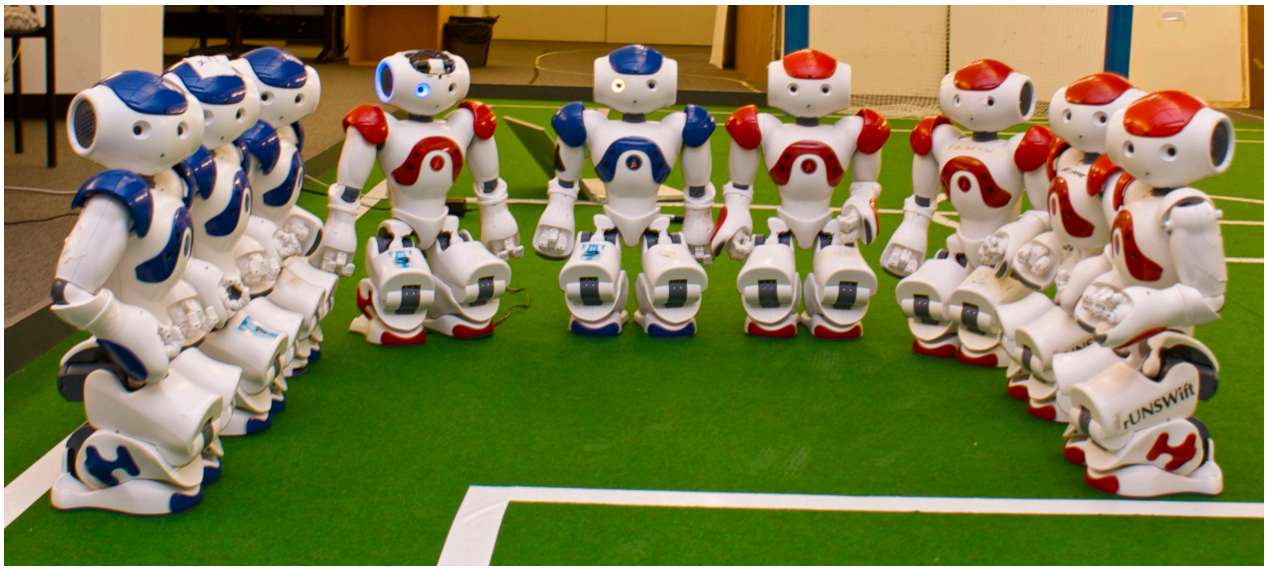
# Robocup 2009: Localisation.

Submitted for COMP3902: Special Project B

Mitchell Currie
Z3187468




The University of New South Wales.
&
ARC Centre for Autonomous Study

Assessor: Prof. Claude Sammut.
Supervisor: A/Prof. Maurice Pagnucco.

# Robocup Localisation Project 2009.

## Overview

**Synopsis of the League**
The Robocup competition since its inception in 1999 has grown substantially and has done much to advance the profile of robotics in the academic community, the commercial world, and the general public at large. The standard platform league (SPL) of the robocup competition is a relatively recent addition, being the successor to the ‘Legged League’ of the Sony Aibos. The ‘Legged League’ was the basis for many of the teams now present in the SPL, however as the league expands, more universities joined, competing for the first time, and has reached a league size of approximately 32 teams who competed in the last grand finals.

The 2009 finals for all leagues were held in Austria’s second largest city, Graz. The previous year was in Suzhou, China and was a very significant year for Robocup as it marked the introduction of the Aldebran Nao (in addition to Aibos), which will be discussed later . The 2009 competition was also important for SPL however, because it was the first year not to run the Aibo ‘Legged League’, officially making the SPL league what is arguably the most high profile league for the competition's publicity.

**Aims and purpose of this document**
This report will endeavour to document the efforts of the University of New South Wales team “rUNSWift” leading up to and including the 2009 competition. The main focus will be on localisation, as that is what I as the author was mainly responsible for, but it will of course be given in perspective to the other areas worked on, to full appreciate the undertaking, this is necessary.

# Specifications

### The Nao Platform

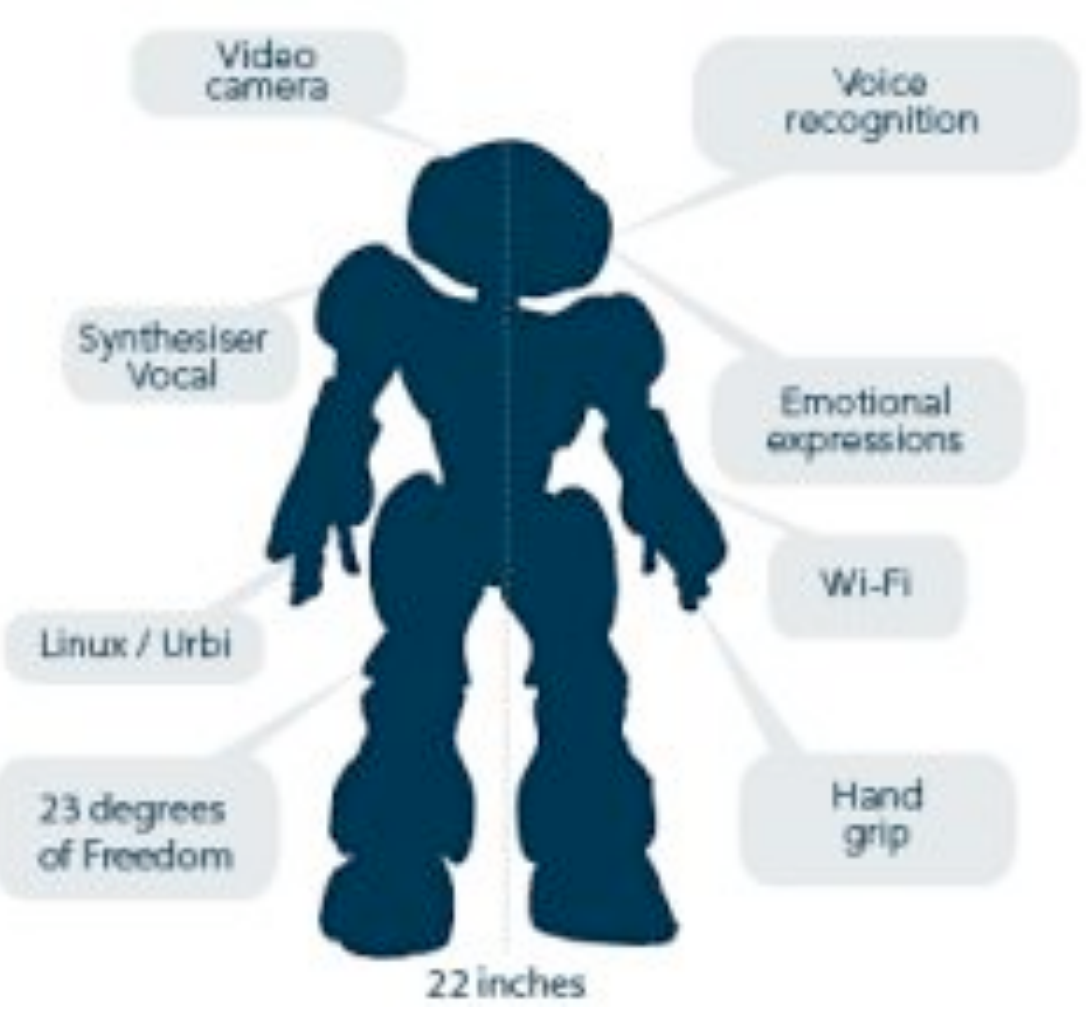



*Figure 1. 'Nao overview' (Nao Humanoid Robot by French Aldebaran, n.d.)*

Aldebaran Robotics, the French based manufacturer and designer of the Nao robot have released so far three revisions of the Nao to date, each with significantly practical implications for the Standard Platform League (SPL) participants. A brief outline of the most important changes relevant to the development this report documents.

| Revision | First appeared | Significant Changes |
|---|---|---|
| V1 | 2008:pre-finals | None, release version. |
| V2 | 2008:finals | Overall enhanced robustness and reliability. |
| V3 | 2009:finals | Second camera in chin. Motor strength to 'get-up' |
| V3+ | 2010:tentative | Faster CPU in chest |

### The Field and playing environment

A standardised playing field is used for the NAO's to compete it, the specifications of which are set out below.

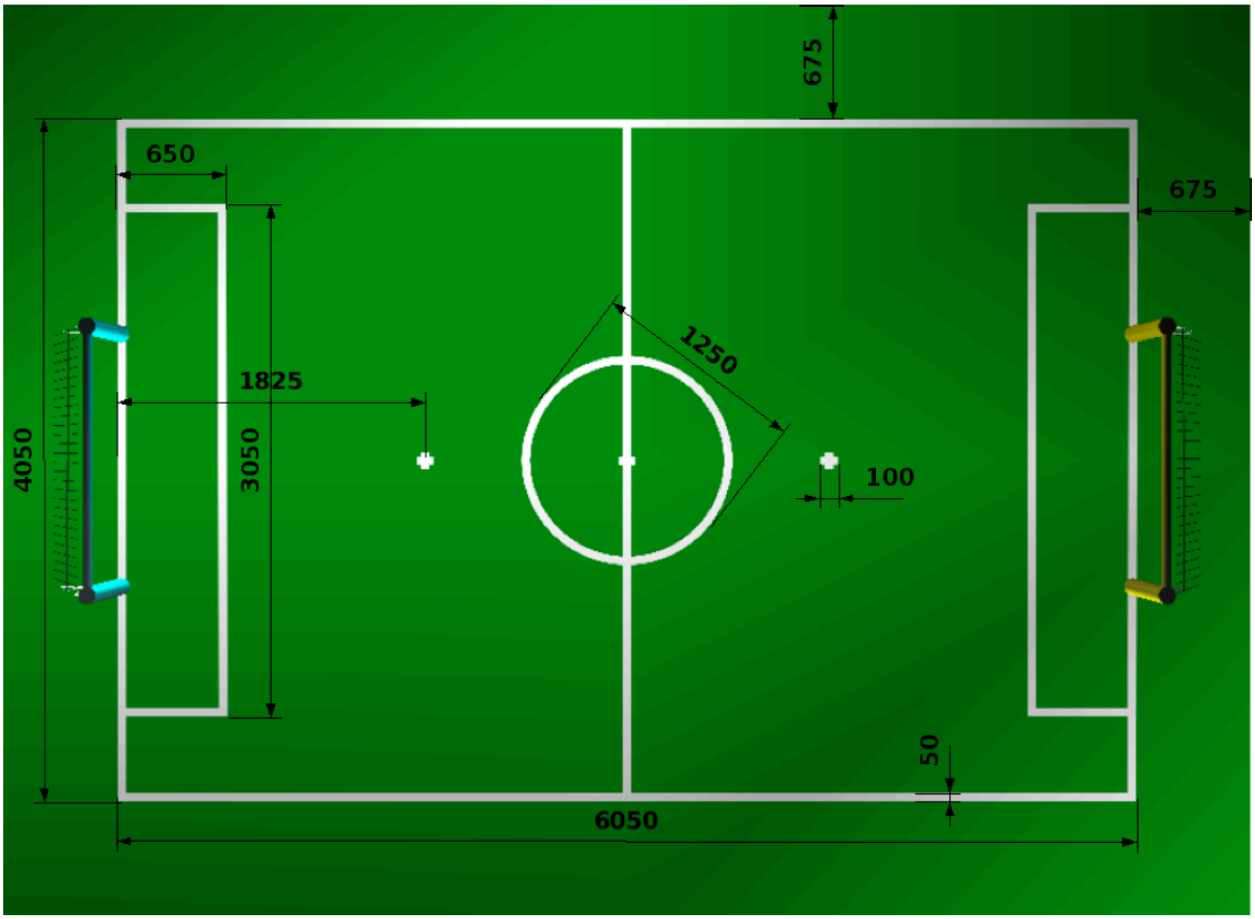



*Figure 2. 'SPL field specification' (RoboCup Standard Platform League (Nao) Rule Book, n.d.)*

# Strategy and method

### Aim

The overall goal of achieving competition success is a complex one and must be broken down to evaluate properly. Since 2008 saw the first Nao competition, insight had been gained from previous experiences, and we had developed certain criteria to achieve for what we believed was vital to successful performance at Graz.

### Design

The actual code-base that was used was inherited from the 2008 team, and a significant proportion of which was ported from earlier teams Aibo code. A consensus was reached that the code-base was to be extended upon, rather than re-written, which placed certain constraints on the design and modularity of our task implementation.
The language of implementation was virtually entirely C++,

In terms of modules the code-base can be divided into the following.

| Module | Description |
|---|---|
| Vision & Sensory | Processing of camera input into information for localisation, very CPU intensive operation. |
| Actuation & Locomotion | Enables the robot to walk and kick, as well as such rudimentary actions as panning the head/ |
| Perception & Localisation | Responds to input from vision as well as the output of locomotion to determine where the robot and objects are. |
| Behaviours | Goal-directed behaviours and Environment-driven responses involving localisation,vision and actuation. |

# Details of Localisation Component

**Requirements and Design**

The localisation that was present at the start of this year's effort was incapable of anything more than reporting (rather crudely) the heading of the opponent goal in terms of the robot, when vision has returned a match. It was apparent that in order to have a chance of successful performance in Graz we would require:
The ability to determine the robots location on the field, with an accurate bearing.
Incorporation of walking and turning data from locomotion for when visual localisation is temporarily unavailable.
Filtering of input to smoothen out spurious sightings of visual landmarks, and increase the overall reliability of the measure, whilst decreasing the dependency on perfect information from vision.

An input filter was chosen for the prediction of the robot's 'belief state' of where it currently believes it is located in the world model.

The core features of this system were the following:
•An observational confidence for vision.
•A belief state certainty.
•A rate of belief state decay every cycle. = Process update.
•The ability to use apparently unreliable data from locomotion based on walking odometer, without adversely affecting the estimation from vision.

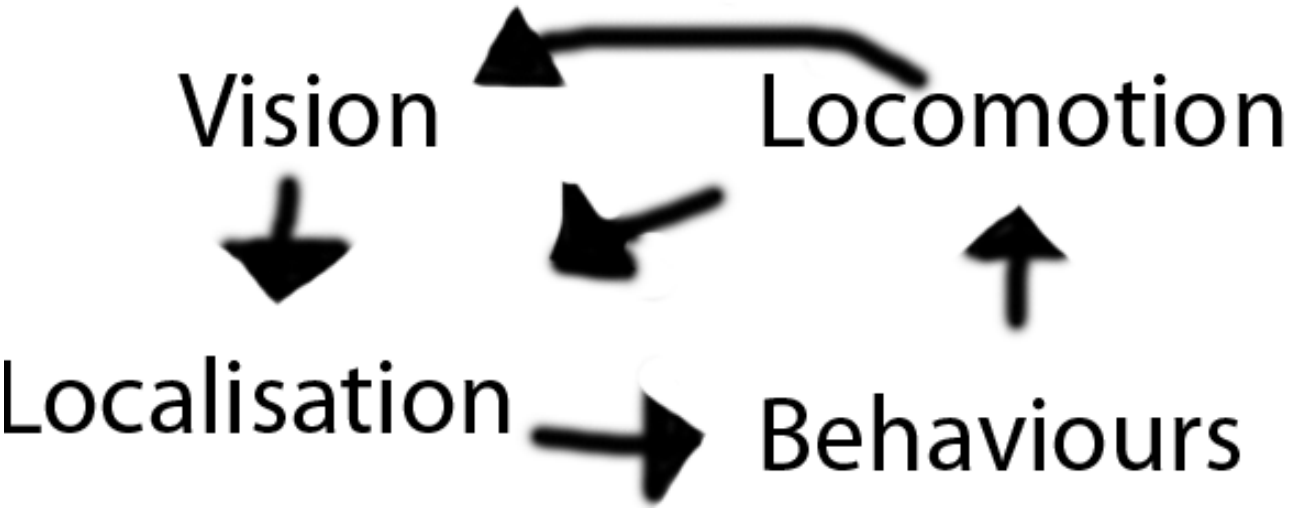



*Figure 3. Diagram of Dependency and Affect.*

**Implementation**
The system was implemented in the manner of the previous localisation module and in accordance with the modular C++ design of the project. Localisation was a class that communicated to other threads/modules via the 'blackboard' (shared, pre-defined data structure for communication).

The filter applied to the observations is one part of the localisation pipeline, but prior to that it is necessary to turn the data from vision into specific numbers and measurements... namely an estimation or observation of where the robot could be based on this data, and then it can be pushed into the filter.
The data from vision can be vary, depending on the type, and quantity of landmarks or objects visible, and this changes constantly. It may also be the case that there is no meaningful or useful information that vision has passed to localisation module.

**Filter principles**

There is a tuneable variable called “uncertaintyK” in the codebase, which is a constant, and it describes how uncertain the system is, or in other words, how quickly the system should deteriorate confidence when updating. This value was determined by empirical analysis, determining the value that slows updates enough not to jitter around the place, but a value which isn’t so low as to make the robot take a minute to update once it has been picked up and moved.

Order of operation for filter system of localisation observations

**1. Process Update**
This is an update which is performed every cycle of localisation.

*confidence = confidence/( confidence * uncertaintyK + 1.0f );*

**2. MotionUpdate**
The localisation will apply the update from the locomotion engine that reads how far the odometer reads for position changes, and simply adds this value to the belief state.
No change for any confidence is required here as this is accounted for by the process update, and will decay without vision updates.

*selfPosition = selfPosition + deltaFromOdometer;*

**3. VisionUpdate**
This update occurs when vision has reported information such as the location of landmarks including goal posts. Using any of the available methods to estimate the position of the robot, it will obtain an observsation value to perform the update, at which stage the following update is applied.

*currentLearningRate = observationConfidence/(beliefConfidence + observationConfidence)*
*beliefState = ( (1.0f - learningRate) * value) + (learningRate * observationValue);*
*newConfidence = oldConfidence+observationConfidence;*

**Calculating the distance to landmarks**

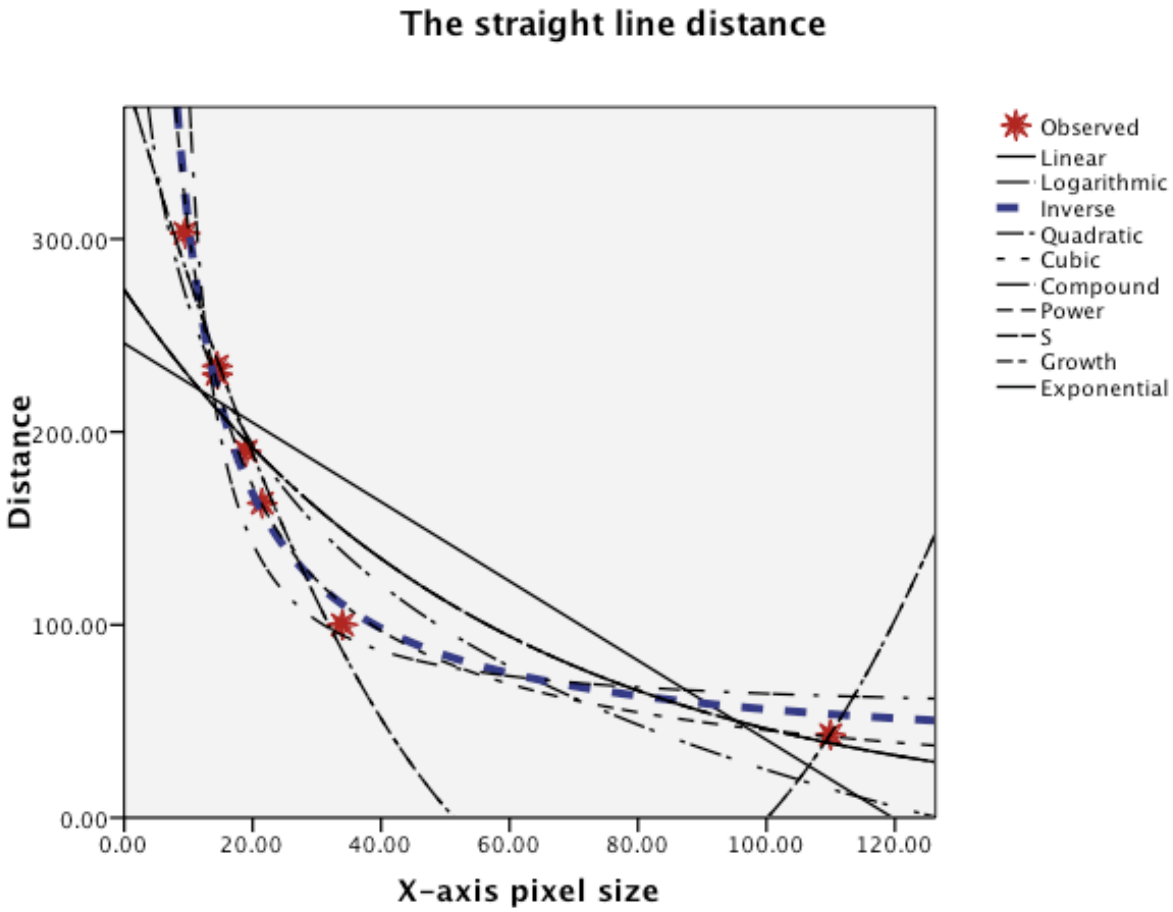



*Figure 4. Curve fitting to goal distance as a function of pixel size.*

It is necessary for the robot to be able to determine its distance from any goal post sighted in vision, as this is the most important landmark to currently consider. Fortunately for us, we can exploit the cylindrical property of these goal posts, which leads to them becoming 'beacons' as such, no matter which angle they are seen from they will always have a width proportional to their distance to the viewer (robot camera). Due to the cylindrical nature we can use the relationship between width of object and distance to it, regardless of direction, and with this a model was developed with an inverse regression function as demonstrated by figure(5) which compares possible function candidates. This function was used as an empirical basis for determining criterion validity of goal-post pixel width measurement; The measure was both valid and reliable, to the extend that consistently accurate predictions were given from consistently reliable measurements, however the unreliable measurements will be discussed below.

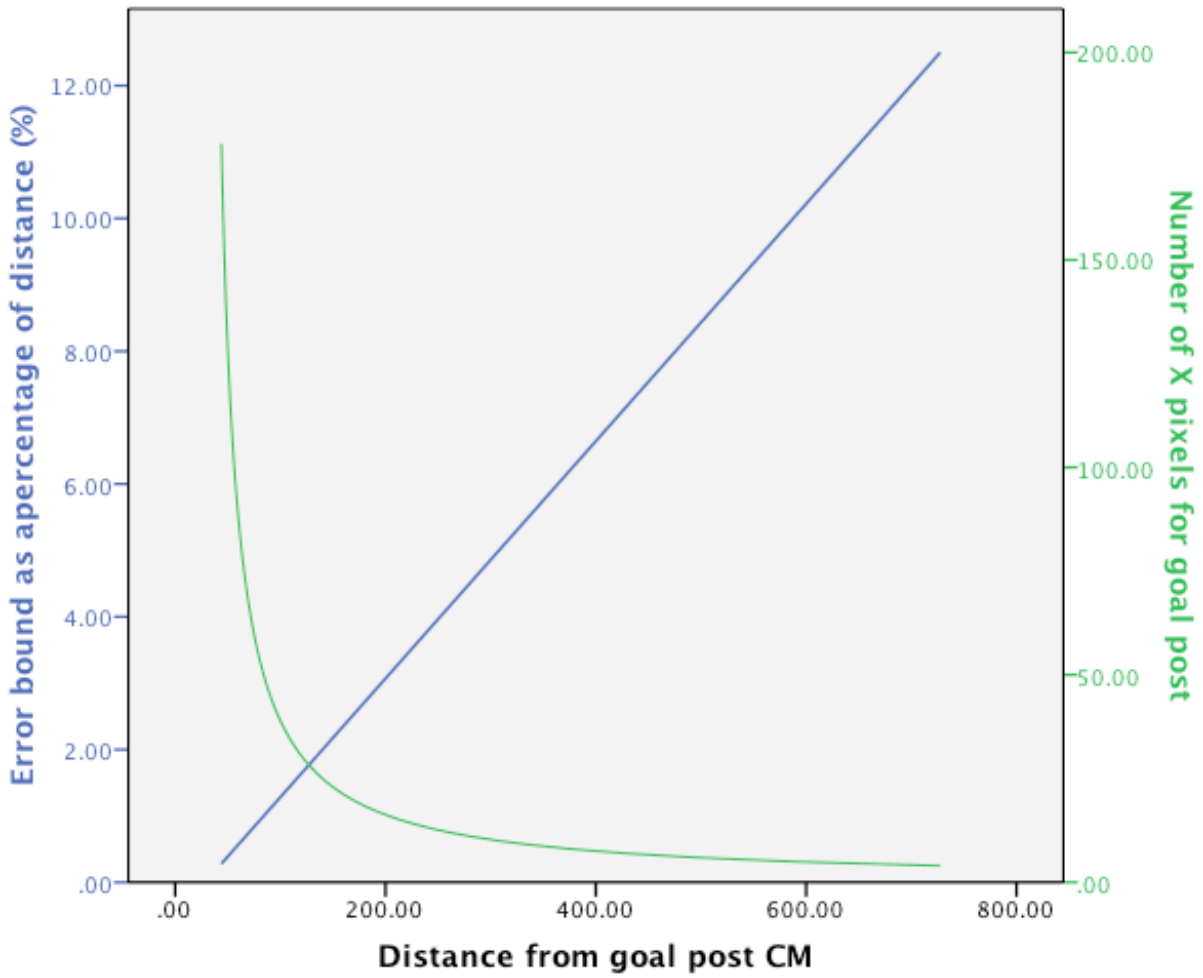



*Figure 5. The relationship between error/granularity vs distance.*

The above figure(5) shows two very important facts:
Firstly, the amount of pixels that a given goal post will imprint upon the camera's sensor is drastically reduced with the further away you are.
Secondly, the bound on error (as a percentage of your distance from the object) resulting in the camera reporting half a pixel off (due to inability resolve anymore granularity in the image) is a linear function of distance.
Essentially, all of this means, that at about 600cm (6.0,m) away from a goal, even an error of half a pixel in the camera's capture will lead to a 10% error in location... meaning the distance from the object will most likely be reported as 600cm +/- 60cm.
This error is unavoidable due to the camera's resolution at 320x240 (there might be better results at 640x480, but we did not have the resources to cope with this at the time).
There will also be other error due to noise and lighting conditions that affect the reported width of a goal, and this relationship between error and distance still holds; the previous example given of 6m away, could also be one pixel wrong in goal width (entirely common with these cameras) - which is an error of 20% (120cm), leading to a total bound of very likely error of 1.8 metres.

After it had been determined that this function worked, it was later changed to use a function involving the camera's X resolution and the Horizontal field of view to calculate the distance to a post. This is demonstrated in the following code example. Where the subtended angle is the angle that the goal post subtends in the camera's field of view.

```
float subtendedAngle = calculateSubtendedAngle(goalPostPixelWidth);
float distanceToPost = postRadiusCM/(tan(subtendedAngle/2.0f));
```

It should be worth mentioning at this point, that previous methods to calculate distances to objects were used such as projection by inverse kinematics, however these proved to be unreliable, and in comparison quite slow due to the computational requirements of these matrix operations. All in all, the rationale for the choice of the described process was that it was relatively reliable, and also simple to compute, both necessary requirements in such a performance critical environment.

**Triangulating the robot's position estimate**

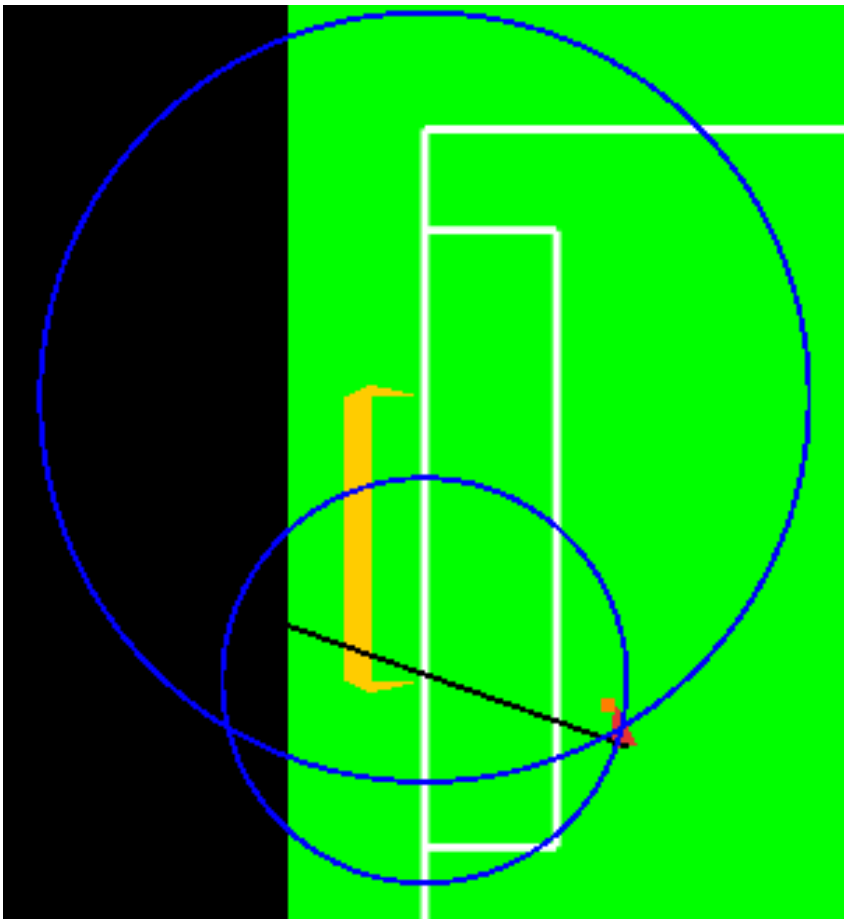


*Figure 6. The triangulation of a robots position from goal posts.*

The ideal situation is when vision module has reported sightings of both goals, in which cause we have two distances (one for each post), which we may call r1 and r2. Constructed in the above image is a plotting of the red arrowhead representing the robot and the direction his body is facing. The method used by localisation module to determine the current estimate of the robot's observed location is the following:

Solve the simultaneous system of equations for where on the field the robot must be in relation to r1 and r2 occur from the respective fixed goal post locations on the field.

Now there are two possible locations where the robot could be (one is behind the goals), but because we have knowledge of which goal post we are looking at, only the location taken on-field is used. For reasons discussed, this method can deliver satisfactory results up to about a distance of 4-metres away from the goals.
In this case it was discovered that the smallest goal was likely to produce more erroneous measurements as it was further away. A solution to the fore-mentioned problem is that using some simple trig we can calculate the distance of the smallest goal using the subtended angle of the entire goal and the distance of the closest goal. The end result is the equation necessary to give a more accurate measurement for the distance to the furthest post becomes a quadratic, as shown from code below.

```
lengthBetweenPostCentres = fabsf(closestCentre - furthestCentre);
subtendedAngle = calculateSubtendedAngle(lengthBetweenPostCentres);
distanceClosest = calculateGoalPostDistance(closestPostWidth);
a = 1.0f;
b = -2.0f*distanceClosest*cos(subtendedAngle);//Minus is essential.
bSquared = pow(b,2);
c = (pow(distanceClosest,2) - pow((float)GOAL_WIDTH,2) );
solution1 = ( -b + sqrt( bSquared - 4.0f * a * c ) ) / (2.0f * a);
```

In the event that only one goal post is sighted, it is necessary to know not only the distance to the goal post, but it would be extremely useful to have an estimate of which goal post it actually is, otherwise we do not know where to offset the distance from. Such an estimation was implemented in this system, it uses the robot belief state to calculate the distance to each possible goal from where the robot thinks it is currently, whichever goal works out to be closer to the measured distance is taken as the post observed. Once that has been resolved, we have reduced the problem to finding the closest point on the radius of d1 (the calculated distance) around the observed goal post to where the robot's belief state is currently set to. This is the estimated position for this observation.

The previous method for dealing with only one goal post isn't perfect though, if the robot is equal distance away from both goal posts, then it's just as likely to get this estimate wrong. If the localisation module detected that the goals were likely to be within the same distance to the robot as each other, then a method of using the robot's estimated heading to predict which goal post better fits, but this was only useful when we knew the goal posts were about equal distance away from the robot, since it could mess up the other method unnecessarily.

**Measures of confidence**
Whilst getting an estimate for a particular measurement or observation is crucial, it is also necessary to develop some measure of confidence (a quantitative measure of quality). There were many ideas thrown around for this, however many are problematic because if you know the confidence for them you would know the answer you're trying to find in the first place.

Eventually it was observed that the most reliable, and valid predictor of confidence was obtained from the measurement of two goal posts. The exact details of this confidence was as follows:
From the largest goal post visible, estimate the distance to the furthest one based on trigonometry with the subtended angle in the field of view. The difference between the directly measured distance, and distance calculated from the aforementioned method provides a means of confidence: take the difference between the measurements as the inverse of confidence - the larger this discrepancy, the less confident we are.
Unfortunately it was not so possible to develop a measure of confidence for when one goal-post only was seen, there are too many degrees of freedom in the unknowns in this situation, and ambiguity constrains us to simply counting the number of scan lines or pieces of evidence for this goal.

As it was previously mentioned, the goal posts are represented by far too few pixels in the image to be useful when the distance exceeds about 4 metres or so. A solution to this was to look at both goal posts and treat them as one landmark, essentially using the same sort of function based on subtended angles and ratios to calculate how far the robot is likely to be from this large super-post (which has many more pixels now). This produces an accurate distance measurement, however the only problem is that we only have one distance, which places us somewhere on a circle, it's the same situation as one goal post only being visible, and so we take the closest point to this (what is actually an) ellipsoid shape.

**Tools to aid process**
One of the key requirements in the design of a system such as this is visibility. That is, visibility of the system and the transparency to see what it is doing, because a system such as this may produce results that arise because of many different problems, so to understand what is occurring in the process pipelines I created some simple but very useful aides.

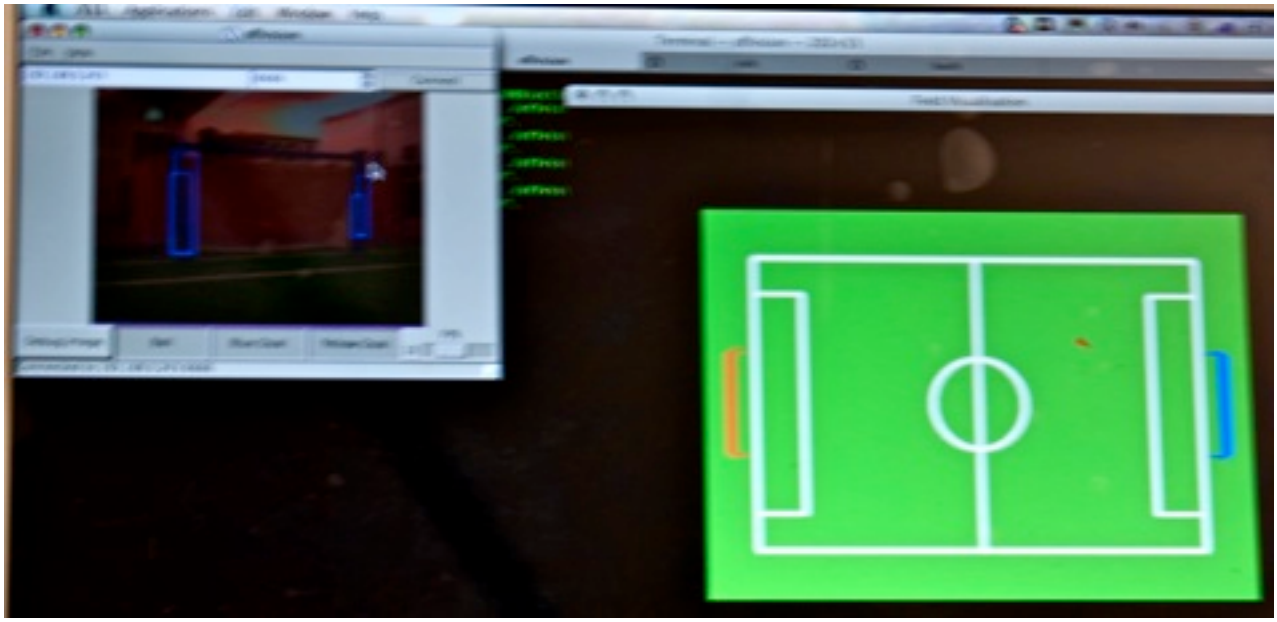


*Figure 7. Photo of actual tools in usage.*

Figure(7) shows the set-up used in this year's project to debug and verify localisation module. The key tools are ability to stream video (which is a vision utility) and the ability to see where the robot's belief state location and orientation are (via TCP streaming with a field diagram), it is much clearer to use visual aids than spools of numeric output. This aides in not just tuning the system, but working out where any possible bugs or shortfalls are, by allowing observation in actual game-play situations.

# Results and Observation

**Actual performance**

The localisation module is just one part of the nao robocup framework, and its performance is inter-linked with performance of other modules, so the best observation of how well it performed was during the finals in Graz.
Whilst much was lacking from the desired functionality, there was ability to have a reasonable localisation of the robot at most times on the field.
On average it took a couple of seconds for the robot to localise where it was properly on the field from start, which was not such an issue because there is time when the robots are in set before the game.

The filter model for this robot was also quite successful in dealing with the robot being displaced (by penalty or game-reset perhaps), as it took only a couple of seconds for it to adjust its belief state to where it actually was (providing it received some observations during this time).

Eventually the behaviours model had interacted better with localisation, it had became evident that the confidence of localisation was important for behaviours, localisation could not always deal with whatever the robot was doing at the time as sufficient input for localising... sometimes behaviours needed to help ensure good conditions for optimal localisation such as slowing the movement, or looking around for goals.

As a part of the challenges we entered the localisation challenge: where robots were required to walk to some arbitrarily given 3 points and signal its arrival at each point.
The attempt for this did not work at all as reliably as it had during testing, indicating that it was far too rushed and unpredictable. This challenge however, was probably the biggest indicator of what was needed for localisation to function correctly, and is recommended for all future teams to try and accomplish this - it really brings all the pieces together.

# Conclusions, Extensions and Future-work

**Shared world model**

Co-operative localisation model between agents.
Tracking the ball in a global world model will be essential for co-operative game play.

**Some more landmarks**

A useful tactic for initial localisation of the robot at kick-off. If the robot can see the ball, at initial state you know the ball is at field centre, and using the measured radius to obtain distance, can help resolve initial localisation.

**Automation & observation**
There was no time to implement this, but to aid localisation - A possibility is to use the overhead cameras in the labs to monitor and decode the detected blobs into field co-ordinates to give an exact position: useful for determining error and also when the localisation is likely to be off.

**Field Lines**
Although attempts were made to use field lines for the purpose of localisation, by the time it was necessary to have an implementation vision could not produce the necessary output and time was lost. These field lines provide invaluable means of localisation because there are more of them, and can be seen from more angles and positions than goals; especially when looking down at the ball (when goals are usually not visible).

# References

*Nao Humanoid Robot by French Aldebaran* [image] (n.d.). Retrieved August 14, 2009, from http://www.i4u.com/article6119.html

*RoboCup Standard Platform League (Nao) Rule Book* [image] (n.d.). Retrieved May 15, 2009 from *http://www.tzi.de/spl/pub/Website/Downloads/Rules2009.pdf*