A Description of the rUNSWift 2003 Legged Robot Soccer Team

Jin Chen[1], Eric Chung[1], Ross Edwards[1], Nathan Wong[1], Eileen Mak[1], Raymond Sheh[1], Min Sub Kim[1], Alex Tang[1], Nicodemus Sutanto[1], Bernhard Hengst[1,2], Claude Sammut[1], and William Uther[2]

[1] School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW, 2052, Australia
[2] National ICT Australia
{hengst,claude,willu}@cse.unsw.edu.au

Abstract. This paper describes the 2003 world champion legged robot soccer team, rUNSWift. The 2003 rUNSWift team is enhanced in a number of ways over previous teams; both long and short range collaboration between team members was carefully crafted, a new method of ball localization was used when the ball was close, new tools were developed to assist the construction of object filters in the vision code, self-localization was simplified, opponent localization was significantly improved with distributed data fusion, ball tracking was improved to account for the ball's velocity, edges and lines on the field were used to assist in self-localization, and automatic gait optimization was used to improve the forward walking speed of the robots. The result of these changes is a team that is significantly improved over previous teams.

1 Introduction

This paper describes the rUNSWift 2003 legged robot soccer team. This team was based upon the rUNSWift 2002 team [1], with significant extensions. In particular, the architecture of the code was changed from a single monolithic object in the operating system into a number of interacting modules. At the same time, many of those modules were enhanced. As a result of these changes, rUNSWift was the world champion team in the legged league of the RoboCup 2003 international robotic soccer competition.

Due to space limitations, not all the details of the technologies developed are described in this paper. Readers are referred to the undergraduate theses of the student members of the team, which will be released as UNSW technical reports in late 2003, for complete details (also see [2]). In particular, rUNSWift performed a major restructuring the source code this year, and developed a number of technologies for completing the three technical challenges in the competition that are described in the relevant theses.

This paper describes a number of distinct changes to the rUNSWift team. In Sections 2 and 3 we describe the team coordination used. Vision changes are described in Section 4 before moving to object tracking and data fusion in Section 5. Finally, Section 6 gives an overview of the automatic gait optimisation.

2 Behaviours

The consolidation of behaviours into a single hierarchy accompanied a major reworking of the behaviour of the rUNSWift team. While many aspects of the rUNSWift design philosophy for individual players remained the same, much more emphasis was placed on team co-operation. It is important to note that we do *not* equate team cooperation solely with a passing game. The close quarter cooperation among the robots was integral to behaviour of the team and is described in detail in the next section (Section 3). However, first we will describe the large scale behaviour of the robots.

rUNSWift's strategy relating to forwards was centred on aggressive play. Two design principles underlie the rUNSWift strategy: first, at least one forward must reach the ball quickly, and second, a forward should only attempt fine control of the ball if it has enough of a lead on the opposition to carry out that fine control, otherwise that player should concentrate on moving the ball upfield fast without worrying overly about the exact direction of movement. This is preferable to slowing down to reliably grab the ball for a front kick, because moving the ball upfield behind the opposing team stochastically increases rUNSWift's chances of scoring.

This aggressive chasing of the ball by the forwards is tempered by a mechanism which facilitates collaboration and minimises interference among the forwards. In this "dynamic role determination", each of the three forwards assumes a role of attacker, supporter or striker depending on information that is shared by the robots, via wireless ethernet as well as visually, regarding each robot's location, whether they see the ball, their proximity to the ball and their current role. An attacker is a robot that is currently playing the ball. A supporter is a robot that is supplying close-in support to an attacking robot. The striker is the robot providing long distance support, sometimes as a wing player on the far side of the field, and sometimes as a back.

When the Sony AIBO robots are in close proximity, they have a tendency for their legs to become entangled. To try and avoid this, an automatic "backoff" mechanism (see Section 3) is triggered when a forward's vision system detects that a teammate is in very close proximity. It is this backoff mechanism that controls the roles of the two robots nearest the ball.

The robot furthest from the ball plays the striker role, and moves swiftly towards a point on the field, determined by whether the ball is in the offensive half and whether it is in the left or right half of the field. If the ball is in its own (defensive) half, the point lies back from the ball near the robot's own goal; if the ball is in the target half, the point lies in the offensive quadrant opposite to where the ball is located.

rUNSWift's play strategy does not adopt the popular strategy of permanently having a forward remaining back in its own half defensively but instead favours the principle that defence is best achieved through offence and that it would be undesirable to lose the resources of an active third forward. This strategy leaves rUNSWift exposed to an extremely problematic situation that arises when all three forwards are on the wrong side of the ball and opponent robots have possession of the ball. In the Robocup 2003 final against UPenn, rUNSWift found itself in such a situation several times because the UPenn forwards had a kick that successfully flicked the ball past rUNSWift's forwards.

rUNSWift developed a defensive behaviour to counter this problem: the forward with the greatest offset across the field from the ball quickly rushes to position itself behind the ball, while the two other forwards rush directly towards the ball. This defensive behaviour is dynamically triggered when all three forwards are on the wrong side of the ball. The behaviour is effective because when this situation occurs, an opponent robot is likely to be either closer to the ball than the forwards or is between the forwards and the ball. If all three forwards rush directly towards the ball, they will either become blocked by opponent robots or become entangled with one another. In order to ensure effective defence, one of the forwards needs to quickly get back to its own half, between the ball and its own goal to block a possible goal. As it is important that this robot does not become entangled or taken off the field, it circles around other robots and travels along the boundary of the goal box if it finds that its path would otherwise travel through it.

3 Local Robot Interactions

In addition to possessing a global form of cooperation between the robots, where a third robot is placed apart from the main two forwards, the close-in cooperation and local interactions are an important feature of the UNSW/NICTA strategy.

The two robots closest to the ball do a lot of shifting and reconfiguring of their positions relative to the ball and one another as they chase the ball. Collectively, this allows them to better maintain possession and control of the ball, as the robots support one another and take turns attacking the ball. Typically one robot will be attacking the ball, and the other will be closely supporting, getting ready to take over the attack. This is mostly used in overcoming the opponents' defensive moves. When the original attacking robot runs into the opponents' resistance, the supporting robot will then take over the attack.

As delays of 500ms were common in the wireless communication system, this close-in cooperation was heavily reliant upon vision to verify the last known position of a robot's teammates. If a robot cannot see any of its teammates, it will attack the ball. However, when a robot can see a teammate, a decision is made to determine whether it should attack the ball, or support the attack. This decision is made based on three aspects: the robots position relative to the field, the ball and its teammates. The overall effect is subtle but exceptionally effective.

This strategy uses the concept of "Desired Kick Direction" (DKD) from previous UNSW teams. Depending on where the ball relative to the field, the

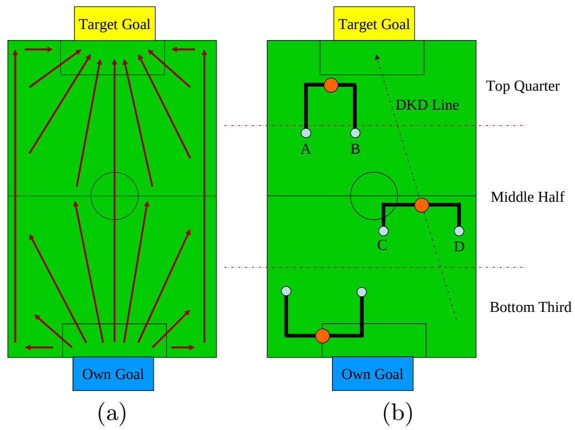

Fig. 1. (a) Desired Kick Directions (DKD). (b) Various "L" support positions.

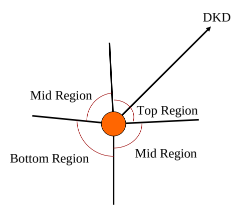

Fig. 2. Star, dividing the area around the ball into regions

robots will want to move the ball in a different direction (See Fig. 1a). This desired direction will help in determining which robot is in the better position relative to the ball, and thus which robot should attack the ball. Generally, the DKD is away from the robots own goal, and towards the target goal.

After the DKD has been determined, we draw a star, centred on the ball and pointing in the direction of the DKD (see Fig. 2). This splits the area around the ball into regions, and it is the teammate's positions relative to each other on this star that determines which is to attack and which is to support. The details of this algorithm are shown in Table 1.

The final major component in our local cooperation strategy is the close-in support/backoff position. If the robot is not attacking the ball, it will position itself to prevent the ball from travelling towards the defending goal, and to get ready to take over the attack towards the target goal. This position is based on where you are on the field, and relative to the position of the ball (See Fig. 1b).

If the ball is in the top quarter of the field, the supporting robot will position itself in an "L" shape to the side and behind the ball, such as that depicted by points A and B. The robot will go to point A or B depending on which side of the DKD line it is on. If it is to the left of the line, it will go to the left most position, *i.e.* A, otherwise it will go to point B. The reasoning behind this is that this will keep the dog from obstructing goals by travelling between the ball

- If a robot is in the bottom region of the star, and the other robot is not, then that robot will attack the ball and the other robot will go to the supporting position.
- If both robots are in the top region, then the one who is closer will attack the ball, and the other will get out of the way as the closer robot will probably perform a 180 degree kick. After getting out of the way, that robot will move into the support position.
- If both robots are in the bottom region, then the robot to attack the ball is determined based on the following:
  - Which robot can actually see the ball. (as opposed to wireless knowledge).
  - Which robot is closer to the ball.
  - Which robot is closer to the DKD.
  - Which robot has the lower player number.
- In all other cases both robots will closely circle around the ball towards the DKD until one stops seeing its teammate, and thus is free to attack the ball, or one enters the bottom/attacking region. Circling around closely allows the robots to maintain a defensive position relative to the ball.

Table 1. The algorithm for deciding the role of a robot.

and the target goal. Probabilistically, it will also have the dog travelling to the closer support position, and it will have the dog closer to the centre of the field, and between the ball and its own goal. The support position is back more than it is to the side. This is so that the supporting robot will more often be behind any loose balls, ready to take over the attack.

If the ball is in the middle half of the field, the supporting position will be in a wider "L" shape such as that depicted by points C and D relative to the ball. Again the robot will choose between points C and D depending on which side of the DKD line it is on. The "L" is wider here so that the supporting robot is wider, ready to intercept sideways passes.

In the bottom quarter of the field, *i.e.* the defensive quarter, the support position is in an "L" shape in front of the ball. Instead of positioning itself defensively between the ball and the defending goal, it is away and in front of the ball. This is so that the support robot stays out of the attacking robots way, minimizing the chance of getting leg locked with that robot and inhibiting the defence. It is also then able to capitalize on any opportunities where the ball becomes momentarily loose from the opponent attacker.

An emergent property of the up field "L" shape in the bottom third of the field is that the two robots tend to move in a circular path around the bottom corners. Typically, the attacking robot will be in the in the bottom corner trying to stop an opponent from progressing any further with the ball. Meanwhile, the supporting robot will wait in the supporting position. As the first robot loses control of the defence and moves out of the bottom region of the "star shape", the supporting robot will take over the attack and the first robot will then assume the support position. In this way, the robots exchange positions, moving in a circular fashion until the ball is out of the corner.

A special case that needs to be discussed is what happens to the supporting position when the ball is near the side edges. In this case, the supporting position will be against the side edge and behind the ball. Often the attacking robot will get into a scrum on the side wall with one of the opponent robots. In the case where the opponent robot wins and gets the ball past the attacking robot, the supporting robot is there as a second line of defence to stop the ball from advancing to far towards the defensive goal, and as a second chance to clear the ball.

The behaviour of local cooperation in the UNSW/NICTA strategy is a result of having Desired Kick Directions, Star shaped region dividers, L shaped support positions and the ability to visually see teammates and opponents. It is the combination of local cooperation skills and global cooperation skills that results in an effective team strategy. Although to the unfamiliar eye, the behaviour may not be obvious, the behaviour, subtle as it is, is quite deliberate, planned and effective.

4 Vision Overview

The vision system used in 2003 is very similar to that used previously. Information from the CMOS camera is processed by first identifying the colours, then forming coloured blobs, combining the coloured blobs to form objects and finally calculating the location of objects relative to the robot.

rUNSWift 2003 used replaced the nearest neighbour algorithm from 2002, with a table generated using the C4.5 decision tree inducer [3] as it was found to be less susceptible to lighting changes. Neither method was found to be entirely satisfactory, so it was also necessary to fine-tune the result with a manual classifier.

Ball localisation close to the robot gained more importance this year due to the heavy emphasis on close-in interactions. As the close-in interactions require accurate calculation of the ball's position when a large portion of the ball resides outside of the camera frame, a new method was introduced that could determine the centre of the ball by applying circle geometry and edge detection. Once the centre is accurately determined, the ball's location is calculated by projecting the ball centre onto a plane parallel with the ground plane. For far away balls, distance is determined by the width of the bounding box.

While detection of beacons and other objects is largely unchanged from previous years, the filtering of false positive identifications was significantly changed. These filters have traditionally been hand crafted after watching the behaviour of the robots. This year a more structured approach was taken. The vision module was ported for offline execution, using as input logs taken from the robots. This allowed the object filters to be tested with large sets of sample images. This system shows both over and under-filtering quickly, as well as allowing the identification of which rules are causing over-filtering. Originally intended for development of object filters checks, this offline port has introduced improvements to development and debugging time in other areas, including both the black & white ball and edge detection challenge.

5 Object Tracking

The agent's internal model of the world allows it to track it's own location on the field, as well as the location of the ball, it's teammates and opponents. This section of the rUNSWift team was entirely re-written this year using extended Kalman filters. The location of the robot itself was tracked with a single 3D Kalman filter over X, Y, and θ. The location of the ball was tracked, along with its velocity, in a pair of 2D Kalman filters. Finally, the opponents were tracked using a set of four filters on each robot. These final filters shared information between allied robots using distributed data fusion.

Tracking opponent robots is inherently difficult since the visual detection of robots is relatively poor, and the observation frequency is low. Our aim is to track the approximate position of the opposition team, hence we are not as interested in fine accuracy as we would be with our own position, or the ball. For these reasons we developed a distributed sensor network for opponent tracking. This consists of multiple sensor nodes (our team's robots), which are connected with a temporal constraint. In our case the temporal constraint is that our robots do not communicate every frame (the cycle period of the sensor), but rather wait for a period of several frames before sending any information. The processing of the network is also done in a distributed fashion, with each node keeping it's own model of the opponents' positions. The model we developed is based on Information Form Kalman Filtering.

5.1 Information Form Kalman Filter (IFKF) Algorithm

For a full introduction into Information Form Kalman Filtering see [4]. This work extends that work through adaptation to non-linear observations. The information form Kalman filter algorithm is a recasting of the basic Kalman filtering algorithm. The data contained in a Kalman filter, expected value $\underline{x}$ and covariance matrix $\underline{\underline{Q}}$, can be represented in the form of inverse covariance times expected value $\underline{y} = \underline{\underline{Q}}^{-1}\underline{x}$ and inverse covariance $\underline{\underline{Y}} = \underline{\underline{Q}}^{-1}$, known as information form. By doing this, and rearranging the update equations accordingly, we obtain a system mathematically identical to a Kalman filter model where updates are of the form

$$\underline{y}(k|k) = \underline{y}(k|k-1) + \underline{i}(k)$$

$$\underline{\underline{Y}}(k|k) = \underline{\underline{Y}}(k|k-1) + \underline{\underline{I}}(k)$$

where $\underline{i}(k)$ and $\underline{\underline{I}}(k)$ are the updates due to observations at time k. This provides a distinct advantage over regular Kalman filtering, in that estimations are formed from linear combinations of observation information. Hence if a sensor wishes to send distributed update data for the last α timesteps, then it sends the sum of the information gathered over those timesteps to the other sensors: $\sum_{j=k-\alpha+1}^{k} \underline{i}(j)$ and $\sum_{j=k-\alpha+1}^{k} \underline{\underline{I}}(j)$. Therefore, over time, all sensors should hold the same data (assuming N sensors)

$$\underline{y}(k) \;=\; \underline{y}(k-\alpha) \;+\; \sum_{n=1}^{N}(\sum_{j=k-\alpha+1}^{k} \underline{i}_n(j))$$

and similar for $\underline{\underline{I}}$. This solves a key problem in decentralised data fusion, all nodes hold equivalent data in the long term. We also mentioned that the information form of the Kalman filter is mathematically equivalent to the regular Kalman filter, therefore the estimation held by all the nodes is equivalent to the estimation made if a single node was to receive all observations by all nodes combined.

5.2 Observation Matching

The algorithm described above solves the problem of having multiple sensor nodes with our temporal constraint, however the task of tracking opponent positions still holds other issues. Tracking a team of four opponents requires the estimation of four positional vectors, this indicates that four separate information Kalman filter models need to be used, or a similar approach applied. This leads to a very elementary problem, given an observation, and that all opponents look identical, which model should be updated? There are two simple solutions to this problem, both of which were tried.

The first solution is arguably the logical one, choose the most probable model. This approach can be realised through the equation

$$P(O|G_n) \;=\; \frac{1}{2\Pi\sqrt{|\underline{\underline{K}}|}}e^{-\frac{\underline{x}^T\underline{\underline{K}}\underline{x}}{2}}$$

where O represents the observation, G_n the nth gaussian probability distribution, $\underline{x}$ the innovation vector and $\underline{\underline{K}}$ the gaussian covariance projected to observation space. By choosing the gaussian with the largest of these probabilities, and updating it's Kalman filter model with the observation, we can match individual observations to individual models. Although this solution is perhaps the mathematically precise one, it suffers from several problems, the most severe of which is that some gaussians reach outer limits, growing very large, and end up not winning any observations, while other gaussians win many observations and hence "bounce" between observations.

A variation of the above is to use the same algorithm, except instead of using the probability of the observation given the gaussian, use the number of standard variations away from the expected value the observation is, this is given by $e^{-\frac{\underline{x}^T\underline{\underline{K}}\underline{x}}{2}}$. By using this value instead gaussians that do not attract observations, grow in variance, and hence have a much greater chance of "winning" an observation. However this method still does not offer adaptation to an opponent's positions quick enough, suffering especially in the kidnapped robot situation.

The second simple solution to the problem of observation matching is to "share" the observation around. We can apply a fraction of an observation to a gaussian in the IFKF by multiplying the variance of the observation by the inverse of the fraction. Hence we can assign a weight for each gaussian (n)

$$w_n = \frac{P(O|G_n)}{\sum_{i=1}^{N} P(O|G_i)}$$

and apply the fraction of the given observation to the IFKF. This solution leads to a set of models that can very quickly adapt to changes in the opposition's positions as gaussians quickly move from areas where there are no observations to places of high observations. However the solution also leads to problems of multiple gaussian distributions becoming extremely similar, at which point they are very unlikely to separate.

After analysing the two simple solutions discussed, we can see that both have their problems. Ideally the solution should have the ability for models to adapt to sudden changes in opposition position's that the second method offers, while maintaining a reasonable spread of gaussian distributions that the first method has. The logical step was to try to create a hybrid method which combined the two, and hopefully inherits both ideal behaviours. By using the calculated weights above, we can create a new weighting system taking both methods into account. The new weight v can be calculated by

$$v_n = (1 - \alpha)w_n + \delta_{nj}\alpha$$

where w_n is the original weight, j is the "winning" gaussian calculated using standard deviations, δ is the Kronecker delta and α is a constant between 0 and 1. In fact if we take α as 1 we get the first solution, and as 0 we get the second. We can see this solution takes a portion of the observation and assigns it to the "winner", then breaks the compliment portion into pieces according to the second solution. This hybrid solution works extremely well, providing a highly adaptive set of distributions while maintaining a spread over all observation areas. It of course still inherits some of the problems that the initial solutions had, and it certainly would not be reasonable where acute accuracy of positions was required, but it succeeds in our initial goals, providing a good approximation of the opponent's positions, representing observations from all teammates.

6 Automatic Gait Optimisation

A major innovation made by the UNSW/NICTA entry this year was automatic gait optimisation (see [2] for more details). This was the first such technique seen in the competition and resulted in improvements in speed and stability over previous hand-crafted gaits. The technique learned fast enough to run on-site at the event during the allocated practice days before the start of competition matches.

This work was built on top of the locomotion module from previous years, originally developed by the year 2000 UNSW team. This was designed on the concept of a rectangular walk locus; this is the trajectory traced out by the robot foot as it walked [5]. Parameters were used to define the position of the four corners of the rectangular locus, as well as the position of the loci relative to the robot body. Inverse kinematics would then translate the loci to joint angles fed to the motors.

A variation of the walk locus used by the 2002 UNSW team was a trapezoidal locus. This change resulted in a significant speed increase of approximately 12% over the rectangular locus, but it was found that it was much less manoeuvrable and not very smooth. For these reasons, both gaits were used, with the strategy determining when it was suitable to use a particular gait.

This work builds on these previous gaits by exploring the space of quadrilateral loci to find an effective gait. As with the 2002 architecture, the new walk was only used for walking forward, with no sideways movement component and less than $15°\mathrm{s}^{-1}$ turn.

6.1 Problem Representation

The problem representation was formulated with the intent of restricting exploration to small local changes to the walk locus. The rationale for this is that small changes both increase search speed and minimise disruption to other tasks, such as localisation, that may be dependent on characteristics of the previous walking system.

The new quadrilateral walk locus is described by four offsets from the original rectangular locus, and the speed of the robot foot around the locus is constant. Using the original locus as a base allows all of the existing speed and turn controls to carry over to the new gait. In order to explore this search space, each of the offsets can be moved in three dimensions – either forward or backward, sideways, and up or down. Since there are four corner points in a quadrilateral locus, this produces a 12-dimensional vector representation. This was then extended by using different loci for front and rear pairs of legs, resulting in a 24-dimensional vector representation.

6.2 Experiment Environment

The experiment environment was largely based on the setup used by Hornby et al. [6, 7]. The robot walks back and forth between two landmarks, and the optimisation method tries to minimise the time required. While the robot walks mostly straight forward to a landmark, small heading corrections are constantly applied. These small corrections help the system learn a walk robust to small amounts of turn.

The time measurement was based on the camera frame rate and execution cycle of the robot. 25 camera frames are processed per second; minimising the number of camera frames processed (or time taken) as the robot walks between the landmarks corresponds to maximising speed.

There were two main sources of stochasticity in this experiment. Specifically, they were:

– Error in distance measurement from sensors – this may cause the robot to stop either too early or too late in front of the target landmark. As a consequence, the distance walked on each trial is not exactly the same.
– Error in positioning – the distance covered could also turn out to be different if the robot ventured slightly off course.

Bouncy walks that shook the robot a lot had the side effect of underestimated distance measurements. Thus, the robot stopped briefly to settle after each trial, and re-measured its distance so as to penalise trials that fell shorter than a predefined threshold.

A single evaluation of a locus was implemented as four separate runs between the landmarks, taking the average of the two median readings as the evaluated time. The trade-off for the resulting longer trial times was that the stochasticity in the problem was significantly reduced.

6.3 Optimisation Method

We used Powell's (direction set) method for multidimensional minimisation [8]. Powell's method minimises along each of a given set of directions in the multidimensional space in turn, and then uses the results to derive new directions, aiming to find directions that minimise the function quickly.

Instead of using the set of unit vectors in the search space as the initial direction set, we chose a different set with the intention of exploring the search space more efficiently. This was a linear transform of the set of unit vectors, i.e. we can reach any point in the space using a linear combination of this set of vectors. For a complete description of the set of vectors used, see [2].

6.4 Results

Within the preparation time constraints at the competition, we were not able to run the method through to complete convergence, but minimised along 22 of the 24 directions. At one point the robot seemed to be stuck in a poor local minima, and so we restarted that particular line minimisation. It did not get caught in that local minima the second time; we believe the minima was caused by noise in the sensor measurements.

At the competition, the gait speed was measured at $27\mathrm{cm\,s}^{-1}$, which was significantly faster than the hand-developed trapezoidal locus on that particular surface. Furthermore, the locus produced a much smoother walk that kept the robots camera steadier, which had helpful effects on other tasks such as localisation.

Following the competition, the minimised walk from the competition was compared with the trapezoidal locus and rectangular locus walks, on a similar carpet surface to that used at RoboCup. 50 trials were timed for each walk over a distance of 1 metre. As expected, the rectangular walk was slowest, with an average speed of $22.69 \pm 0.45\mathrm{cm\,s}^{-1}$. The trapezoidal locus was next, measured at $25.42 \pm 0.50\mathrm{cm\,s}^{-1}$. The walk produced at the competition came out as the fastest, measured at $26.99 \pm 0.50\mathrm{cm\,s}^{-1}$.

7 Conclusion

Significant development occurred in the rUNSWift team this year along a number of lines. Software engineering process improvements as well as code re-structuring allowed the team to progress in a number of different directions.

The overall strategy of the team was re-implemented with a strong focus on cooperation of two forwards working in close proximity. A new method of ball localization was used when the ball was close, and hence partially out of the camera frame of the robot. New tools were developed for building filters for objects returned by the vision code. Tracking of objects was made mainstream using Kalman filters. In addition, opponent localization was significantly improved with distributed data fusion. For the challenges, a system using edges and lines on the field to localise the robot was developed and used to aid localisation during games. Finally, automatic gait optimization was used to improve the forward walking speed of the robots. The result of all these improvements was a high effective, world champion, robot soccer team.

References

1. Wang, D., Wong, J., Tam, T., Leung, B., Kim, M.S., Brooks, J., Chang, A., Huben, N.V.: The UNSW RoboCup 2002 Legged League Team. Undergraduate thesis in computer and software engineering, University of New South Wales (2002)
2. Kim, M.S., Uther, W.: Automatic gait optimisation for quadruped robots. In: Australasian Conference on Robotics and Automation, Brisbane (2003) Under Submission.
3. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1992)
4. Durrant-Whyte, H., Stevens, M.: Data fusion in decentralized sensing networks. In: 4th International Conference on Information Fusion, Montreal, Canada (2001)
5. Hengst, B., Ibbotson, D., Pham, S.B., Sammut, C.: Omnidirectional Locomotion for Quadruped Robots. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: Lecture Notes in Computer Science, RoboCup 2001: Robot Soccer World Cup V, Springer (2002) 368–373
6. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous Evolution of Gaits with the Sony Quadruped Robot. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 1297–1304
7. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Evolving Robust Gaits with AIBO. In: IEEE International Conference on Robotics and Automation. (2000) 3040–3045
8. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press (1992)