Robot World Cup Soccer: The Power of UTS Unleashed!

Nicholas Agnew, Peter Brownlow, Gamini Dissanayake, Yohan Hartanto,
Shannan Heinitz, Alankar Karol, Christopher Stanton, Michael Trieu,
Mary-Anne Williams and Astrid Zeman

Innovation and Technology Research Laboratory
Faculty of Information Technology
University of Technology, Sydney
Australia
Mary-Anne@it.uts.edu.au

Abstract. UTS Unleashed! joined the SONY Four Legged League in 2003, and in a matter of months designed and developed a competitive team of soccer playing robots from scratch. The team only lost one soccer match, and came *fifth* overall in the *Soccer Challenges* at RoboCup 2003.

The main objective of UTS Unleashed! for RoboCup 2003 was to build an agile and competitive team, so that we could gain automatic entry to RoboCup 2004 by performing well and being amongst the very best teams. We achieved our objective and can now focus on our longer term goals and aspirations which include building a robot soccer team that *knows* it is playing soccer, and exploring innovative knowledge representation solutions to managing complex information in dynamic environments.

In this paper we describe the overall architecture and the individual components of the UTS Unleashed! Robot Soccer System. The architecture implements a classical *sense-think-act-cycle* that relies on modules dedicated to the main soccer functions, namely *locomotion, sensory information management, localisation, actions and behaviours*.

1 Introduction

UTS Unleashed! joined the SONY Four Legged League in 2003, and in a matter of months designed and developed a competitive team of robots capable of playing soccer. The team only lost one soccer match during the *Championship* to the competition runners-up, and came *fifth* overall in the *Challenges at RoboCup 2003* by beating many of the veteran teams in both the *Obstacle Avoidance Challenge* and the *Black and White Ball Challenge*.

The success of the UTS Unleashed! can be attributed to a highly motivated, talented and cohesive multidisciplinary team that involved academics, postgraduates and undergraduates from business, computer science, electrical engineering, and mathematics. Robot soccer was initiated at the University of Technology, Sydney as a high-tech high-fun-factor project.

The goal of UTS Unleashed! for 2003 was to build an agile and highly competitive team, and to qualify for RoboCup 2004 by being amongst the top teams at RoboCup 2003. We achieved our objective and can now focus on our longer term goals and aspirations which include building a robot soccer team that *knows* it is playing soccer, and exploring novel knowledge representation solutions to managing complex information in dynamic environments [1, 2].

In this paper we describe the overall architecture and the individual components of the UTS Unleashed! Robot Soccer System. The architecture implements a *sense-think-act-cycle* that relies on modules dedicated to the main functions, namely *locomotion, sensory information management, localisation, actions and behaviours*.

The power of the UTS Unleashed! Robot Soccer System stemmed from (i) efficient and robust vision and locomotion modules, (ii) effective localisation and world modeling modules, (iii) an efficient ball chaser behaviour which formed the backbone of almost all the soccer behaviours, (iv) soccer playing strategies based on adaptive positional play, and (v) fundamental collaborative team strategies.

We describe the overall architecture of the UTS Unleashed! Robot Soccer System in section 2, and in sections 3 through 6 we describe each of the system modules. We conclude with a summary of our research interests in section 7.

2 The System Architecture

Our system is still under construction, and in this section we will describe the design and architecture used for 2003 rather than outline our plans for the future. Our architecture was designed with two principal aims in mind. First we wanted to create a simple framework that was easily extensible, and that would allow different implementations of core modules to be interchanged so that a *plug and play* approach to module development could be supported. In this way key soccer systems and subsystems can be interchanged with different implementations by recompiling, and then tested during actual robot soccer matches. This allows different versions of the same module to be put through rigorous side-by-side tests, while at the system level different combinations of modules could be trialed. Second, we wanted to have system parameters and constants used for calibration of the system stored in a single location, so as to make calibration and testing of the system as efficient as possible.

The four main functional modules are:

Locomotion: controls the AIBO's effectors for the purpose of walking, kicking;

Vision: recognizes physical objects from YUV images;

Sound: allows communication between AIBOs by playing sequences of tones;

Localisation: models the position and heading of relevant objects, such as a robot, the robot's team-mates, opposition players, and the ball;

Messenger: allows communication over the wireless network.

Actions and Behaviour: supports high level decision making.

3 Locomotion

All locomotive movement of the robots is calculated and controlled by the Unleashed! Walk Engine. Being a new team, we designed and built the engine from scratch. Our design aimed for flexibility and extensibility since the locomotion engine is part of the basic infrastructure which will be used by future generations of team members. Once the foundation of locomotion was complete, speed and stability became of main priority.

The robots walk with the common bent fore-elbow stance adopted by most teams in the legged-league since it was first introduced [11]. The engine uses a static gait for all walking motions on the field. This involves the two diagonally paired legs moving synchronously, i.e. the front left paw moves in sync with the back right paw while the other pair moves out of phase by half a step as illustrated in Figure 1.

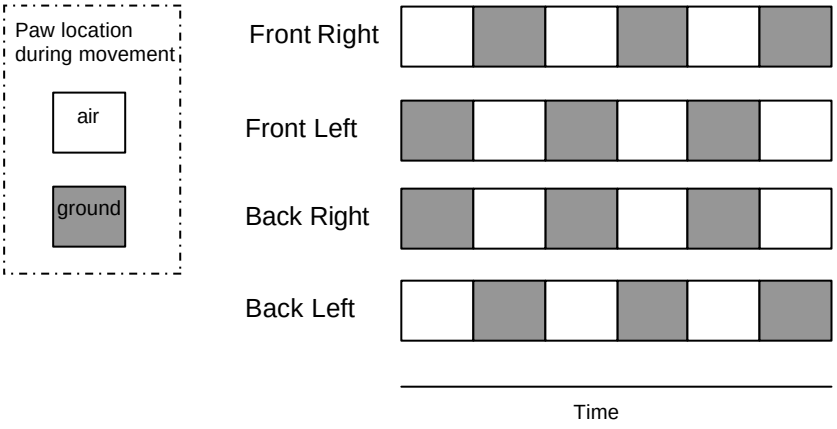

Figure 1 – Static gait

Each paw follows a particular locus and can only be interrupted at two specific points as shown in Figure 2. A full step is when the paw moves around the locus and returns to its initial position and a half step is when the paw moves from one interruptible position to the other.

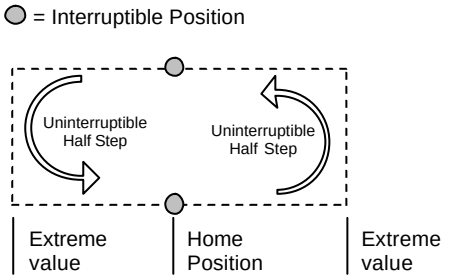

Figure 2 – Rectangular locus in detail

This simple yet effective gait allows for the best combination of speed and stability as there are always two legs in contact with the ground at any given time. It is sometimes required that the robots move to a stable position before performing an action such as a kick. This stable position occurs when all four paws are situated at the home positions of their corresponding loci. The home position is the interruptible position located on the ground, which lies between the two extreme locus values as shown in Figure 2. Controlling the walk stance, direction and speed requires five input parameters. These inputs are determined by an external module, which commands the Unleashed! Engine to complete an action. The input parameters are: *type of walk*, *forward motion*, *strafe* (side-wards) *motion*, *turning motion*, and *speed*. Each walk type corresponds to a particular set of unique parameters. The path that each foot takes when in the air depends on the particular locus used. The locus path guides the robots paw through the air in a particular pattern and along the ground. Several loci were trialed with different walk types and tested for speed and stability.

3.1 Actions

Actions are an essential part of robot soccer. We use actions to get up after falling down or to dance after scoring a goal. Furthermore, we can use actions to develop kicks, which are used to score goals, to push the ball up the field and to pass the ball between robots.

Kicks can be used in a wide variety of situations, so it is necessary to define each kind of situation and develop a kick that is both consistent and fast for that situation. For example, when kicking towards a goal, it would be advantageous to have a number of kicks to choose from, so that we can choose the appropriate angle for each kick. It would also be advantageous to be able to choose the strength needed for each kick, so that the ball will travel the required distance.

When developing actions, there are three main criteria: *power*, *speed*, and *consistency*.

Actions can be broken down into two main categories: Non-Kicking Actions and Kicking Actions. Non-kicking actions include actions such as the grab and the getup actions. Kicking actions include both forward and side kicks. Several example actions are described below.

Grab: The grab is an extremely useful action if you want to execute kicks consistently and also eliminate false kick triggers. Kicks executed after grabbing have proven much more successful than those that do not include an initial ball grab. Unfortunately, grabbing takes time, which is an important factor in deciding to use this kick in any situation during a game.

Paw Kick with Grab: Difficulties can arise if an AIBO does not grab and secure the ball before executing the paw kick. A paw kick with grab proved quite effective and was the main kick used for the Australian Open 2003.

Smack Down: The smack down is a powerful, length-of-field kick that uses the AIBO's weight for power. The AIBO's weight is first transferred to the back legs and then the AIBO falls on top of the ball using its front arms to support its fall. This kick proved very powerful and consistent; however it takes a significant amount of time

compared to most kicks. This means that during game play, opposition players can assault the robot before the kick could be executed. The Smack Down kick is most suitable for the goalie, when the ball needs to be cleared away and there aren't any opponent robots nearby. This kick should be used with care because it strains robot motors, and could easily backfire by ricocheting off a robot for example and lead to a home goal.

Head Kick Left and Right: This head kick uses the head to push the ball at a 90-degree angle either left or right. More power can be delivered by not only going from one side to the other with the head, but also by swooping down at an angle. The head kick we used had a head action that went from a top right position to a bottom left position for a left head kick, and vice versa for a right head kick. CMU also uses the AIBO's body weight to sway from side to side to get more power out of the kick.

Throw Left and Right: The throw is a kick that locks the ball between one arm and the head, and then throws the ball at a 70-degree angle. Variations in the timing of positions can allow the kick to go from 45 degrees to 90 degrees.

Spin Kick: The AIBO first turns with the ball by using its head to hold the ball down and then lifts its head up to release the ball. It is a fast kick because it does not require a grab before it is executed, however it heavily relies on localisation.

4 Vision

The task of the vision system is to receive each raw camera frame and produce a list of objects that have been identified in that vision frame. These objects must have properties that allow the calculation of their positions relative to the robot. Raw vision events arrive from the operating system twenty five times every second and are handled in a special method in the main OPEN-R object. The event object that is the argument passed to this method contains the raw vision frame. This data (or more conveniently, simply the event object) is passed to the vision system, which processes the raw data and makes available the objects that have been identified in the current frame.

Each raw vision frame is 176 pixels high and 144 pixels wide, with the horizontal and vertical zeroes in the top left corner. The logical distance from the camera's focal point to the screen is 158.4 pixels, giving the robot a very limited viewing angle both horizontally and vertically. In addition, image quality is low, and while altering the camera parameters using calls to OPEN-R libraries (white balance, gain and shutter speed) can produce a noticeable improvement in colour range and distinction, the quality of the images is always poor. In particular, adjacent colours blur into each other and movement of the robot's head results in skewed and stretched shapes. Thankfully, the fish-eye effect apparent in the lens does not cause many problems, although it does degrade the image quality in the corners of a camera image. Despite these technical limitations, it is possible for the robots to actively use their vision to play fast, dynamic soccer and for the vision system to find and identify sane, surprisingly accurately described objects gleaned from the vision frames.

4.1 Colour Classification

The entry point to colour classification is an array of raw pixel data from the robot's digital camera. The exit point is an array of symbolic colours.

The first step in dealing with each vision frame is to classify each pixel in the grid into the colours that the robot is interested in and can recognise. This must be performed pixel by pixel, and the most efficient way to do this is to classify the entire image immediately. Because of the large number of pixels, this step can be the most time consuming and must involve the minimum of processing. The most common way to classify pixels is to use a lookup table.

In the raw data from the digital camera, each pixel is represented by a Y, U and V value combination. The YUV colour-space is based on the *luminance* and the *chrominance* characteristics of colours. These are discrete values, a fact that allows a processing saving manoeuvre. Imagine a 3-dimensional array with Y, U and V axes. This array holds colours. All that is required to determine the colour of each pixel is to use its Y, U and V values as indices in this array. This array is referred to as the lookup table, and is static and merely needs to be established once, during start up. The array is loaded from a file on the robot's memory stick.

The colours represented are those of interest in the robot soccer domain. i.e. the colour of the ball, beacons, goals, robots, and walls. An "unknown" symbolic colour label is used to represent all the Y-U-V combinations that the robot should ignore, e.g. *purple*.

4.2 Basic Blob Formation

The entry point to basic blob formation is an array of symbolic colours (a classified camera image). The exit point is a list of solid (i.e. 4-connected, meaning horizontally or vertically adjacent pixels) regions of these colours having minimum and maximum x and y extents, and a pixel count.

The most efficient method of forming basic blobs is *run length encoding*. A run is a horizontal line of pixels, all of the same colour. First, examine each horizontal line (stating at the top) in the colour-classified image from left to right, and on each line build a list of runs. Each run is given a line (i.e. y) value, an x starting point, an x end point, a colour and a label that uniquely identifies 4-connected regions of the same colour. Labels have an order typically a numeric order. Any runs that are 4-connected by one or more pixels when their formation ends are given the same label. Some runs that are ultimately 4-connected will be given different labels in this process. For example, imagine two runs of the same colour on the same line with a gap between them and a third run of this colour spanning the entire line underneath. When the first line is processed its two runs will be given two different labels because they are not touching. When the second line is processed its run overlaps with both on the line above and so must be given then same label as one of them. This label must be the label of the leftmost run on the line above – this is important to the second part of *run* formation.

Once this first pass of run formation ends in the lower right corner of the image, a backward pass is made (i.e. starting on the bottom line and moving right to left). This time the runs are examined, not the image. The parent label of each label is recorded –

this is the first label assigned to a 4-connected region of colour before any additional labels were assigned to runs in this region. It is possible for this to go through multiple levels of parenting. For example, consider a number of blue runs. Some are labelled '1', some '2', some '3' and others '4'. Those labelled '1', '2' and '3' are ultimately all connected to each other. The parent label of all three is '1' and the parent label of the unconnected '4' is '4'. In the backward pass these parenting relationships are recorded and updated, and runs are re-labelled. It is implicit in the algorithm that strict top to bottom, left to right parsing in the first pass, followed by the reverse in the second pass produces correct labeling.

4.3 Advanced Blobs

The entry point for creating advanced blobs is a list of basic blobs. The exit point is a list of sanity-checked blobs, possibly with additional perimeter properties.

Blobs must be checked because occasionally things will be postulated by processing pixels colours that do not match configurations in the real world. These typically fall into two categories; colour hallucinations and shape distortions. For example, as a result of their similarity yellow and red blobs may be seen within the orange soccer ball and orange blobs may be seen within the yellow goal. Blobs that fail the sanity tests are discarded.

Blob amalgamation is an important part of the vision system. Quite often, blobs of colour that are actually part if the same object will not be 4-connected and may be entirely separated. They can be logically joined into a larger blob encompassing both regions and the space between. For this to occur the two blobs must have the same colour and overlapping extents (or very close) in both *x* and *y* directions. In addition, specifying that their combined pixel counts must cover a reasonable percentage of their combined bounding box is sensible. If basic blob formation has been performed properly and blob joining is not too aggressive then this will not result in small regions of noise being combined into large regions of legitimate-appearing but harmful blobs.

Perimeters are a useful method of calculating extended information about blobs. It is quite possible to produce reasonable results without calculating blob perimeters at all, but they do allow far more accurate and consistent results. Once a perimeter has been established a more accurate blob area can be calculated as a set of triangles and the shape of the blob can be examined. For example, beacons have a rectangular shape with a known height and width ratio, and a beacon blob that is too short for its width can have its properties corrected. Note – when deciding what is vertical and what is horizontal (e.g. beacon height and width) is important to take into account how much the image is rolled by the position of the head. It is a simple calculation to rotate points around the centre of the image and inexpensive for the small number of points involved in blobs.

4.4 Objects

The entry point for objects is a list of advanced blobs and the exit point is a list of real-world objects at calculated distances and relative positions.

Each blob is transformed into an object. Some objects are discarded and an image quality confidence value is assigned to each object. Beacon objects are composed of two blobs of different colours. Image quality is higher when the blob is closer to the centre of the image and larger.

Distance is simple to calculate – the perceived dimensions of a blob (e.g. height) diminish proportionally to the distance of the corresponding object from the camera.

Relative position to the camera is also simple to calculate. Imagine the image presented by the camera in the *x-y plane* and the focal point of the camera sitting on the *z-axis* 158.4 pixels away from the screen. Project the blob's centre co-ordinates vertically and horizontally onto the *x* and *y* axes. Two triangles are formed – a horizontal triangle between the focal point, the projection onto the *x* axis and the origin, and a vertical triangle between the focal point, the projection onto the *y* axis and the origin. The angles at the focal point are the horizontal and vertical angles (in the camera co-ordinate system) from the line of sight passing from the focal point through the centre of the image. The lengths of all three sides of both triangles are known. Because the real-world distance to object has been calculated, the relative position of the object (in real-world units but camera co-ordinate system) can be calculated easily, using similar triangles.

5 Localisation

A successful game of soccer revolving around cooperative play and strategy is highly dependent upon the player being able to determine its pose on the field. The ability of a mobile robot to determine its coordinates and its orientation in a global coordinate system is called *Self-Localisation*. This estimate of the robot's pose with respect to its environment is deemed to be the most important problem in autonomous robot navigation [5].

The problem of localisation presents itself in various guises. There is the simple and exhaustively researched problem of *position tracking* in which the robot knows about its initial position and has to compensate for errors in odometry as it moves. Algorithms for position tracking often make restrictive assumptions on the size of the error and the shape of the robot's uncertainty, as required by a range of existing localisation algorithms [3]. Then there is the more challenging task of *global localisation* in which the robot has no knowledge of its initial state and has to determine its pose from scratch. The global localisation problem is more difficult since the errors in the robot's estimate cannot be assumed to be small. Consequently the robot should be able to handle multiple, distinct hypothesis [4, 12, 18]. Finally there is the more difficult kidnapped robot problem [5, 6] in which a well localised robot is teleported to another location without being told. This problem is often used to test a robot's recovery from catastrophic localisation failures. These problems are made particularly difficult in a dynamic environment where there exist chances of corrupt sensor measurements.

In a typical game of robot soccer it is hard to determine the initial pose of the robot since this is based entirely on the kick-off situation. For example, if the opponent team is kicking then our robots are placed strategically for defense and depends a lot on the opponent teams positioning. Moreover, a player can be penalized and is then "kid-

napped" and moved to the centre of the field. Therefore the localisation algorithm has to solve the problem of global localisation and also has to be robust enough to handle random kidnappings.

No matter which way the problem of self-localisation presents itself, it is normally tackled by Estimation Theory which provides a basic set of tools for position estimation and environment modeling. These tools provide a formally sound method for combining internal and external sensor information from different sources.

UTS Unleashed! 2003 concentrated on the state-space approach to modeling dynamics systems, and focused on the discrete-time formulation of the problem. This approach is convenient for handling multivariate data and nonlinear processes and it provides a significant advantage over traditional time-series techniques [20]. So we used difference equations to model the evolution of the system with time and measurements also available at discrete times. This approach to time-series modeling focuses on the state vector of a system which contains all information required to describe the system, for example in our case the kinematics of the robot. The measurement vector represents noisy observations that are related to the state vector and in our case this corresponds to the information gathered from vision and other sensors like the infra red sensor.

In order to analyse and make inferences about a dynamic system, at least two models are required: The *system model* which describes the evolution of the state with time and secondly the *measurement model* relating the noisy measurements to the state. We assume that these models are available in probabilistic form which are ideally suited for the Bayesian approach. In the Bayesian approach to dynamic state estimation the aim is to construct the posterior probability density function of the state based on all available information, including the set of received measurements. Since the probability density function contains all statistical information it is said to be a complete solution to the estimation problem [15]. In principle an optimal estimate of the state and the accuracy of the estimate can be extracted from the probability density function.

In our localisation problem an estimate is required every time a new measurement is received which calls for a solution comprising of a recursive filter. This means that the data can be processed sequentially rather than as a batch which greatly enhances the execution of the filter algorithm since it is not necessary to store the complete data set nor is it necessary to reprocess the existing data when new measurements become available. Such a filter consists of essentially two stages: *prediction* and *update*. The prediction stage uses the system model to predict the state probability density function from one measurement time to the next. The state is usually subject to unknown disturbances which are modeled as random noise hence the prediction stage generally translates, deforms and spreads the state probability density function. The update operation uses the latest sensor measurements to modify the predicted probability density function. This is achieved using Bayes theorem.

For 2003, UTS Unleashed! used extended Kalman filters to solve the localisation problem. We converted the global localisation problem into a tracking problem by initialising the state vector and specifying a large covariance matrix to make the solu-

tion feasible. This also helped us address the issue of the kidnapped robot problem without much computational expense.

The state model of the system describes the change in the pose of the robot as it moves. The movement of the robot is controlled by control parameters, i.e. the parameters that the robot receives from the UTS Unleashed! Walk Engine. The control parameters basically consist of the three commands: *forward*, *left*, and *turn*. The forward command forced the robot to move forward in the original heading, the left command forces the robot to strafe in a direction perpendicular to the heading and the turn command forces the robot to turn on the spot in an anticlockwise direction.

For the purpose of localisation, whenever the vision module identifies a landmark it sends the distance, the bearing and elevation of the landmark to the localisation module. However these three quantities are measured from the camera point and we are trying to localise the robot by using the body center. So we have to transform the coordinates and hence the measurements from the camera point to the body point. This is achieved by matrix translations and rotations.

6 Behaviour and Decision Making

Components of our behaviour include:

- low-level skills, such as chasing, choosing kicks, and performing kicks;
- high-level planning and decision making, such as when to chase, when not to chase, where on the field to position the robot;
- team coordination.

At RoboCup 2003, UTS Unleashed! coordinated their players by allocating a fixed, unique role to each of the four robots on the team. These roles were goalie, fullback, left roamer, and right roamer. A roamer could chase the ball anywhere on the field, while the fullback was limited to the defensive half of the field. Left-sided roamers spend more time searching for the ball on the left side of the field, while right-sided roamers search for the ball on the right side of the field.

Numerous other sets of positions were trialed and experimented with before the competition, including:

- non-overlapping position/grid players
- overlapping grid/position players
- 3 roamers (all field players can chase the ball anywhere on the field, except for the defending penalty box).

A difficult task in coordinating a robot soccer team is balancing the need to be first to the ball, with the skills and knowledge available to prevent “spoiling” - interfering with a team-mate who is in ‘better position’ to kick or get to the ball. UTS Unleashed! adopted a strategy in which individual robots always chase the ball if they can see it,

and if it is in area of the field in which they should chase it. With overlapping regions, multiple players may be chasing the ball, and thus a mechanism for adjudicating possession of the ball is required, otherwise team-mates would end up fighting each other over the ball. We used "*Backing-off*" techniques to ensure that a robot backs away from the ball because it believes it may interfere with a team-mate who is already very close to, or in contact with, the ball. Backing-off was developed to avoid situations where two or more players of the same team fight over the ball. Back-off triggers using vision, sound and wireless communication were investigated and used at the competition.

7 Research Interests

Our major research interests lie in designing and developing expressive and robust representations of world models, and effective collaborative robot soccer strategies. We are also interested in building a robot soccer team that *knows* it is playing soccer! Our research objectives are twofold: first to apply intelligent techniques in the 4-legged robot league domain, and second to use this domain as a test bed to develop agent-oriented solutions for wider application in other complex domains.

The 4-legged robot league domain provides a rich problem space for us to explore deep and fundamental issues related to perception, sensor data fusion, sophisticated information management, high level deliberation, and agent-to-agent communication. For an agent to be successful, in even moderately, complex environments they require a concept modeling and management capability. Concept management is particularly important if the agent must interact and communicate with other agents to achieve its goals. In order for agents to communicate effectively they must share concepts, and attribute the same meaning to those shared concepts.

Conceptual spaces [10] provide the infrastructure for linking the perceptual and the deliberative layers, in other words they provide mechanisms for integrating information gained from sensors with higher level strategic knowledge for reasoning. The solutions that need to be developed for soccer playing robots must be fast, effective and suitable for a limited processing power on the AIBO which during the course of a soccer match must react quickly and not spend long periods of time deliberating on the best strategy, or building the best plan. To this end we have been exploring the use of game plays and case-based reasoning [13].

We use the conceptual space framework to ground symbolic representations used in high level reasoning in percepts [9, 19] where we use a computationally effective representation of a multidimensional conceptual space as our modeling tool [10]. Grounding sensory and symbolic information in robots involves linking the internal representation of the world to real world objects. Grounding is an important issue in all agents from rodents to primates. Some of our work has been directed at modeling the concept of *colour* in RGB and HSI colour spaces. From our experiments the HSI colour space turns out to be particularly useful mainly because it resembles the way human's perceive colour [8, 16, 17], and it assists the communication between humans and robots on topics related to colour concepts.

References

1. Benferhat, S., Kaci, S., Le Berre, D., Williams, M-A, Weakening Conflicting Information for Iterated Revision and Knowledge Integration, *Artificial Intelligence Journal*, in press.
2. Benferhat, S., Dubois, D., Prade, H., and Williams, M-A., A Practical Approach to Fusing Prioritized Knowledge Bases, *Studia Logica: An International Journal for Symbolic Logic*, Kluwer, 70(1): 105-130, 2002.
3. Borenstein, J., Everett, B. and Feng, L., Navigating Mobile Robots: Systems and Techniques. ed. Peters, A.K, Wellesley, MA., 1996.
4. Burgard, W., Derr, A., Fox, D. and Cremers, A.B., *Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Localisation Approach*. Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, 1998.
5. Cox, I.J., *Blanche -- An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle.* IEEE Transactions on Robotics and Automation, 7 (2), 193—204, 1991.
6. Engelson, S. and McDermott, D., *Error Correction in Mobile Robot Map Learning*. Proc. IEEE International Conference on Robotics and Automation., 2555-2560, 1992.
7. Fox, D., Burgard, W. and Thrun, S., Markov Localisation for Mobile Robots in a Dynamic Environment, *Journal Artificial Intelligence Research*., 11, 391—427, 1999..
8. Gonzalez, R. and R. Woods, R., Digital Image Processing, Addison Wesley, 1992
9. Gärdenfors, P. and Williams, M-A, *Building Rich and Grounded Robot World Models from Sensors and Knowledge Resources: A Conceptual Spaces Approach*, in the Proc. of 2nd International Sym on Autonomous Mini-Robots for Research & Edutainment, 34 – 45, 2003.
10. Gärdenfors, P. and Williams, M-A, *Reasoning about Categories in Conceptual Spaces*, in the Proceedings of the Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 385 – 392, 2001.
11. Hengst B, Ibbotson D, Pham SB, Dalgliesh JD, Lawther M, Byrnes-Preston PJ, Sammut C, *The UNSW RoboCup 2000 Sony Legged League Team*, in the Proc. of RoboCup 2000: Robot Soccer World Cup IV, Springer-Verlag, Berlin Heidelberg New York, 64-75, 2001.
12. Jensfelt, P. and Kristensen, S., *Active Global Localisation for a Mobile Robot Using Multiple Hypothesis Tracking*, in the Proceedings IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation, Stockholm, Sweden, 13—22, 1999.
13. Karol, A., Nebel, B., Stanton, C., and Williams, M-A, *Case-Based Game Play in the RoboCup Four-Legged League: Part I The Theoretical Model*, in the Proceedings of the Robot World Cup Symposium, Springer Verlag , 2003 (in press).
14. Liu, W, and Williams, M-A, *Trustworthiness of Information Sources and Information Pedigree* Intelligent Agents VIII, Series: LNCS Volume 2333, 290 – 306, Springer, 2002.
15. Maybeck, Peter S., *Stochastic Models, Estimation and Control: Volume I*, Acad Press, 1979.
16. Prasser, D., *Vision Software for a Humanoid Soccer Robot,* Honours Thesis University of Queeensland, October 2001.
17. Pratt, W., *Digital Image Processing*, Second Edition, Wiley-Interscience, New York, 1991.
18. Roumeliotis, S.I. and Bekey, G.A., Bayseian Estimation and Kalman Filtering: *A unified framework for Mobile Robot Localisation.* Proc. IEEE International Conference on Robotics and Automation (ICRA-2000), San Francisco, CA, 2985—2992, 2000.
19. Stanton, C. and Williams, M-A, *Grounding Robot Sensory and Symbolic Information using the Semantic Web*, in the Proceedings of the Robot World Cup Symposium, Springer Verlag, 2003 (in press).
20. West, M. and Harrison, J., Bayesian Forecasting and Dynamic Models. Springer Series in Statistics, 2nd ed., New York: Springer--Verlag, 1997.