The Evolution of a Robot Soccer Team

Claude Sammut and Bernhard Hengst

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

Abstract. This paper traces four years of evolution of the UNSW team in the RoboCup Sony legged robot league. The lessons learned in the creation of a competitive team are instructive for a wide range of applications of robotics. We describe the development of vision and localisation procedures for robot soccer, as well as innovations in locomotion and the design of game play strategies. Since all teams in the competition are required to use identical hardware, the key factor to success in this league is the creativity of the software designers in programming the robots to perform skills that the robots were not originally intended to do and to perform them in a highly dynamic and non-deterministic environment.

Introduction

In 1998, the RoboCup competition introduced the Sony legged robot league as a demonstration and a full competition started in 1999. Some initial reactions where that these robots could not be made to play soccer with any real competence. In 2000, the team from the University of New South Wales (UNSW) proved otherwise.

In the first years of the competition it appeared that the vision system of the robots could not reliably detect objects on the field. Their locomotion was too clumsy to control the ball well and they could not act as a team to play an interesting game. However, it was clear that the designers of the robots believed that they could be programmed to play credibly. Unlike other RoboCup leagues, in which teams can design and build their own hardware, the competition in the legged league is based entirely on the creativity of the programmers to make robots do things that they were never originally intended for. This is an exercise that is interesting for robotics research since it tests the boundaries between what should be done in hardware and what can be done in software.

In 2000, the UNSW team introduced a vision system that was fast and reliable even in the confusion of soccer match; localisation that was robust to frequent occlusions and buffeting from other robots; locomotion that was both agile and stable and behaviours that made the robots play with apparent purpose and with the ability to recover from failures. These developments were the result of many

hours of analysis of the robots' capabilities and the study of many specific game play situations.

Each year, following our initial entry into the competition in 1999, the teams built upon the successful ideas of the previous year and added their own innovations so that the 2000 and 2001 teams became champions of the league with decisive wins. In this paper, we first give a brief history of the technical developments involved in the RoboCup software. We then give details of the vision, localisation and locomotion systems. We also discuss communication between robots, first using sound signals and then using the Ethernet introduced in the 2002 competition. Finally, we describe the behaviours required to make a legged robot play a good game of soccer.

The Sony Legged Robot League

The aim of RoboCup is to utilise the competitive spirit of the participants to stimulate research in robotics. The Sony legged robot league is one of four leagues that currently form the RoboCup soccer competition. In some leagues, the teams are required to design and build the robot hardware themselves. In the case of the Sony legged robot league, all teams use the same robotic platform, manufactured by Sony. Since all teams use the same hardware, the difference lies in the methods they devise to program the robots.

From 1999 to 2001, each team in the Sony legged robot league consisted of three robots with the matches of two 10-minute halves. In 2002 the team size was increased to four, with a larger field.

The robots used in the legged league are a slightly modified version of the Sony AIBO entertainment robot. Although designed for the entertainment market, the Sony robots are extremely sophisticated machines, with an on board MIPS R4000 processor, colour camera, accelerometers, contact sensors, speaker and stereo microphones. Each of the four legs has three degrees of freedom, as does the head. Programs are written off-board in C++ and loaded onto a memory stick that is inserted into the robot. All of Sony's ERS robots run a proprietary operating system called Aperios, which provides an interface for the user-level programs to the sensors and actuators.

The field used in the Sony legged league up to 2000 measured 2 metres wide by 3 metres long. For the 2002 competition the dimensions were increased by 150%. The field has an angled white border designed to keep the ball within the field. The game ball is coloured orange and the two goals are coloured yellow and blue. The field also contains six coloured landmark poles to aid the robots in localising themselves in the field. In front of each goal there is the penalty region that only the goalie is allowed to defend.

History

In 1998, three teams from Carnegie-Mellon University (CMU), the Laboratoire de Robotique de Paris (LRP) and Osaka University played demonstration matches in the RoboCup competition. The robots used then were only prototypes and their capabilities were still a matter of exploration for the teams. The biggest problem they faced was programming the vision system to see the ball reliably. The CMU team won that competition largely because their robots were able to find the ball.

UNSW's first entry into the competition was in 1999. Based on the advice of Will Uther, from CMU's 1998 team, we concentrated on developing the vision system first. Our approach to strategy was to take the simplest possible behaviour and make it work with reasonable reliability. Being our first entry and being pressed for time, we made no attempt to implement any sophisticated theory. We simply wanted to find the ball, get behind it and then walk towards the goal. We had no goalie until a simple one was written at the competition itself. However, our "keep-it-simple" approach was sufficient to get us to the final, where we were defeated by the LRP who had developed a superior locomotion method. We had used the walking routines provided by Sony whereas LRP developed a walk that was faster and more stable.

The 1999 team consisted of two undergraduate students: John Dalgliesh and Mike Lawther. Their efforts laid the foundations for the 2000 team: Bernhard Hengst, Son Bao Pham and Darren Ibbotson. They built on the 1999 vision system and world model, significantly improving the localisation method. They also developed an entirely new locomotion method that was very agile and stable. Keeping to the principle of "keep-it-simple", the vision, localisation and locomotion were much faster than those of the other teams in the 2000 competition.

One of the biggest lessons learnt in 1999 was the importance of testing new behaviours under competition conditions, that is, with several robots getting in the way. Because of lack of time in the first year, all behaviours were tested using a single robot. Seeing all the robots interacting on the field at the competition was a revelation. In 2000 we resolved to always test with several robots on the field and to run practice games as often as possible. Our method for developing strategies was to observe and analyse as many game situations as possible and build situation-specific heuristics where needed.

In the early days of Artificial Intelligence research, attempts were made to build general purpose problems solvers. These worked well for small-scale problems, but were rarely successful in practical applications. Later, "knowledge-based" systems took the place of the general purpose problem solver when it was realised that domain specific heuristics are more powerful than general methods. We quickly made the same transition in designing strategies for our robots. The key to good performance, has been to have simple behaviours that are tailored to suit particular situations. Using this approach, the 2000 team became the legged league champions, never scoring less that 10 goals in every match and only having one goal scored against them throughout the tournament.

After each competition, the teams publish their methods and most also release their source code, as we did. So we expected the 2001 competition to be somewhat tougher. Indeed, many of the teams adopted or were inspired by UNSW's 2000 locomotion method and several teams also used similar localisation methods. In particular, CMU developed skills that were very similar in performance to our own. UNSW's 2001 team: Spencer Chen, Martin Siu, Tom Volgelgesang and Tak Fai Yik, developed several new manoeuvres for the robots. But the outcome of the final against CMU was certainly not a foregone conclusion. It was somewhat surprising that the score at the end of the game was 9-2 in UNSW's favour. The main difference appeared to be that we had a larger range of situation-specific behaviours and through exhaustive testing, we had reduced the number of mistakes made by the robots during the match.

At the time of writing, the 2002 competition is only a few weeks away. So much of the basic development for this year's team has been completed. The major changes to our software are due to changes in the game itself. The size of the field has been increased, an additional robot has been added to the team and wireless Ethernet communication (WLAN) between the robots is now possible.

In the following sections we describe the development of each of the major skills required by the robot and conclude with a discussion of the development of game play behaviours. Throughout the evolution of the UNSW software, the same basic architecture has been used. A layer of infrastructure modules provides the basic skills: vision localisation and locomotion. These are combined to produce the behaviours for playing the game. The behaviours are, in turn, combined to create different roles such as the goal, forward, mid-fielder, etc.

Vision

Colour Classification

A colour camera is the robot's primary sensor. The current robot, the ERS-2100, has a 196x144 resolution camera mounted in the "snout" of the robot. This delivers 25 frames per second. The information in each pixel is in YUV format, where Y is the brightness and U and V are the colour components. Since all the objects on the field are colour coded, the aim of the first stage of the vision system is to classify each pixel into the eight colours on the field: orange for the ball, blue and yellow for the goals and beacons, pink and green for the beacons, light green for the field carpet, dark red and blue for the robot uniforms.

The Sony robots have an onboard hardware colour look up table. However, for reasons that will be explained later, we chose to perform the colour detection entirely in software. Because colour detection can be seriously affected by lighting conditions and other variations in the environment, we need a vision system that can be easily adapted to new surroundings. One way of doing this is by using a simple form of learning. This was originally developed by the 1999 team and extended by all the subsequent teams.

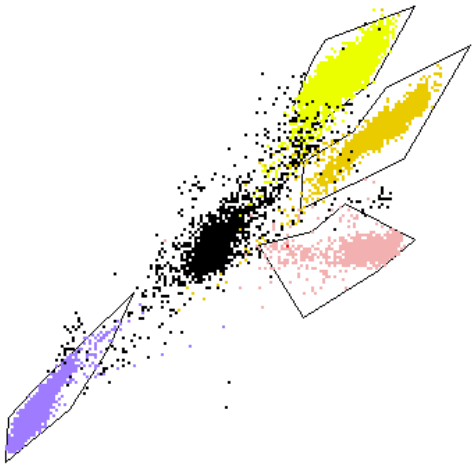

Fig. 1. A polygon growing program automatically finds regions of pixels with the same colour in one plane.

The first step is to take about 25 snapshots of different objects at different locations on the field. Then for each image, every pixel is classified by "colouring in" the image by hand. All these pixels form the set of training data for a learning algorithm. Each pixel can be visualised as a point in a 3 dimensional space.

In 1999 team's software, all these pixels were projected onto one plane by simply ignoring the Y value. Pixels were then group according to their manually classified colours. For each colour, a polygon that best fits the training data for this colour was automatically constructed. An unseen pixel could then be classified by looking at its UV values to determine in which polygons it lies. As the polygons can overlap, one pixel could be classified as more than one colour.

Figure 1 shows a screen grab at the end of a run of the polygon growing algorithm. It illustrates why we chose to use polygonal regions rather than the rectangles used by the colour lookup in the hardware. We believe that polygonal regions give greater colour classification accuracy.

For the 2000 competition, we kept the learning algorithm but changed the way it was used. By slicing the colour space into different Y planes, the colours are more separated in UV space. Initially, Y values were divided into 8 equally sized intervals. All pixels with Y values in the same interval belong to the same plane. For each plane, we ran the algorithm described above to find polygons for all colours.

Discretisation of the Y values into eight equal intervals was a considerable improvement, however, the robots were still unable recognise the red and blue colours of robots. Those colours appear very dark in the camera images. The Y values of all pixels where examined to try to group them in such a way that dark red and blue pixels can be distinguished from black pixels. We did this manually and settled on 14 planes of unequal sizes. In this configuration, we have more

planes for lower Y values, reflecting the fact that dark colours are harder to separate. With these 14 planes, the robots could recognize the colour of the robot uniforms with reasonable accuracy.

This method of colour classification worked well in 2000 and during the development of the 2001 team. However, on arrival at the 2001 competition in Seattle, we found that the lights were so bright that some colours started washing into others. A particular source of difficulty was that the orange of the ball started to look yellow in some spots. This threw out the robot's localisation because it thought it was seeing the yellow goal in places it should not. Fortunately, we had experimented with an alternative approach to learning colour classification, using Quinlan's C4.5 decision tree learning program [3]. C4.5's input is a file of examples where each example consists of a set of attribute values followed by a class value. In this case, the attributes are the Y, U and V values of a pixel and the class value is the colour manually assigned to that pixel. C4.5 turns the training data into a set of classification rules that should be able to determine the colour of a new, unclassified pixel.

Indeed, this method did prove reliable in Seattle and resulted in a considerable improvement in performance of our robots. The success of C4.5 encouraged us to investigate other, general purpose learning algorithms. For the 2002 competition we will be using a nearest-neighbour algorithm which seems to improve robot recognition further.

Object Recognition

Once colour classification is completed, the object recognition module takes over to identify the objects in the image. The possible objects are: the goals, the beacons, the ball and the blue and red robots. A blob formation algorithm links neighbouring pixels of the same colour in the image to form blobs. Although straightforward, the blob formation algorithm is the most time consuming operation in the robot's cycle of sensing, decision making and action. In successive years, we have made the implementations faster. Based on the blobs, we identify the objects, along with their distance, heading and elevation relative to the camera and the neck of the robot.

Objects are identified in the order: beacons first, then goals, the ball and finally the robots. The reason the ball is not detected first is because we use some "sanity checks" for the ball based on the location of the beacons and goals. Since the colour uniquely determines the identity of an object, once we have found the bounding box around each colour blob, we have enough information to identify the object and compute its parameters.

Because we know the actual size of the object and the bounding box determines the apparent size, we can calculate the distance from the snout of the robot (where the camera is mounted) to the object. We also calculate heading and elevation relative to the nose of the robot and the bounding box's centroid. However to create a world model, needed for strategy and planning, measurements must be relative to a fixed point. The neck of the robot is chosen for this purpose. Distance,

elevations and headings relative to the camera are converted into neck relative information by a 3D transformation using the tilt, pan, and roll of the head.

A crucial part of the vision system is a collection of "sanity checks". These are situation-specific heuristics that are used in object recognition. They were developed in the course of running practice matching and observing when object recognition failed. Some examples follow.

Every beacon is a combination of a pink blob directly above or below a green, blue or yellow blob. One side of the field has the pink on the top of the colour in the beacons, while the other has it below. The beacons are detected by examining each pink blob and combining it with the closest blob of blue, yellow or green. This simple strategy was used with reasonable success in 1999, but was found to fail occasionally. For example, when the robot can just see the lower pink part of a beacon and the blue goal, it may combine these two blobs and call it a beacon. A simple check to overcome this problem is to ensure that the bounding boxes of the two blobs are of similar size and the two centroids are not too far apart. The relative sizes of the bounding boxes and their distance determine the confidence in identifying a particular beacon.

During development, it was sometimes noticed that the robot would try to kick the ball into a corner. The cause was the robot seeing only the lower blue part of the beacon in the corner and identifying that as the goal. When the robot is near the corner and only sees part of the blue half of the beacon, the blue blob may appear large, having its width roughly twice as long as its height, which matches a feature of the goal. To avoid this misidentification, we require the goal to be directly on top of the green of the field.

It is clear that object recognition requires further development. Either a more systematic approach using model-based vision is required or a method for learning situation-specific heuristics is needed.

Robot recognition presents a particular problem because: they are irregular shapes; they have sub-parts that move; they can adopt different orientations and they are very often occluded by other robots, possibly of the same team and therefore there is a confusion of blobs.

This has been an ongoing difficulty in all the competitions. Improved colour classification and blob formation in the 2002 team has improved the separation of the blobs that form robots and a robot's orientation can roughly be detected as heading left, right, away or towards the observing robot. Distance measurements can also be made by using the number of pixels from the bottom of the image to the blob of the robot. Knowing the height and angle of the head, we can compute an approximate distance. This, however, is still not very reliable.

Localisation

The Object Recognition module passes to the Localisation module the set of objects in the current camera image, along with their distances, headings and elevations relative to the robot's neck. Localisation tries to determine where the robot is

on the field and where the others objects are. It does so by combining its current world model with the new information received from the Object Recognition module. Since all beacons and goals are static, we only need to store the positions of the robots and the ball.

The world model maintains three variables: the x and y coordinates of the robot and it's heading. The robots first attempt to localise using only the objects detected in the current image. Being stationary, beacons and goals serve as the landmarks to calculate a robot's position. Because of the camera's narrow field of view, it is almost impossible to see three landmarks at a time, so any algorithm that requires more than two landmarks is not relevant. If two landmarks are visible, the robot's position is estimated using a triangulation algorithm.

More information can be gathered by combining information from several images. Thus, the localisation algorithm can be improved by noting that if the robot can see two different landmarks in two consecutive images while the robot is stationary, then triangulation can be applied. Typically, this situation occurs when the robot stops to look around to find the ball.

The world model also receives feedback from the locomotion/action module, *Pwalk*, to adjust the robot's position. The feedback is in the form (*dx*, *dy*, *dh*) where *dx* and *dy* are the distances, in centimetres, that the robot is estimated to have moved in the *x* and *y* directions and *dh* is the number of degrees through which the robot is estimated to have turned. Odometry information is clearly not very accurate and small errors in each step accumulate to eventually give very large inaccuracies. Another problem occurs when the robot gets stuck, for example, by being blocked by another robot. *Pwalk* is not aware of this and keeps feeding wrong information to the world model.

The methods described above cannot provide reliable localisation because the robot usually sees only one landmark in an image and the robot is moving constantly. In 2000, we introduced a method for updating the robot's position using only one landmark. Suppose the landmark has coordinates (*xb,yb*) and its distance and heading relative to the robot are *db* and *hb* respectively, as shown in Figure 2. We draw a line between the (*xb,yb*) and the estimated current position of the robot in the world model, (*x_cur,y_cur*). The robot's perceived position, relative to the landmark, is the point on that line *d cm* away from (*xb,yb*) and on the same side as the current robot position, in the world model. The localisation algorithm works by "nudging" the estimated position in the world model towards the perceived position relative to the landmark.

With a camera frame rate of 25 frames/second, this algorithm converges quite quickly and accurately. Single-landmark update overcomes many of the problems caused by odometry error. Even when the robot's movement is blocked and the odometry information is incorrect, if the robot can see one landmark it will readjust its position based on that landmark.

One problem remains due to the perception of landmarks. Since we mostly use the size of the object to estimate distance, noise in the image, occlusions, motion and other factors make distance measurements unreliable.

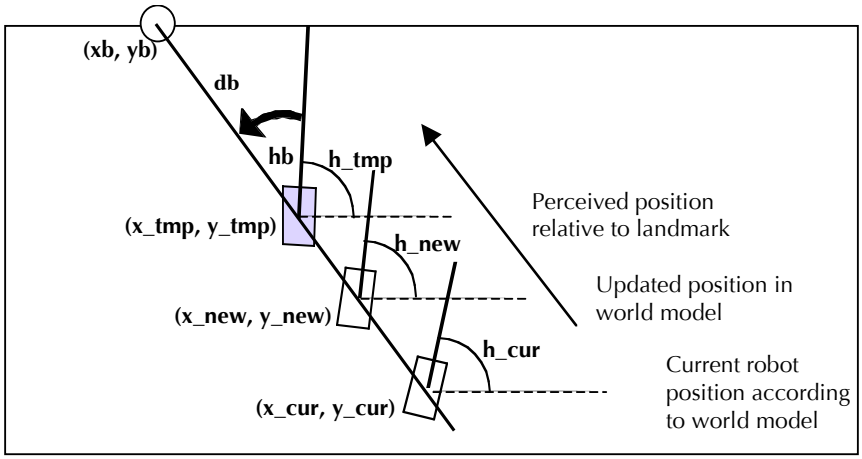

Fig. 2. Update of position based on single beacon.

Beginning in 1999, a confidence factor was associated with each dynamic object in the world model. This was used when incorporating new data into the existing model. To avoid the problem of error accumulation, the confidence factors of the objects were decayed after every camera frame. It was later realised that our method was actually a crude form of Kalman filtering [2]. Therefore, in 2002, we have cleaned up the localisation method and replaced the ad hoc confidence factors by the variance obtained from the Kalman filter.

The 2000 team adopted the philosophy of passive localization, where the robot does not actively look around to localise itself during match play. This allows the robot to concentrate on gaining control of the ball instead of wasting time locating itself all the time. This strategy resulted in our players being able to maintain their focus on the ball and move it down the field while opposition robots would take their eyes off the ball to localise themselves. Thus, for the majority of the game, we had control of the ball.

Our robots must still have accurate estimates of their positions. This information is obtained from occasional glimpses of the beacons and the goals while we are competing for the ball. The frequency of these glimpses is sufficient for the world model to maintain a fairly accurate representation of the game state. An advantage of the low stance of our robots is that they can often see beacons without having to look up. In tuning the locomotion parameters, special care was taken to maintain this benefit. Another factor that increases the frequency of seeing landmarks is the fact that the robot inevitably looses track of the ball, for example when challenged by an opponent. During execution of the find ball routine, the tilt of the scanning head was also tuned so that the robot can see the beacons as it scans for the ball. So the Find Ball routine also acts simultaneously as a method of active localisation.

In 2001 a more active localisation method was introduced that forced the robots to look about and localise when the confidence levels in the world model fell be-

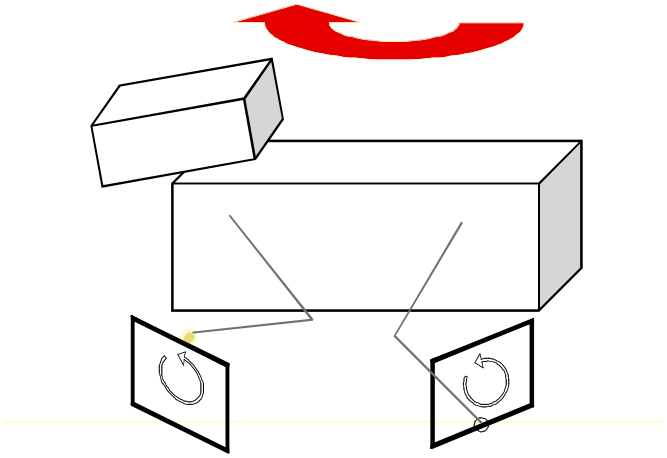

Fig. 3. The robot's paw describes a rectangular locus. The robot's walk can easily be changed by adjusting the dimensions and orientation of the rectangle

low a specified threshold. In 2002 this has been modified so that when the variance from the Kalman filter is large, active localisation is triggered. A major change in 2002 is the introduction of wireless Ethernet communication between the robots. This enables the robots to share their world models and offers the potential to improve the accuracy of the positioning of objects in the model.

Locomotion

In 1999, we used the walking routines provided by Sony. These were not designed for playing soccer. A library contained routines for a variety of simple walks and turns. Our biggest problem was that there was no smooth transition from a straight walk to a turn, so the robots were very clumsy in manoeuvring and frequently fell over in transitions from one type of walk to another. The 2000 team wanted to be able to drive the robots as if controlled by a three degree of freedom joystick. That is, the robot should be able to travel in any direction, regardless of which way it was facing. To achieve this, they devised a parameterised walk. The paw of each leg describes a rectangle as it moves (Fig. 3). A particular kind of walk or turn is specified by giving the dimensions and orientations of the rectangles. The joint angles are automatically computed by inverse kinematics.

By continuously varying the parameters, the robots can make smooth transitions from a straight walk to a turn to a sideways walk, etc. Thus, the robots are now highly manoeuvrable. A further refinement was to improve the speed and stability of the robots. After considerable experimentation with different parameter settings, it was found that a long, low stride gave the greatest speed while also being very difficult to push over during a game.

Three control parameters, one speed parameter and 8 stance parameters influence the leg movements for a particular walk style. We could try automating the search for the best parameter settings by a method such as gradient ascent on a performance measure (e.g. forward speed), incrementally adjusting the parameters over many test runs. We chose not to automate the search because it would take considerable time and resources to set up the learning environment. We were also concerned about the wear and tear on the robot motors and leg joints given the long periods of training required. Hornby, et al[1], report continuous training times of 25 hours per evolutionary run using their genetic algorithm (GA).

The approach we adopted was to manually adjust the parameters after a number of runs of observing and measuring the performance. Unlike gradient ascent or a GA we were able to adjust the parameters using our judgement and knowledge about the robot's dynamics. These considerations include:

1. The home positions of the legs must be adjusted so that the robot dynamically balances on the two legs that touch the ground.
2. A longer stride requires the robot body to be close to the ground to allow the legs to reach further.
3. The front and back legs must not interfere with each other.
4. The legs need some sideways spread to allow the robot to move its legs laterally so it can walk sideways.

Besides the ground speed, we would also judge the reliability of the walk and the deviation from the intended path as a part of the performance measure. In this way we found that there was a trade off between the frequency of the gait and the length of the stride. Maximum ground speed was achieved with a longer stride and a slower gait frequency. This manual approach used about 10 minutes of robot running time and only about 12 iterations of parameter adjustments to find a good performance setting resulting in the characteristic low forward leaning stance of the UNSW robots.

A further refinement that increased ground speed considerably was the introduction of a canter action. The canter action sinusoidally raises and lowers the robot's body by 10mm synchronised with the trot cycle. The parameters could then be manually tuned so that the robot was able to reach speeds of 1200cm/min. This compares to 900cm/min achieved using a genetic algorithm approach (Hornby et al 2000). The camera is not as steady in this type of walk because of the additional canter movement.

Head and walk parameters are varied dynamically allowing interesting skills to be developed in the behaviour module such as ball tracking, ball circling and walking towards an object. To track the ball, for example, we would take the horizontal and vertical displacement of the ball from the centre of the camera image and use parameters proportional to these measures to drive the head movement relatively. In this way, the head can be made to move to keep the ball in the centre to the image. As another example, to make the robot walk towards the ball in a straight line, we first track the ball as above and then use the degree of twist in the neck from straight ahead to feed the turn parameter of the walk. This will ensure that the robot will turn towards the ball if it should veer to either side as it walks

towards it. These types of applications of the action module form the skills that are more fully described in the section on behaviours.

The locomotion method described above gave us a great advantage in the 2000 competition. In 2001 several teams copied us, either by adopting our source code or by reproducing our gait in a different implementation. So in 2002, we looked for a ways to speed up the walk further. We found that by changing the locus from a rectangle to a trapezoid and reworking the trot cycle time, we were able to obtain a 25% increase in speed.

Communication

Team coordination amongst the robots is quite difficult. Until 2002, the only communication possible was by sound and this was quite unreliable. The Sony robots have onboard speakers and stereo microphones. However, during a competition match, there are large crowds cheering, creating loud background noise and the robot motors also generate noise through the body. So the only reliable signal that could be sent was a single, high-pitched tone. As we shall see in the next section, this was sufficient to provide some useful information to team mates.

In 2002 a wireless Ethernet link became available. The robots have an onboard 802.11b PCMCIA card that enables them to become nodes on a Local Area Network. It was agreed by all the teams that the wireless communication should only be used between robots and should not be used for centralised control from another computer. Since the WLAN has not yet seen full use in a competition, there is still much exploration needed to properly exploit its capabilities.

At present, the primary use for the WLAN during a game is for transmission of the world models from one robot to all the team mates. In theory, this should approximate global vision since four robots should be able to see most of the field. However, all the robots have significant localisation errors, so combining world models is rather tricky. We still prefer to act upon sensor data rather than the world model so rather than combining world models from different robots, we use the onboard sensor data, if it is available and only use the data from other robots if nothing else is available.

Perhaps the greatest benefit of the WLAN, so far, has not been during game play, but during debugging. We are able to transmit real-time images from the robots so that we can see what they are seeing when something goes wrong. We can also transmit real-time displays of the world model so that we can detect mistakes in the localisation routines.

Game Play Behaviours

To play a game of soccer, we require a library of behaviours that can be combined to create a player. These skills include: find the ball; get behind the ball; dribble

on the chest; push with the chest; kick with the paw; a power kick and a side-ways kick while turning with the ball. The players are the forwards and the goalie.

The basic strategy for the forward is simple: the robot tries to align itself behind the ball, and then selects an appropriate attacking skill to move the ball towards the target goal. However, the details of the strategy are modified by the robot's location on the field and its relation to other robots.

An important innovation in 2000 was keeping team mates separated so that they did not interfere with each other. For example, robot 1 is dribbling the ball. Its team mate, robot 2, sees the ball and starts running at the ball. When robot 2 gets close to robot 1 it backs up, giving way to robot 1. This behaviour also reduces the occurrence of rugby-type "scrums" when a pack of robots all fight for the ball, mostly just getting in each other's way. The backup behaviour tends to keep one robot on the "wing" of its team mate, which effectively makes one robot wait outside a scrum for the ball to pop out. Once that happens, the "wing" can attack the ball in clear space.

In 2001, sound communication was added when a robot backed away from its team mate. A high-pitched tone signalled to the front player that a team mate was behind. Using the robots' stereo microphones, it was sometimes possible to know which side the supporting robot was on. Thus, if an opponent challenged the front player, the ball could be kicked to the side of the supporting player.

With the introduction of a fourth player and the WLAN in 2002, the backing away strategy has been extended to include the notion of a "potential field". That is, each robot on the same team tends to repel the other robots so that they stay well spread. Special conditions apply as the robots approach the ball since we do not want a deadlock to occur if two team mates are attacking. In general, the robot that has the best position (eg, facing the goal) has the right-of-way.

The behaviour of the forward is also modified depending on its position in the field. For example, when the robot has the ball near the target goal, it is worth taking time to line up on the goal. However, if the robot is far from the target, it is more effective to simply knock the ball into the opponents half. This wastes little time and does not allow opponents the chance to take the ball away. If the robot is approaching the goal and the ball is in the clear, it is worth lining up carefully. However, if an opponent robot is nearby, it is better to simply attack and take control away from the opponent.

The goalkeeper strategy employed by our teams since 2000 utilises three behaviours to defend its goal: find the ball; track the ball and acquire a defensive position; clear the ball. Clearing the ball is activated when the ball gets close to the goal or the ball enters the penalty area. The mode finishes when the ball is kicked, lost, or moves more than a set distance from the defended goal. Upon deciding to clear the ball the robot will evaluate its position relative to the ball and determine whether it can directly attack the ball or should reposition behind it.

Ideally, we would like to have a game in which team mates pass the ball to each other. This is still illusive since it takes considerable accuracy in vision and localisation and quick skills in the receiving robot to track and catch the ball.

Conclusion

The Sony legged league of RoboCup has been a great source of interesting research problems both for robotics and artificial intelligence. As we mentioned at the outset, it was not clear if it made sense to use these robots for playing soccer. However, the fact that we now have credible play demonstrates that providing a powerful software development environment for a "general purpose" platform allows programmers to be creative in constructing competent behaviours. The performance of a robot will always be limited by the quality of its sensors and actuators but clever programming allows us to get the most out of the available resources and sometimes even surprise the designers of the robots.

The other major lesson learned through our involvement in RoboCup has been the importance of situation-specific behaviours. The reason for much of the success of the UNSW robots is that they are highly reactive and therefore fast. All the planning for the robots is done during countless hours of practice sessions so that when they are on the field, we have anticipated as many as possible different situations that the robots might find themselves in. Clearly this is very labour intensive. Ideally we would like the robots themselves to learn through practice. What is not yet clear is where Machine Learning can be used most effectively.

Acknowledgements

The results described here are due to the efforts of successive UNSW RoboCup teams: John Dalgliesh and Mike Lawther (1999); Bernhard Hengst, Son Bao Pham and Darren Ibbotson (2000); Spencer Chen, Martin Siu,, Tom Vogelgesang and Tak Fai Yik (2001); James Brooks, Albert Chang, Min Sub Kim, Benjamin Leung, Canh Hao Nguyen, Andres Olave, Tim Tam, Nick von Huben, David Wang and James Wong (2002).

References

1. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O. (2000) Evolving Robust Gaits with Aibo. IEEE International Conference on Robotics and Automation. pp. 3040-3045.
2. Kalman, R,E. (1960) A New Approach to Linear Filtering and Prediction Problems, Transactions of the ASME Journal of Basic Engineering, **82**(D), pp 35-45.
3. Quinlan., J.R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.