Bipedal walk and goalie behaviour in Robocup SPL

Thesis B Report

Roger Liu z3291304

January 8, 2013

School of Computer Science & Engineering
University of New South Wales
Sydney 2052, Australia

Supervisors and Assessors

Bernhard Hengst
K17-401E / +61 2 9385 6993

Claude Sammut
K17-401J / +61 2 9385 6932

Contents

Abstract

This thesis documents the work done for the Robocup Standard Platform League. The thesis presents a solution to the problem of IMU filtering using the Kalman filter as well as an alternative solution using the complementary filter. The method to obtain ground truth and calibration of the IMU measurements are also presented. In addition, various motion changes were implemented aimed to improve on the shortcomings of the 2011 walk engine and to adapt to the new Nao robot hardware used for the Robocup SPL. A separate chapter will discuss the goalie behaviour that was redeveloped in 2012 to improve on the 2011 goalie.

Chapter 1

Acknowledgements

I would first like to thank my teammates Peter, Bel, Youssef, Ritwik, Sean, Carl and Sam for their support throughout year 2012. Particularly, I would like to thank Youssef and Peter for doing such an incredible job on localisation which immensely helped me develop my goalie behaviour, Bel for discussing motion ideas with me, Sean for the great work on the vision system, Ritwik and Carl for the new Python infrasture for easier behaviour writing, and Sam for the excellent Defender behaviour. Most of all, thank you for the awesome times in the lab and being able to put up with me.

I would also like to thank Jayen, who helped us solve a motion bug at a stage.

And, of course, I would to thank Bernard for guiding me through the motion aspects of robotics that were completely new to me. Also, your leadership and initiative in week meetings were invaluable to the team.

Lastly, I would like to thank team rUNSWift and school of CSE for giving me this opportunity to compete overseas against outstanding opponents and learn new things from the experience.

Chapter 2

Introduction

2.1 IMU filtering

The inertial measurement unit(IMU) is located in the chest of the Nao robot used for Robocup SPL and consists of a 2-axis gyroscope and a 3-axis accelerometer. The output data can be used to estimate the torsos orientation(pitch and roll) and also angular velocity. In robotic soccer it is imperative to monitor the robots state in space provided by the torsos angle and angular velocity, and balance the robot such that it does not fall due to disturbances that often occur in soccer.

Last years open challenge work featured the bipedal control policy to balance the robot as it moved over uneven terrains[1]. It made use of the foot sensor readings to detect where the pressure currently lies on the support foot and applied ankle control so that the pressure is at the desired place of the foot. One of the flaws of using foot sensors was that they were very unreliable. Robots often had malfunctioning foot sensors.

This years motivation is to use the angle and angular velocity of the torso of the robot as the state of the system such that another policy can be learnt to balance the robot, and minimise the use of foot sensors.

The robots torso orientation can also be used for other applications. The SafetySkill module can use it to detect whether the robot is falling or is already fallen, so that stiffness values in each joint can be turned to zero to soften the impact of falling or execute the getup routine if fallen. For more precise application, torso orientation data are used for camera pose configuration, such that the robot can determine the location of other objects that it sees on the soccer field. As such, there was good incentive to develop an accurate IMU filter.

Aldeberan, the Nao robot providers, provides their own torso orientation estimation, which uses a combination of accelerometer reading and gyroscope reading[2]. However, we decided to develop our own filter for the IMU because:

- We have no control over the already filtered data provided to us. The filter variances Aldeberan used would have been tuned to their own walk. We need the filter to be tuned to our own walk that is faster and noisier.
- We would also like the angular velocity of the torso to be filtered. This and torso lean can be used as Markov decision process states to learn an optimal policy for future work on balancing on one leg while kicking or balancing while walking.

This thesis discusses the applications of both Kalman filter and Complementary filter in IMU filtering, and proposes solutions to how ground truth was obtained and how calibration was done.

2.2 Motion

Bipedal walk is one of the core developments in robotic soccer. Walks developed in the Robocup competition are required to be fast and agile to beat opponents to the ball, as well as stable enough to not fall from common collisions that occur when robots fight over the ball. Agility and stability is not enough however, as the sustainability of the robot must also be considered as a design requirement due to constant wear and tear, and the eventual of overheating.

Development in walk in 2011 by White[4] improved the shortcomings of the walk engine built in 2010 by Hengst[3] while keeping the core engine the same. This thesis aims to further improve the walk engine and address shortcomings. The shortcomings of the 2011 walk engine were mainly: slow acceleration and deceleration due to stability limitation, rough walking motion and camera noise due to lack of smooth coronal (or lateral) rock motion, difficulty in robot avoidance due to arm positioning.

The motion section will discuss changes made to increase the acceleration of the walk, the implementation of coronal rock and arm stiffness control to assist robot avoidance. Additionally, changes necessary for the new Nao version 4 hardware and miscellaneous walk modifications are detailed. As bipedal walking continues to be a research area, methods used in this thesis were mostly empirical. Work on kicks this year are covered by Belinda Teh[5].

2.3 Goalie

The goalies primary role in soccer is to block the goal. It serves as the last line of defense and the only robot allowed in the goal box. The goalie behaviour developed in 2011[14] added the ability to dive for balls in addition to strategic positioning to block the goal as much as possible. When balls were close to the goal box, the goalie would clear it. It also made use of the goalie as a scout to track the ball and passed the precise location to its team members to take advantage of the fact that goalie moves the least and is less susceptible to noise in localisation.

Further improvements were made in ball filtering and localisation this year[6, 7]. The new Python infrastructure was also implemented to allow easier compilation of behaviour code. As such, the goalie behaviour was rewritten and improved upon to take advantage of these changes. The goalie section will cover the state machine used combine and prioritises all different tasks of the goalie, and the improvements on diving.

Chapter 3

IMU filtering

3.1 Introduction

This chapter documents the filtering method of the IMU component. We first describe the method to obtain ground truth as well as calibration. We then discuss the actual filter applications using Kalman filter and Complementary filter. Lastly, the test results of the filters are looked at to evaluate their effectiveness.

3.2 Ground truth

The first problem of developing an accurate filter for the IMU is to obtain ground truth of torso lean so that it can be compared to the estimates. This was achieved using the robots vision system. While the robot tracked a ball, the position of the ball was reported in pixels of its camera. The torso lean per pixel ratio can be determined as long as the ball was in constant range. To minimise the mechanical error that can occur, the head of the robot was strapped tightly to the body using tape so that it does not rock. The angular velocity of the torso was also derived this way.

3.3 Calibration

The IMU readings provided by Aldeberan are not calibrated. The torso lean is just a straightforward constant offset. However it was found that the gyroscope offset drifts over time, most likely due to temperature changes within the robot.

To handle this drift, we need a good enough indication of whether the torso is currently still (in other words angular velocity equals 0), so that the current gyroscope offset, O_t, will be the same as the raw gyroscope reading provided by Aldeberan.

For each motion cycle, we collect torso lean values so that another angular velocity can be derived using backward difference approximation:

$$\dot{\theta}_t =\sim (\theta_t - \theta_{t-1})/dt \tag{3.1}$$

The condition that we believe the angular velocity is 0 is: foot sensor reports at least one foot is touching ground and $\dot{\theta}_t$ is close to 0. The reasoning behind this was that we shouldnt trust $\dot{\theta}_t$ if both of the robots foot is not on the ground as it could be falling, but the angular velocity of the robot can still be 0 even if only one foot is on the ground since we still need to update the current offset when the robot walks.

This was a very crude way to obtain the offset O_t. The following steps were taken to refine this value:

- If we believe current angular velocity is 0:
  - Add O_t to update current sample average gyroscope reading
  - Filter this average in a simple one-dimensional Kalman filter, the filtered value is the current offset O_t

3.4 Filters

Kalman filter is known as the optimal linear estimator provided that all noise is Gaussian. It requires an accurate process model to compute the estimate for the current state given the estimated state from the previous time step and current measurement.

Complementary filter is a popular alternative to Kalman filter. It is much simpler implement compared to the Kalman filter and is not very processor intensive. In most applications, the estimate that the complementary filter computes can be substituted for ones computed by the Kalman filter.

Both Kalman filter and complementary filter were trialled in this thesis.

3.4.1 Kalman filter

For our Kalman filter, we assume the robot has a relatively straight pose and model it as the inverted pendulum with pivot located at the centre of pressure of its flat support foot as shown in Figure 3.1.

θ is the torso lean that we require, φ is the pendulum angle, α is the angle in between the torso lean and the pendulum lean, h is the height of the robot, l is the height of the pendulum, f is the position of the pivot relative to the ankle position. f can be roughly estimated from the centre of pressure of the support foot, which is computed from the foot sensors.

The force acting on the centre of mass of an inverted pendulum in the direction of motion is $F = mg\sin\varphi$. We linearise this by assuming $\sin\varphi =\sim \varphi$ as this is true with φ being small ($|\varphi| < 30$ degrees). We arrive at $\ddot{\varphi} = g\varphi/L$ using the angular acceleration definition.

The other system equations are:

- $\alpha = \arctan(f/h)$
- $\varphi = -\alpha + \theta$
- $l = \sqrt{f^2 + h^2}$
- $\ddot{\theta} = \ddot{\varphi}$

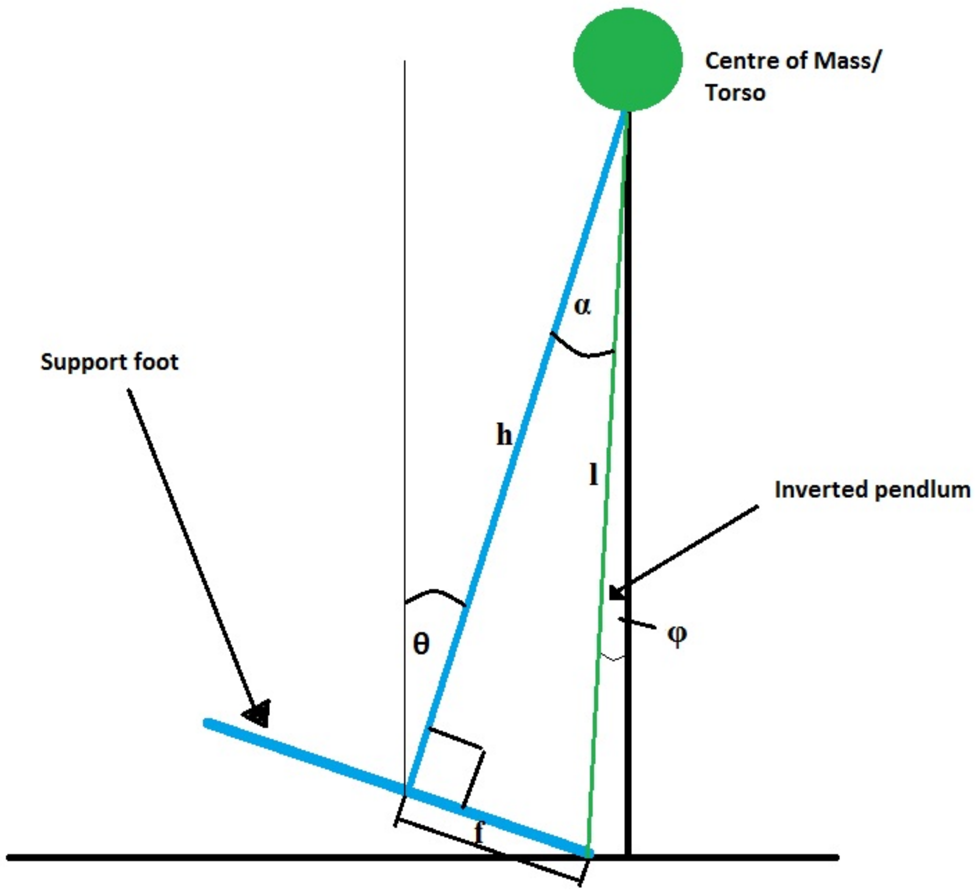

Figure 3.1: The blue stick construction is the robot's support foot and the pivot of the inverted pendulum is where the foot touches the ground.

Thus, the state of the system $(\theta_{t+1}, \dot{\theta_{t+1}})$ with time indexed by t and dt as the time step can be computed:

- $\theta_{t+1} = \theta_t + \dot{\theta}_t dt + \ddot{\theta}_t dt^2/2$
- $\dot{\theta_{t+1}} = \dot{\theta}_t + \ddot{\theta}_t dt$

The standard Kalman filter procedure can then be implemented with those equations used for process update and calibrated readings from IMU used for observation update.

3.4.2 Results

To test the effectiveness of the Kalman filter, the robot collected ground truth data to compare with the filtered data. The robot was made to walk on the spot and force was applied such that the torso rocked. Note that the Runswift vision thread runs in 30 frames per sound while the motion thread runs in 100 frames per sound. So the vision data collected are expected to have a slight lag compared to the IMU data in the motion thread.

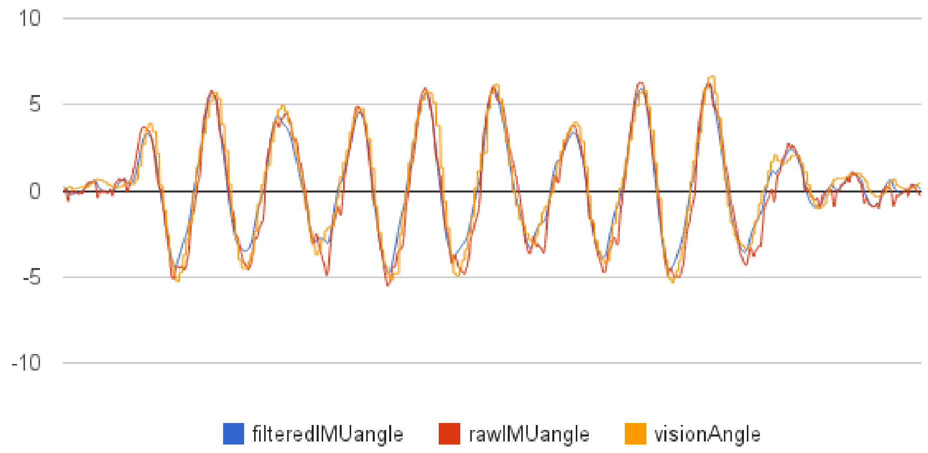

Figure 3.2: The filtered torso angle is super-positioned against the raw torso angle and the ground truth from vision data.

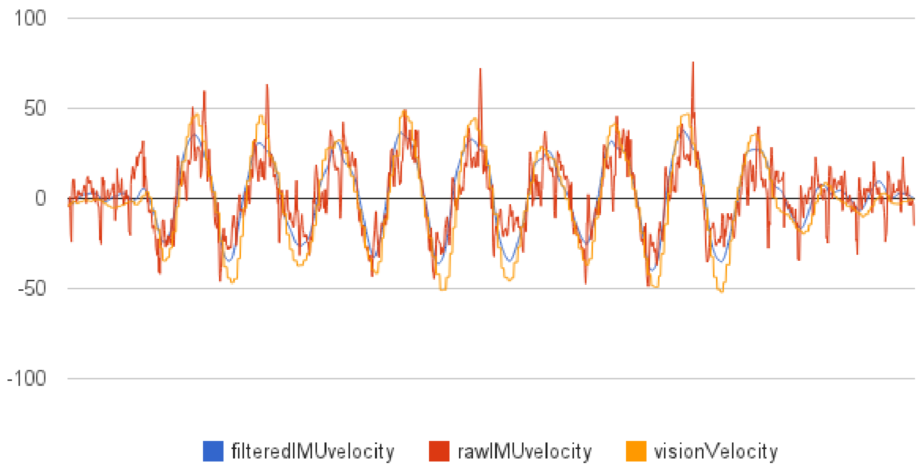

Figure 3.3: The filtered torso angular velocity is super-positioned against the raw angle angular velocity and the ground truth from vision data.

The filter had little effects on the angle readings and was evidently more effective on the gyroscope readings. It was able to filter out the noise in the gyroscope readings and provide accurate estimations. However, it was observed that as walking gaits changed, inaccuracies were introduced. The critical assumption for the process model to work was that the pose of the robot was relative straight such that it would form an inverted pendulum where the centre of mass is at the torso. This assumption seemed to be too crude and other solutions were sought.

3.4.3 Complementary filter

The complementary filter for the IMU was developed as an alternative to the Kalman filter. It does not assume a process model and will therefore be more robust if the walking gaits change.

The idea of the complementary filter is to fuses the accelerometer data, which are more accurate in static conditions, and gyroscope data, which are more accurate in dynamic conditions, to obtain the torso lean.

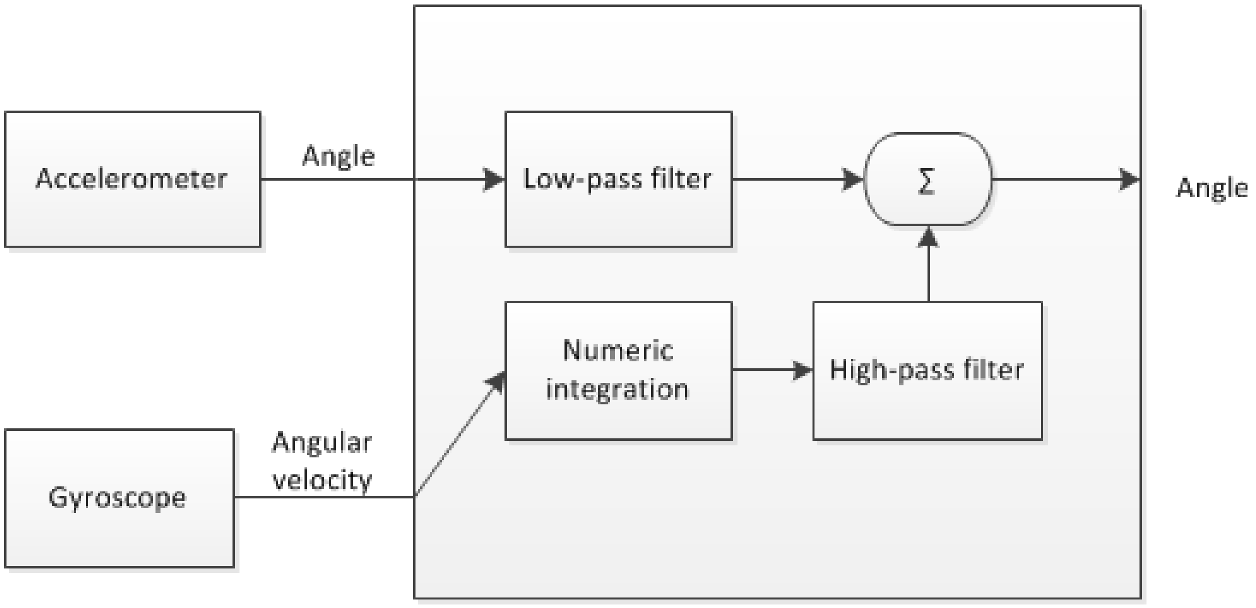

Figure 3.4: The mechanics of the complementary filter.

The initial torso angle is computed from the accelerometer data where $accZ$ represents the downward acceleration and $accX$ represents the acceleration in the axis that we are currently solving for:

$$\theta_a = -atan2(accX, accZ) \tag{3.2}$$

The final filtered angle is obtained from passing θ_a through a low-pass filter and combined with high-pass filtered angle obtained from integrating gyroscope data:

$$\theta = (1 - \alpha) * (\theta + gyro * dt) + \alpha * \theta_a \tag{3.3}$$

The α used was 0.02. Alpha defines the boundary where the gyroscope readings take precedence over the accelerometer readings and vice-versa. This prevents drifting of accuracy when numeric integration is continuously performed on gyroscope data and filters out noise from the accelerometer data.

3.4.4 Result

The complementary filter was tested via the same method as the Kalman filter. Initial tests showed promising results, as there seemed to be a minimal lag in the filtered data and it was quite accurate. However, further tests showed that whenever the robot was turning in a circle, the accuracy of the filter drifted away significantly. I was unsure of the reasons for this. The complementary filter was not used due to this flaw.

3.5 Conclusion

This chapter has documented the Kalman filter and complementary filter applications to the problem of filtering IMU, as well as the method to obtaining ground truth and IMU calibration. The Kalman filter developed for the Nao was effective to a point. However, the process model must be more sophisticated to account for changes in the walking gaits to obtain accurate results. I believe future work should look into further development for the complementary filter as it showed promising results and is much simpler to develop. As a starting point, the inaccuracy that happens when the robot turns in a circle must be solved.

The filtered IMU data was used in camera pose configuration and also feedback control for walk balance. Future work can also make use of the IMU data, torso angle and angular velocity, as the system states for machine learning a policy to balance the robot.

Chapter 4

Motion

4.1 Introduction

This section discusses the implementations to address the shortcomings of the 2011 walk engine. We start with describing the method to accelerate the walk and overcome its stability limitation. We then look at the implementation of coronal rock and how it creates a smoother walk motion. Following this we will look at the addition of arm stiffness control to assist robot avoidance. Lastly, changes made to adjust to the new Nao 4 hardware and additional walk modifications are documented. We will also look at the results for individual components to evaluate their effectiveness.

4.2 Walk acceleration

A key feature of the 2010-2011 walk engine was that walk parameters are adjusted slowly to match the input parameters, which helped with stabilisation of the robot. The problem was that a 1-2 seconds was required for the robot to accelerate to full speed or to decelerate to a stop. This affected the agility of the walk and behaviours such lining up to the ball was more difficult and had to be slowed down. Increasing the rate that walk parameters adjusted to reach their input would cause the walk to be unstable.

A simple solution was implemented this year to mend this flaw. To counter act the effects of accelerating and decelerating such that a higher rate of walk parameter adjustment can be handled without destabilising the robot, a small bending motion was added to the hip pitch of the robot. This bend assists the torso of the robot to lean into the direction that it is accelerating to, so that the push back effect of acceleration is nullified.

Before the ActionCommand request is passed to the WalkEngine module, the forward step component is recorded. The difference between the current forward step and previous forward step is calculated and knowing the maximum limit of the forward component, a percent in the change of step size can be calculated:

$changePercent = (curForward - prevForward)/limForwardPatter$

This change percentage is multiplied by an arbitrary angle, and before it is added to the hip pitch, its value is interpolated at a constant rate to ensure abrupt changes wont occur:

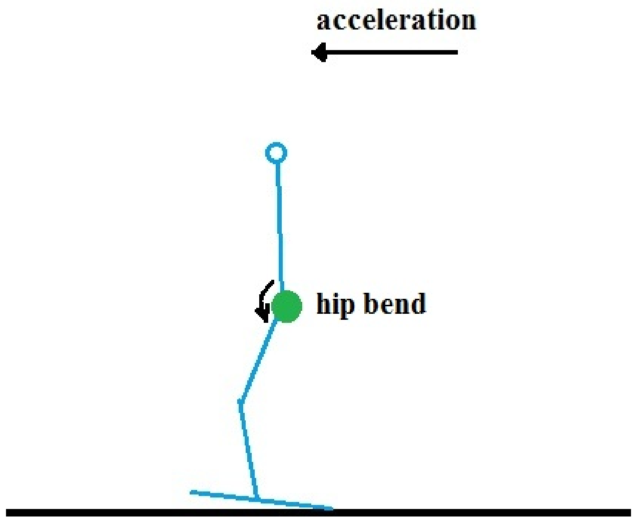

Figure 4.1: Bending the hip such that the torso leans into the direction of acceleration.

$fall = -changePercent * DEG2RAD(10)$
$toFall+ = 0.1 * (fall - toFall)$
$request.jointOffset[PITCH][HIP]+ = toFall$

4.2.1 Results

Here is a comparison of the time it takes for the robot to accelerate to full speed and decelerate from full speed to stop before and after walk acceleration was added:

	Time to full speed(sec)	Time to stop(sec)
No acceleration	2.2	2.2
With acceleration	1.3	0.7

The robot remained relatively stable while gaining 40-70% higher rate of acceleration and deceleration.

Notice that the time taking to stop was significantly cut down. This meant that the time the robot took to slow down and line up against the ball was also reduced. This was a significant edge that allowed our robots to kick the ball before enemy robots.

The ability to suddenly stop and not destabilise the robot also gave birth to another type of kick. The robot was able to bump into the ball to dribble it without having to stop and transition into a kick stance. However, the accuracy of this kick was a big problem and it was not used. This is discussed in more detail in[5].

4.2.2 Conclusion

The bending motion added to the hip of the robot was an effective way to achieve a much quicker rate of acceleration and deceleration. This has also resulted in speed up of lining up against the ball to take initiative against enemy robots.

4.3 Coronal rock

While the robot walks, it naturally rocks in the coronal (or lateral) plane as the centre of pressure shifts between support feet. However, without any addition rocking motion in the hip, the weight of the robot impacts on the edges of the feet as the support foot lands and creates a rough walking motion. This also increased the difficulty to process camera images as the rough impact and sway produced noise in the images. To create a smoother walk, an additional rocking motion in the coronal plane was needed to comply the natural rock.

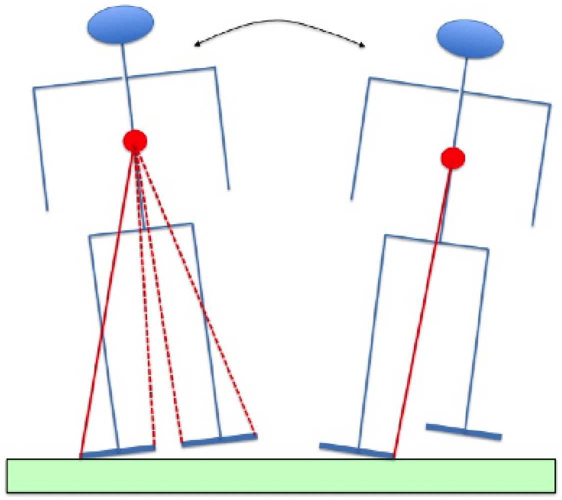

Figure 4.2: Natural coronal rock when no rocking motion is in the hip. The walking motion is rough and the head sways rapidly.

Coronal rock was experimented with last year, but a successful rocking motion was not found. Team B-Human, having developed one of the smoothest walk, modelled the trajectory of the centre of mass as the robot walks using the Linear Inverted Pendulum Model[9]. This meant that the rocking of the hips was dictated by hyperbolic functions of time according to the model. This is demonstrated in the Figure 4.3.

Coronal rock was implemented this year using a simplified concept. A sinusoidal function was used to model how the centre of mass moves during walk. Where T is the period[1] of the walk, t is the current time step within the period, and $rockMagnitude$ is an arbitary value of the magnitude of the rock, the forcing function of the coronal rock is:

$rock = rockMagnitude * \sin(t/T * 2 * \pi)$

[1]The period of the walk is the time it took to complete a step with both feet of the robot.

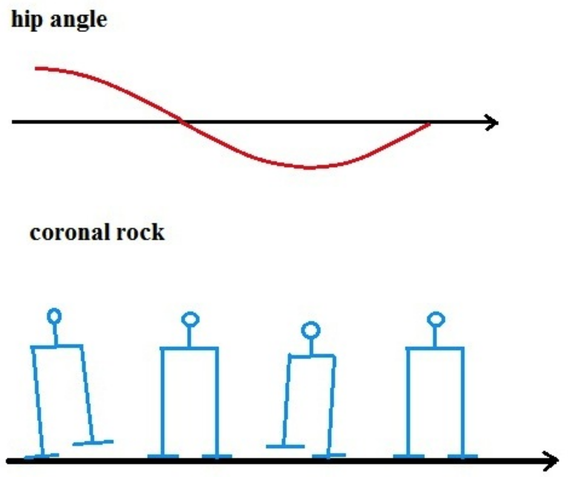

Figure 4.3: The Nao robot stylised as stick figures showing the hip rock as it walks.

However, the forcing function alone is not enough. A degree of compliance with the natural rock must be achieved so that the feet of the robot can remain flat on the ground. This was achieved by using the accelerometer as sensor feedback and its scaled value was added to the rock such that the lateral acceleration was dampened:

$compliance = -bodyModel.getFilAccY() * scale$
$request.jointOffset[ROLL][HIP] = rock + compliance$
$request.jointOffset[ROLL][ANKLE] = -(rock + compliance)$

The compliance creates a smoother walking motion by reducing rough disturbances from the edge of the foot digging into the ground when the support foot switches.

4.3.1 Result

To test the effectiveness of the coronal rock, plots of coronal torso lean while the robot is walking with and without the coronal rock are super-positioned shown in Figure 4.4.

Even though the effect of the coronal rock was not very significant, the sway of the torso from side to side was noticeably lower with coronal rock implemented. This also assisted vision processing as camera images were also swaying less.

Also notice that the coronal rock made the walking period slightly lower than before, as represented by the red line in the plot. As a part of the coronal stabilisation control implemented in 2010[3], the walk is paused if the centre of pressure has not shifted over to the next support foot during the double support phase[2]. This prevented premature alternating of the support foot such that the weight of the robot can first be shifted over to the next support foot. It meant that the actual period of the walk was sometimes longer than the specified period due to natural rocking. What the coronal rock achieved was that the lifted foot was able to be touch down in time due to the coronal rock complying with the natural rock, such that the actual period of the walk closer to the specified period. This wasnt noticeable in practice but it meant that the walk was slightly faster.

[2]Double support phase is where the walk is about to alternate support foot and both feet are on the ground.

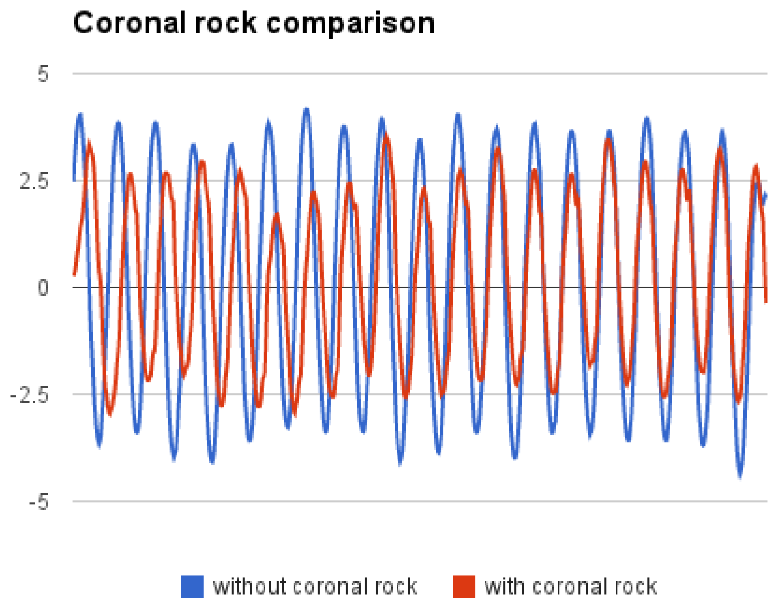

Figure 4.4: The super-positioned plots of coronal torso lean with and without coronal rock.

4.3.2 conclusion

The implementation of the coronal rock motion created a smoother walking motion and reduced swaying of the head of the robot to assist vision processing. However, the effect was not very significant. I believe a more sophisticated system model is required for a more significant effect.

4.4 Arm stiffness control for avoidance

Robot avoidance is an important part of Robocup as colliding with another robot will damage the hardware and causes the robot to fall down. The pushing robot also has the risk of being penalised. Even though the robot makes an attempt to avoid its obstacle, the arm of the robot often brushes against the obstacle. The issue of avoidance is mainly addressed in the behaviour module, however the motion module makes use of arm stiffness control to assist with avoidance.

The arms of the robot normally perform alternating swings to counter balance weight shift from side to side caused when walking. However, the arms occupy about 40% of the total width of the robot. This makes robot avoidance a challenging task. An obvious solution is to keep the arms at the back of the robot. However, the arms will no longer counter balance weight shifts and this affects the stability of the robot. Team HTWK[10] uses a unique walking style to compensate for the shortcoming of this solution.

An alternate solution is to let the arms of the robot continue to swing normally, and when robot avoidance is required, the stiffness values of the arms are turned off, allowing the arm of robot to slide past its obstacle much easier. Controlling stiffness is a better solution to shifting the arms to the back of the robot because the act of shift may destabilise the robot.

Note that there is a lag in visual odometry and that the change in heading of the robot wont be detected immediately. This means that the stiffness of the arms wont be turned off right away. To account for this, the arms are not 100% stiff by default. Shoulders have only 10% stiffness where as elbows have 30% stiffness. This achieves a degree of compliance to soften the collision whilst still being able to swing the arms.

4.4.1 Result

The arm stiffness control effectively assisted robot avoidance as the rate that the robot falls down due to colliding with an obstacle on the side is significantly lowered. The robot often brushes past the obstacle using the limpness of its arms.

However, visual odometry isnt the best sensor to use for the purpose of detecting collision. Many false-positives were seen in the tournament. Luckily, the use of stiffness control did not destabilise the robot even when there was a false-positive. Visual odometry was only used because arm stiffness control was introduced at a later stage of the development and other sensing methods were never experimented with.

4.4.2 Conclusion

The method of arm stiffness control to assist robot avoidance was document in this section. With the use of visual odometry for detecting collision, arm stiffness control was effective in preventing the robot from falling when colliding with obstacles on its arm. However, a better method for detecting collision should be investigated for future work to minimise false-positives and improve accuracy.

4.5 Back getup routine

Getup routines had to be redesigned this year due to the upgrade of Nao version 4 that introduced bulkier hand shapes and different weights for each component[11]. The redesign of front getup routine was covered by Teh[5]. This section details the redesign of the back getup routine.

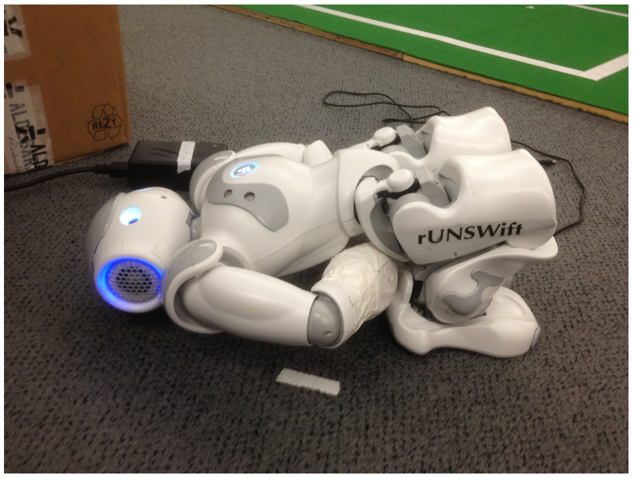

Figure 4.5: Legs are curled down to create space under the back for the arms to shift in and support the weight of the robot.

The main problem of old back getup was that the bulkier hands of the robot got stuck on the ground when trying to move to the back of the robot and could not reach a position to support the weight of the robot while attempting to get up. The fix for this was to curl the legs of the robot to create space behind the back for the hands to easily shift under, therefore creating a pivot where the weight of the robot could be supported when getting up. The getup routine pos file was updated with these initial steps and minor changes were applied to the steps after to connect with the initial steps. The redesign was inspired by popular back getup routines used by teams such as B-Human and RoboEireann[13].

4.6 Walk modifications

With the introduction of the Nao version 4, the new hardware does not overheat as often and the motors were changed from plastic to metal such that every motion experienced an increased torque[12]. Several changes were introduced to adjust to and take advantage of the new hardware. Other minor changes were also described in this section.

4.6.1 Walk parameters

The modifications to walk last year[**?**] featured switching between 2 modes: a fast walk with a higher step frequency and low stance, and a conservative walk with a lower step frequency and high stance. The role of the conservative walk was to walk at a higher stance to reduce overheating of the robot cause by a stressful large knee bend in fast walk. However, the conservative walk could not handle higher speeds so it was necessary to transition between the modes. The transition time between the modes was unfavourable in competitive soccer play due to slow start-up phase and slow reactions after kicking.

Fortunately, the Nao version 4 did not overheat as often. This meant that the robots were less dependent on the conservative mode of the walk to reduce overheating, and it would be more beneficial to cut down the transition times. The two walk modes were replaced with just one and walk parameters were retuned and kept constant such that no transition occurred. The differences between the walk parameters are documented below:

	Step period(sec)	Knee bend(degrees)
Fast Walk (old)	0.38	40
Conservative Walk(old)	0.5	20
Walk	0.5	20

4.6.2 Lifting function

Additionally, the lifting function of the moving foot was also changed from using an inverted parabola to a skewed sinusoidal function after some testing was done to determine which worked better. We define x as the percentage within the time allowed to take a step, and $L(x)$ as the height above ground the foot had lifted. The function is defined as:

$$L(x) = \begin{cases} \cos((x-\pi)*5)/2+0.5) & : x \leq 0.6 \\ \cos((x-\pi/2.2)*7.6)/2+0.5) & : x > 0.6 \end{cases}$$

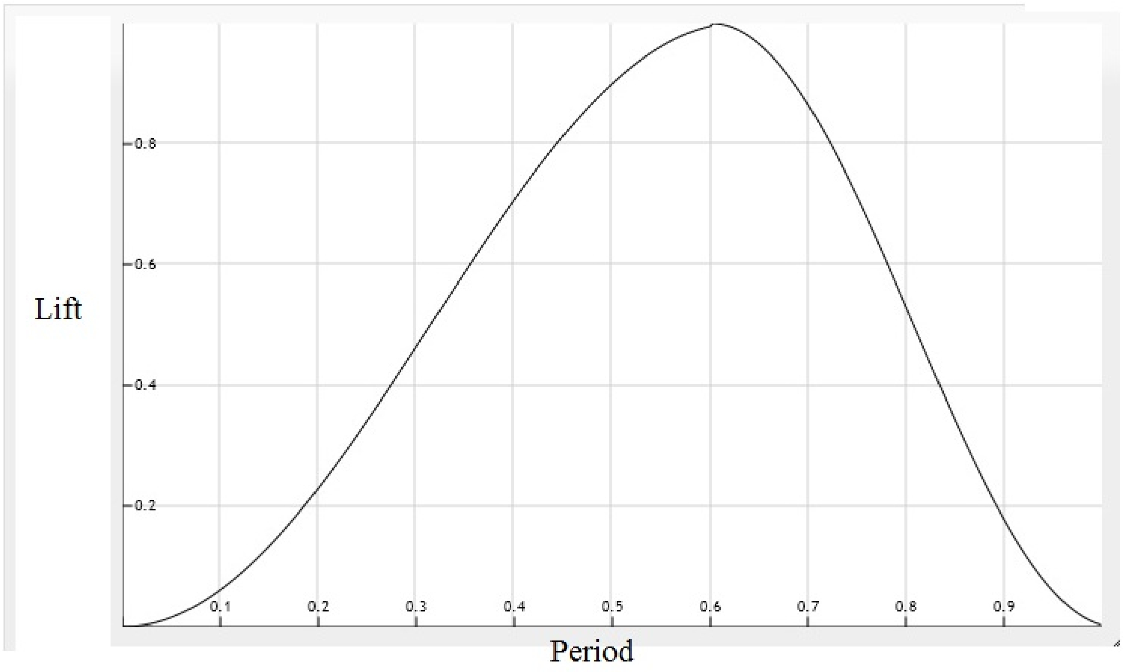

Figure 4.6: piecewise super-position of the lifting function of the foot.

The rationale for choosing such a skewed sinusoidal function was to move the phase where the foot is placed down towards the end of the step period. This was done to avoid the case where the foot kicks into the ground during landing, due to disturbances, which causes the robot to fall.

4.6.3 Sagittal feedback control

Sagittal (or forward) stabilisation was implemented in 2010[3]. It used feedback components based on the centre of pressure computed from foot sensors of the robot. This was replaced by IMU feedback control to achieve the same effect and not rely on foot sensors (due to its unreliability) for such critical component.

The feedback components are:

- High and low frequency responses to torso lean
  $highAng = 0.8 * highAng + 0.2 * angleY$
  $lowAng = 0.95 * lowAng + 0.05 * angleY$
- Filtered acceleration in x-direction:
  $filAccX += 0.04 * (DEG2RAD(accX * 0.2) - filAccX)$
- Filtered gyroscope reading directly from the Sensor module:
  $filGyro = sensors.sensors[Sensors :: InertialSensor\_GyrY]$

These components were used to counter-balance unwanted sagittal rock and was applied to the hip pitch as follows:

$hipOffset = (-highAng + lowAng)/6.0 - filAccX + filGyro/400.0 - DEFAULT\_LEAN$

The scales for each component and filter magnitudes and gains were empirically tuned such that the rocked sagittal lean of the robot stabilised in minimal number of steps. A slight default lean forward was added to the hip pitch to bring the centre of mass closer to the centre of the feet such that it was easier for the robot to balance if it was rocked backwards.

4.6.4 Result

With all modifications to the walk this year, the robots were able to reach a forward speed of 26-30cm/s, side step speed of 13cm/s and turn speed of 70 degrees/s. The speed aspect of the walk showed to be crucial in the Robocup tournament as our robots were able to out pace most components. The walk seemed to be equal in speed against other teams with a well-developed walk such as team B-Human. In our pool match against B-Human, our robots were able to race with them side by side toward the ball. However, B-Human seemed to out pace us when making turns for the ball with seemingly instantaneous turns. This showed that our walk was still not as agile compared to them, due to the gradual speed up feature of our walk. On a minor note, we were able to win the unofficial running contest this year, even though there was only one other participant.

The walk also showed its robustness during the competition. We did not have to tune any walk parameters to adjust for the competition surface, where as many other teams struggled to tune their walk for the Mexican carpet. We might have been lucky that our walk adapted to the surface, but I believe one of the benefiting factors was that foot sensor feedback signals were replaced by IMU signals for sagittal stabilisation.

Despite the effort in minimising the use of foot sensor signals, we still had major problems since coronal stabilisation (discussed in the coronal rock section) still made use of centre of pressure calculated from foot sensors. It affected one of our robots by pausing the walk and the malfunctioning happened intermittently even after being fixed by Aldebaren. An attempt was made to account for malfunctioning foot sensor by detecting whether the robot had been in double support phase for too long, and if detected true, the walk resumes regardless of centre of pressure. However, malfunctioning foot sensors could report any value and this fix was not valid.

Overheating and the wear and tear of the robot did not seemed to be a problem before the competition. However, as soccer matches continued daily, the built up stress on the robot began to show. This was worsened by major wireless problems that caused robots to collide with each other and fall because every robot took the striker role and fought over the ball. By the end of the competition, the robots were clearly very unstable and began to sway back and forth even though the walk engine was not changed. The falling rate of the robots increased dramatically and the wear and tear also reduced the success chance of our getup routines dramatically, since it appeared that there wasnt enough torque in the motors to execute the heavy-duty motion. To compensate for this, both the acceleration of the walk and the maximum speed were tuned down. The drop in speed was notable in the semi-finals match against Austin-villa, but the rate that the robots fall over was also decreased. The rumour heard from another team was that there was a change in the IMU component after Aldeberan fixed the robots during the competition. It might have affected our sagittal stabilisation.

4.7 Conclusion

With the changes made to accustom and adjust to the new hardware, the walk was able to reach a higher speed, which was competitive amongst the best teams. Future work should aim to improve the agility of the walk, the ability to swiftly shift between different directions.

Replacing foot sensor feedback with IMU feedback increased the robustness of the walk. However, foot sensor unreliability still caused problems for coronal stabilisation. This can be overcome by correctly detecting the state that the walk is paused and resume the walk regardless of coronal stabilisation. Another solution is to develop a different stabilisation model without the use of foot sensors, which can be difficult. The current solution only detects whether the walk was stuck in a double support phase, which was aimed to free robots that were getting stuck next to the goal post and did not consider the possibility of malfunctioning foot sensors.

Lastly, the stability and sustainability of the robot were overlooked. Even though the new hardware does not overheat as often, the decision to remove conservative walk was questionable.

As for wireless issues, the direct solution was to develop more robust team behaviour to not rely on it, and that was done during the competition. However, future work should also improve on the stability of the walk such that it rarely falls even if collisions are constant.

Chapter 5

Goalie

5.1 Introduction

This chapter will explain takes involved in each state of the goalie state machine. We will then look at the transitions that occur between each state. Lastly, we will evaluate the effectiveness of the goalie in the competition.

5.2 LostScan

The LostScan state is entered if the external isLost function indicates that the robot is lost and requires localisation assistance. The state runs the external lostScanSkill that turns left and right slowly and scan for near field features.

5.3 ReturnOrigin

The ReturnOrigin state returns the goalie to the goal centre following a few simple rules.

1. If the robot is already within 1000mm of the goal, then it will set the *keepFacing* flag to true and passes this to the *walkToPoint*[1] helper function so that the robot will try to face the ball while walking. Dont face the ball if the robot is 1000mm away from the goal. The reasoning was that it is important for the goalie to face and track the ball, but it is more important to get back to cover the goal as quick as you can if you are far away.

```
$keepFacing = False$
**if** $myDistanceToGoal < 1000$ **then**
    $keepFacing = True$
**end if**
```

2. The robot attempts to avoid the goal posts on its way back to goal if it came from the sides. The simply approach for this was to add 400mm to the x-direction[2] of the target if the y-

[1]walkToPoint is an external function that handles all walking tasks and how you want to walk, such as whether you should face the ball or use avoidance for obstacles[8]

[2]The 2 goals on the field are on the x-axis in our coordinate system.

position of the robot is still above or below the goal posts. This way, the goalie will attempt to loop around the goal posts from the side.

```
if $|robotY| - 400 > GOAL\_Y$ then
    $targetX += 400$
end if
```

3. Lastly, if the goalies distance to goal is close enough, the *urgency* flag for *walkToPoint* function is lowered to 0.6. The urgency flag is simply a scale of speed the walk is set to. This acted as a hysteresis such that robot does not overshoot and walk over its destination due to a slight lag in localisation.

```
$urgency = 1$
if $myDistanceToGoal < 300$ then
    $urgency = 0.6$
end if
```

5.4 Intercept

The task of the Intercept state is to have the goalie clear the ball. The intercept state is trigger and if either the *shouldClear* or *shouldBeAggressive* condition is satisfied. Conservative decisions were focused since we believed the ability for goalies dive to save the goal if we did not choose to clear the ball.

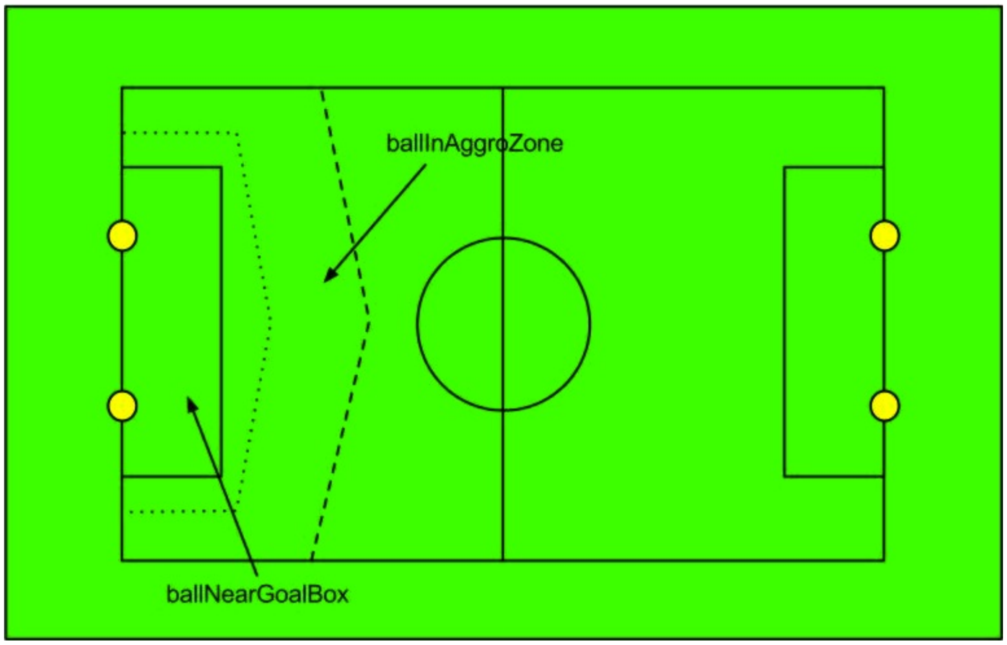

Figure 5.1: Zones which shouldClear and shouldBeAggressive conditions are checked.

5.4.1 shouldclear

Several conditions are first checked for early termination.

1. If the team has no information about the ball, indicated by $ballsVisibleToTeam$, then the goalie cannot clear the ball.

```
if $ballsVisibleToTeam == 0$ then
  return $False$
end if
```

2. Note that the Intercept state will function even if the goalie himself cannot see the ball and can only rely on the team ball information. This is to handle cases where the ball is too close to the goalie that it can only be seen by helpless teammates outside of the goal box. The problem is that the Intercept state will delegate to the searchForBall behaviour and the behaviour makes the robot turn in a circle to search for ball. To prevent the case of the goalie turning in circles outside of the goal area if the ball cannot be seen by goalie, the condition requires the goalie to be close to the goal area.

```
if $\neg isMyBallSeen \wedge \neg isCloseToGoal$ then
  return $False$
end if
```

3. Next if the ball is already in the goal box, indicated by $isBallInGoalBox$, the goalie should clear the ball since the goalie is the only one allowed to be inside.

```
if $isBallInGoalBox()$ then
  return $True$
end if
```

4. The ball has to in the area near the goal box as seen in Figure 5.1. Note that the ball position that the goalie sees is always used if its available, since its the most accurate. If its not available, filtered team ball position is used instead.

```
if $\neg ballNearGoalBox$ then
  return $False$
end if
```

5. The new robot filter[7] can obtain accurate enemy robot data up to a range of 2 meters. We took advantage of this and used the enemy data to aid the goalies decision. The goalie chooses not to clear the ball if its distance to ball is further than 700mm away and an enemy robot is within 300mm of the ball. This is because the cover position is left open for a short time when the robot walks up to clear the ball. The enemy robot can shoot the ball at any time and we wont be close enough to block it.

```
if $myDistToBall > 700$ then
  for all $enemy \in enemyRobots$ do
    if $enemy.distToBall < 300$ then
      return $False$
    end if
  end for
end if
```

6. We also do not want to collide with the striker and compete for the ball. This is checked via the broadcasted team information. The team role behaviour roughly dictates that the closets robot to the ball is the striker. The striker that the goalie is interested in is not fallen over and can see the ball. The *isFallen* flag is checked and if the robot cant see the ball, their *distToBall* variable will be **NaN**. If a valid striker is not found, the goalie will clear the ball.

```
$striker = None$
```

```
for all $teammate \in teammates$ do
  if $\neg teammate.isFallen$ then
    if $\neg isNaN(teammate.distToBall) \wedge (striker == None \vee teammate.distToBall < striker.distToBall$ then
      $striker = teammate$
    end if
  end if
end for
if $striker == None$ then
  return True
end if
```

7. After the striker is found and if the goalie is not already within 300mm of the ball, strikers distance to ball is compared to the goalies distance to ball to roughly estimate who will first get to the ball. The goalie decides that it will get to the ball first if strikers distance is greater plus a 500mm margin in favour of striker. These parameters are tuned to promote conservative play.

```
if $myDistToBall > 300 \wedge striker.distToBall + 500 > myDistToBall$ then
  return $True$
end if
```

8. Lastly, strikers own *isInGoalBox* function is checked to see whether it thinks the ball is in the goal box. Since the *isInGoalBox* function is shared between striker and goalie, the goalie will always clear the ball if striker stops because it thinks the ball is in the goal box, regardless of errors in their ball location.

```
if $striker.isBallInGoalBox()$ then
  return $True$
else
  return $False$
end if
```

5.4.2 shouldBeAggressive

The area that the *shouldBeAggressive* condition is tested is much further away from the goal. Strict conditions are tested and the goalie will only aggressively clear the ball if deemed necessary.

1. First, the goalie must be able to see the ball and the ball must be in the *ballInAggroZone* area shown in Figure 5.1 to clear the ball.

```
if $isMyBallSeen \vee ballInAggroZone$ then
  return $False$
end if
```

2. There is considered to be only one safe condition for goalie to clear the in this area. The goalie will only clear the ball if the closest robot to the ball is further than 500mm of the ball and its distance to the ball is greater than goalies distance. The *getClosestToBall* function obtains the closest robot to ball using both robot filter data and broadcasted information for team locations to get maximum coverage.

```
$closestToBall = getNearestToBall()$
```

```
if $closestToBall.distToBall > 500 \wedge closestToBall.distToBall > myDistToBall$ then
  return $True$
else
  return $False$
end if
```

After either *shouldClear* or *shouldBeAggressive* condtion is passed, the goalie simply makes use of the Striker skill to attack the ball. The Striker skill makes special goalie exceptions and is covered in striker behaviours[5].

5.5 CoverGoal

The goalie behaviour last year switched between 3 different positions to block the goal as much as possible. It could not be significantly more sophisticated without better localisation. However, an improvement was made along with localisation improvements this year.

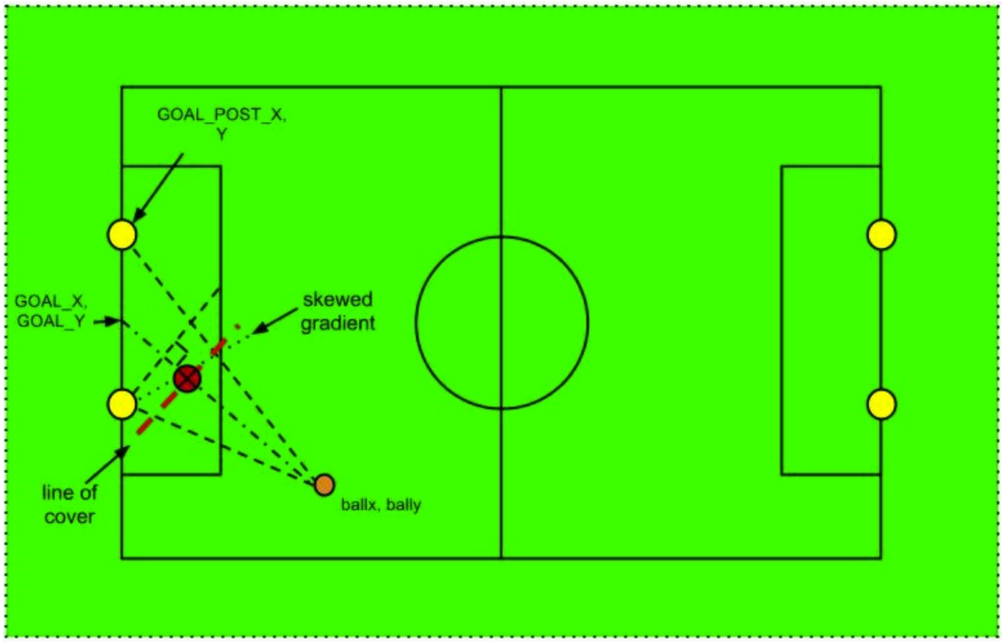

Figure 5.2: Shows how the best cover position is derived.

The best cover position for the goalie to be in is on the midline from the ball to the goal centre, intersecting with the line of the cover perpendicular to the midline. The line of cover needs to be pushed forward pass the goal posts, so that when the ball is blocked, it safely rolls outside of the goal posts. See Figure 5.2. The cover point also needs to be within the goal box so that the goalie can legally cover the ball with its arms. The possible positions of the cover point form an arc, known as the goalies arc. The *CoverGoal* state uses the *getGoalCoverPos* function to determine the best position to cover the goal by using this strategy.

5.5.1 getGoalCoverPos

1. First the function checks if the ball lies near the axis of the goal. If so, it assigns hard-coded cover positions to prevent math error when calculating gradient.

```
if $ballx - GOAL\_X < 100$ then
    $targetX = GOAL\_X + 100$
    $targetY = sign(bally) * (GOAL\_POST\_Y + 50)$
else if $|bally - GOAL\_Y| < 1$ then
    $targetY = GOAL\_Y$
    $targetX = GOAL\_X + 250$
end if
```

2. Next, we find the gradient of the midline from the ball to the centre of goal, m. The line perpendicular to this midline is our line of cover, with gradient $\bar{m}$. The line of cover is pushed forward by adding a skewed gradient, s to $\bar{m}$, such that the angle between $\bar{m} + s$ and $\bar{m}$ is constantly 10 degrees. The reason for using a constant angle for the skew was so that the distance that the line of cover is pushed appropriately adjusts depending on where the position is.

```
$m = bally/(ballx - GOAL\_X)$
$\bar{m} = -1/m$
$angle = sign(m) * 10$
$s = \tan(\arctan(\bar{m}) + angle) - \bar{m}$
$\bar{m} += s$
```

3. Lastly, we use simultaneous equations to work out the intersection point of the line of cover and the midline, as it is the best possible position to cover the goal.

```
$c = bally - m * ballx$
$pY = sign(bally) * GOAL\_POST\_Y$
$pX = GOAL\_X$
$d = pY - \bar{m} * pX$
$targetX = (d - c)/(m - \bar{m})$
$targetY = m * x + c$
```

4. The x-position of the position is capped 150mm before the goal line to avoid hitting the goal posts.

```
if $targetX - GOAL\_X < 150$ then
    $targetX = GOAL|_X + 150$
end if
```

5. After *targetX* and *targetY* is obtained from *getGoalCoverPos*, *CoverGoal* state calls *walkToPoint* by setting the necessary flags. Urgency is set for hysteresis and robot avoidance is off in goalies own box. Heading kept facing the ball while during movement is a must.

```
$urgency = 1$
if $myDistToTarget < 300$ then
    $urgency = 0.6$
end if
$use\_avoidance = False$
$keepFacing = True$
```

5.6 Anticipate

The Anticipate state follows after either ReturnOrigin or CoverGoal is finished, and places the robot in a squat position ready for dive. The idea is that once the goalie is in an ideal position, it is advantageous to squat such that the diving motion is quicker and causes less damage to the robots hardware. The squat motion was implemented using the pos file.

In 2011, a standing motion was used to assist tracking the ball if its far away and switched to a squatting position once the ball was near. However, the vision system was vastly improved in 2012, and the goalie can track the goal relatively far away even when squatting. Therefore, the standing motion did not need to be used any more.

5.7 Dive

The task of the Dive state is straight forward, to dive at the trajectory of the ball. There are two parts of implementing the Dive state. Detecting the need to dive and the actual diving motion.

5.7.1 Detection

The Dive state is triggered if the *ballTimeToPassGoalie* function returns a time lower than the threshold time to dive. The function works as follows:

ballTimeToPassGoalie:

1. The time it takes for the ball intersects the goalies sidelines is calculated using quadratic equation where a is the carpet deceleration, $ballx$ and $ballVelx$ are the filtered robot relative ball position and velocity in the x-axis respectively. The determinant of the quadratic equation is used to detect whether the ball will intersect goalies sidelines, and only returns a valid time if it intersects. Lastly, whether the ball is currently moving is checked using the $ballMoving$ flag, This comes from the ball filter, which has two modes, one for modelling moving balls and one for stationary balls[6]. The function only returns a valid time if the filter had switched to the moving ball mode to prevent false-positives while walking towards the ball.

   ```
   $a = 400$
   $det = ballVelx * ballVelx - 2 * a * ballx$
   **if** $ballMoving \wedge det \geq 0$ **then**
     $t = (-ballVelx - \sqrt{det})/a$
   **else**
     $t = \infty$
   **end if**
   ```

   The modelling of deceleration of the carpet was implemented during the competition because the Mexican carpet made too much of a difference in decelerating the ball.

2. The destination that the ball arrives on the y-axis of the robot, $destY$, is project and two sanity checks are then performed. First, $destY$ further than the length between the goalie post and centre of goal with a leniency scale of 1.1 is checked, such that the robot is prevented to dive for balls that it cannot reach. Next, balls further than 3000mm away (half of the soccer field) from the goalie are also ignored due to increasing inaccuracies the further away the ball.

   ```
   $destY = bally + robotVely * t$
   **if** $|destY| > GOAL\_POST\_Y * 1.1$ **then**
     $t = \infty$
   **end if**
   ```

```
if $ballx > 3000$ then
  $t = \infty$
end if
```

3. It would make sense to check if the trajectory of the ball goes into the goal as to not waste time diving if it does not. However, this required the goalie to know precisely where its own location is and made checking this condition very unreliable due to inaccuracies in localisation. As such, decision to dive was made on robot-relative information alone. This was also an issue back in 2011 described on page 20 of[14].

4. *destY* is used to determine which direction the goalie should dive for. The goalie will be able to reach the ball using *DiveCentre* if *destY* is within 280mm of the goalie. Otherwise, *DiveLeft* or *DiveRight* will be appropriately chosen. The other case where *DiveCentre* is triggered is when the ball is rolling away from the goal indicated by a positive velocity of the ball in x-direction, *absBallVelX*, on absolute coordinate system[3]. The rationale is that there is a chance that the heading of the robot is incorrect due to inaccuracies in localisation just after falling over. The two possible scenarios are if the ball was actually rolling towards the goal and we did not save it, or if another robot was clearing the ball away from the goal and we chose to *DiveLeft* or *DiveRight*, which blocks the ball from clearing and also waste time falling over due to the diving motion. The best trade-off is to use *DiveCentre* to attempt to block the ball and not waste a lot of time since it does not make the robot fall.

```
if $|destY| < 280 \lor absBallVelX > 0$ then
  $DiveCentre()$
else if $destY < 0$ then
  $DiveRight()$
else
  $DiveLeft()$
end if
```

5.7.2 Dive motion

The ability to dive to save the ball is paramount in goalies arsenal. The requirements of the dive are to be fast enough to save the goal, and to be low impact as to minimise the damage caused to the hardware. Three different dives were developed for the goalie. DiveCentre, DiveLeft and DiveRight to cover the centre, left and right respectively. The motions are coded using pos files.

DiveCentre A goalie dive is often performed as a complex falling motion in the left or right direction. However, it is not desirable for the robot to dive often for the following reasons:

- Diving causes damage to the robots hardware
- A long time is needed to perform get up routine and to get back to position to defend, leaving the goal defenceless.
- Inaccuracies are introduced in localisation when the robot falls, especially in its heading.

Last year, a centre block was developed to block the ball rolling towards the centre of the robot such that the robot only needs to dive left and right when necessary to save the goal.

[3]Robot relative coordinate system centers the axis on the robot. Where as absolute coordinate system centers the axis on the soccer field. Robot relative coordinates are often more accurate since the input can be from the vision system without having to rely on localisation.

The possible improvements for it were to increase the width of the centre block up from 34 centimetres, and reduce the transition time for a quicker block.

This year a new centre block, DiveCentre, with an increased width was developed. The new centre block consists of 2 main parts. First, the robot lowers its height and partially opens its hip yaw in preparation. Second, the robot opens its hip yaw completely and squats down to block. The first part allows the robot to be in a more stable stance before transitioning into the final part to improve stability as transition time is sped up. From the squatting position in Anticipate state, the total transition time it takes to get to the final blocking position is only around 0.4 seconds.

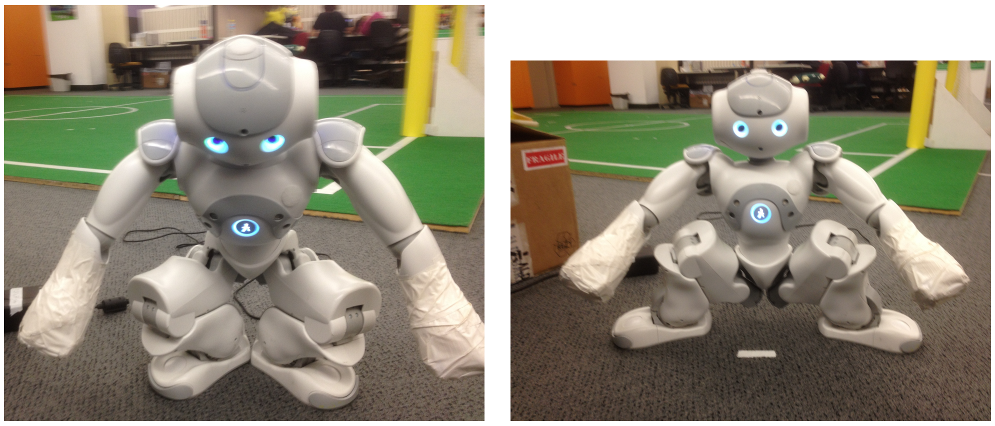

Figure 5.3: The first and second part of DiveCentre.

DiveLeft and DiveRight Even though diving to the side was risky, it was necessary to improve it since it can reach for balls further away.

The work from last year improved on the transition time of the dive as well as reduced the damage done to the hardware by switching off stiffness in the joints. The speed of the dive was further improved this year, with strategy to reduce damage on impact.

The robot opens its hip yaw slightly and angles its arm outwards such that it would roll on its back to dissipate the energy from impact. The stiffness of joints is also turned off to minimise the damage.

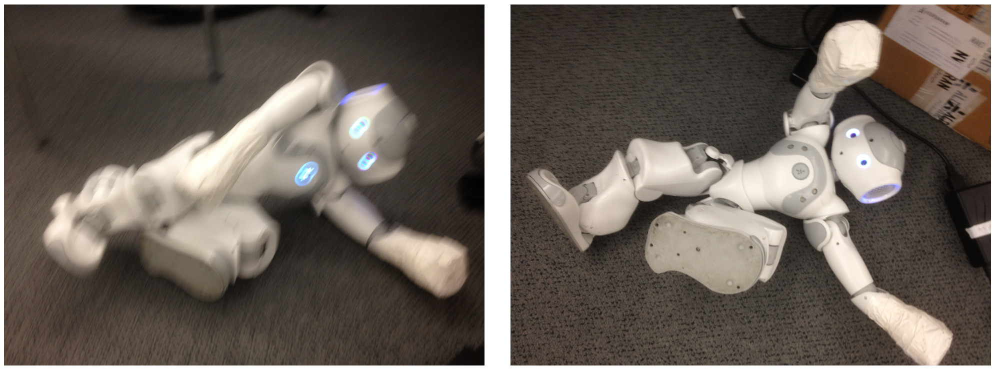

Figure 5.4: The dive motion rolling the robot on its back.

It was found that landing the back of the robot was not ideal, since when the robot got up the ball was most likely to be behind. This meant extra time was needed to turn around and it left the goal defenceless during that time. It proved to be difficult to use similar strategies to roll on the front of the robot while retaining the speed required. This became a trade off of either using the old dive which is a bit slower but lands on the front of the robot, or using the new dive which is faster but lands on the back. Penalty-striker versus goalie test was carried out to make a decision of which dive to use. It was found that the new dive saved significantly more shots than the old dive. It was decided that against better teams who are likely to have stronger kicks, it is better that we at least save the first shot at the goal.

An arm flick was added at the end of the dive to elevate the problem of a longer time needed to get back into defending positioning after a dive. It was found that the ball would stop within range of the arm most of the times after a dive, and so the robot would flick the ball out of the goal box. The ball can be flicked as far as across half of the field. Not only does it assist with clearing the ball, the ball going outside of our goal box also means other team members can have a go at the ball without being penalised for illegal defender.

Lastly, odometry is updated after a dive to reflect a change in heading to assist with localisation. This is simply achieved by adding a constant angle to the current odometry such that the robot knows where its roughly facing.

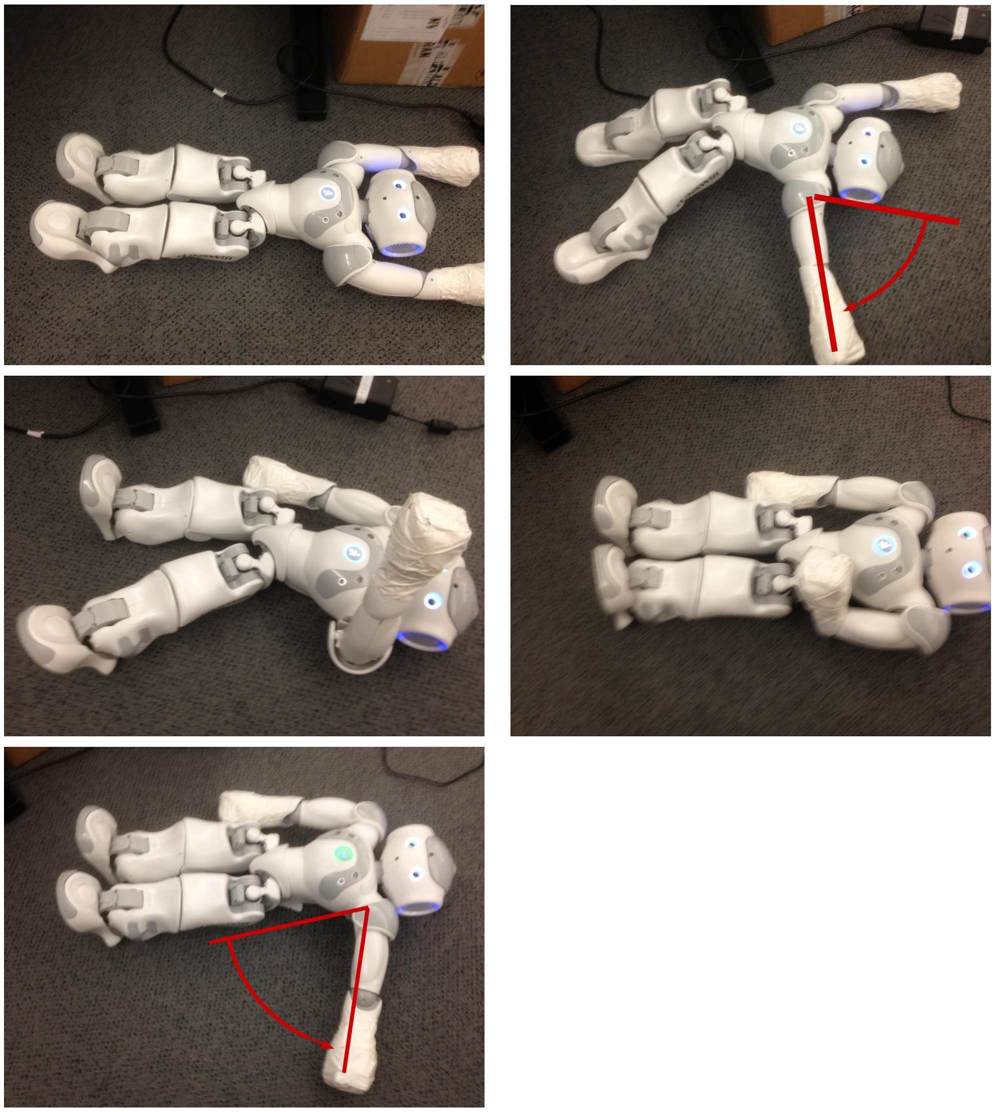

Figure 5.5: The arm flick sequence is ordered from left to right, top to bottom. It is able to reach a large area around the torso of the robot.

5.8 Head action

Each cycle of the goalie state machine start with the function *getHeadAction* for deciding whether to look around and localise or to find and track the ball. Given that the goalie is the most localised robot, the ball information is crucial for the team and can assist the team in localising, so tracking the ball was conditioned to be more favoured than localising. The behaviour that controls the head is external.

getHeadAction:

1. Three quick termination conditions are first checked. Localising is the first priority if the goalie is lost, since finding the ball when lost wont benefit the goalie. Second, the goalie needs to find and track the ball if ball is not in sight. Lastly, keep track of the ball if less than 2 out of 4 teammates including the goalie can see the ball.

```
if $isLost()$ then
  return $localise$
else if $\neg isMyBallSeen$ then
  return $trackBall$
else if $ballsVisibleToTeam < 2$ then
  return $trackBall$
end if
```

2. Enemy distances to the ball are checked and if any enemy robot is within 900mm of the ball, then keep track of the ball as the enemy can get close to the ball any moment.

```
for all $enemy \in enemyRobots$ do
  if $enemy.distToBall < 900$ then
    return $trackBall$
  end if
end for
```

3. Lastly, a short interval is provided for localising. The behaviour module runs in 30 fps and the localising interval is 60 out of every 600 cycles. This means the robot spends 2 seconds localising out of every 20 seconds. In every case above, the localiseMeter is reset to 0 and the interval starts at the 120th cycle so that there is a 4 seconds delay to switch into localising mode to ensure the ball is already kept tracked of. Further more, the localising interval is twice shorter if the ball is on our teams side in order to safely keep track of the ball. Note that the ball information is kept for 30 cycles in the ball filter such that the goalie does not lose track of the ball position when localising unless the ball was moved.

```
$LOCALISE\_INTERVAL = 60$
$localiseMeter += 1$
if $isBallOnOurSide$ then
  $LOCALISE\_INTERVAL = 30$
end if
if $localiseMeter > 600$ then
  $localiseMeter = 0$
end if
if $localiseMeter > 120 \wedge localiseMeter \leq 120 + LOCALISE\_INTERVAL$ then
  return $localise$
else
  return $trackBall$
end if
```

5.9 Transition

State transitions occur at every cycle of the goalie state machine. Every state uses the same transition function,*sharedTransition*, to determine the next state, with exceptions for some states as seen later.

5.9.1 sharedTransition

1. The most prioritised state is of course, Dive. Whatever the case, we want to save the ball even if we are lost.

```
if $ballTimeToPassGoalie() < DIVE\_THRESHOLD$ then
  return $DIve$
end if
```

2. The robot must be localised to perform other tasks, and this makes LostScan the next priority state.

```
if $isLost()$ then
  return LostScan
end if
```

3. The two conditions for entering Intercept state are checked. *shouldClear* and *shouldBeAggressive* condtions are already discussed in the Intercept section.

```
if $shouldClear() \lor shouldBeAggressive()$ then
  return $Intercept$
end if
```

4. Lastly, the CoverGoal state is entered if there are at least two teammates contributing to the team ball or if the goalie can see the ball. This is to ensure reliability of the ball information. The ReturnOrigin state is entered if conditions dont satisfy for other states.

```
if $ballsVisibleToTeam \geq 2 \lor isMyBallSeen$ then
  return $CoverGoal$
else
  return $ReturnOrigin$
end if
```

5.9.2 Anticipate

Transition to the Anticipate state only happens within the ReturnOrigin or CoverGoal state. If the sharedTransition function returns the same state as the current and as long as the distance and heading to the destination is close enough, the Anticipate state is transitioned to.

ReturnOrigin:

```
$nextState = sharedTransition()$
if $nextState == ReturnOrigin$ then
  if $myDistanceToGoal < 150 \land headingDiff < 10 degrees$ then
    $nextState = Anticipate$
  end if
end if
```

CoverGoal:

```
$nextState = sharedTransition()$
if $nextState == CoverGoal$ then
  if $myDistanceToCoverPoint < 150 \land headingDiff < 10 degrees$ then
    $nextState = Anticipate$
  end if
end if
```

As Anticipate is the only state that uses a squatting motion, a larger distance and heading threshold is required for conditions that return to the ReturnOrigin or CoverGoal state for hysteresis. This is to prevent the goalie fluctuate between the squatting and standing due to localisation noise. The other case to consider is that if the ball rolled slow enough towards the goal, the Intercept state would trigger before the Dive state and the robot will stand up walk instead of squatting for a dive. This is handled by checking time returned by *ballTimeToPassGoalie* function and the Intercept state will not be entered if the time to pass goalie is less than 4 seconds.

```
$nextState = sharedTransition()$
**if** $nextState == CoverGoal$ **then**
  **if** $myDistanceToCoverPoint < 450 \wedge headingDiff < 20 degrees$ **then**
    $nextState = Anticipate$
  **end if**
**else if** $nextState == ReturnOrigin$ **then**
  **if** $myDistanceToGoal < 250 \wedge headingDiff < 20$ **then**
    $nextState = Anticipate$
  **end if**
**else if** $nextState == Intercept$ **then**
  **if** $ballTimeToPassGoalie() < 4$ **then**
    $nextState = Anticipate$
  **end if**
**end if**
```

5.9.3 Intercept

The only other state that has extra conditions for transitions is Intercept. First, any extra transitions are skipped if the *nextState* returned from the function *sharedTransition* is a top priority state, Dive or LostScan. For all other cases, the goalie is conditioned to be committed in pursuing the ball if no drastic changes occurred, since giving up half way would most likely waste precious defending time. This is achieved by passing the *transitionOut* flag to the *shouldClear* and *shouldBeAggressive* functions, and it adds an extra buffer to the *ballNearGoalBox* and *ballInAggroZone* areas as well as loosens the distance conditions for goalie to pursue the ball. This way, the goalie should only transition to other states if its distance to the ball has increased significantly.

```
$nextState = sharedTransition()$
**if** $nextState == Dive \vee nextState == LostScan$ **then**
  **return** $nextState$
**else if** $(shouldClear(transitionOut) \wedge isMyBallSeen) \vee shouldBeAggressive(transitionOut)$
**then**
  **return** $Intercept$
**else**
  **return** $nextState$
**end if**
```

5.10 Result

We first analyse the individual matches at the Mexico 2012 Robocup competition. Given each observable situation presented to the goalie, we look the goalies response in terms of the states that it transitioned to, if it was the correct response that an ideal goalie would make, and if the enemy scored from this situation. Matches against teams RoboCanes and Dutch Nao were not accounted for since their team rarely got the ball on our side of the field. We then evaluate the goalies overall behaviour performance and the dives effectiveness.

Austrian-Kangaroos:

Situation	**Goalie response**	**Correct?**	**Scored?**
Ball stopped on our side	CoverGoal→Anticipate	Yes	No
Ball near goal box, no one near ball	Intercept, cleared away	Yes	No
Ball shot at left side of goal	DiveLeft, arm flick too early	No	Yes

We did really well against Austrian-Kangaroos and only allowed the ball to be in our side of the field 3 times. They were able to score the 3rd time. The dive was triggered on time, but unfortunately the arm of the robot lifted for the arm flick sequence and the ball was able to go right through. The problem was elevated right after the match by delaying the timing of the arm flick sequence such that the arms stay stretched out for a longer while.

B-Human:

Situation	**Goalie response**	**Correct?**	**Scored?**
Ball stopped in ballInAggroZone, robots near ball	CoverGoal→Anticipate	Yes	No
Ball shot right of goal, ball shot a second time after save	DiveRight, arm flick ball back to enemy	Yes	Yes
Robots blocked goalies ball vision	ReturnOrigin, head search for ball	Yes	No
Ball stopped near goal box, enemy near ball	CoverGoal→Anticipate	Yes	No
Ball shot at goalie, ball shot a second time after save	DiveCentre twice, one false positive	No	Yes
Ball shot from other side of field	Head localise scan	No	Yes
Ball stopped in goal box	Intercept, cleared away	Yes	No
Ball shot from other side of field, missed goal	CoverGoal→Anticipate	Yes	No
Ball shot from half line to right side of goal	DiveRight	Yes	No
Ball shot at free goal	Goal net caught goalie after previous dive, made localisation way off	No	Yes

We had a good match against team B-Human. The CoverGoal positions were all good in this match, and the goalie did not unnecessarily enter the Intercept state. The time that the goalie entered the Intercept state was appropriate as it cleared the ball away before the enemy got near.

The first goal we didnt save was unfortunate as the arm flick sequence after dive flicked the ball right back to the attacker and they were able to shot again.

DiveCentre was triggered twice during the second goal, only one of which was able to block the ball and the other was a false positive. B-Human was able to shoot a second time to score.

The third goal that the goalie failed to save was shot from the other side of the field. The goalie could not detect the enemy robot near the ball as it was too far away, and the goalie looked away for a localisation scan thinking it was safe and failed to see the ball coming. The last goal that the goalie did not save was right after a previous dive that saved the goal. Unfortunately the goalie was caught in the net that the localisation heading flip around. The goalie walked off the field as a result and left the goal free.

Despite the goals that the goalie failed to save. All goalie responses were appropriate. One reason for the DiveCentre false positive to occur might have been because of the transition between standing and squatting created a spike in the ball filter and the ball was thought to be moving. The ball filter was tweaked to be more accurate during the competition to accustom to the Mexican carpet condition. This might have affected the dive trigger sensitive too. Also considering that the teams are not capable of scoring from the opposite side of the field, a more sophisticated method is required to detect a safe opportunity for a localisation scan. One method is to use team ball position difference to detect if the ball was kicked while the goalie looks away.

UPenn:

Situation	**Goalie response**	**Correct?**	**Scored?**
Ball stopped in $ballInAggroZone$	Position was off at first, but got back to CoverGoal position	Yes	No
Ball shot at centre of goal, but slowed down and curved inside the goal box	DiveCentre, then switched to Intercept and cleared ball	Yes	No
Ball stopped on our goal line	Couldnt see ball, ReturnOrigin and avoided own goal	Yes	No
Ball still on our goal line	Remained in Anticipate	No	No

Team UPenn was not able to score against rUNSWift. There were no behavioural problems but the main issue in this game was that the goalie could not see the ball over his shoulder pads and the ball remained on the goal line. This couldnt be helped since the team ball information was not available to the goalie due to wireless issues.

Austin-Villa:

Situation	Goalie response	Correct?	Scored?
Ball stopped near goal box, enemy not in line of sight to goalie	Intercept entered, backed off after ball got further away	Yes	No
Ball shot from previous position while goalie still on edge of goal box	DiveLeft, arm flicked ball away	Yes	No
Ball shot again and stopped near goal line	ReturnOrigin from previous dive→Intercept and cleared→ReturnOrigin	Yes	No
Ball was shot from half line and missed goal	Remained in Anticipate	Yes	No
Ball stopped near goal box, teammate near	Remained in Anticipate	Yes	No
Ball stopped in goal box	Intercept and cleared	Yes	No
Ball stopped in goal box	Intercept but wasnt fast enough to clear ball, got stuck next to goal post	No	Yes
Ball stopped near goal box, enemy not near yet	Intercept but slipped on tape marking and got beaten to the ball	No	Yes
Ball stopped in *ballInAggroZone*, teammate near	CoverGoal→Anticipate	Yes	No
Ball stopped near goal box	Walked off field from back of goal	No	Yes
Ball shot in free goal	No goalie	No	Yes
Ball stopped in goal box	Intercept and cleared	Yes	No
Ball stopped near goal box	Intercept and cleared	Yes	No
Ball stopped in *ballInAggroZone*, teammates near	CoverGoal→Anticipate	Yes	No
Ball stopped in *ballInAggroZone*, enemy near	Intercept, but backed off after seeing the enemy	Yes	No
Ball stopped in goal box	Intercept, missed and fell to ground	No	Yes
Ball stopped near goal box	Walked off field from back of goal	No	Yes
Ball stopped in *ballInAggroZone*, teammate near	CoverGoal→Anticipate	Yes	No
Ball stopped in goal box	DiveCentre false positive and fell to ground	No	Yes
Ball stopped in *ballInAggroZone*, robots near	CoverGoal→Anticipate	Yes	No

The semi-finals match was against team Austin-Villa. A lot of back and forth actions were in this match and the ball was often on our side of the field. The goalie did a brilliant job in the first half of the match, despite the 2 goals not saved because of getting stuck next to the goal post and slipping on tape marking.

However, the goalie allowed 5 goals to be scored against us in the second half. 2 of the goals scored were because the goalie walked off the field from the back of the goal. The only possible explanation for this was that the localisation filter mistakenly saw the goal box line as a half circle, which matched its self against the centre circle and therefore pushing its position forward. This happened only in the second half of the match suggests that the vision system may not be calibrated for the particular lighting in this half of the field. The goalie completely switching sides had also occurred, although it did not cost us any goals. The free goal was due to goalie being sent off the field as it kept falling due to wear and tear of the robot. The other 2 goals were also due to the wear and tear of the goalie, making it fall from kicking the ball in the Intercept state and during the DiveCentre motion. The DiveCentre false positive that occurred was for the same reason as in the B-Human match.

HTWK:

Situation	Goalie response	Correct?	Scored?
Ball stopped in $ballInAggroZone$, teammate near	CoverGoal→Anticipate	Yes	No
Ball rolled slowly to goal line	Intercept→DiveCentre and fell	Yes	No
Ball on the goal line, behind goalies vision	ReturnOrigin avoided own goal	Yes	No
Ball stopped in $ballInAggroZone$, teammates near	Intercept and did not clear	No	Yes

The 3rd place play off was against team HTWK and they were able to score once against us. The goalie mysteriously entered the Intercept state to clear the ball even though there were clearly teammates already at the ball, which had resulted in a goal against us. The reason for this was not known.

The other issue was that the goalie stood up and caused the subsequent DiveCentre to fall from standing when the ball rolled slowly into the goal box. Entering the Intercept state when the ball got near caused this. Although there was a check for not entering the Intercept state when the ball was detected to be rolling towards the goalie by checking the ballTimeToPassGoalie function, but it seemed that the time threshold was not sensitive enough to trigger the detection.

The goalie also fluctuated between squatting and standing in this match. There might have been a change in the localisation noise, which caused the margins implemented for hysteresis ineffective.

Lastly, the goalie could not clear the ball that landed on the goal line for the same reason as the UPenn match.

5.10.1 Overall behaviour

11 goals were scored against the goalie. Out of 41 observable situations that were presented to the goalie by the enemy teams, the goalie responded to 29 situations with ideal responses. Out of 12 less than ideal responses, 1 response was due to lack of team ball, 4 responses could be categorised as motion shortcomings, and 3 responses were due to localisation errors. This leaves 4 less than ideal behavioural responses. These were: two DiveCentre false positives, unnecessarily entering Intercept state and untimely head localisation scan. In other words, the behavioural response was over 90% correct.

5.10.2 Dive effectiveness

The dive was able to save 4/5 goals. All necessary dives were triggered and there were two false positive DiveCentre detections due to transition between squatting and standing motions. Diving for balls missing the goal was not attempted. Only the first shot at the goal were included in these numbers.

DiveCentre was quite successful in that the width had been increased from 34 centimetres to 48 centimetres, and it was able to save 2 of the goals.

DiveLeft and DiveRight were fast enough and never failed to trigger on time. Unfortunately, the arm flick sequence allowed the ball to pass through in one case.

As for the actual arm flick sequence, it was effective in that it was able to clear the ball twice. However, the ball was flicked back to the enemy striker in one case.

5.11 Conclusion

The goalie showed great potential and many improvements were made to the rUNSWifts 2011 goalie. It made use of the improved localisation to devise an optimal position to defend the goal according to the ball location, and improved diving motions were developed that saved many goals. The ability to defend the goal was also enhanced by clearing the ball before the enemy robot, and made use the new robot detection filter to aid its decision making.

Chapter 6

Future work

6.1 IMU

Two future work fields are possible for IMU filtering. One is to develop a more sophisticated system model for the Kalman filter to improve its accuracy regardless of the body structure. The other is to find a solution to the inaccuracy in the complementary filter while the robot is turning.

6.2 Motion

The top priority of future work in motion is to improve the stability of the walk by developing a more sophisticated walk method. The major drawback this year was that if the walk was more stable, it would have prevented the robot from falling due to collisions that happened too often because of wireless issues. This in turn would reduce the wear and tear of the robots such that they would perform better in later more critical matches.

Work was started this year on two different approaches. One was to reinforcement learn a coronal stabilisation strategy using the angle and angular velocity of the torso as system states, and make policies to adjust the hip and ankle bends to control the coronal rock. The coronal rock will adjust to different periods of the walk, which will result in a smoother walk motion and improved stability. This meant that it would also improve the agility of the walk as rapid changes in speed and direction could be handled. The other approach was to use the Linear Inverted Pendulum model[9] to model the legs of the robot as two switching pendulums and predict the movement of the centre of mass, located at the torso, using hyperbolic equations. An open looped solution was developed, but its speed was not competitive. There was not enough time to complete a closed loop solution.

The other future work field is to start making use of the joint sensors of the Nao. The Nao reports the current of every joint. The current of a joint would increase if a force were directed on it. This is useful for 3 things. First, the joint sensors can provide feedback against a sudden push to develop push resistance in the walk. Second, calibration of the joints can be achieved by tuning the joint angles until minimum current is drawn. We noticed some robots were very good in their walk motion, where as some robots were very poor and could not walk straight. It was theorised that the joint calibrations were poor on these robots, and the joint sensors should allow us to calibrate the joints ourselves. Lastly, joint sensors is a much better alternative to visual odometry for detecting if the robot is getting caught on its arm, and loosen arm stiffness to assist avoidance accordingly.

6.3 Goalie

The goalie this year integrated almost every needed future improvement discussed in 2011[14]. The only future work that was left out was actions such as high kicks and throw-ins. I believe that the current hardware still limits us from developing such high level motion with acceptable reliability and speed.

A possible improvement is to integrate the dives in the motion engine instead of hardcoding joint sequences in pos files. It means that it would be possible to develop an omni-directional dive where the dive can reach any direction. This would be advantageous since the goalie would no longer have to keep its heading toward the ball and limit itself to side and centre dives, and the goalie would be able to take a more direct route to the target location to cover the goal.

Chapter 7

Conclusion

This thesis discussed the Kalman filter and complementary filter applications to the problem of filtering IMU, as well as the method of obtaining ground truth and IMU calibration. Results showed accurate estimations from both the Kalman filter and complementary filter.

Various motion changes were introduced in 2012 to accustom to the new robot hardware. The walk was able to reach a higher speed that was competitive amongst the best teams. The agility and the stability of the walk were lacking, and should be worked on for future work. Motion changes also include the method of detecting collision near the robots arm and loosening arm stiffness that successfully assisted robot avoidance.

Lastly, goalie was redeveloped in 2012 to integrate with improvements in localisation and the addition of robot detection filter. The dives were also improved, which saved many goals during the competition. The performance of the goalie was greatly improved as a result.

Bibliography

[1] Bernhard Hengst, Brock White, and Manuel Lange. *Learning to control a biped with feet.* Humanoids 2011 - Accepted, 2011

[2] Aldeberan. *Aldeberan IMU documentation.* http://www.aldebaran-robotics.com/documentation/nao/hardware/inertial.html?highlight=inertialsensor

[3] Adrian Ratter, Bernhard Hengst, Brad Hall, Brock White, Benjamine Vance, David Claridge, Hung Nguyen, Jayen Ashar, Stuart Robinson, and Yanjin Zhu. *runswift team report/* 2010.

[4] Brock White. *Humanoid Omni-directional Locomotion.* 2011

[5] Belinda Teh. *Dynamic Omnidirectional Kicks on Humanoid Robots.* 2012

[6] Youssef Hunter. *Humanoid Robot Localisation for the RoboCup Standard Platform League.* 2012

[7] Peter Anderson. *New Methods for Improving Perception in RoboCup SPL.* 2012

[8] Ritwik Roy. *Low Level Behaviours For Soccer-Playing Robots.* 2012

[9] Colin Graf and Thomas Roefer. *A closed-loop 3d-lipm gait for the robocup standard platform league humanoid.* 2010

[10] HTWK Team. *HTWK's Youtube channel.* http://www.youtube.com/user/naoteamhtwk.

[11] Aldeberan. *Aldeberan Nao v4 component mass documentation.* http://www.aldebaran-robotics.com/documentation/family/robots/masses_robot.html

[12] Aldeberan *Aldeberan Nao v4 motor torque documentation.* http://www.aldebaran-robotics.com/documentation/family/robots/motors_robot.html

[13] RoboEireann and B-Human. *RoboEireann vs B-Human in 2012 German open.* http://www.youtube.com/watch?v=Ox9Z8XyLgbY

[14] Belinda Teh. *Ball Modelling and its Application in Robot Goalie Behaviours.* 2011