is

necessary to form a strong team. For instance, professional soccer team trains their players. However, without strategy decisions, each player should be robust enough to be able to play soccer. In order to understand a common 'language', the form of communication and the 'language' itself has to be simple and contains no behaviour specific information. For example, the information about the next move of the players is not provided, but other players can predict the intention by looking at its destination. Based on simple information, the aim of pickup soccer is that each player is able to interpret that information to higher level behaviour and play soccer as a team.

2.2.2 UDP Packet

Just like human, the communication between players is language, in robots, the communication is based on UDP packet. The proposed UDP packet structure is shown below.

```
#ifndef PICKUP_H
#define PICKUP_H

static const int PICKUP_UDP_PORT = 9000;

struct PickUpSoccerBroadcastInfo {
    char header[4];    // "PkUp"
    int playerNum;     // 1-4
    int team;          // 0 is red 1 is blue

    // position of robot
    float posx, posy;
    // same coordinate system as the localization challenge
    float posh;
    // degrees, 0 along the +x axis, increasing counterclockwise

    // variance of position of robot (n.b. not full covariance matrix)
```

```
    float posVar, hVar;
    // pos variance is maximum of individual dimension variances

    // position of ball
    float ballx;
    float bally;

    // variance of position of ball
    float ballvar;

    // destination of robot
    float destx, desty;

    // If visualy see the ball
    bool ballSeen;
};
#endif // PICKUP_H
```

As it describes, without being tuned for any behaviour, the generic information only consist of position and heading of a player at destination and position of the ball. Section 2.4 discuss how to interpreting the UDP packet for this challenge.

Note that the position and heading is based on global coordinate system. In order all players to interpret data the same way, they require some agreements. For simplicity, the coordinate system is the same with coordinate system on Almost SLAM Challenge, which is discussed later in chapter 4. The coordinate system is originally come form the first localisation challenge for RoboCup four legged league, written by CMU.

As shown in figure 2.1, the center of the coordinate system is in the center of the field where x axis goes positive to the yellow goal. Angle zero in positive x axis and increases counter clockwise. Positive y axis is 90 degrees angle from positive x axis.

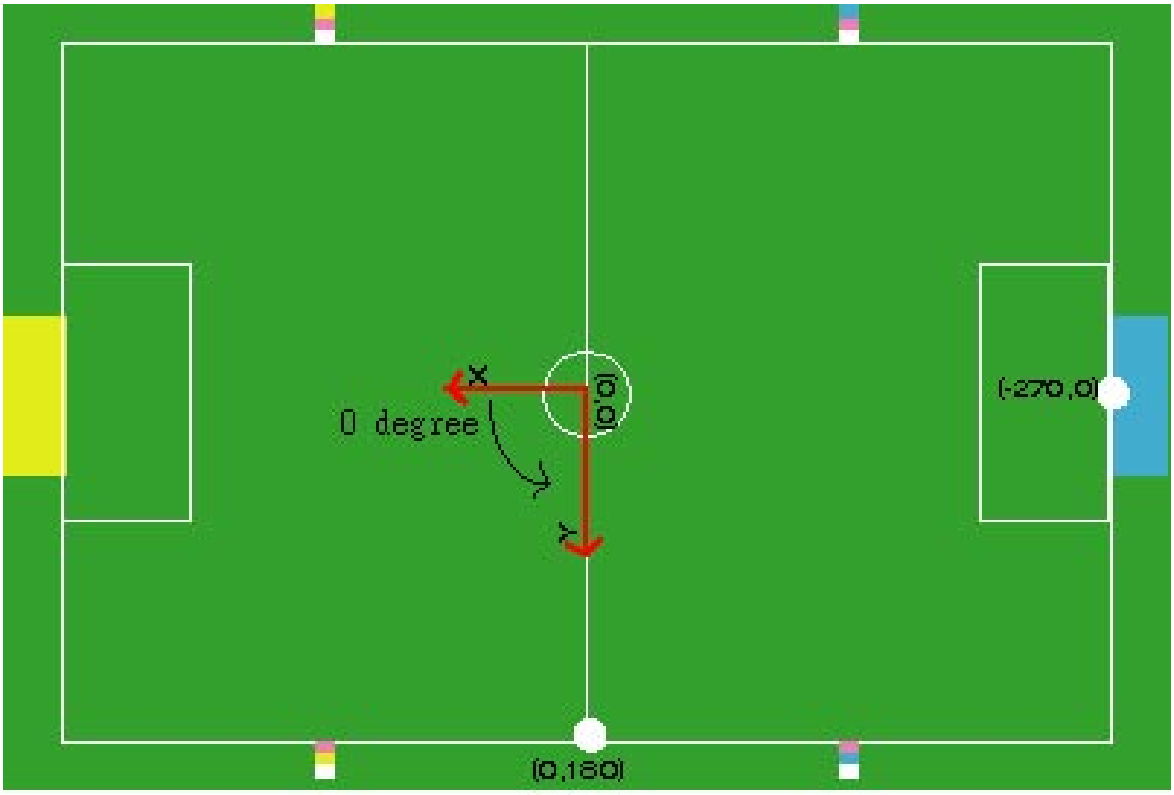


Figure 2.1: Diagram of global coordinate that is used for Open Challenge

2.2.3 Rules

Since we want the open challenge to be considered as playing soccer like normal game, we use the normal rules [Com05a]. In addition, rUNSWift did not engage in any other cooperation except agreeing on the UDP packet for this challenge. Rules of the game can be found in [Com05a].

2.3 Problem Statement - Changes to the code

After we defined the requirements, the next point is to identify the changes required for the current system (rUNSWift 2005). There are numbers of potential problems identified, from simple to complex:

1. Changes in global coordinate system.

2. Changes in current code for UDP packet structure to enable the reading and writing the Pickup Challenge Packet

3. Changes in behaviour interpretation since the packet is simple and does not send rUNSWift behaviour information such as type the of roles, signal other team members when the attacker attempts to shoot, and so on.

4. Changes to support rUNSWift team behaviour and formation. If the changes are compatible to the main code then it can avoid the cases such as all players chase the ball or all players avoid the ball as they assume that other player hold control of the ball, and so on.

5. Kick off position problem, since the packet does not contain kick off role.

2.4 Proposed Solution

Figure 2.2 is showing the changes to the rUNSWift 2005 code in performing this challenge. Modification to the UDP constructor for changing the code to receive and generate Pickup packet. It also filters the incoming UDP packet that are not pickup packet, and not from the same team. The rest of the solution are divided into three subsections. Next section provides solutions to problem 1, 2 and 3 which interpreting the packet data to low level code. Next, section 2.4.2 shows a way which using the packet information to solve high level behaviour information. Once the changes are all compatible to the role switching code, picking a formation to use and the reason is explained in section 2.4.3. Section 2.4.4 shows the solution for the remaining problem, kickoff position.

2.4.1 Intercept and reconstruct Pickup UDP packet

The way for solving the coordinate system without modify underlying calculations and algorithm is by injection of layers. This is one of the way to get global coordinate work with our main code.

To solve the different coordinate system, from rUNSWift 2005 coordinate system to pickup challenge coordinate system is basically the translation of a point. Coordinate system for rUNSWift is dependent with the team color, when the team is blue (as shown in figure 2.3) and when the team is red (as shown in figure 2.4). Let position from the packet is ($Packet_x$,

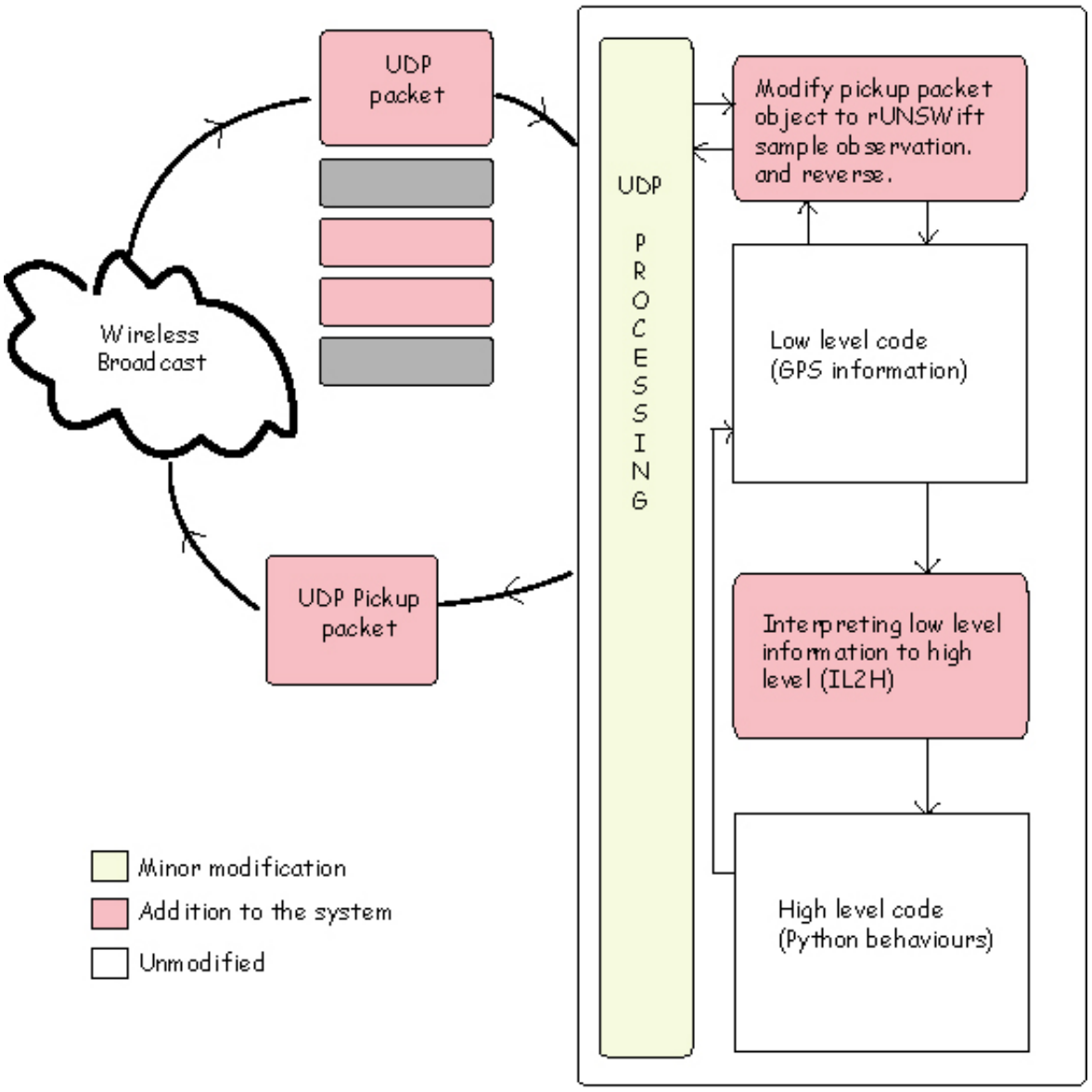


Figure 2.2: The proposed solution, which contains minor changes at three points in the system.

$Packet_y$) and position in rUNSWift coordinate system is (Pos_x,$Posy$). Suppose the team color is blue,

$$\begin{aligned} Packet_x &= Pos_y - 0.5\,FIELD\_LENGTH \\ Packet_y &= 0.5\,FIELD\_WIDTH - Pos_x \end{aligned} \tag{2.1}$$

Where the FIELD_LENGTH is the length of the field, and FIELD_WIDTH is the width of the field. Note, as section 2.2.2 mentioned that the coordinate system is symmetrical, depends

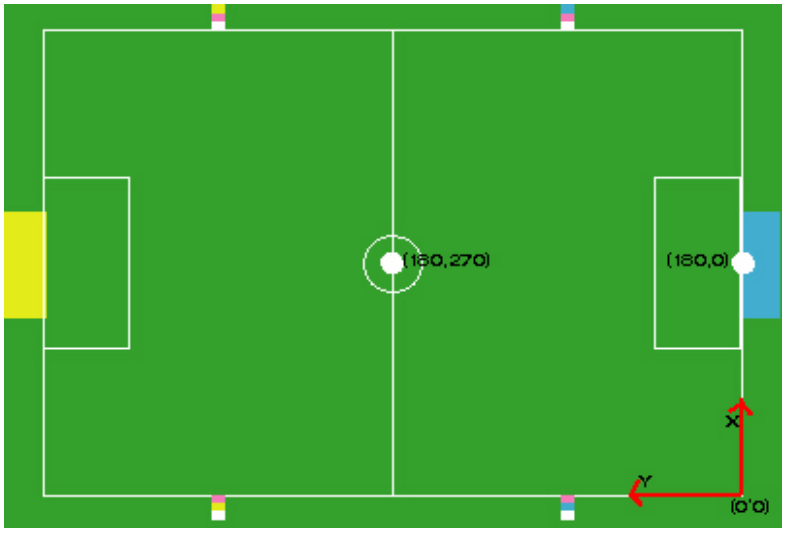


Figure 2.3: rUNSWift coordinate system when the team is blue.

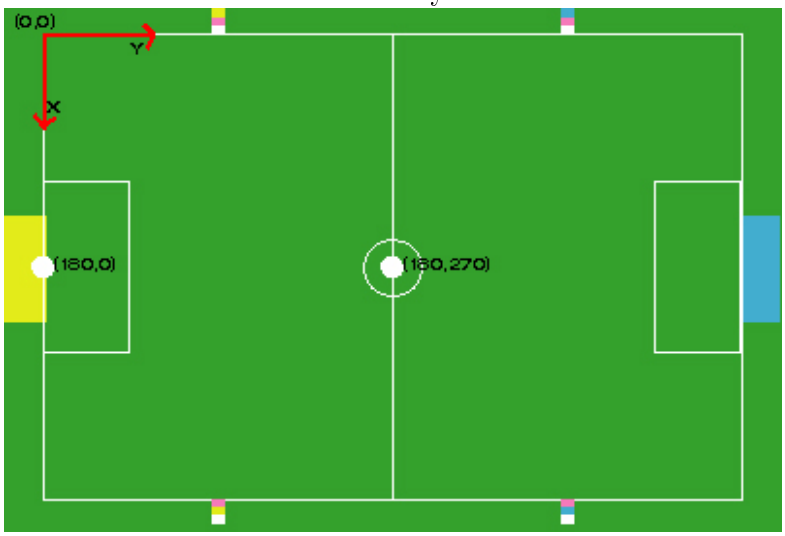


Figure 2.4: rUNSWift coordinate system when the team is red.

on the team color. Therefore, we have two conditions whether our team is blue or red. If team color is red,

$$\begin{aligned} Packet_x &= 0.5\,FIELD\_LENGTH - Pos_y \\ Packet_y &= Pos_x - 0.5\,FIELD\_WIDTH \end{aligned} \tag{2.2}$$

In the opposite, from challenge coordinate system to our coordinate system, the transformation apply as follows,

if receiving as blue team:

$$\begin{aligned} Pos_x &= 0.5\,FIELD\_WIDTH - Packet_y \\ Pos_y &= 0.5\,FIELD\_LENGTH + Packet_x \end{aligned} \tag{2.3}$$

if receiving as red team:

$$\begin{aligned} Pos_x &= 0.5\,FIELD\_WIDTH + Packet_y \\ Pos_y &= 0.5\,FIELD\_LENGTH - Packet_x \end{aligned} \tag{2.4}$$

Variances for position and heading can be directly feed into GPS system. However, ball position need to be filtered. Since the ball position given by other agent might not correct and not come from visual observation, and our algorithm will work better if given observation data, not precomputed or prediction. For example, other system might set the ball position to some undesirable value when the AIBO is not seeing a visual ball or other system might increase ball variance only by a small value on the other hand, when the ball is rolling fast or get moved, and etc. Those examples are all undesirable observation sample, by removing those we will have less sample error.

The 'ball seen' flag can be use as indication that the position is not an observation data. To make less changes to the internal code, the wrapper is able to filter these data by changing the ball variance to a large number when the agent is not sensing a visual ball. This case will then filtered out from sample observation. Therefore, it will not update ball position using data based on other agent's prediction instead taking only valid data(visually identify the ball). Therefore, this solution will exclude all uncertain value that is given by other system and gives better ball predictions.

2.4.2 Analysing Pickup packet for behaviour and formation

As previous section mentioned that the changes is integrable to the main code, the game playing behaviour could use the current 2005 rUNSWift formation. The dynamic and high level behaviour implementation can be found in [Mor05]. This section largely contains implementation of pickup packet in supporting rUNSWift roles positioning and briefly explains the formation that is used for Pickup Soccer Challenge.

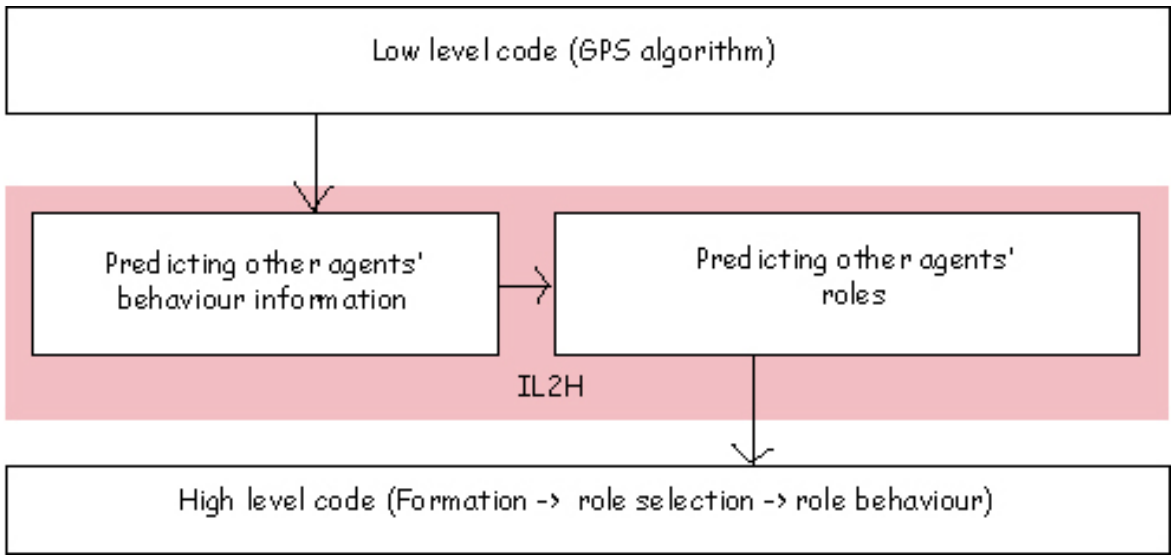


Figure 2.5: Another layer for predicting high level information of other agents using low level information.

Previous section shows that for each of the information in the packet has been matched to our structure. Nonetheless, rUNSWift behaviour code requires other agent's behaviour information that is not given by the Pickup packet. The main stream code need behaviour information such as the role, the 'time to reach ball' of other teammates.

The list of residual high level behavioural information that have not been described:

Has seen ball This information is set to True when the agent see visual ball. This can be retracted from the Pickup packet.

Has lost ball The opposite from 'Has seen ball', the information is set to True when the agent unable to see visual ball.

Time to Reach Ball This information contains prediction on distance between the agent and the ball. This is largely necessary for role switching. The function that is used in the code could be found in [Mor05]. The function to calculate the time to reach ball takes current position, current heading, destination position, and the heading at the destination location. The function returning a number to estimate the time to move to the destination location. For Pickup challenge, if an agent(A) needs to calculate other agent(OA) then by using the function above, passing OA's position, OA's heading, ball position and heading

at the ball location. Note that ball position is selected by using the most accurate ones (best variance between OA's estimate of the ball, A's estimate of shared ball position, A's estimate of the visual ball).

Has grabbed ball This information is set to true if an agent grab the ball. This is used mainly for 'get out of the way' behaviour so the attacking agent can have clear shooting range. In this challenge, there is no information in the Pickup Packet that can interpolate an agent grabbed the ball. However, testings show that it is enough to estimate the value by distance between the agent and the ball location. Note, the value is not corresponding to the agent grabbed the ball but corresponding to the time when an agent should execute the 'get out of the way' behaviour. Therefore, it is set to true when an attacker agent is really close to the ball, which could means grabbing or kicking the ball, and everyone has to 'get out of the way'.

Roles Type of role. For this challenge, we use Attacker, Defender, and Striker. Since the pickup packet does not specify these information, to be able to use rUNSWift formation the roles has to be specify by prediction. This is done in sequence so no overlap occurs. The way for predicting the role is shown in pseudocode 1.

Since the destination is specified, we can predict the intention of the agent by checking the destination. First, assess whether it is an attacker by checking if the agent moving to ball position. Note, the ball position is selected using the best variance as mentioned in Time to Reach Ball prediction. This is due to the attacker agent might not see the ball in every frame (eg. localising while walking to the ball). The value sent by the attacker might not be correct or inaccurate. Therefore, we can not use the exact position of the ball to indicate the attacker walking to the ball. That is why using variance of the ball to indicate the attacker walking to somewhere close the ball.

Role counter The number of frames to indicate the agent having the same role. In main rUNSWift code, every agent sends its' role counter. Since the Pickup Packet does not pass this information, the receiving agent creates the local copy of the counter for other agents.

Therefore, by using above solutions we can determine the behaviour and role of other agents, as shown in pseudocode 1.

2.4.3 The game play

After the behaviour information is predicted by the method described above, those changes are compatible for using rUNSWift formation which has dynamic role switching. This section broadly describing the chosen formation and explaining the reason, more detail of formation can be found in [Mor05].

In this challenge, we use Striker/Defender formation. Using this formation gives agents' position to spread across the field so the agents are able to switch role and attack or support the other player easily. Choosing this formation rather than Supporter/Defender formation(formation that is use main rUNSWift code) is due to the main formation has close position between attacker and supporter. Though it has some advantages, for example, handle large area near the ball, the supporter has to know the possible behaviour of attacker and both of them are needed to works closely (behaviour derived from the test and modified so they will not block each other). The Striker/Defender formation gives less coverage near the ball because the position spreads, however, it has more coverage of the field. Moreover, the striker would be able to support the attacker or defender easily. It works by trusting the agent to handle the area, and if enemy blocks the ball or teammate pass the ball to the other side of the field, the striker is ready to intercept and attack.

2.4.4 Kickoff formation

In addition to the game play, the kickoff position is important. Since no other source information apart than using the Pickup Packet structure, this problem also being resolved by estimating kickoff position of other agent by looking at the target location.

There are some kickoff formations allowed by the rules [Com05a] and used in rUNSWift kickoff formation. rUNSWift has two kickoff formation when kicking off, displayed in figure 2.8(a) and figure 2.8(b). There is one formation when receiving kickoff, shown in figure 2.8(c). Firstly, selecting the formation depending on the team state, whether is kicking off or receiving the

Algorithm 1: Predicting other agents information and role

```
begin
    for aForward ∈ otherValidForwards do
        destination ⟵ aForward.getDestination()
        ballPosition ⟵ aForward.getBallPosition()
        playerPosition ⟵ aForward.getPosition()
        ballVariance ⟵ aForward.getBallVariance()
        if ballVariance > FIELD_DIAGONAL then
            aForward.hasSeenBall ⟵ False
            if sharedBallisbetter then
                ballPosition, ballVariance ⟵ sharedBallPosition, sharedBallVariance
            else
                ballPosition, ballVariance ⟵ myBallPosition, myBallVariance
        else
            aForward.hasSeenBall ⟵ True
        aForward.hasLostBall ⟵ not(aForward.hasSeenBall)
        prevRole ⟵ TeamRoles[aForward]
        if distance(destination, ballPosition) < ballVariance then
            aForward.setRole(ATTACKER)
        else if destinationtothedefendingfield then
            aForward.setRole(DEFENDER)
        else
            aForward.setRole(STRIKER)
        TeamRoles[aForward] ⟵ aForward.getRole()
        dkd ⟵ getHeadingBetween(ballPosition, TARGET_GOAL_X, TARGET_GOAL_Y)
        bonus ⟵ getBonus(aForward)
        t =
        timeToReachPoint(ballPosition, dkd, playerPosition, aForward.getHeading()) −
        bonus
        aForward.setTimeToReachBall(t)
        if prevRole = TeamRoles[aForward] then
            aForward.setRoleCounter(aForward.getRoleCounter() + 1)
        else
            aForward.setRoleCounter(1)
end
```

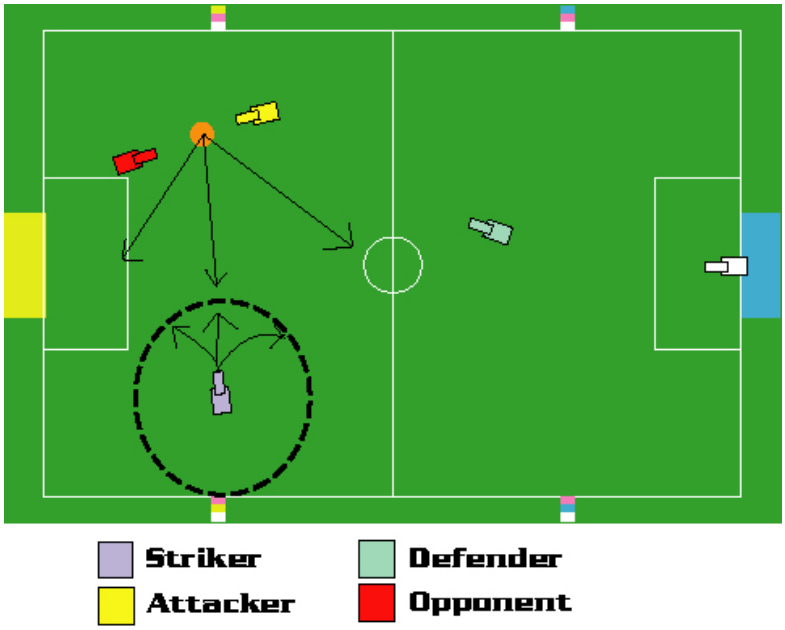


Figure 2.6: An example of striker formation in attacking half. Attacking yellow goal.

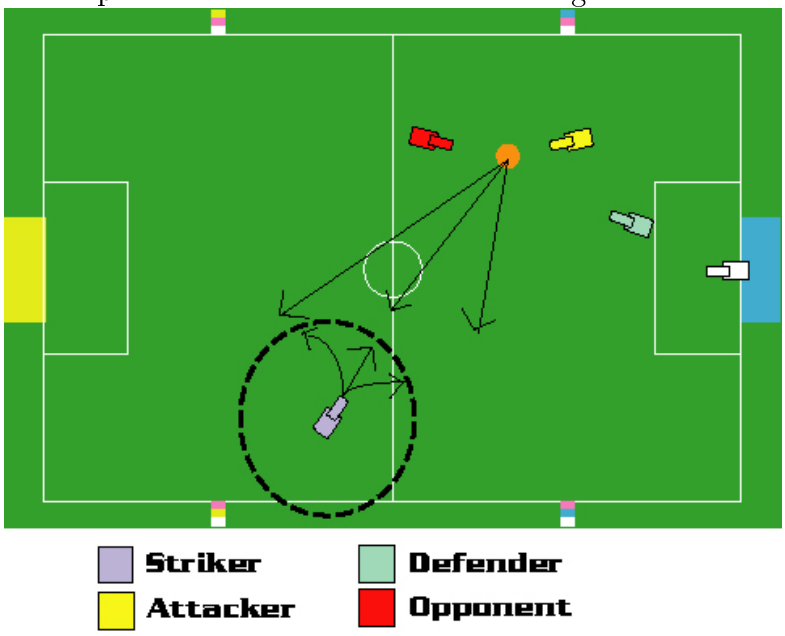
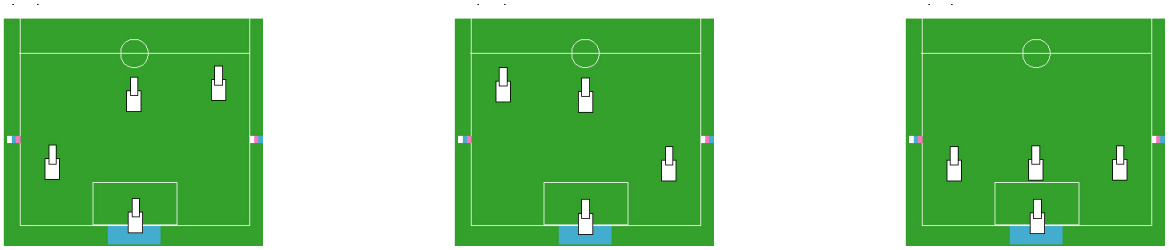


Figure 2.7: Striker formation in defending example. Attacking yellow goal.

kickoff. From those, an agent decides on a position by the selecting the nearest position from the agent location.

In this challenge, the kickoff position is selected by using other agents' destination. Primarily to identify the position that other agents' selected. By separating other agents' position to three



(a) **(b)** **(c)**

Figure 2.8: The rUNSWift kickoff formation.

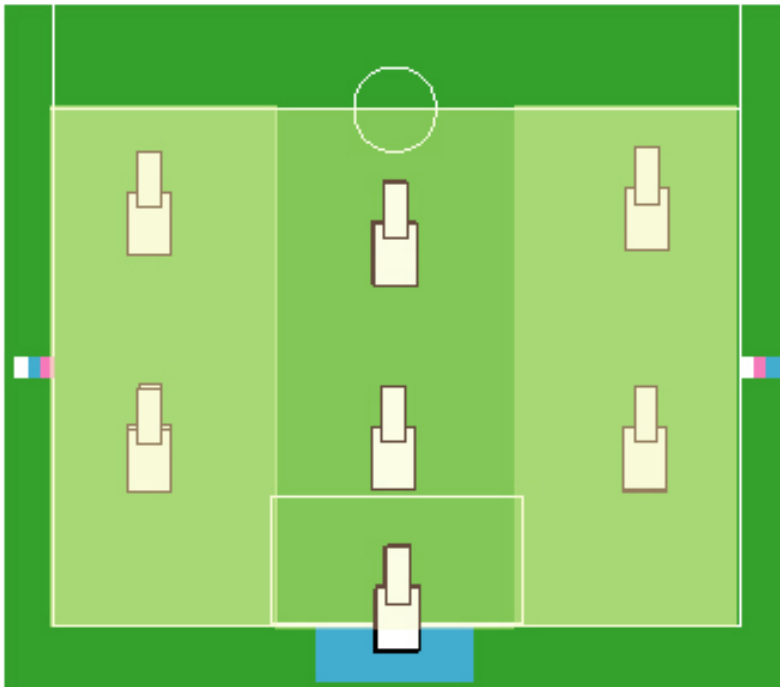


Figure 2.9: Possible location area for kickoff.

areas would identify which position has been taken, shown in figure 2.9. Based on that, if any of other agents have taken the position then change position to an unattended position on random. This randomness is introduced to avoid constantly changing the position while the collision agent also change position on the same rate. Finally, checking the only two forward positions. The rule states that when kicking off there can only be two players positioning in front of the beacon position. Therefore if number of forward positions less than two then switch the position to

forward on random time. The implementation can be found in the code(pReady.py).

2.5 Evaluation

Before the open challenge, we have done tests in the laboratory which are essential to ensure the packet is sent and received correctly, and the role is switching as expected.

On the open challenge showcase, we set up 8 AIBOs with 4 rUNSWift and 4 NUbots codes. Audience then pick which 4 players that play on the game, which resulting 1 forward and 1 goalkeeper from rUNSWift. For the opponent, we would like to thanks to German Team providing the opponent team.

2.6 Results

Team rUNSWift and NUbots played againts German Team (the winner of RoboCup 2005). We were able to create few chances and hold the score 0-0 for 6 minutes game. We coordinate well as a team, and goalkeeper successfully hold the GermanTeam attacker to conceive a goal. The final result is a mark of 21.5, which is the second highest mark at RoboCup Open Challenge 2005.

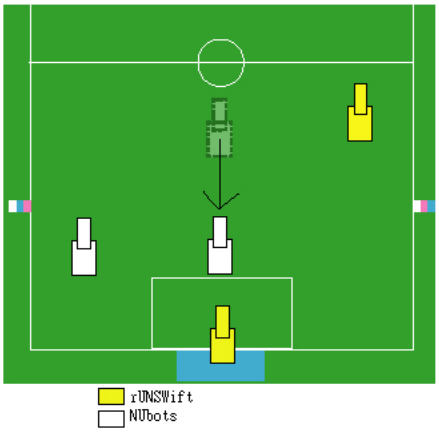


Figure 2.10: Kickoff position at RoboCup 2005 Open Challenge.

There is an unexpected kickoff positioning, during kick off the position of two NUbots AIBOs are behind the beacon and the center player is not move to kickoff position as shown in figure 2.10.

Our player stood still in the position as expected, due to no other teammates were taking right area. We later found that NUbots did not send destination information, which might be the caused for kickoff problem. This arose interesting discussion of why the players still perform well. One of the possible reasons is the criteria of time to reach ball, as explained in section 2.4.2 and [Mor05].

2.7 Conclusion

With this challenge, we have shown the integration in terms of game played between different teams' code, behaviour and strategy to play soccer as a single solid team is achievable. This is done by only agreeing on a UDP packet that contains common information without describing other team's behavioural information. In the challenge also demonstrated that rUNSWift code is robust, as players give well performance in the pickup soccer challenge and also laboratory tests.

2.7.1 Future works

The perfomance is could be increase by improving the overall system. In figure 2.10, if the rUNSWift would like to take cover the center part, it could also be handled by introducing tighter check for kicking off position. In other words, the middle area could be tighter by checking only the top part(middle area in front of beacon), as shown in figure 2.11.

Pickup Challenge could also be extended to more broader views. There are a great amount of ideas for the extension, some of the examples are :

- With the field is getting bigger and number of AIBO player in the team increasing, an all-star game with mixed players from different teams is easily achievable.
- Boosting up the marketing and commercial point of view, by giving away or selling memory stick containing random teams' code with pickup soccer setting so any user can play without knowing the details of the code. This will increase the public knowledge of RoboCup, the hardware company (Sony AIBO), the teams, as well as each individuals.

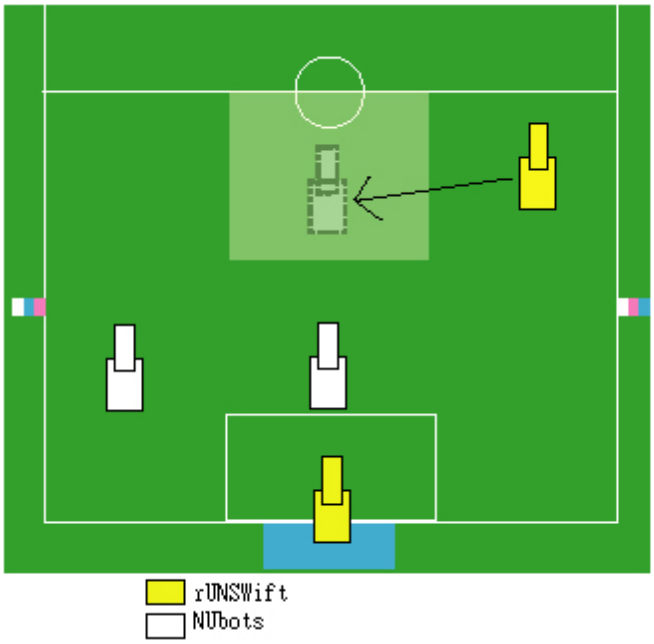


Figure 2.11: Possible change to take cover the center position.

Chapter 3

Variable Lighting Challenge

3.1 Introduction

As RoboCup aim is to beat world best soccer players by the year of 2050, one of the target is able to play outdoor. The outdoor environment has shadows, lighting temperature and lighting direction are mixed and random. The current soccer games for 4-legged league are still in indoor and with stable lighting temperatures and a maintained brightness level. Instead of trying to solve the entire problem, section 3.2 would discuss on the problem and the constrains that variable lighting challenge trying to solve. Variable lighting challenge is a challenge to test the robustness of the system with different lighting conditions. This challenge could be used to evaluate different sectors in rUNSWift code. While different lighting intensity would test the vision level, with undetected beacons or goals causing less sample observation to the localisation, and also performance of rUNSWift behaviour where chasing the ball that appear less frequently. In the section 3.3 would suggests a way to restoring the quality and quantity using rUNSWift vision. In order to test the solution, in section 3.4 would contains evaluation on different lighting condition of vision algorithm for rUNSWift 2005 which is developed by N.Alex [Nor05]. Before the competition, we mainly focused on low intensity problem, where in the competition, the lighting condition is brighter than normal game conditions. Apart from the white lights for the RoboCup game, there were 4 extra lights that have different lighting temperature and the color is a bit orange.

3.2 Problem Statement

The lighting condition for the challenge would be unknown for all the teams until just before the challenge. In addition, there would be a schedule that is just decided before the competition. With the schedule, the challenge not only have different lighting condition than the normal game, but also the lighting change according to the schedule on different part of the field, there might produce shadows as well by covering some of the lights.

As mentioned in the Challenge Rule 2005 [Com05b], the evaluation based on the performance of scoring the goals with a special penalty shoot rule. On this challenge, the participant would use a blue uniform AIBO and being placed at the center of the field. There would be two red uniform AIBOs, one placed somewhere on goalie position and another one is placed outside the goal box area. Both of the red AIBOs are placed just before the competition and it is stationary for the whole challenge. The challenge is to score as much as possible in 3 minutes time limit. At the beginning and every goal, the ball is moved back to the center position and back button is pressed. When the ball leaves the field, it would be placed at the center without any indication sent to the blue robot.

Different lighting intensity and direction affecting the colors, which then impact on recognising goals, balls, and landmarks. In order to recognise an object, vision algorithm starts by finding the feature points. Feature points is selected and grouped depending on the color classifications. Color classification is normally being pre-calculate offline, depending on the lighting conditions. With brighter lighting, the color value of the red uniform might be shifted to orange's range(orange ball).

The problems that we would like to focus on:

- Different level of brightness on different part of the field.
- Shadows, which reduce the light that is reflected from objects(landmarks, ball, goals) to the camera.
- Behaviour that need to be changed, reflected to the rule and the proposed solution.

3.3 Proposed Solution

In 2005, rUNSWift rewrite the vision algorithm, more on the algorithm is explained in [Nor05]. With the new vision, we hope it would produce more robust vision. From the list of problems listed above, all affected if vision does not perform adequately. On the other hand, if vision quality and quantity can be restored or could perform similar to normal lighting condition, the problems might reduced to vision problem. However if the quality and quantity are changed quite dramatically, we need to identify the effects on other system from localisation up to behaviour level. Therefore, we need to tackle the problem by priority and individually. Methodology start by solving vision difficulties, localisation and finally the behaviour.

To identify the solution, we gather sample data on the different intensity. With the sample data, we start with testing the performance by changing the camera settings for the AIBOs. The camera settings that AIBOs have are,

- White balance. This is for adjusting the white balance, usually different lighting temperature could affect the overall color. Contains selection of Indoor mode, Outdoor mode, and Fluorescent mode. At the laboratory, we use Indoor mode.
- Camera gain. This contains selection of low, medium, and high. The higher the more detail('graininess').
- Shutter speed. This contains selection of slow, medium, and fast.
- Automatic white balance. This setting for automatically adjusting for white balance before taking an image.
- Automatic exposure. This setting for automatically try to find the best aperture and shutter speed.

The automatic settings have disadvantages, which are low frame rate due to mechanical adjustment and unstable results due to different exposures. The settings that are used for the normal game condition in the laboratory are not using the automatic setting because of the problem stated above. The current settings are indoor white balance, high gain and fast shutter speed.

In the next section, would show that by experiments with camera setting, it is sufficient to shift the color to the normal setting so the vision algorithm able to recognise the important feature points. Next section would also consist the methodology of evaluating the different camera setting and the overall performance by running the challenge.

Behaviour of running the challenge is similar to the behaviour for normal attacker. The player is modified to ignore the penalised command and listening to the back button that indicate the ball being placed back to the center. The GPS ball position also being reset to the center position to incorporate with finding ball behaviour.

3.4 Evaluation

Testing the camera settings are done by the following,

1. Set a lighting environment
2. Prepare set of position and heading for an agent to take images. For examples, facing the goal from the center, looking at the ball and the red robot, etc.
3. For each set of camera setting, run YUV logs for the positions above.
4. For each lighting environment changes, repeat above step.
5. Evaluate the YUV logs on the number of landmarks, balls, and goals detected. And also evaluate on the error of misclassified the object.

If a set of setting is sufficiently have less error rate, further test to the setting is done by performing the 3 minutes challenge. In addition, for the overall evaluation, apart from the laboratory tests, we also have friendly match test with NUbots and at the competition in Osaka.

3.5 Results

Since we want to have the same detailness, the gain was not changed. The white balance is using indoor which the best for the laboratory setting. Therefore we conduct a experiment on different

intensity with different shutter speed. Each of the test with an object contains approximately 4 - 5 images.

Results for testing the camera setting

The lighting environments are defined as follows,

Normal lighting	The normal lighting condition in the laboratory. All the lights is on.
Turn off main lights	The 4 projection lights is turned off.
Turn off white lights	The main lights is off. And the white lights also is turned off.

Camera setting	Environment	Ball detected	Beacons detected	Goals detected
Low shutter speed	Normal lighting	80%	100%	100%
Low shutter speed	Turn off main lights	80%	80%	80%
Low shutter speed	Turn off white lights	20%	20%	60%
Mid shutter speed	Normal lighting	80%	60%	100%
Mid shutter speed	Turn off main lights	50%	80%	100%
Mid shutter speed	Turn off white lights	0%	20%	50%
High shutter speed	Normal lighting	100%	80%	100%
High shutter speed	Turn off main lights	50%	75%	50%
High shutter speed	Turn off white lights	0%	0%	0%

Camera setting	Environment	Misclassification
Low shutter speed	Normal lighting	Goal in beacon, beacon on top of the goal, beacon on the red robot
Low shutter speed	Turn off main lights	Ball in red robot
Low shutter speed	Turn off white lights	None
Mid shutter speed	Normal lighting	Beacon on the red robot
Mid shutter speed	Turn off main lights	None
Mid shutter speed	Turn off white lights	None
High shutter speed	Normal lighting	None
High shutter speed	Turn off main lights	None
High shutter speed	Turn off white lights	None

Note, the low shutter speed have high probability of recognising the ball, however, the size of the object is not the size of the ball(mixed with random orange feature on the image). Also, with low shutter speed and the dark environments, it tends not to classify the white under the beacon as white color, as the result the 'insanity' check failed to removed the goal in beacon. The check is using long white under the beacon to indicate if it the goal(yellow or blue group) is belong to a beacon. By the experiments it shows that the medium shutter speed could detect the ball and landmark better on the overall different intensity, while having less misclassified objects. Therefore, prepared to use medium shutter speed for the competiton.

Results for the overall performance

At the friendly challenge test with NUbots we score 3 goals. At that time, the shooting behaviour is dependent on the vision goal. Some of the cases where we dribbled from the center and unable to see the best gap, so it failed to shot. In Osaka, at variable lighting challenge rUNSWift score 3 goals in 3 minutes. This result is the second highest, where the highest score is 4 goals. From the observation, we could score more if we did not have 'ball in red robot' problem. Before the competition, we prepared for low lighting condition, but, the lighting condition at the competition is much brighter than normal. In other words, the color would appear more brighter than normal. We tried to reconstruct the problem of ball in red robot which happening

frequently at the challenge in Osaka. We placed a red uniform robot in front of the agent, and logs the number of ball seen. The experiment is done by adding extra light intentionally to brighten the red robot, as figure 3.1 shown.

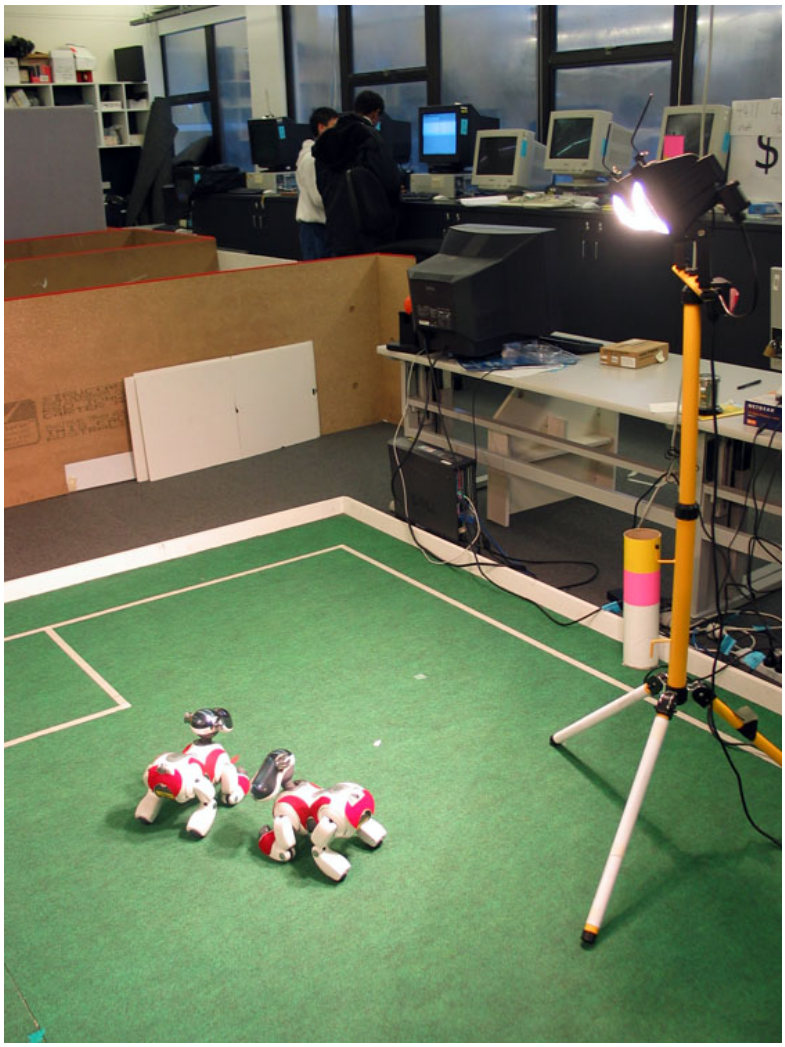


Figure 3.1: Trying to reproduce 'ball in red robot' problem.

From the experiments in the laboratory (running in total of 20 minutes), the distance near to 40 cm between the two robots causing some detected balls in red robot but not sufficiently enough to keep focus. In other words, although is detected, but it is rarely on consecutive frames. We also tried to make the robot looking at different side of the red robot. Other distances (nearer or further than 40cm) unable to detect the ball in red robot in any frame. The experiment unable to reproduce the contantly seeing ball in red robot as in Osaka. The possible

reasons are the lighting temperature is different than in the laboratory, the camera setting that we changed to fluorescent mode in Osaka due to 'blueish' color by the Osaka game lights, the high intensity of the extra lights, or might be the color of the extra lights.

3.6 Conclusion

In conclusion, rUNSWift code (vision, localisation, and behaviour) is capable to play on different lighting intensity as tested above. However, it might not work well on other environment that have different lighting temparature, as an example, in the laboratory we failed to replicate variable lighting environment at Osaka that sometimes cause misclassified the red robot with the ball. The proposed solution by changing the camera setting, also shown that the vision algorithm is robust enough on handling the problem.

Future work of using average YUV value [Xu05] could make it dynamically change the intensity, however, it might be computationly expensive and causing dropping frames and also it might not preserve the original color.

Chapter 4

The Almost SLAM Challenge

4.1 Background

The essence for Almost Simultaneous Localisation and Mapping(SLAM) challenge is the same as other challenges, to make the RoboCup moving toward common nature environment. Explanation of the rule of the challenge can be found in [Com05b]. This challenge is divided into two co-related part. The first part of the challenge is building a map of the environment, while the second part is moving to five location by localising using the map built by the first stage. In this challenge, rUNSWift building a map of extra pink landmarks. Extra landmarks are provided as the exchange to the normal beacons. Pink landmarks are used, because it is certain there would be a minimum of three landmark containing pink patches. The extra landmarks are being placed outside the field within 1 meter. Firstly, the robot start paused and being placed by the referee anywhere in the field. Once the robot activated, the robot then have to complete the task within 1 minute. Section 4.2 explored several ways and proposing some solutions for the first part, from the pink mapping algorithm until finding the most effective behaviour to do the exploration. Before the time limit, the robot should paused itself. The normal beacons and goals then removed or covered. The robot will be moved to a random position on the field and activated to start the second stage. At the second part, the robot have 2 minutes of time to localise itself and move between 5 positions. The positions are decided and given just before the competition. Section 4.3 described a way of using pink positions and field line updates for localising and maintaining the localisation while moving to the points. The total points

is marked based on the precision and the time to complete the 5 positions. The performance measurement of the proposed solution is separated into each section and overall evaluation and results in shown in section 4.4.

4.2 Landmark detection and analysing the location - Part 1

First part of the challenge is to analyse and memorize the position of colored landmarks. Landmarks may contain different colors, sizes and shapes. In this challenge, we would use pink color only, same reason as mentioned in previous year, it is certain to have pink patches on at least three of the extra landmarks. The rules [Com05b] also mentions that the pink patch is at least 10cm across, which is at least the size of pink patches on beacons. This section is separated to three subsection which discuss the detection of pink, the usage of pink to approximate the location of the new landmarks, and the last subsection will show the behaviour of an agent to ensure identifying all extra pink landmarks on the field.

4.2.1 Detecting pink patches

This year implementation is not the same as previous year since this year's vision algorithm has been changed. The rUNSWift 2005 vision algorithm implemented by N.Alex[Nor05] is using features points detection, instead of blobbing the colors. However, the base idea from Derrick[Wha05] is still work for implementation of the solution. The idea is as follows, using the half-beacon pinks which are not on the top or bottom of the blue or yellow half-beacons.

In rUNSWift 2005 vision algorithm m[Nor05], there is no blobs of half-beacon. However, the algorithm for finding the pink patches could works by finding the pink patches that is not belonging to a beacon, after that trying to classify it as an extra pink landmark. To understand the logic of identifying a pink patch that is not belong to a beacon, need to understand broadly the algorithm of finding a beacon.

As shown in figure 4.1, at first, features point get classified into specific colors. Next, the features points get grouped together based on the close image location and the same color. For

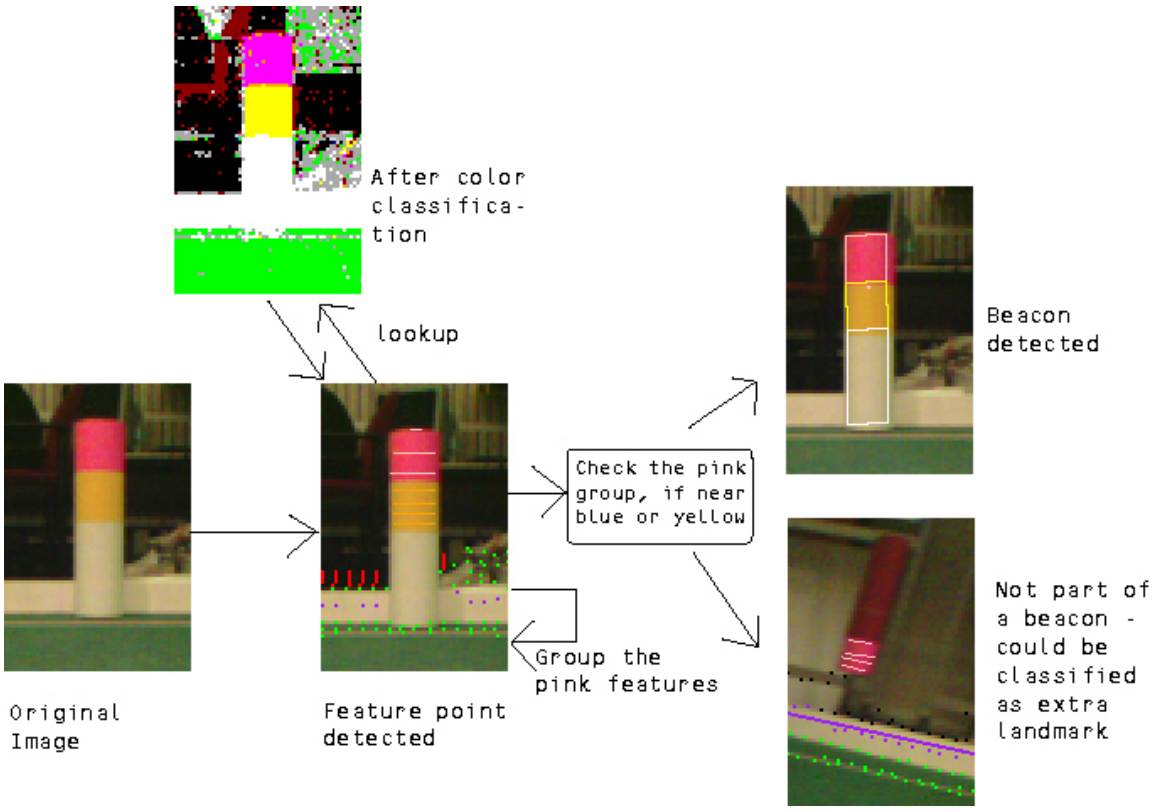


Figure 4.1: The sequence for detecting a beacon and pink landmark.

every pink group, the algorithm then check if it is near blue or yellow and check the criteria considered as a valid beacon.

This year's implementation of recognising pink patches is basically intercept the ones that are not pink from the beacons and try to classify it as pink objects (objects that considered as extra landmark). Note, the pink that are selected, have gone through some checks when vision picking the feature points. Apart than the color in the pixels have to fall between color range of pink, the other criteria for pink feature is the feature at least have few consecutive pixels. The number then being tweaked to recognise the normal beacon in the same time reducing the unwanted pink randomly appear outside the field.

4.2.2 Estimating the location of extra pink landmarks

After the low level vision able to recognise the pink patches, the next problem that be examined is the estimation of the extra pink position. Commonly used technique of estimating the position

of an object is by distance and angle information. In this case, distance information cannot be obtained by looking on an image, since the size of the pink patch is not specified.

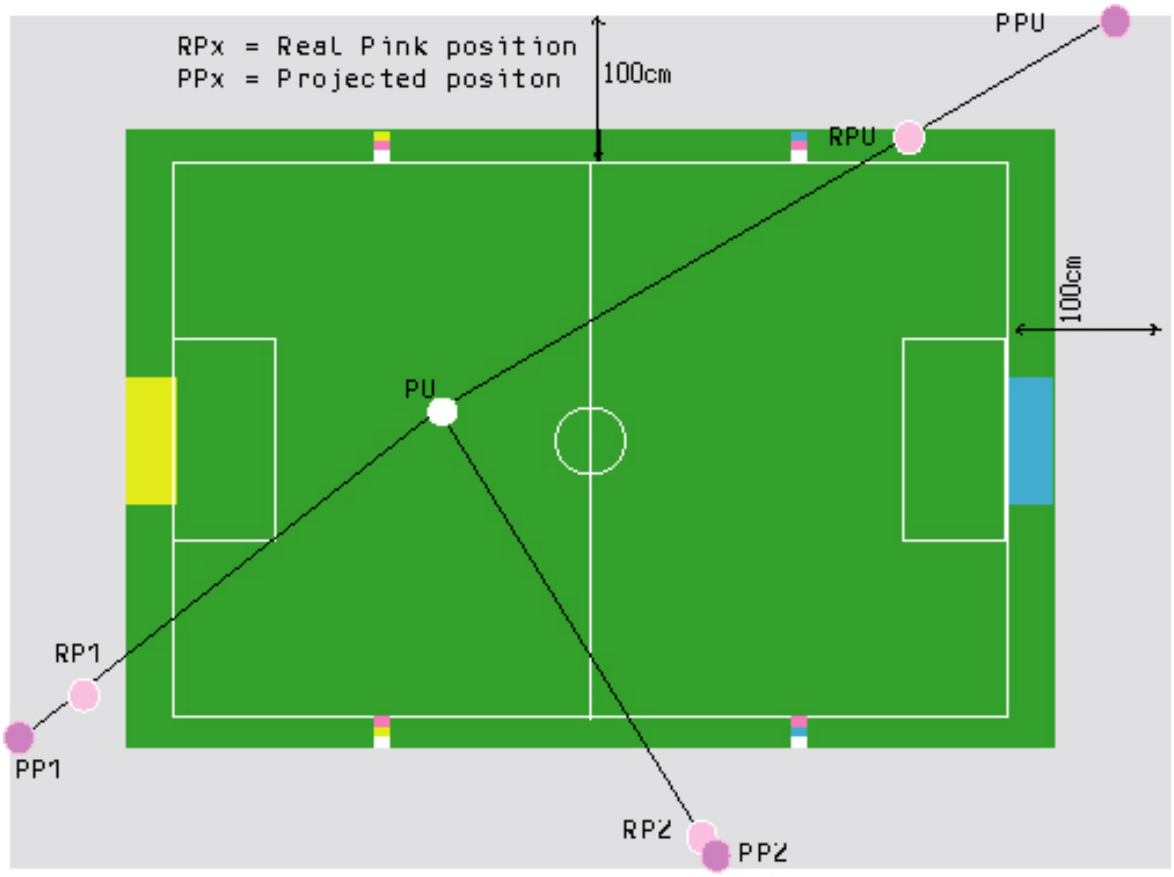


Figure 4.2: Derrick's solution for rUNSWift 2004 for estimate position of the pink.

Previous solution in [Wha05], every extra pink detected projected to the wall, as shown in figure 4.2. The result of the projection indicates the position of the pink. This solution works for last year case, where the extra pinks are located exactly on the wall. In 2005, the rules [Com05b] stated that the position of the extra landmarks could be anywhere outside the field within 1 meter. Hence the previous solution could not estimate the location well. For example, a pink object located near the side of the wall would get projected far from the real position, as shown in figure 4.2. Changing the boundary of the projection to some number between the side of the field to the wall could result the same problem. An example shown in figure4.3.dThere always be some cases that the projection gives large error. This error would then carried to the second stage, localisation using these extra pinks. The localisation would not work well since the source

of localising is not on the right place. Figure 4.4 displays one of the cases where wrong estimate of the pink location would gives error update for heading.

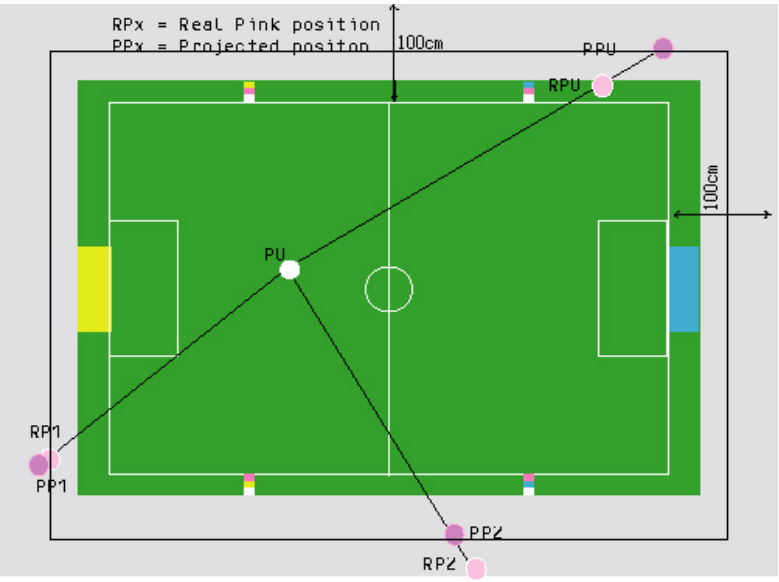


Figure 4.3: Derrick's solution using different boundary projection. Boundary is approximately in the middle between side line and wall. PP0 and PP2 is shifted for at least 50cm(distance between the projection boundary and sideline).

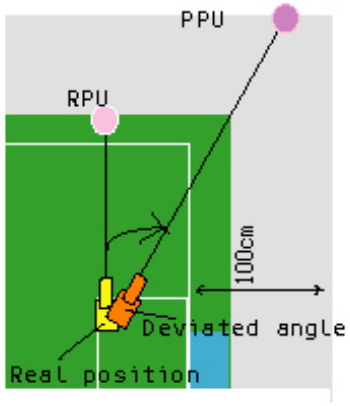


Figure 4.4: When the real position saw a pink with 0 angle, it then update the localisation to with 0 angle to the recorded pink. Since the recorded pink is not in the correct place, the update would shifted the localisation to an angle.

Rays on the field

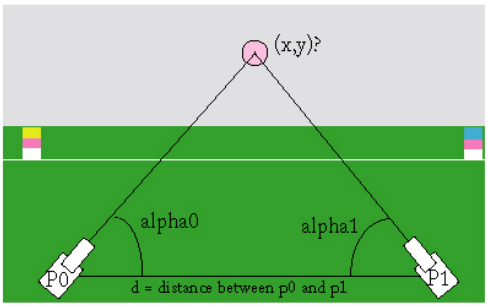


Figure 4.5: Looking at a pink object from 2 location.

Basic idea of the proposed solution is using two source of rays to estimate the position of the pink, as shown in figure 4.5. Knowing the position of the two places and the angle to the pink could compute the position of the pink. Figure 4.6 shows one of the way to solve the pink position.

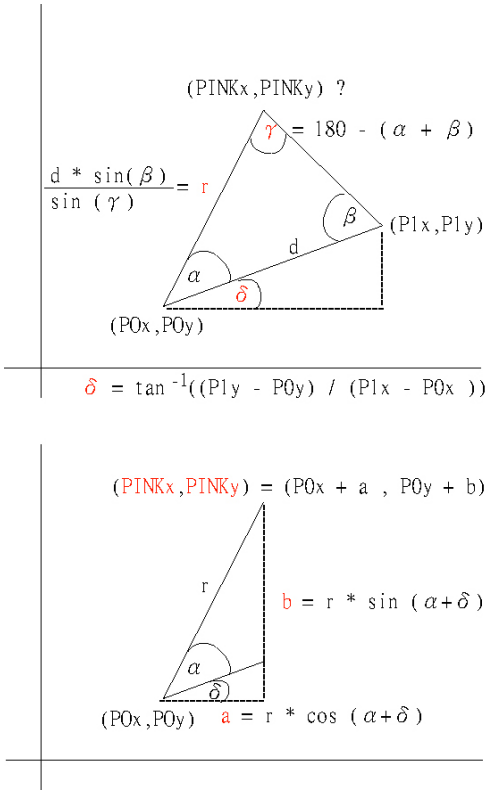


Figure 4.6: Computation the position of the pink.

By focusing on a pink and move between two location could works for any pink location specified in the rules [Com05b]. A practical problem arouse from that, it is not possible to move in two places for each pink in the given time limit.

Since the constraints is the movement, one possible solution is lock the two location and focusing on solving the combined data from the two location. The solution that we would like here is where the data is collected at P0 can be combined with data collected at P1, and it produced the result close to the real position at the end.

From that assumption, arises another problem, keeping knowledge which subset data from P0 is correspond to the same pink object to subset data from P1.

This problem could solved by looking at top view. Instead of recording the angles, it store the data as rays. With rays, combination of the data could provide the pink solution. As shown in figure 4.7, each intersection of the rays is a possible pink location. With this solution, the information that needed to be stored is only the rays. Since the rays could be represented as a line and a direction. The line is represented as a gradient and a constant. The direction of the ray could be identify by which side of the side line does the ray intersect.

The gradient and constant is calculated as follows,

$$\begin{aligned} rayHeading &= myHeading + pinkHeading \\ gradient &= tan(rayHeading) \\ constant &= myPosition.y - gradient * myPosition.x \end{aligned} \tag{4.1}$$

$$\tag{4.2}$$

Where rayHeading is the heading of the body plus the heading of the pink from the neck of the robot. With some checking to identify the which side does the line intersect, the data is stored for later use.

Intersection of rays

After data is gathered from the two positions, one way to find points where the rays intersect could be calculated by comparing an element with the rest of the rays and repeat the process until the list is empty. However, instead of comparing a pair of elements which have O(n2), another way which also be implemented is run on O(n).

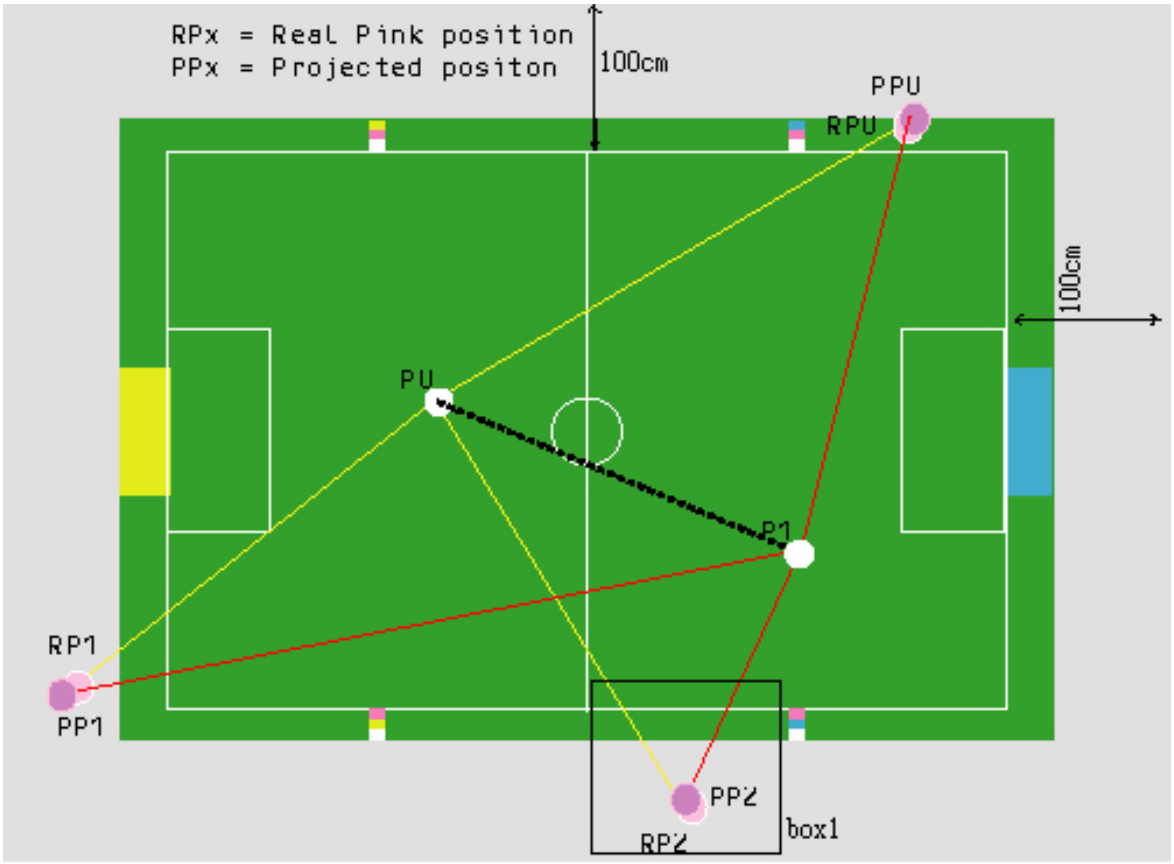


Figure 4.7: Looking at a pink object from 2 location.

The second way solving the problem is dividing the area outside the field into boxes, as shown in figure 4.8. Each of the boxes contains a counter where it increases every time a ray pass through the box. At last, we could group or merge the boxes with high counter together to find the best match. Therefore, the performance instead of O(n2) by using pairs of rays, drawing boxes to indicate the rays passing through is O(n) and since the number of boxes is fixed then selecting the points takes O(1). The advantage of using boxes rather than the first way is that faster to calculate when the number of data is large (by experiment, shows that scanning a pink patch average produce 80 rays). Another advantage is able to locate a pink position while no data coming from one of the source (no intersection but the number of in the box is high). However, this is resulting less precise than calculating the each pairs. Precision is already lowered by angle errors from returned by the motors, and more by approximate the intersection to the box size.

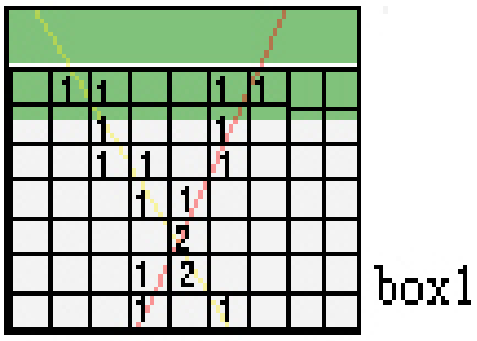


Figure 4.8: Boxes outside the field,each box contains a counter.

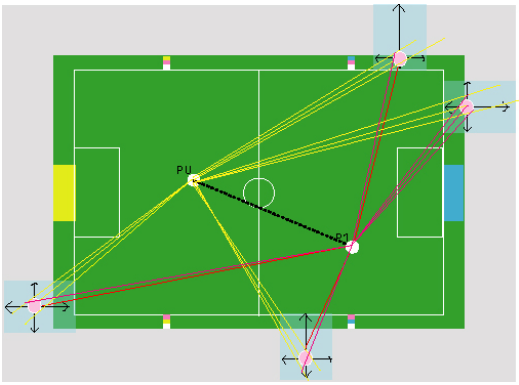


Figure 4.9: Merging area. Pink points outside the field behind the goal are merged with any points between X cm along x-axis does not concern on the y position, while left and right side merged anything between X cm along y-axis. The box shows that the point would merged any other point inside the area.

After the first part completed (robot paused itself) and before the second part begins (removing / covering the beacons, move the robot to the position), there is a gap of at least a few seconds, for later, this gap we called 'extra' time. Since this finding the intersection is calculated on the 'extra' time slots, the time where the system is not running any heavy computation (Vision, GPS updates and locomotion is off), we could utilise the memory space and processor. As experiments shows that it is capable to do the pairs calculation, therefore, for the challenge we used the more precise results which is checking pairs of rays. For each intersection detected, we try to merged with the points calculated previously. Merging the points is not using the distance between two points. The criteria for merging is using distance between two points on only one

of the axis, the distance between x-axis is used if the points are on the top and bottom of the field (behind the goals) and y-axis for the side of the field (behind the beacons). For example, the different merging criteria of different location is shown in figure 4.9. This grouping is used by making the assumption that no other pink landmark that located behind a pink landmark. With this assumption, the merging also helps cases where the points that are separated because the points merged to two separated points although it corresponds to the same pink landmark. Each time a merging is done, increase the confidence value of the point by one. At the end, the confidence value could be used for filtering the low confidence value results, for example, random pinks outside the field. The remaining points that can not be merged together are the points that are used for pink locations.

As described above, the proposed solution of using two source points and intersection of rays give more advantages on performance and accuracy rather than previous solution. Instead of looking at a frame and estimate the position of the pink, we collects the data and computed those at the end. By less costly computation at the process of scanning, the advantage is dropping less frames and able to collect data as much as possible. Another advantage is heavy computations are done in the time between end of the first stage and preparing for second stage. However, due to time limit the data could not be achieved without efficient behaviour is described in the next section. The evaluations of this algorithm is shown later in section 4.2.3.

4.2.3 Behaviours for mapping pink

As the proposed solution above requires sample data of rays that can creates numbers of intersection to indicate the pink position. From Derrick's thesis[Wha05], last year behaviour is described as following steps,

1. At first, localising to recognise the current location.
2. While doing active localise to update the GPS(position), move to the center.
3. When time to move finishes or when stopping because the agent already near the center, do active localise to give more precise position on the field.
4. Do a slow 180 degrees scan four times, rotate the heading 90 degrees between the scan.

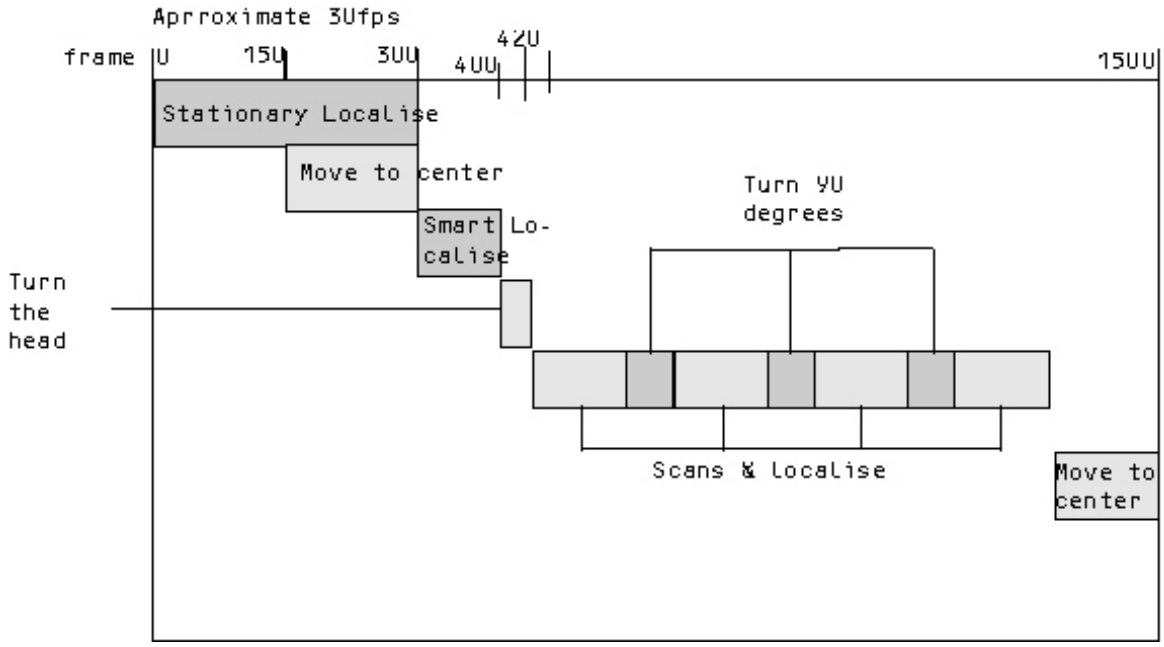


Figure 4.10: rUNSWift 2004 behaviour sequence for SLAM challenge.

5. Stationary localise while moving to the center.

Although with slow scan could get nearly accurate angle reading, the time is consumed on the scan and above behaviour gives a single source of rays. Section 4.2 above shows that single source of rays is not sufficient with 2005 rules, as it might not capture cases where position of the extra landmark is between the wall and the field, as shown figure 4.2. Hence, above behaviour is not sufficient to provide the data for the proposed solution.

Methodology of finding the behaviour

In order to have two source of rays with the time limit is less than 60 seconds, there are major changes to the previous behaviour. First of all there should be an allocated time to support walking between two far source points. With that, the limit is reduced to 1 minute minus the time to walk. With that short limit, it is required that the rest of the time should be efficiently used. Then, lists the critical behaviour that has to be performed for mapping algorithm to work and separate them into sections. The time for each section is measured by its performance and usefulness. For example, the time to localise is tweaked to the minimum, where the time required to be able to localise in the worst case (figure 4.12). Another example, removing the

last behaviour (behaviour number 5 from last year solution) which localising while moving to the center because it does not gives any advantages, and more discussed later in the section. The values are being tweaked at few points of time between the development of rUNSWift 2005. Tweaking is done largely due to improvement of localisation and faster walk. Changes in localisation could be affected by vision is better at detecting landmark, edge updates gives better results, and so on. The faster to localise, the less time necessary for initial localisation, therefore this time slot could be used to handle other cases. For example, sparing the time to do behaviour of moving away from the side of the field. Doing that will allow the scans to detect the high pink position near the side of the field.

Effective Mapping

After the many tweaking and iterations, the 2005 rUNSWift proposed solution for the mapping the pink is shown in figure 4.11.

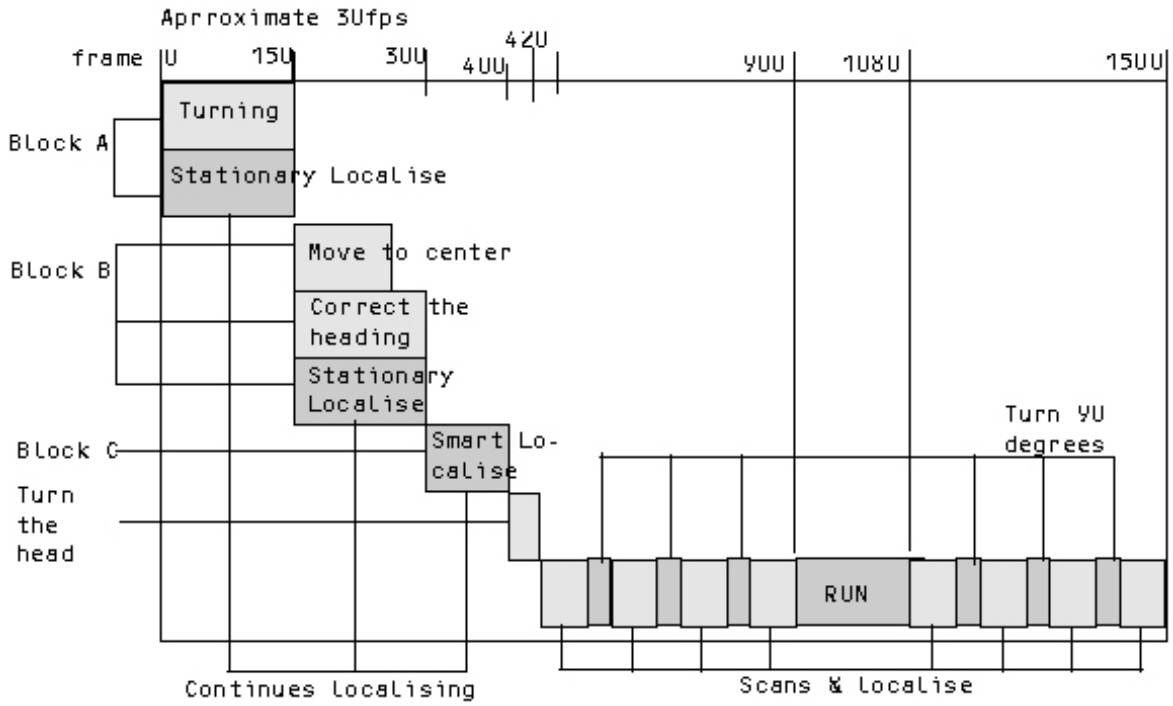


Figure 4.11: rUNSWift 2005 behaviour sequence for SLAM challenge.

Block A

Firstly as shown in pseudocode 2, since the rule does not mention the placement of the

Algorithm 2: Block A (frame less than 150)

```
begin
    if surveyCounter < 150 then
        if turnIndicator = 0 and countGreenPoints() < MIN_NUM_GREEN_COUNT
        then
            set the heading to see the front
            movement(turn)
        else
            stationaryLocalise(10)
end
```

robot and the heading, the first behaviour is turning towards the field. The advantage of this is cases when the agent is being placed near the side of the field and facing outside the field. On contrary, if turning is not executed, it will waste the time of initial localisation and getting only small number of observation. Before turning behaviour is executed, each frame is checking the number of greens points in visual image. At each frame in block A, the number of green points has to be sufficient that indicates the agent is not at the side and facing outside the field. If the criteria fails, then executes turning until facing the field. At this point, we are sure that it is facing the field. There are cases where at this point the scan to the side might still resulting less green points. For example, in the corner position where it turns until facing the field but left or right scan might resulting small number of green points.In order not to repeatedly turning, turn indicator is used to make sure this behaviour is run only once and stop when it is facing the field. Note, the diagram show that turning start at the same time with initial localisation. If the turning is not necessary, it will start with the stationary localise. The time value is tweaked by experimenting how long an agent to localise given 'bad' positions, some examples shown figure 4.12. The time is experimented such that start with 'bad' position, and check if the given time limit is able to recognise the position to which quadrant of the field and the heading. The value from the localisation is not to be precise, since the next block will continue further localisation.

Block B

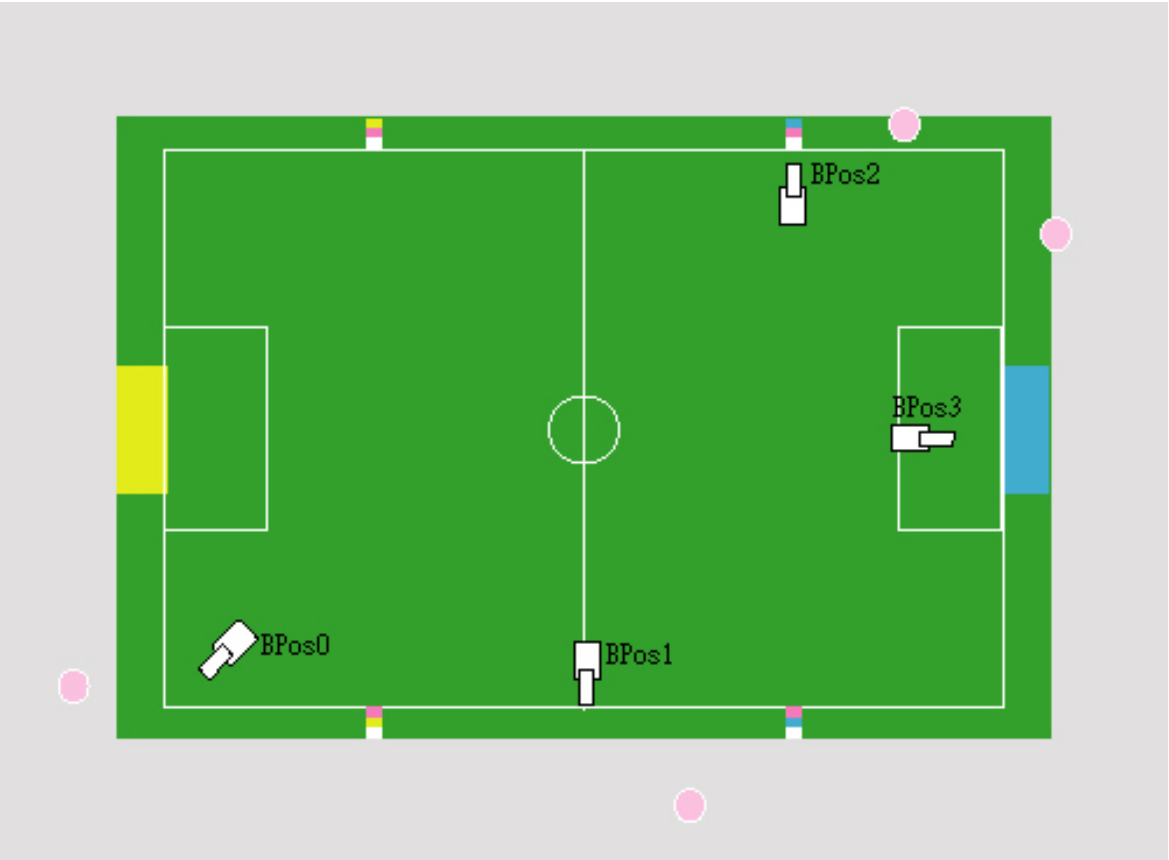


Figure 4.12: Examples of bad starting positions.

As in pseudocode 3, the aim for this block is to move the dog to the correct heading and away from the side of the field. Since we are going to do 360 degrees scan, we have to make sure the location is far enough from the side to capture pink patches near the field. The number for the distance is adjusted depending on the camera position. With the different walk stance, the positions of the camera view is change.

If the distance criteria is satisfied, then turn the heading to 90 degrees towards the second point for initial scanning, as mentioned in figure 4.13.

Block C

Before start scanning, stop the body movement and smart localise to increase the precision for position and heading, as shown in pseudocode 4. This precision is critical for building the rays as mentioned in section 4.2.2.

Scanning section

For preserving large number of sample input, the implementation is done by scanning at 4

Algorithm 3: Block B (where 150 ¡ frame ¡ 300)

```
begin
    if surveyCounter < 300 then
        stationaryLocalise(10)
        if distancetotheside < minimumsafetydistance then
            movement(tothecenterofthefield)
        else
            calculatethesecondpoint
            movement(headingtothesecondpoint + 90)
end
```

Algorithm 4: Block C (where 300 ¡ frame ¡ 400)

```
begin
    if surveyCounter < 400 then
        movement(stop)
        if (surveyCounter − 300)mod25) = 0 then
            findnearestbeaconstolocalise
        localisebylookingatnearestbeacons
end
```

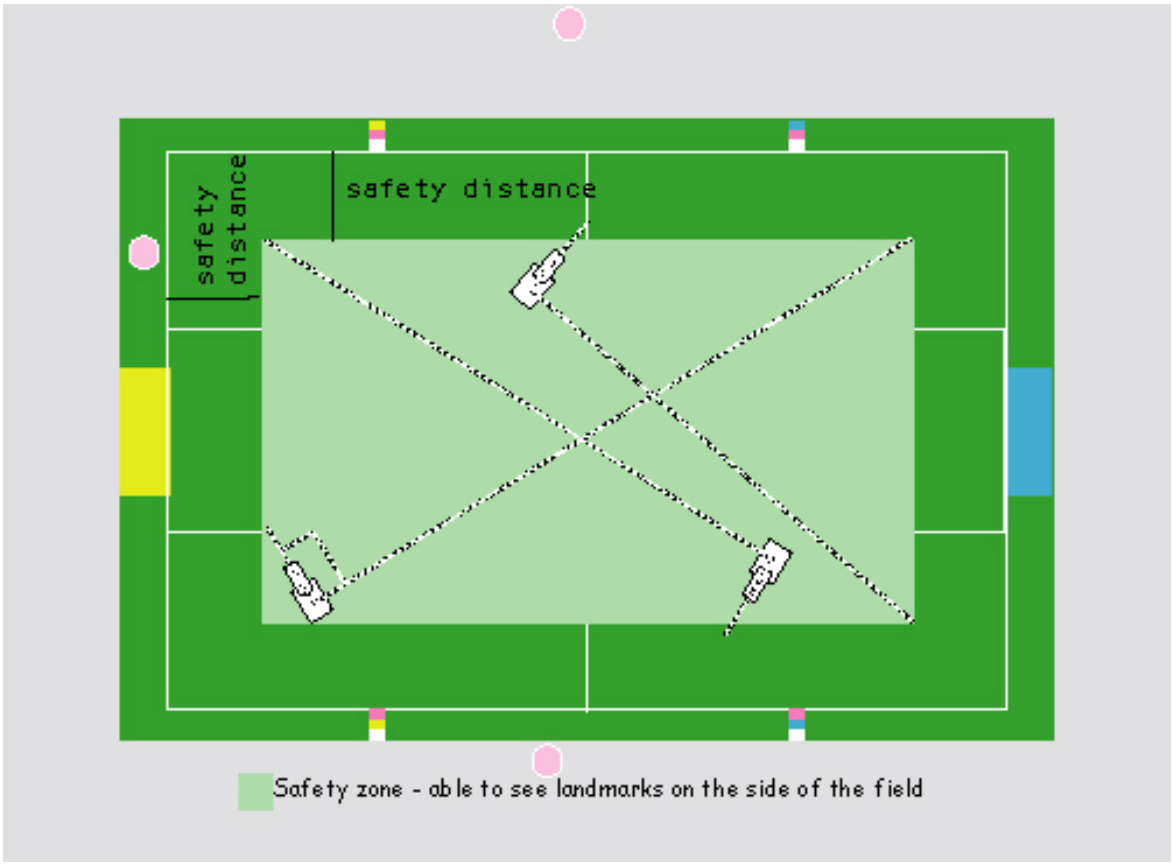


Figure 4.13: Some examples of starting position before scanning stage. They are positioning 90 degrees to their second position.

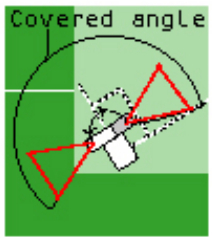


Figure 4.14: Scan part 1. Start scanning by panning 180 degrees to the left and right. Note the angle that is covered could be more than 180 degrees.

direction in each of the two source points. For efficiency, some block of time are multitasks, explained later in this section. The scanning sequences is shown in figures 4.14, 4.15, 4.16, 4.17, and 4.18.

The first point is the current position and the second point is the target location for second

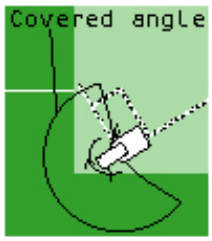


Figure 4.15: Scan part 2. After rotate the body 90 degrees, do the scanning again. Note it turn away from direction to the second point.

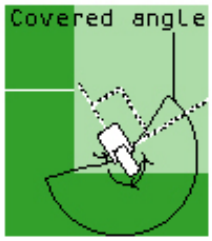


Figure 4.16: Scan part 3. After rotate the body 90 degrees, do the scanning again.

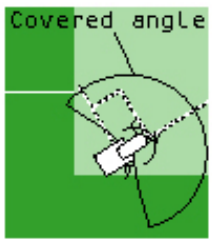


Figure 4.17: Scan part 4. After rotate the body 90 degrees, do the scanning again.

scan sequence. The second target location is always on the opposite corner position, this enhance the probability on intersecting pink rays. As shown in figure 4.13, the starting point is always 90 degrees away from the direction of the second point, it is done to optimise the number of turning, 8 directions scans in 6 turns.

At the starting position shown in figure 4.14, start the scanning by doing head movement of 180 degrees pan(implemented in speed of five). Speed on different AIBOs may vary depending on the neck motor, however, with time limit maximum 3 seconds should cover the worse case. The speed is faster than last year's solution because it is tweaked such that the current vision still

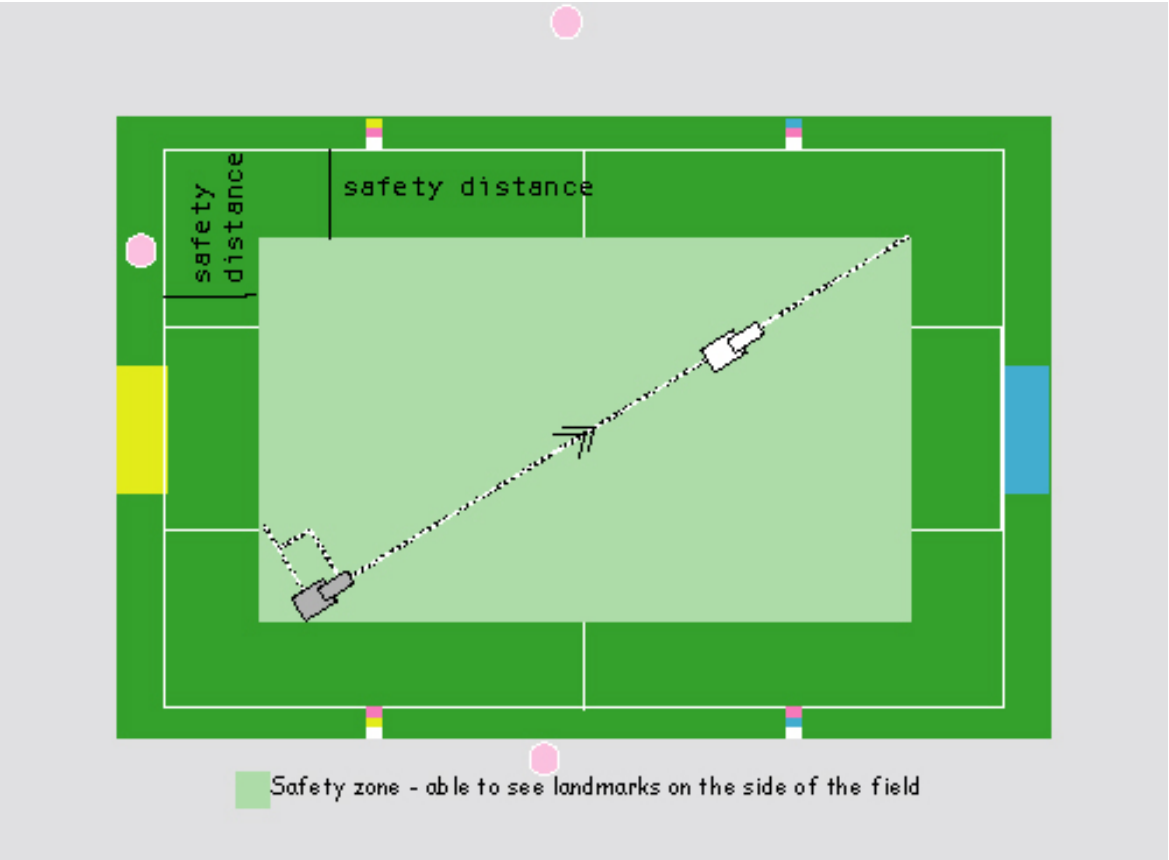


Figure 4.18: Scan part 5. Scanning part 4 completed, run to the second position.

be able to recognise the pink and the precision pan value from the neck motor is satisfactory. After that, rotate the body 90 degrees away from the direction to the second point, shown in figure 4.15. The reason is to have the last scan facing the second point, therefore could run straight after the last scan. After repeating the scan and turn until finishes the scan facing the second point(figure 4.16 and 4.17), run using the fastest walk type to the second point(figure 4.18). Depending on the speed of the runs, the final position of the second point is limited by the allocated time, so that, have spare time to be able to do the second sequence. At the second point, repeat the scan and turn at the final position until the final time reached.

Evaluation

Apart than evaluating the performance on the time blocks mentioned above, overall performance evaluation is measured by running the first part of the challenge many times. The measurement

is done in the laboratory with laboratory settings for colors in vision and walk type in locomotion.

Methodology of testing is as follows:

1. Set environments in the code to specify that is going to be Slam Challenge mode (Flag in Challenge.h). This flag changes the vision criteria for number of consecutive pink. Note, in the laboratory, more than 20-30cm outside the field is not covered by the main light. The value is changed to detect more pink, since we test pink landmarks located between side of the field until 1 meter. The flag also turn off the removal of line points above obstacles. The removal is used for removing line points that detected on the AIBOs body. Since no other AIBOs will be on the field, we turn it off. After few experiments, decided that it is necessary, some of the cases where random obstacles are detected on the field and line points gets removed because of that.

2. Change the flag in Python code to indicate the process will start from mapping the pink (useMinimalMode = False). If it set to be true, it will run on the second stage of the challenge.

3. Change the walk type in the python code to use the walk type that has the best calibration for the field. Since the lines are distance apart and seeing a pink landmark is not frequent, we would like the walk position update to have less deviation on the angle and distance.

4. To log the evaluation, for every test store the telnet session on different files. The slam challenge code will print the equations of the line that it detected. This is done due to avoiding computation on the agent to send the vision packet information, which might cause dropping frames. With the equation of the line is recorded we then be able to play back the same function to do computation on the subvision debugging tool.

5. The extra pink landmarks is placed outside the field and measure the position for result comparison.

6. Testing different starting position. This is for testing if the initial localisation is well enough.

7. After the tests are done, we play back the logs in the subvision debugging tool, comparing the estimate with the original position and table of results is constructed.

The aim for the evaluation is to check if the algorithm is able to minimise the average error of distance between estimation and the original position.

Results

To begin with, with the current algorithm need to make sure that with no extra pink landmark, it would not produce any extra pink landmark. Ten times runs of the mapping stage with no extra pink landmark does not result any intersections. However, sometimes have rays which are coming from beacon, but projected to the side of the beacon. This data mostly happen when after running, the angle is a bit shifted, and at that time the vision is not recognise the pink from the beacon as a beacon object. From 10 runs for 1 minutes each, there are 5 rays with no intersection. Where normally where pink is an extra landmark would produce average of 80 rays.

With extra pink landmarks, running the first part of the challenge 25 times with different starting position. Test is done in the laboratory with laboratory setting (laboratory's color calibration and laboratory's locomotion calibration). Extra pink landmarks are placed on (310,-50), (-50,230), and (45, 555).

As shown in figure 4.19, the points are clustered near the correct position. The high confidence position is projected near real y position, as shown on figure 4.21. The x position also has similar results shown in figure 4.20.

Although the research shows that it is sufficient, this solution might not be the most effective solution, since the effectiveness is dependent to the low level implementation. The solution able to be improved when low level improved such as vision performance, localisation results, locomotion changes, and so on.

4.3 Localisation using given landmark - Part 2

At this stage, the robot is being placed randomly inside the field to start with. The robot then has to localise, and move to five positions on the field. This section discusses localising the agent

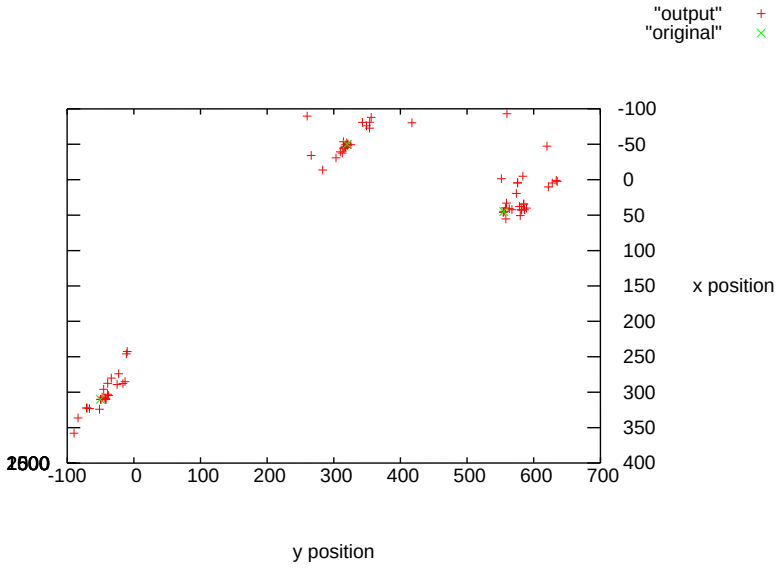


Figure 4.19: X-Y view of data, plotted from all of the results. Original points are the real position of the pink landmark.

and moving to the points while maintaining the localisation.

4.3.1 Localising the agent

Localising the agent is very critical to be able to complete the challenge. If the localisation is not resulting any good enough position, the movement to other points will be incorrect. The problem that need to be solved is localising an agent only using line points and extra pink landmarks.

Related work

Last year proposed solution, Derrick[Wha05] stated that using stationary mapping able pin point to two location which symmetrical, and the choosing between the two is easily being done by the angle updates when detecting extra pink landmark. Since the field is larger than last

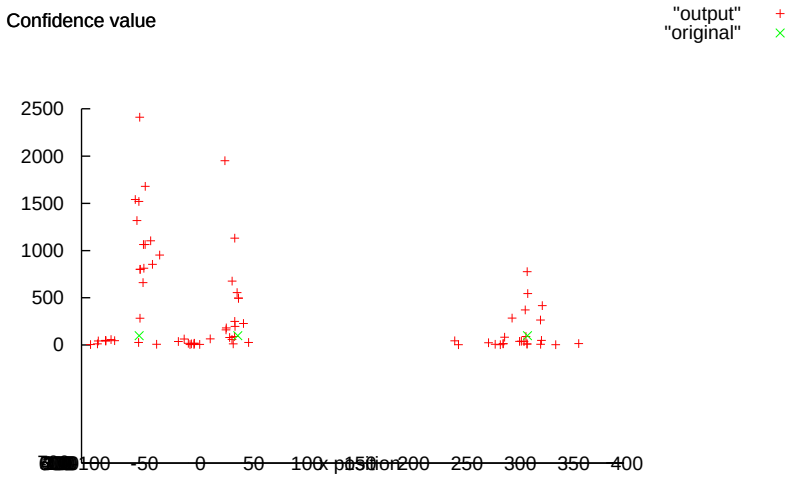


Figure 4.20: X-Confidence view of data, plotted from all of the results. Original points are the real position of the pink landmark. The graph show that the high confidence points lying close to the original x position. Note, this is X axis in coordinate system against Convidence value.

year, this reduces the performance of the solution. The impact on the stationary mapping is the line distance. With a random position and heading, most of the places stationary mapping could not construct a good line map that could be used. If there is not enough line points, the result from stationary mapping could be map to anywhere on the field.

From the behaviour point of view, the field size also impact the behaviour performance. Derrick's method of slow scan on low and high tilt results is not much different than using fast scan. The fast high scan would detect faster on the pink landmarks and the line points without getting large error. Moreover, the low tilt scan is not performing well, since the field is larger and the possible location where it could see lines below are small (most of the cases seeing the green points on the field).

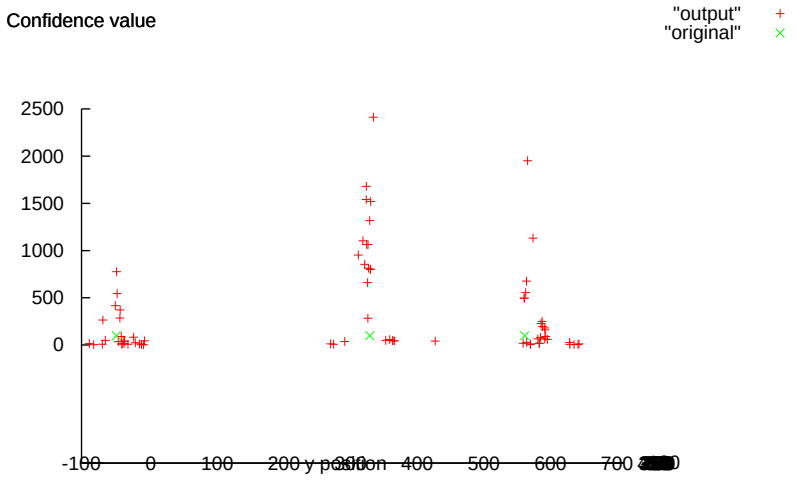


Figure 4.21: Y-Confidence view of data, plotted from all of the results. Original points are the real position of the pink landmark. The graph show that the high confidence points lying close to the original y position. Note, this is Y axis in coordinate system against Convidence value.

Proposed solution

The proposed solution is similar to last year, which using field line updates and pink lanmark. The localisation updates is performed by using field lines observation, and the pink landmarks. Updates by using field lines is the same as performed for main rUNSWift code. It uses the projection of the field lines to be feed into gradient accent to find the the best match to the nearest line. Depending on the match, the possible duplicates of the positions are created as observation data. After that, the edge update creates few Gaussian from observation data that is applied to the previous Gaussian distributions. For the final distributions, few top weights are selected from the two sets (previous distribution and distributions affected by the observation). Gaussian that have the best weight is used to indicate the location and heading of the agent.

By detecting a pink landmark, in this case, is not possible to give position update as explained

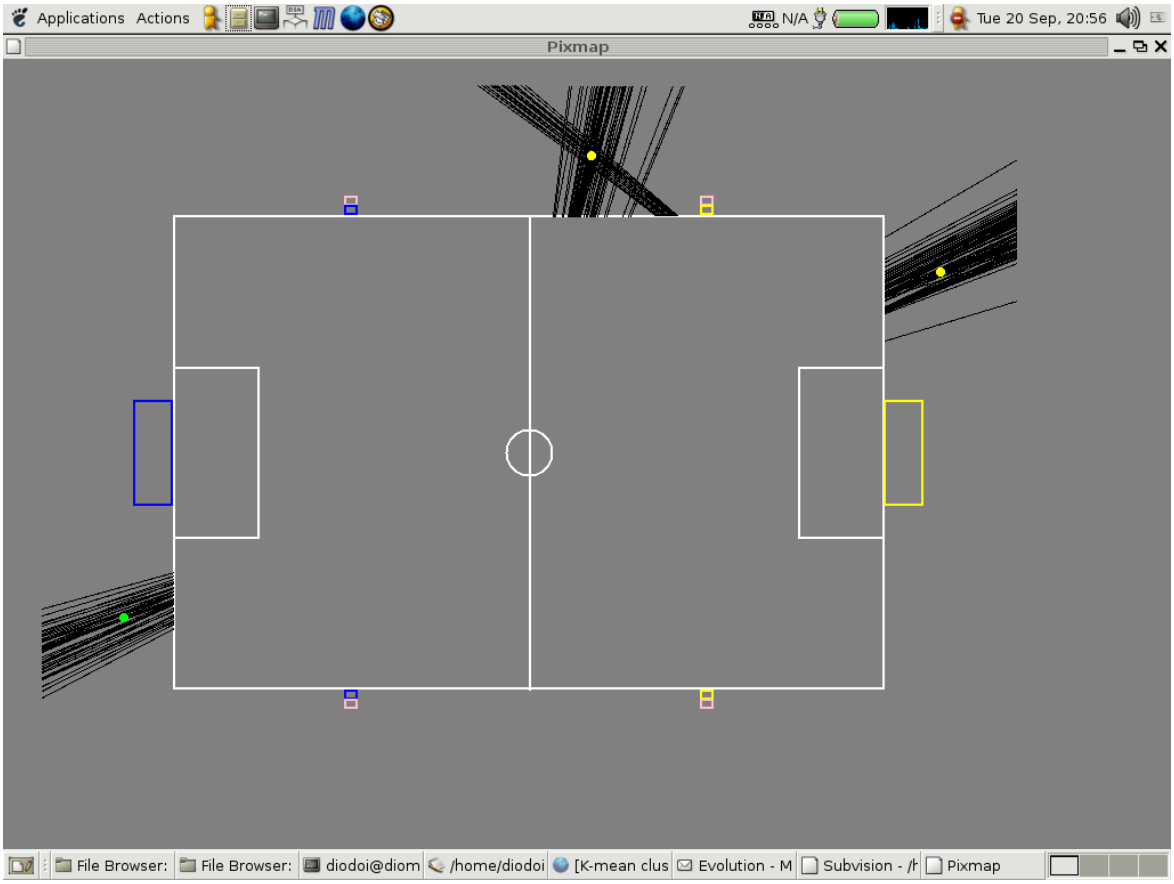


Figure 4.22: An example top view from playback rays in subvision debugging tool.

in section 4.2, that the pink patches do not have an unique identification to make it differ from the rest. However, it could be used to give angle updates on the Gaussian. The method is similar to the field updates, instead giving observation on location and heading, the pink angles gives observation for the angles to the pink. In other words, at every position from the previous prediction, apply an angle to each of the pink positions. If it produces better weight, inserts the heading to the distribution.

The behaviour for the second part is based on the previous solution [Wha05], first few seconds would localise itself and after that, start moving to the nearest given position while focusing the camera on the nearest edge and once in a while look for pink landmarks to localise. The difference is the slow scan on the low tilt is removed, incorporates the green field detection, and stuck detection.

Apart from local field lines update and angle updates by pink landmarks, the problem of

maintaining the localisation while moving between the points is more relying on the locomotion update. Since it is not often to have field lines and pink landmark to give reasonable updates, the locomotion update is critical on the several steps in between the next good vision or edge update.

Evaluation

For the evaluation to be more accurate, we contain the second part of the challenge by giving the real positions of the pink landmarks. Running the part 2 of the challenge multiple times by starting on different position. The measurement is based on number of location visited within 50cm and the time to complete the 5 locations. The time limit is 2 minutes, the same with the challenge part 2. To get overall result on different starting position, each test has different starting position.

Results

Pink location : (310,-50), (-50,230), and (45, 555)
Points to travel are : (40,100), (245, 65), (180,270), (180,540), (40,440).

Test	number of location visited	time min.sec	start position
1	5/5	2.40	outside the goalbox facing the blue goal
2	1/5	+3.00	outside the goalbox facing the yellow goal
3	1/5	+3.00	center position facing yellow goal
4	3/5	+3.00	center position facing blue goal
5	4/5	close to 3.00	(40,100) with 0 degree heading

Test	Comment
1	Successfully complete 5 locations with average 27 cm off
2	The robot was mislocalise after stuck in the goal
3	Stop at the center, after that, unable to localise(moving to the wall)
4	Walk to the first 3 places and unable to maintain localisation when moving to the other part of the field
5	One of the points is off than 50 cm able to regain localisation after doing green backoff.

The table shows an average of 14 out of 25 points visited, while most of the cases pass the time limit before stopping 5 times. Observation shows that the locomotion update is affecting the misalignment to the destination, as an example, the robot moving in an angle but the worldviewer debugging tool shows that it walking straight. Another interesting event is that the robot after doing turning(green backoff) sometimes still turn back and try to walk outside the field (due to still mislocalise). This could be improved in the future, by knowing the previous action was green backoff, it could move towards opposite of the wall.

4.4 Overall Results

At the RoboCup competition in Osaka, at second phase, unfortunately we failed to move to any point. The robot was wandering around without stopping, it seems that the agent unable to localise. Later in the laboratory found one of the possible reasons is the distance to the sideline when doing the first stage. Since we changed the walk type to 'fast forward' walk type at the competition which have bigger angle on body tilt compare to 'normal' walk type that was used in the laboratory before the competition. While 'normal' walk have 60 cm safety distance, the 'fast forward' walk required minimum of 130cm. As shown in figure 4.23 the 'fast forward' have bigger the body tilt, impact on the minimum distance necessary to view the 50 cm off the ground.

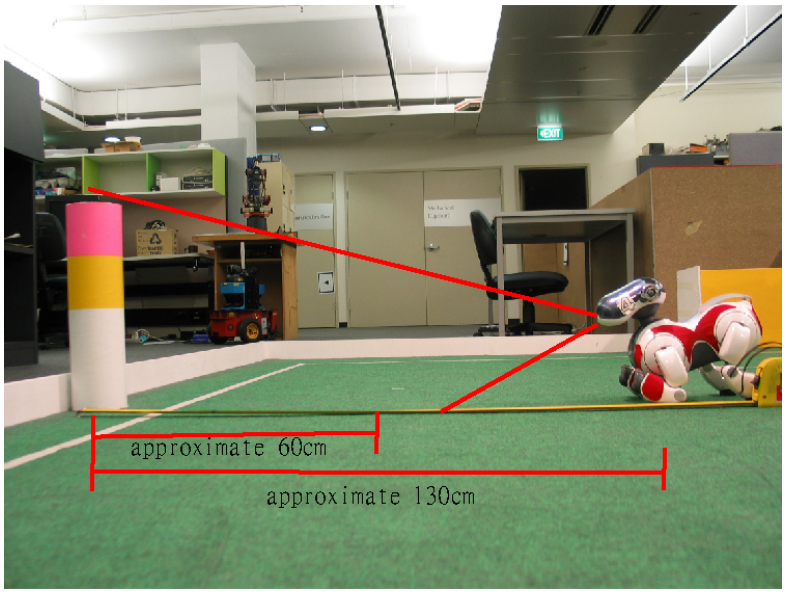


Figure 4.23: Fast forward stance with minimum distance of 130 cm to be able to capture the top part of the beacon.

4.5 Conclusion

In conclusion, two source of rays have perform better results than previous work, however, to complete the challenge with better success rate, further work and test on the second part is needed.

In other words, although the experiments shows that the two source of rays algorithm has good estimate of the pink location, the algorithm for localising at the beginning of the second part seems to be the weakest link.

Chapter 5

Conclusion

This paper shows wider capabilities of rUNSWift by exploring three of the challenges. Those challenges cover extensive tests on several part of the system. From vision algorithm, localisation, locomotion and behaviours. In the World RoboCup Competition 2005 in Osaka, we ranked 4th for the overall challenges, with two of the challenges placed second. Our team (rUNSWift) is 3rd in the world for RoboCup Soccer Competition 2005 in Osaka, JAPAN.

5.1 Future Work

Ideas of possible future works are separated in each sections, please see section 2.7.1, 3.6, and 4.5.

Bibliography

[Com05a] RoboCup Technical Committee. Sony four legged robot football league rule book. 2005.

[Com05b] RoboCup Technical Committee. Technical challenges for robocup 2005 legged league competition. 2005.

[DWU05] rUNSWift Dr. Will Uther. runswift/nubots combined 2005 four legged robocup open challenge entry. 2005.

[Mor05] Nobuyuki Morioka. Road to robocup 2005 : Behaviour module design and implementation, system integration. 2005.

[Nor05] Alex North. Object recognition from sub-sampled image processing. 2005.

[Sha05] Josh Shammay. Real-time shared obstacle probability grid mapping and avoidance for mobile swarms. 2005.

[She05] Raymond K. Sheh. Visual feature detection for robotic soccer. 2005.

[Wha05] Derrick Whaite. Robot soccer - the sony legged robot league the localisation system used by the 2004 unsw robocup team. 2005.

[Xu05] Jing Xu. 2004 runswift thesis - low level vision. 2005.