Remote Control and Global Obstacle Detection for the RoboCup SPL

Benjamin Vance

bvance@cse.unsw.edu.au

August 24, 2011

Abstract

In the RoboCup SPL, autonomous robots compete in soccer matches. However, during development and testing phases, a remote-control system whereby a human operator can directly control a robot is rather useful. Hence, a remote-control system for this purpose was implemented, tested and used during said development and testing phases. Separately, a system to globally track obstacles - such as enemy robots - was also developed. This system is based upon a set of corroborated per-robot global grids.

Contents

Chapter 1

Introduction

This is a report for rUNSWift work performed in pursuit of a standard 6-unit Special Project A course (COMP3901) in Semester 1, 2010, assessment continuing to Semester 1, 2011. Assessors are Maurice Pagnucco and Bernhard Hengst.

1.1 Remote Control

During the RoboCup SPL competition itself, the robots on each team can communicate with each other, but are, for obvious reasons (the competition is all about autonomous robots), not allowed to have any information conveyed to them from an 'outside' source. Thus, a remote-control system is not directly and immediately useful in the competition itself.

However, if one takes a step back, then it becomes somewhat apparent that a remote-control system is useful for a myriad of purposes - enabling easier configuration of walk parameters for new surfaces, for instance, and in fact many other kinds of calibration as well - say one wishes to get the head at a particular angle to calibrate some aspect of the camera image settings. Further, a remote control system is useful during development in a vast array of scenarios - basically, any scene where the robots need to be in a particular place, be looking at a particular item, or perform a particular action, in order to test everything from vision tasks to behavioural reactions. Remote-control systems can also be used to aid machine learning.

Thus, a remote-control system was desired by the rUNSWift RoboCup SPL team. The system thus produced allows control of a robot in a familiar fashion (e.g. use of the WASD keys to move the robot in an intuitive way) from the already-existing rUNSWift 'Off-Nao' runtime-diagnostics / debugging tool. The system was integrated into existing infrastructure, and sends a standard serialised motion-command (including movement of both the body and the head and camera selection) to a fixed UDP port opened on the robot for just this purpose.

1.2 Global Obstacle Tracking using Grids

A RoboCup SPL match involves 4 robots on each side, often all moving at once. 7 other robots on the field apart from the current robot in question can be rather difficult to track, even with inter-robot communication from other robots on the same team providing cues about both friendly and enemy robots. Often, it is not necessary for a robot to track the positions of all other robots, even if this would be an essential situation. In particular, information about robot locations relevant to immediate goals (such as robots between the robot in question and the goal, when the robot has possession of the ball) is rightly prioritised over less immediately relevant information. Further, as robots are often moving about the field, their positions can change rapidly enough to quickly invalidate specific observation information.

With these considerations in mind, a grid-based obstacle mapping system was architectured in such as fashion soas to allow all robots to have a general idea about the state-of-affairs, whilst still prioritising information more immediately relevant to itself - and keeping other factors in mind such as errors in localisation not just in regards to the current robot, but also all the other robots on the same team.

Each robot maintains its own grid-based obstacle map in a grid kept in global coordinates, with adjustments made based on changes in odometry over a limited timeframe before being rolled into the aforementioned global grid for good, slowly decaying soas to reflect increasing uncertianity about the relevance of information due both to the constant movement of robots about the field and also errors in localisation, robot detection and odometry.

These global-grids are then distributed between all the robots on the team, and blended - each robot prioritising its own grid above others'. The grids have the purpose of informing behaviours, both on a general level ("there are lots of enemy robots to the right of the enemy goal") and a more specific level ("there's a robot moving to intercept our striker"), but not the purpose of very specific observations - say, for instance, being relied on to line up a shot on goal rather than the robot observing the goal itself more accurately determining the relative positions of the goal with respect to itself. Further, the grids can be used to inform robots uncertain about their position to rely on teammates' observations to more accurately choose a position hypothesis - this was implemented and successfully demonstrated.

Chapter 2

Remote Control

2.1 Off-Nao Interface

The Remote-Control piece was designed from the start to integrate into the existing diagnostic / visualisation aid 'Off-Nao' tool (This tool had been in existence since 2010 [3]), and as such, a tab was created containing information about the supported remote-control commands, and an option added to the existing menu-bars to turn remote-control on and off when connected to a robot.

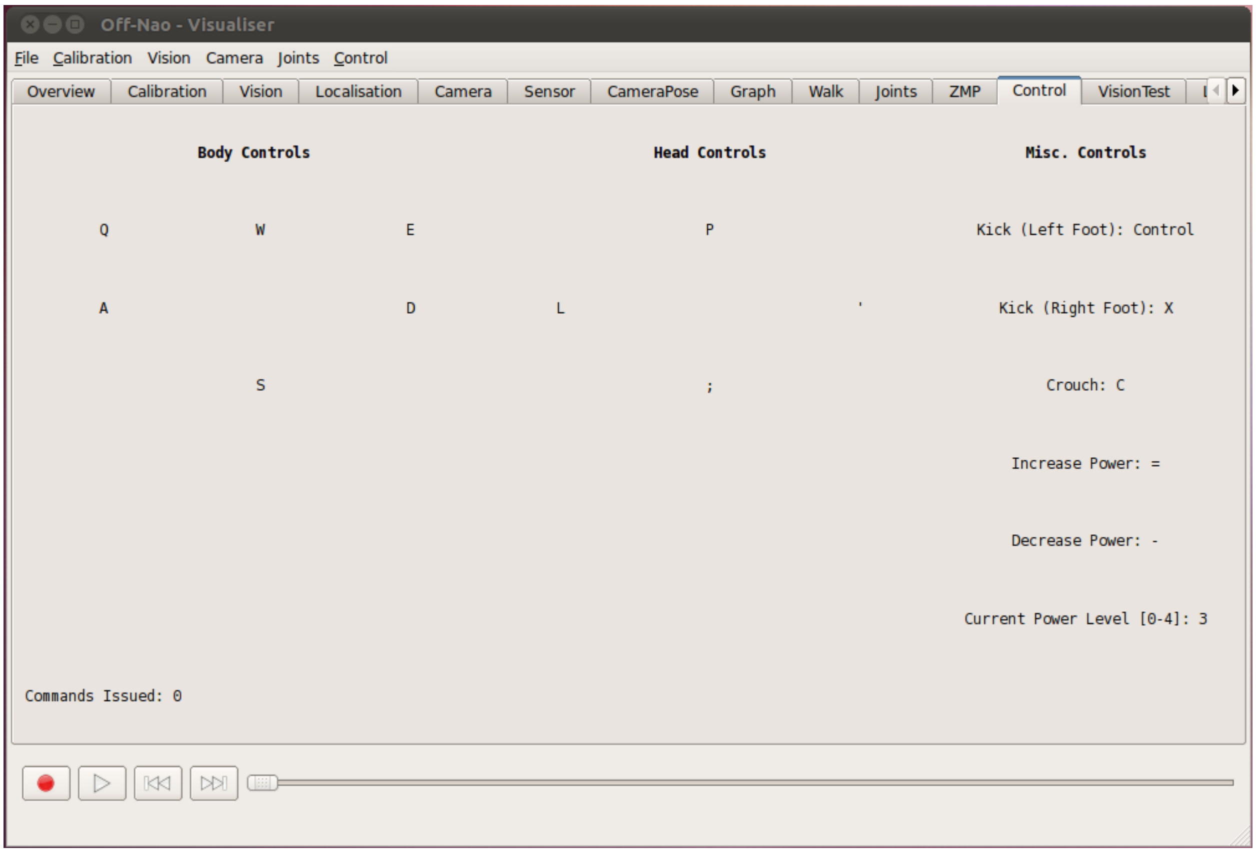
Keys are intercepted regardless of which screen the user is focused on on the Off-Nao application, allowing for the remote-control functionality to be conveniently used concurrently with vision-calibration tabs, sensor tabs or just the overview tab (usually used to view a high-framerate saliency image for basic, low-latency visual feedback). A wide variety of commands are supproted (see Fig 2.1), and as many as the robot can physically perform at once can be sent concurrently. Varying power levels / speeds can be used.

The Off-Nao application itself then creates the appropriate action-command as required, serialises said action-command, and sends it to the relevant robot over UDP port 4000. As latency is more important in this situation than accounting for packet loss, no provision was made to account for lost packets - in the worst case, one can usually just trigger the same command to be sent once more, and as explained above, the Remote-Control system is not used directly during competitions, nor is such a purpose useful.

2.2 On-Robot Components

The Remote-Control piece listens on port 4000 for UDP traffic. If a recieved packet successfully deserialises into a valid action-command object, said object is then used to override any existing action-commands being sent to the motion thread. To account for concurrency, these actions are double-buffered to ensure an action-command is not being read and written at once. Action-commands are only sent when the command changes; this is to avoid unnecessary network bandwidth use, as well as to reduce the complexity of both the sender and receiver (which is of more impact on the receeiver side due to the tigher hardware and software limitations in place).

Expected latency is at most the amount of time taken for the remote-control listener to run (it is locked when there are no incoming messages) and the motion-executor thread to run (plus network latencies, network card latencies, ethernet stack latencies, et cetera), plus reasonable thread-scheduling operation, given (in particular) the 500Mhz Geode [4] processor on-board the Nao SPL


Off-Nao - Visualiser
File Calibration Vision Camera Joints Control
Overview Calibration Vision Localisation Camera Sensor CameraPose Graph Walk Joints ZMP Control VisionTest

```
                 Body Controls                          Head Controls                 Misc. Controls

      Q               W              E                        P              Kick (Left Foot): Control

      A                              D             L                    '      Kick (Right Foot): X

                      S                                       ;                      Crouch: C

                                                                               Increase Power: =

                                                                               Decrease Power: -

                                                                           Current Power Level [0-4]: 3

Commands Issued: 0
```

Figure 2.1: The remote-control help / information tab, integrated into the existing Off-Nao tool.

robots. Given the vagaries of thread-scheduling, this it not specific. Generally, however, the remote-control system was found to run with a more than acceptable latency in practice.

Failsafe procedures are in place to ensure that the robot at no time recieves a blank motion-command, as this would cause the robot to collapse on the spot. The remote-control system itself times out after a configurable timeframe (60 seconds by default).

Chapter 3

Global Obstacle Tracking using Grids

The Obstacle Tracking Grids are expressly designed to track both allied and enemy robots, and not any other obstacles. The system has been designed with often somewhat-large input error (from the robot-detection piece) as well as changes in the state of the robots on the field over time in mind. This approach differs from previous approaches [5] used by the rUNSWift teams over the years by utilising and focusing on a grid in the global (i.e. field) coordinate space, and sharing these grids amongst the team. The robot begins by building a local Global Obstacle Tracking Grid.

3.1 The local Global Obstacle Tracking Grid

The robot considers all of the robot-observations it recieves, then, accounting for error, keeps track of robot-observations for configurable a period of time, adjusted for odometry for a period of time (by default, 30 frames). The odometry adjustment is in place to maintain accurate observations while the robot is moving rapidly, or, in particular, rotating its head. After this threshold, observations are locked-in to a particular grid square/squares. This odometry adjustment was found to greatly add to the accuracy of the produced grid. The 'base' grid containing all finalised observations is kept separately to the odometry-adjusted observations (kept in a vector of robot-relative coordinated, for transform speed and accuracy, until finalised).

The grid is subject to decay, both for odometry-adjusted observations and observations permenently locked into the grid. The decay rate is configurable, and is by default 4%, applied every 5 frames. Observations are stackable in each grid-square - that is, a single observation cannot saturate a grid-cell, and multiple observations make a grid-square 'more likely to contain an obstacle'. This grid, then, is the local Global Obstacle Tracking Grid.

Before this grid is sent out to teammates, however, it is merged with data from the robot's teammates (if available). This process is described in the next section.

3.2 The team Global Obstacle Tracking Grid

If the robot detects any Global Obstacle Tracking Grids incoming from its teammates, it then merges the grid with its own grid, on a ratio of 50

This reinforcement method is desirable as detection errors are unlikely to be of a nature whereby multiple robots have a spurious detection in roughly the same global (i.e. field-relative) position at roughly the same time, and thus multiple observations in the same area by different robots are highly likely to have substance.

Note that the result of this is that the team Global Obstacle Tracking Grid is different for each robot. The robot then sends out this team Global Obstacle Tracking Grid to its teammates over the standard team-data-synch system, which runs at roughly 5Hz [3]. This results in a slight propagation latency, which actually further strengthens the rejection of bad candiate detections, except in the case of very fast-moving robots. In the case of fast-moving robots - either allied or enemy robots - even the locally-detected locations are usually inaccurate due to detection error, or the information becoming very quickly out-of-date.

Optionally, the robot can filter out its own known location (if said location is known accurately enough) out of the grid it sends. When this feature is used, the use of team robot localisation data (in behaviours) is critical, as otherwise accurate observations of well-localised robots on the same team will not be reinforced sufficiently to support the use of this system to track teammates from these Global Obstacle Tracking Grids. This feature is especially useful due to the high distance error in robot detections - in particular, distances are often reported as being shorter than 30cm for objects a metre or so away). Further, the sonar can be used for robot detection, but this is quite inaccuracte and often less useful in a global-grid context, especially given the importance of inter-robot sharing and the intended uses of the grid filter.

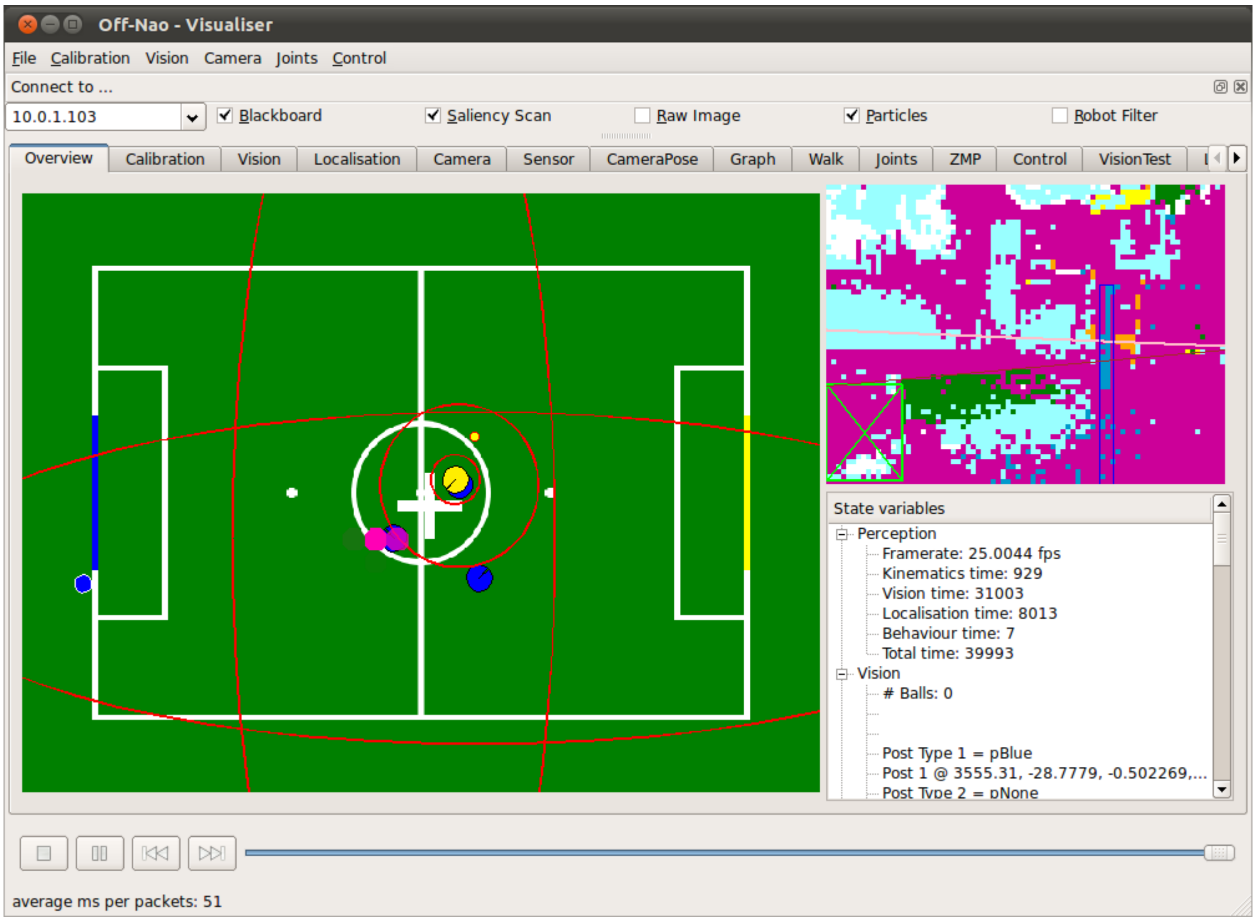
3.3 Use of the team Global Obstacle Tracking Grids to refine localisation hypotheses

A feature was developed utilising the team Global Obstacle Tracking Grids of all but the current robot to improve the robot's hypothesis of its own location (using the rUNSWift 2011 localisation system as the source of said hypotheses [1] by biasing the robot hypothesis choice based on these observations taken from the other robots on the same team. This feature weights all received grids equally. This feature is usually run with the robot filtering out its own location (as mentioned in the previous paragraph), to prevent inaccurate hypotheses from being locked.

Various metrics are used when deciding how to shift the primary hypothesis. Usually, this pools obstacle likelyhoods in the area immediately around the centre of the hypothesis, and takes the hypothesis variance into account, changing the primary hypothesis if the obstacle likelyhood is sufficiently higher than other likelyhoods to compensate for variance differences between hypothesis. Currently, this is based on a simple linear relationship, and will change to any hypothesis if the variance is sufficiently low. This method could certainly be a target for improvement in the future.

3.4 The use of Off-Nao to examine the grid/s

A feature was added to Off-Nao to display the currently connected robot's team Global Obstacle Tracking Grid. This is overlaid on top of the field display which also shows localisation / ball detection data, and so forth. Hypothesis-based robot detections (i.e. detections maintained using a particle-type filter [2]) can also be simultaneously displayed. As the obstacle likelyhood increases for a particular grid cell, it shows up more strongly in the GUI. This proved extremely useful when used for diagnostic / development purposes.


Off-Nao - Visualiser
File Calibration Vision Camera Joints Control
Connect to ...
Overview Calibration Vision Localisation Camera Sensor CameraPose Graph Walk Joints ZMP Control VisionTest

10.0.1.103
[✓] Blackboard [✓] Saliency Scan [ ] Raw Image [✓] Particles [ ] Robot Filter

State variables
Perception
Framerate: 25.0044 fps
Kinematics time: 929
Vision time: 31003
Localisation time: 8013
Behaviour time: 7
Total time: 39993
Vision
Balls: 0
Post Type 1 = pBlue
Post 1 @ 3555.31, -28.7779, -0.502269,...
Post Type 2 = pNone

average ms per packets: 51

Figure 3.1: The grid-data as it appears in Off-Nao. The pink dots indicate locations where a obstacle is likely located.

Bibliography

[1] David G. Claridge. Multi-hypothesis localisation for the nao humanoid robot in robocup spl. August 2011.

[2] Jimmy Kurniawan. Multi-modal machine-learned robot detection for robocup spl. August 2011.

[3] Adrian Ratter, Bernhard Hengst, Brad Hall, Brock White, Benjamin Vance, Claude Sammut, David Claridge, Hung Nguyen, Jayen Ashar, Maurice Pagnucco, Stuart Robinson, and Yanjin Zhu. Runswift team report 2010: Robocup standard platform league. October 2010.

[4] Aldebaran Robotics. Developer documentation for the nao (green). http://users.aldebaran-robotics.com/docs/site_en/greendoc/getting_started/motherboard.html, July 2010.

[5] Oleg Sushkov. *Robot Localisation Using a Distributed Multi-Modal Kalman Filter, and Friends.* Honours thesis, The University of New South Wales, 2006.