Abstract. This paper summarises the fourth year of software development by the UNSW team competing in the RoboCup 2002 Sony legged league. The team won the RoboCup 2002 Challenge and was placed second in the soccer tournament. We describe new developments in vision and localisation procedures for robot soccer, as well as innovations in locomotion and the design of game play and challenge 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.

1 Introduction

The purpose of the RoboCup is to utilise the competitive spirit of researchers and students to stimulate research in robotics. The Sony legged robot league is one of four leagues that currently form the RoboCup soccer competition. Each year, following our initial entry into the competition in 1999, the new team built upon the successful ideas of the previous year and added its own innovations. The UNSW has an excellent achievement record being runners up in 1999 and champions in 2000 and 2001. The 2002 team again performed strongly finishing runners up in the soccer match against 18 other international teams and winning the challenge competition outright.

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 robot team size was increased to four, with a larger field. The robots used in the legged league are identical to the Sony AIBO entertainment robot available commercially. 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. A major change for 2002 was the introduction of wireless
Ethernet communication (WLAN) between the robots.

The field used in the Sony legged league up to 2001 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 distinguishable landmark poles (or
beacons) to aid the robots in localising themselves in the field. In front of each
goal there is the penalty region which only the goalie is allowed to defend.

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 skills for playing
the game. The skills are, in turn, combined to create different roles such as the
goalie, forward, mid-fielder, etc.

In the rest of this paper we will present the main innovations for 2002 for
each of the main modules for game-play and the challenge competition.

2 Vision

Every other module in the RoboCup code infrastructure is dependent on in-
formation provided by the vision system. It is crucial that the vision system
functions well. The goal of the vision system is to be able to recognise objects on
the field, for example to see and distinguish a blue-pink beacon from a pink-blue
beacon. This is achieved by feeding training data for the colours of objects into
a learning algorithm. The algorithm produces a colour lookup table that will
give the learned colour of each image pixel from its Y U V value. The table is
stored in the robot’s memory, and on every camera frame the robot translates
the camera image into colours it recognises by using the lookup table. Using blob
formation (looking at groups of pixels of the same colour) objects are recognised
and bounding boxes are drawn around them.

2.1 Colour Calibration

RoboCup 2001 used two different vision learning techniques: polygon learning
and decision tree. Polygon learning was used in the early rounds of the com-
petition and the decision tree was used in the finals. It was found that these
techniques were slow in learning. A small mistake in the calibration process
would result in waiting for excessive periods of time while the learning algo-

390

rithm was re-run. A new vision learning algorithm that ran quickly and allowed
for changes to take effect with minimum delay was desired.

Both former vision learning techniques produced colour lookup tables which
mapped any YUV coordinate to one of the nine colours of the field. Although
a YUV coordinate can only have one colour in most cases, there are situations
where it can have two. For example consider the colours background and dark
blue. A YUV coordinate that is somewhere between these two colours has to be classified as either one. Using the old techniques, a pixel with this YUV value would get classified as background regardless of its location in the image. However because this YUV coordinate is a borderline case i.e. it can be either blue robot or background it would make sense to classify it as background if the pixel is in a region of the image which is predominantly background and classify it as blue robot if it’s in a region of the image which is a part of a blue robot. A vision learning algorithm that takes into account both the YUV coordinate of the pixel and the location of the pixel within the image to determine the colour would improve the vision.

The new vision learning technique is a nearest neighbour algorithm. It populates the YUV colourspace (see figure 1), marking specific coordinates as colours determined from the training pixels. The colour learning process generalised each unmarked pixel colour by setting it as the colour of the closest marked pixel. If the unmarked pixel is not within a defined threshold of a training pixel then it is set as the background colour. This new technique means the colour lookup table is much larger as we are now storing the entire YUV colourspace. Due to memory restrictions on the ERS-210 the colourspace is compressed into 128x128x128 coordinates. The lookup table size is then $128 \times 128 \times 128 \times 1 = 2.0MB$ as the colour can be stored as an unsigned char of one byte.

Fig. 1. Colour Cube
To teach the robot how to recognise the specific colours, training samples of colours must be collected. As the colour of the objects greatly depends on factors such as the position of the main light sources and amount of ambient light, the training samples must be recollected every time there is a change in the lighting conditions. The training examples are collected in the form of a BFL image and pixels in the image are marked to be one of the nine possible colours. The camera inside the robot captures a 176 x 144 pixel image. The development of wireless debugging tools allowed the robot to capture a BFL image in an arbitrary position on the field and have it sent over wireless to be saved on a workstation. The calibration process saves several of these images creating training data after each pixel in the images has been manually classified. The training data is provided to the nearest neighbour algorithm to generate the colour lookup table.

One of the limitations of the vision techniques from the previous years was that each YUV coordinate could only be assigned one colour. However, the colour of a YUV coordinate is not constant, it is possible that it changes from picture to picture. The multi-colour pixels technique allows a YUV coordinate to be classified as any combination of the nine colours. It then uses the colour of the nine surrounding pixels to classify the colour of a particular pixel within an image.

From previous years it was found that the colour of the blue robot was consistently classified the worst. This is due to the YUV coordinates of the colour being close to the YUV coordinates of background colour. Initially the multi-colour pixels technique was implemented for all nine colours. However it was found that it introduced an unacceptable performance degradation. In the final implementation only multi-colour pixels being labelled as blue robot or background colour were allowed.

In our the implementation this year there are 10 colours. The first nine are the initial colours and the 10th colour is one that can be blue robot or background. The multi-colour pixels are generated during the execution of the nearest neighbour algorithm when an unmarked pixel is within the threshold distance of a blue robot training pixel and also within the threshold distance of a background training pixel. Without multi-colour the unmarked pixel would have been labelled as either blue robot or background depending on which training pixel was closer.

When a pixel within an image needs to be classified, its colour is determined using the colour lookup table. However if the colour is blue robot or background the colours of the nine surrounding pixels are examined. The number of neighbouring pixels which have the blue robot colour and the number of pixels which have the background colour are determined. Based on which count has a higher value, the original pixel is marked as that colour.

In practise the whole calibration process took on average one hour from wireless picture taking to calibration verification. The manual classifier tool already contains efficient painting techniques such as wand and brush so that each pixel does not have to be individually marked.
It is rare that a calibration is effective the first time. New BFL images need to be taken and coloured in and the nearest neighbour algorithm re-run on the new training data. The algorithm runs extremely fast taking approximately 10 seconds. This is a substantial speed increase from the vision techniques from the previous years (the decision tree took 45 minutes to run) and hence allowed for fine-tuning of the vision to be done with relative ease.

Each time verification takes place the new lookup table must be copied onto a memory stick and the robot rebooted. This can be tedious and take up time when performed multiple times. The ability to send the lookup table over wireless and straight into the robot was developed. This improved the speed of the calibration verification process. Once the lookup table was finalised it only need to be copied onto the memory stick once.

2.2 Multiple Robot Detection

Multiple robot recognition did not exist prior to 2002. The 2001 code detects the largest red or blue blob in the image. This works well in most cases but often the largest blob does not represent the robot nor does the center of the blob’s bounding box necessarily represent the center of the robot.

For 2002 a heuristic is used to group together blobs to form individual robots. The horizontal extremities of the blobs that belong to a robot always overlap or are at most separated by a small distance. Blobs that are too far apart horizontally most likely belong to a different robot. The usual noise filtering process applies to all blue red blobs with elevations greater than the top of the beacons and goals removed.

Given the dimensions of the bounding box, it is now possible to calculate the distance of the robot to the camera. Due to the non uniform shape of the robot when viewed from different angle, it is difficult to apply the standard bounding box method to resolve the distance. The method used in the 2002 code uses a relationship between the angle of deviation of the object and the flat distance of the object from the camera (see figure 2). Robot distance is calculated via the angle of deviation of the bottom edge of the bounding box from the field plane. The benefit of using this method is that the distance to a robot can be calculated without seeing the whole object. Since the robot is expected to spend most time tracking the ball, it generally only sees the legs patches of other robots.

Variance for robot objects is calculated differently to other visual objects since there is no expected area or other constant values. Confidence level is based on the height of the bounding box in proportion to the screen (176 x 144). Height is probably the most invariant property of a robot as it is viewed from different angles. As the robot gets closer to the camera, its height increases and as such, confidence in its distance increases. Variance for a robot is calculated using the average of the distance measured if the bounding box is 2 pixels lower (i.e. closer to the robot ) and if the bounding box is 1 pixel higher (i.e. further from the robot). The difference in the number of pixels used when lowering and lifting the bounding box is to balance out the hyperbolic relationship between the angle of deviation and the distance.
Robot recognition improved considerably in 2002. The variance measure was required for the revised localisation software.

3 Localisation

Information from the robot’s vision system, as well as odometry from the locomotion module, are combined to create a world model. Since the dimensions of the field are known in advance, the world model is constructed by placing the mobile objects on a map of the field. As objects are observed to move, their positions in the map are updated. However, some apparent motions may only be due to noise in the image. It is therefore unwise to instantaneously update the position based on a single measurement. Until the 2002 competition, the update method adopted was to compromise between the objects current position in the world model and the perceived position by gradually ”nudging” the position away from the current position towards the perceived position. An ad hoc confidence factor was attached to the location of an object in the world model. This described a belief in the accuracy of the estimated distance to the object. The confidence factor was based on the percentage of the object seen and other criteria. It was used to determine the weight of the contribution of the object’s perceived distance to the update of the estimated position of the object. This localization method worked well in previous competitions. With the increase in field size the measurements to distant objects became more unreliable and the ad hoc method broke down. This method was a simple version of a more principled approach to updating sensor measurements, the Kalman Filter.

3.1 Kalman Filter

In the Kalman filter, there are two sets of equations: time update equations and measurement update equations [WB01]. The time update equations are re-
sponsible for projecting forward (in time) an estimated state, and are used when odometry information is received from the locomotion module. The measurement update equations on the other hand are responsible for feedback by incorporating new measurements and are used when information from the vision system is received. The two set of equations form an ongoing discrete Kalman filter cycle. For the purposes of exposition, we will briefly explain how the Kalman filter works by describing how a robot can localise itself relative to fixed landmarks. The variables for robot localisation are the x and y coordinates, the variance of the location, the heading and the variance of the heading. The origin is at the bottom left corner of the field. The x axis runs along the edge of the field with the robot’s own goal below the axis and the opponent’s goal above the axis. The y-axis runs along the left edge of the field with the goals to the right of the axis. Robot localisation uses beacons and goals as the landmarks and receives odometry from the locomotion module. When the robot sees a beacon or a goal, the identified object is passed to the world model. This module then uses the position measurements and the current estimated position and heading to build a new position and heading where it believes the robot should be. The robot first checks if it can see two beacons. If it can, it uses the two beacons to triangulate its location. If it can only see one beacon, it will use the Kalman Filter to move its estimated position towards the beacon so that the world model distance to the beacon is closer to the measured distance.

To cater for Kalman filters the vision system must produce a variance measurement for each object identified. In order to escape from ad hoc values like the confidence factors, the variances of measurements are derived by recording the error in the estimated distance. By using statically collected variance measurements not only have we moved away from magic numbers, we have also made the confidence of distant measurement distant dependent. This means far objects will have much higher variance than those of closer objects due to the physical limitations of the camera. This should make the world model much more stable and resilient to error.

As well as localising itself, a robot must also estimate the positions of the ball and the other robots. The Kalman filter update algorithm is the same but a complication is that there are several robots. It is not always obvious which robot position to update. The Kalman filter uses variances to represent the confidence of object’s location. By assuming a Gaussian error distribution and using 2 standard deviation radius circle we update the Visual Model from the vision system with the World Model robots that have overlapping circles (see figure 3).

To predict the next location of a ball or a robot we must build a world model that records the direction and velocity of the object. Unfortunately, these measurements are difficult to obtain with the current sensors. Therefore, the time update in the Kalman filter algorithm is simulated by having an artificially high movement variance that assumes the object will move in any direction randomly. This time update simulation is carried out just before the measurement updates.
3.2 The Wireless Model

With the introduction of wireless communication between robots, a new level of representation, the wireless model, was added. This model is a representation of the world as given by a robot’s team mates. It is kept separate from the robot’s own world model. We associate with each team mate a variance which represents our confidence in the accuracy of the information from that robot. The variance of an object in the wireless model is the sum of the team mate’s variance, the object’s variance, as believed by the team mate, and a small variance for the latency over the wireless network. During a game, each robot on the field receives information from every other robot. This means that a robot must combine the world models of all three other robots and its own. The technique is the same except this time the inputs will be four world models, one for each team mate. Where the variances of two robots overlap, they are combined into one. Once the information is merged, the best (lowest variance) four robots of each colour are stored in the wireless model. The same process is also applied to combine information for balls. Only the best ball is selected. In the wireless model, three extra robots are also stored. These describe the location of the team mates according to their own belief.

Fig. 3. Sensor Fusion of Robots

Despite the many improvements obtained through the use of a principled technique such as the Kalman Filter localisation remains a difficult problem. Er-
ratic positioning and frequent errors can be reduced, but not eliminated. Com-
bining information from other robots can be of use on some occasions, but in
others situations, it is better to ignore the information coming from team mates
and rely on onboard sensors. For these reasons, the game play software contains
many heuristics that attempt to use "common sense" in the application of lo-
calisation techniques. Unfortunately, the heuristics are also ad hoc and prone to
error. Ideally, one would like to be able to place robots on the field and have
them learn to tune parameters and acquire or discard heuristics. When dealing
with physical systems one cannot afford to run the thousands of trials that are
often required by learning systems. Learning in simulation has not transferred
well to the physical world. Many challenges remain in finding effective methods
for dealing with uncertainty and noise in physical systems.

4  Locomotion

The locomotion module is responsible for handling the motion of the robot, and
can broadly be divided into walking and kicking routines. All of the walking styles
used employ the trot gait, in which the diagonally opposite legs are synchronised.
This means that as the robot walks, the weight of the body is balanced diagonally
across the robot, and shifts with each step taken. The walking routines are very
flexible, operating on a set of supplied parameters. These parameters can be
defined to produce particular stances and walking styles [HIPS01]

4.1  Trapezoidal Locus

Initially, three variations were tried on the past years’ walks. Two of these
changed the shape of the paw locus, and the other changed the timing of the
movement around the walk locus. One of the dominant problems found with
varying the locus shape was that the claws on the rear paws would drag on the
field, rendering the walk ineffective. The regular walk with the rectangular locus
generally avoids this problem, as the paw is lifted directly up without trying to
move it forward, thus avoiding getting the claw stuck. The top edge of the locus
is the longer one (see figure 4). The timing of the trapezoidal locus is upside-
down compared to the rectangular locus. That is, the duration covered by the
top edge of the trapezoidal locus is the same as that for the time taken to drop
the paw, to move it back, and to lift it. This timing means that there are times
in the walk cycle when there are no paws on the bottom edge of the walk locus,
producing a bouncing effect. The disadvantages of the trapezoidal walk are that
the camera tends to shake more and it is somewhat less controllable than the
regular walk. This makes it less suitable when approaching the ball at a close
range, since the robot is more likely to lose sight of the ball due to the camera
shaking. It also has more difficulty when turning or walking backwards.

By significantly reducing the drag caused by the robot claw, the trapezoidal
walk allows the robot to maintain momentum and move quickly. Combined with
the low stance, the trapezoidal walk is also stable, making it good for charging.
The main advantage is that it is able to increase forward speed of the robot by 30\% over that achieved in previous years.

4.2 Kicks

The forward kick originated from the UNSW 2000 team. The kick is based on a scissors lever action, by dropping the front legs of the robot on the ball in order to shoot it forwards. This idea has served as the basis of many kicking actions developed for 2002. The \textit{chunky kick} is a reliable and powerful kick. Based on the old forward kick, this was a second attempt at utilising the rear legs during the kicking motion to produce more power. The \textit{fast kick} was devised late in development, originally intended to fill the range gap between the chest push and the strong forward kick. The chest push executes quickly but has a short range, while the strong forward kick is relatively slower but has very long range. The fast kick was intended to be a middle range kick with faster execution than the strong forward kick. The result was a fairly powerful kick that was slightly faster than the chest push, but requiring a more restricted setup. Its surprisingly fast execution time earned it the nickname \textit{"lightning kick"} among the team. The \textit{aimable forward kick} is the most useful version of initial developments based on the old forward kick. As its name suggests, this kick can be aimed to a certain extent, providing more flexibility and accuracy.

Another skill introduced by last year’s team, the chest push is a useful short range kicking skill. It was left unchanged from last year, and was used heavily in strategy.

The sideways swipe was devised as means of short range passing. It was designed to allow the robot to make short passes to teammate robots beside it, thus moving the ball around opposition robots before they can recover to defend. The turn kick can also produce a similar effect, but it is generally too powerful for short ranges. In general, the sideways swipe was not useful at all. Its major downfall was its instability, which seemed worse than the aimable forward kick. Like the aimable forward kick, the side swipe necessarily loses a bit of balance when swinging the robot body, and in the process, it becomes easier to push over.
5 Skills

In addition to the basic skills handled by the locomotion module, there is an additional set of skills that are covered by the strategy module. These skills are more complicated than the skills provided by the locomotion module, as they are designed to interact more with the game. This is the main reason for these skills not being handled by the locomotion infrastructure; they require access to game world objects and interact with them. Skills in the locomotion infrastructure are atomic and simply move the robot without any regard for the game environment.

Trapping the ball can be considered to be the most fundamental prerequisite to consistently executing kicking skills. The basic setup for virtually all the kicking skills is to trap the ball directly in front of the robot, with the ball between the robot’s front legs, up against the robot’s chest. Consequently, being able to perform this skill reliably and consistently is critical to the overall strategy.

In addition to reliability, for this skill to be useful in strategy, there is the secondary requirement of speed. In match play, if the robot takes too long to gain control of the ball, it gives too much time for the opposition robots to surround the ball. This renders the kicking skills useless, as it is virtually impossible to kick the ball when surrounded. Another reason for this skill to be fast is that there is a ball holding time limit applied in match play, allowing forwards to hold the ball for 3 seconds, and the goalie to hold the ball for 5 seconds. Since the ball holding time limit starts from when the robot’s front paws are around the ball, the robot has little time left after trapping the ball to decide and execute a kick. Trapping the ball must be both reliable and efficient.

The critical parameter in this skill is the trigger value for the head tilt, used to decide when the robot has trapped the ball close enough to execute a kick. This parameter requires a fair degree of tuning, and is further complicated by different ball tracking skills and different walking styles.

The paw kick is a reworking of the paw dribble skill from last year. In short, the paw kick works by simply walking the robot in the direction of the ball with a small offset to the side, so as to hit the ball with the robot’s paw. Since the paws face forward with the regular low stance, the ball is hit consistently forward, with deviations resulting only due to inconsistencies on the ball or field.

In addition to the paw kick, there is a second version called the lateral kick. The lateral kick is virtually identical to the paw kick, except that it hits the ball while walking sideways, thus hitting it with the side of the lower leg.

The main advantages of the paw kick are its minimal setup time and speed. The paw kick is more appropriate than most other kicking skills when contesting the ball against opposition robots along the edges of the field [Ola02].

The turnkick is a movement that a robot can perform in order to kick the ball at a large angle (greater than 90 degrees) from the current facing direction. The kick execution time is relatively small and there is not much setup time or complexity in the setup manoeuvres. It gives the robot the advantage of being able to approach the ball from almost any direction and without much manoeuvring move the ball at speed in the desired direction. In 2002 the locomotion
module was used to provide the correct starting locations for the legs increasing
the reliability of the kick dramatically.

6 Strategy

Strategy is developed by running weekly games with a full team competing
against previous version of code. The stochastic nature of the game makes it often
difficult see clear progress. As another guide we keep statics on the percentage
of time the ball spends on each half of the field. We also keep track of the scores
over the weeks to ensure progress.

In strategy, there are three roles, an aggressor, an assistor and a goalie. An
aggressor will actively attack the ball while an assistor will assist in the aggressor
by positioning in the right position. Dynamic role determination decides the role
of three the robots by comparing their believed distance to the ball. The goalie’s
role is predetermined.

6.1 Forwards

The aggressor is the robot that believes its distance to the ball is the shortest
in comparison to all other team-mates beliefs of their distance to the ball. From
random errors in vision and delays in wireless, it is possible for robots to all
believe they are all aggressors or assistors. If all robots are assistors then no one
will go for the ball. To void this stalemate a threshold in the distance comparison
is used. A robot will become an assistor if its distance to the ball is more than
the distance between the ball and closest robot to the ball plus the threshold. In
the final strategy, this technique was actually not adopted. We found the robots
were too scattered and usually only one robot attacked or defended at a time.
A robot was set to aggressor by default and used a back-off strategy to avoid
conflict.

During game play when the robot decides to kick the ball or get behind a
ball, a general guidance must be given in the direction to execute the actions.
This guidance is given in the form of a graph that provides indications of the
general flow of the ball. The robots were given an angle anywhere on the field
indicating the direction they should kick the ball relative to its current heading.
This angle is called the Desired Kicking Direction DKD (see figure 5). When the
robot tries to get behind the ball, it will rotate behind the ball lining itself up
with the DKD.

The aim of the strategy is to get the ball up the field as fast as possible.
The flow of the ball is away from its own goal and into the opponent’s goal.
When the robot is at the upper half of the field, it will try to kick the ball into
the opponent’s goal. If the ball is on its own half, it will try to clear the ball
into safety by kicking it away from its own goal. When the ball is along the
edge of the field, the robot will try to slide the ball along the edge of the field.
From experiments we found by sliding the ball along the edge using a turn kick,
the ball can travel much further. Dribbling the ball along the edge also has the
advantage of avoiding opponents because they normally don’t come along the edge of the field.

Fig. 5. Desired Kick Direction  
Fig. 6. Force Field

Force field (potential field) is a technique used to keep the forwards uniformly separated over the game field by placing forces of repulsion on each robot and forces of repulsion from the edges of the field (see figure 6). Force field positioning was initially developed to solve a forward clumping problem (where all the robots would go for the ball and get in each others way). Upon implementation it was found that it was an excellent technique in keeping the robots spread. However the accuracy of this positioning technique greatly relies on the accuracy of a robot’s knowledge of its team-mates’ positions. This positioning technique was replaced with visual back-off. It was used in wireless positioning since the teammate positional data is highly accurate with the wireless network. It is also used to keep the robots separated during the find ball routine. There are three forwards in a team. The robot that is the furthest from the ball out of all three forwards, positions itself using the force field model. The robot that is the second closest to the ball out of three forwards positions itself in a wing position to the teammate closest to the ball.

6.2 Goalie

The larger field has made the goal keeper’s localisation more difficult. The forwards use the robot’s distance to the beacons to determine their localisation, the goal keeper is in such a position that using the distance to the beacon is unreliable (and also unstable). The goal keeper is at the extreme end of the field and so the difference of even one pixel of a beacon results in an error that is quite significant. The goal keeper’s two beacon localisation differs to the localisation that the forwards use. The goal keeper uses the angle between two beacons to determine the localisation of the robot. Angles appear to be more reliable than
distances. To determine the localisation of the robot, the center and radius of
the circle that passes through the two beacons and the robot is first calculated.
After the equation of the circle has been determined, the next step is to draw a
line from the robot’s approximate position to the center of the circle and deter-
mine the closest point where the line intersects the circle. The point where the
line intersects the circle is where the robot is determined to be. This method
is quite accurate but it requires the approximate position of the robot to be
fairly reasonable. The code actually uses the last position of the robot as the
approximate position. This is a fair approximation because the position of the
robot is unlikely to have changed much within 1/25 of a second. It is a problem
when the approximation is significantly wrong in the first place.

Considerable effort was spent in compromising between aggressive and de-
defensive behaviour in covering the long rectangular penalty area.

7 Challenges

The three challenges were shape recognition, collaboration and ball collection.

The shape recognition challenge involved recognising 2D checkerboard geo-
metric shapes placed above the beacons and goal. Step 1 was to extract the
checkerboard pattern from the image. The goal of this process is to extract a
shape that resembles as close as possible the shape physically placed on top of
the landmark. The method of feature extraction used works at the blob level. It
generated the outline of the shape by drawing a line between the centers of adja-
cent blobs producing a mesh. The space bounded by this outline is filled using a
spacing filling algorithm producing a shape that is almost identical to the card-
board cut out. Once the checker box pattern is extracted from the background,
a signature must be generated in order to classify the detected shape. Moment
invariants based on the extracted shapes were used to feed into the decision tree
learner c4.5. The success rate of accurately recognizing Square, Rectangle and L
Shape is about 95 percent. The most troublesome shapes is T Shape and Trian-
gle with an accuracy of about 80 percent. In the competition, it is the Triangle
that UNSW failed to recognize. The triangle was placed with its top pointing to
the right, and it was misclassified as a T Shape. The most likely cause of this
misclassification is probably due to the Japanese kanji characters on the poster
which Sony used to put behind the shapes.

In the collaboration challenge two robots were required to work together to
rotate a foam bar. Instead of using wireless to enable communication between
the two robots, it was decided to use visual triggers to make state changes. The
success of UNSW in this challenge is largely attributed to the deviation method
of calculating distance to the end of the bar. Because distances were calculated
accurately, this helped in preventing the bar from slipping when being moved
by the robot and also allow the robot code to operate relative to the end of the
bar similar to robot recognition. The UNSW solution focused on moving the bar
around as oppose to position robots accurately on the field. We came first in
this challenge.
The ball collection challenge required two robots to "collect" ten balls by shooting them into either goal. The challenge had a time limit of three minutes, and teams were ranked according to how many of the ten balls they managed to put into the goals. The initial configuration of the challenge saw the two robots at the center line, facing the middle of the field. The ten balls would be randomly placed on one half of the field; the exact placement was not revealed until the challenge was ready to begin, but the placement itself is the same for all teams. While the solution of shooting the ball along the edge seemed reasonable, it sometimes seemed to waste a lot of time as the robot would chase the ball after shooting it. The second solution devised was to walk sideways with the ball to move it away from the edge, so the robot would make room to turn to aim at the goal. To do this, the robot holds the ball, turns until it is facing parallel to the edge, and then goes into a steady sideways walk, with the head down to keep the ball in possession. Once it has moved far enough from the edge, it then holds the ball again, localises, and then aims and shoots at the goal. UNSW ranked first in the ball collection challenge, shooting all ten balls in just over 2 minutes 18 seconds, well under the time limit, and was the only team to clear the challenge. The win secured UNSW's victory in the challenge competition overall, for the fourth consecutive year.

8 Tools

A simulator was used to accelerate strategy development. Simulation was also used to observe the behaviour of potential fields. After analysing each of three strategies, a single concept was extracted which seemed to summarise the best approach to playing a game of Robocup soccer. The Simulator supported the idea that constant movement of the ball was essential to a successful game of soccer. If a strategy took too long to move the ball to a new location it was invariably swamped by opposing players often leading to a lost ball or a failure to take advantage of the situation. It was concluded that speed was more important than accuracy.

The introduction of wireless into RoboCup legged league has opened up many possibilities. Its convenience of setup and usage, its fast transmission speed compared to the serial cable, and its flexibility to send any data across the wireless network has motivated the development of a number of wireless debugging tools. This includes the tools to print out debugging statements, tools to display what the robot sees and tools to display the world model. They can give the developer real time feed back of the robots, visualisation of the state of the robot and real time manual control of the robot. Wireless capabilities provided many opportunities to explore different possibilities in the RoboCup2002 competition. It allowed robots to share their world model. We have used a unique connection model to implement the wireless capability and successfully solved many problems such as message identity. We also integrated the communication to the Sony Game Controller allowing the robots to be controlled and communicate
under the Sony connection model. Wireless has limitations in terms of latency, reliability and it may affect the processing of the robot.

9 Conclusion and Future Work

The Robocup project presents and will continue to present many areas for further improvement and development as the Legged-League Competition progresses. 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 teams is that they have been highly reactive and therefore very fast. All the planning for the robots was done during countless hours of practice sessions so that when they were on the field, we had 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.

References


