Plymouth Humanoids Team Description Paper for RoboCup 2012

Peter Gibbons, Phil F. Culverhouse, Guido Bugmann, Julian Tilbury, Paul Eastham, Arron Griffiths, Clare Simpson.

Centre for Robotics and Neural Systems (CRNS),
School of Computing and Mathematics,
Faculty of Science and Technology,
Plymouth University, Plymouth, UK.

Abstract. The Plymouth Humanoids are the first bipedal robot football team to be developed in the UK. This team description paper provides an overview of the latest developments of our small humanoid robots we will use to compete in this year's RoboCup competition. The robot design is based around the platform we successfully competed with at the 2011 FIRA RoboWorld Cup competition held in Kaohsiung in Taiwan. In the HuroCup we won both the sprint and the marathon events setting new world records in the process. The robot hardware uses components of the Bioloid robot platform sold by company Robotis. Adapting the existing servo frames we use RX-28 Dynamixel servos in the legs and body and AX-12 servos for the arms. We have designed and build our own low cost light weight body and rapid prototyped our own unique head design. We use the CM700 microcontroller for direct servo control, but have added an embedded linux board (the Beagleboard running Ubuntu, to provide vision processing, planning and high-level servo control. We have developed our own robot framework which supports real-time communications across a heterogeneous network. The vision system is fed from a Phillips SPC900NC.

1 Introduction

Plymouth University has been competing in the FIRA RoboWorld Cup since 1997. In the first 10 years we entered the MiroSot league until 2007 when we began to develop a team of small humanoid robots and we competed in the HuroCup league [1-4]. In 2011 we won the sprint and the marathon events setting a new world record in each. This achievement was a combination of continual development and an active outreach program to inspire our next generation robot football developers. Our current Plymouth Humanoids team comprises a group of dedicated students, staff and technicians, some

adfa, p. 1, 2011.

of whom were at FIRA2011 and are shown in Figure 1. We have developed a low-cost robust small bipedal robot platform with the aim of entering the RoboCup 2012 competition.

Fig. 1. The Plymouth Humanoids team at the FIRA RoboWorld Cup 2011.

2 Hardware

The Plymouth humanoid robot is built around the Bioloid frame system sold by the South Korean company Robotis. We specially adapted the frames to fit the Dynamixel RX-28 servos used for the legs and the body. The arms and the head pan and tilt system use the lighter AX-12 Dynamixel servos. Altogether the robot has 20 degrees of freedom. Probably the most noticeable feature of our current hardware robot design is the innovative use of the footwear on our robot. The trainers give our robot excellent grip on a variety surfaces which results in a more efficient gait by preventing the feet from slipping. The body is made from 3mm think Acrylic sheet and the head is rapid prototyped. All the components are designed to be robust, lightweight, low cost and to also minimise maintenance time. The total robot weight, including batteries, is 2.6 Kg. The robot head is protected by a 3mm wire frame which also acts as a means of safely handling the robot. The vision is captured at 30 frames per second using a Phillips SPC900NC webcam con-

nected via a USB link to a BeagleBoard OMAP3530 platform running Linux Ubuntu 10.4. The servos are controlled using a CM-700 controller sold by Robotis which uses a real-time ATMEL 2561 processor. We use the controller to update the servos at a frequency of 50 Hz. We also use it to implement control of the low level dynamic gaits parameters. It also provides feedback from both the servos and a Sparkfun 5 degrees of freedom inertial measurement unit.

Fig. 2. The Plymouth Humanoids robot Team.

3 Robot Framework

Our robot framework software has been written within a C++ template framework. This achieves the modularity of an object orientated framework but with the advantage that the compiler strips off this abstraction layer. The C++ compiles into very efficient machine code, and with appropriate operating system abstraction layers the same code runs on CM700, OMAP, MS Windows and Linux computers.

The framework consists of four abstract classes - modules, stations, buffers and blocks. Examples of Blocks include images, arrays of servo commands, or lists of corners. A Block only appears as a template parameter and can be any C++ type or class. A buffer is an abstract model of a ring buffer with ac-

cess guarded by semaphores. Each element in the ring buffer contains book keeping information a pointer to a Block. The abstraction can be instantiated with a real inter-thread ring buffer, but most instantiations are to a single ring buffer element without semaphores whose address is passed to all modules on a single thread. A module is an abstract base class with a “Tick” method. Every tick one input buffer and one output buffer are locked, the module does its processing and unlocks the buffers. For example, the input block may be an image and the output a list of corners. In general a module can lock any number of buffers and read or write to any of them.

The framework is designed for flexibility. Modules should be able to be inserted anywhere. For instance, we may start with a simple producer-consumer scheme running between two threads where one module produces a binary image and another destructively extracts a boundary. Later in the development another Module (on another thread) needs read-only access to the binary image so we have a producer-lurker-consumer model. The buffer abstraction needs another instantiation but the interface to the modules also changes. In the first version, the producer's lock method must semaphore-wait for the consumer and semaphore-notify the consumer. In the second version, it must semaphore-notify the lurker. We introduced another abstraction layer, the station, that provides an "adaptor plug" between modules and buffers. Modules interface to stations using "Lock", "Unlock" The stations interface to buffers with methods "Lock_Producer", and "Unlock_Consumer".

The scheme described above allows us to stitch modules together in different arrangements on one execution thread on one core, multiple threads on one core, or multiple threads over multiple cores that share memory. However, it does not allow us to split over different processor cards with their own memory. Previous work on the Bunny Robot platform [3] investigated a multi-card system of communicating via an Ethernet backplane. A TCP/IP layer has been incorporated allowing a buffer to be split across two or more processors using a TCP/IP bridge. As blocks can be any C++ class there is no requirement for them to be contiguous in memory. In order to copy them across the memory layout needs to be known but we abstracted this knowledge away from the TCP/IP bridge itself.

4 Vision System

The vision system uses the OpenCV v2.2 library to process the image. The image is split into two streams. One stream is converted to HSV and colour blob detection algorithms applied. This produces blobs for the ball, the goals and the players. The ball is identified by its approximate shape and colour, the centre is estimated and its approximate position on the pitch is calculated on a 2D map. The other stream is converted to grey scale where the spherical aberration is corrected to produce a "pinhole camera" image. A Canny edge detector and Hough transform are applied to detect the straight white lines on the pitch, as shown in Figure 3. The intercepts of the lines are calculated and are plotted on a very crude 3 by 4 element map of the image. A histogram of 12 bins is looked up in a database of pre-calculated views of the pitch and the ambiguities of the symmetric pitch resolved by using the last known location and the goal coloured blobs in pre-calculated regions. Once lines, ball, goals, and localisation are complete the remaining colour blobs that are close to shirt colours are examined.

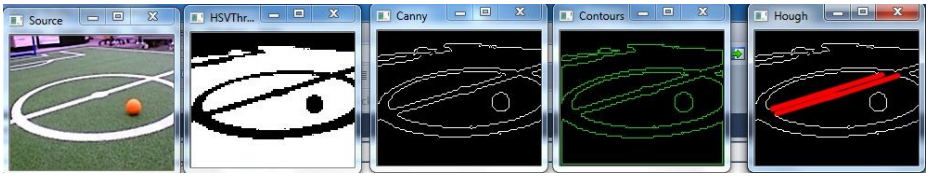


Fig. 3. Raw image with Canny edge detection and Hough transform.

5 Dynamic Gait

The dynamic gait system uses a simple two link inverse kinematic model of the legs to create a Cartesian coordinate grid for path planning [4]. The kinematic plane is segmented into a 70x150 mm grid with each point containing the pre-calculated joint angles saved as a lookup table, see figure 4. The desired X (horizontal) and Y (vertical) coordinates are calculated from a gait function then the joint angles are retrieved from the lookup table and applied to the hip, knee and foot pitch servo positions. If the hip yaw servo is rotated then the kinematic plane is also rotated. The basic path used by the dynamic gait is a sinusoid mapped to the X and Y coordinates of each foot.

This uses the gait parameters of stride amplitude and period. The coordinates of the first foot are calculated using the following function:

$$X = Stride\_Length * \cos(\frac{2\pi t}{T}) + X\_Offset \quad (1)$$

$$Y = Stride\_Height * \sin(\frac{2\pi t}{T}) + Y\_Offset \quad (2)$$

The functions create an ellipsoid with a horizontal axis of two times the stride length and a vertical axis of two times the stride height. The second foot is mapped out of phase by π radians from the first foot. Once the joint angles are calculated, the servo velocities are calculated by determining the rotation angle of each servo and then ensuring that all servos take the same time to complete the movement. This basic sinusoidal function serves as the building block for the dynamic gait, but a number of other dynamic and static parameters are also implemented.

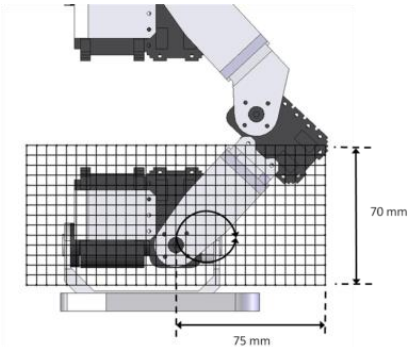
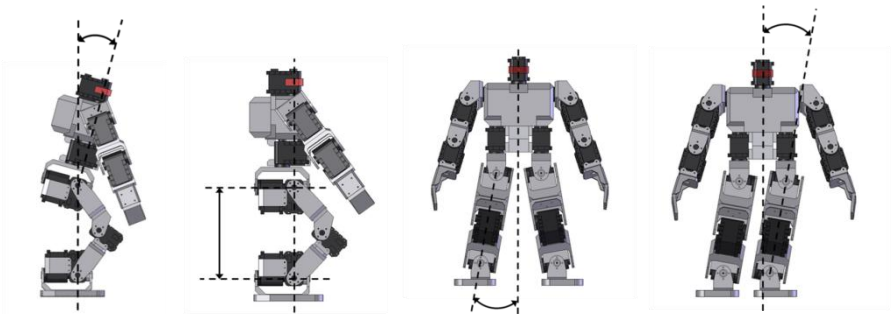


Fig. 4. Kinematic grid with curve indicating the trajectory of the foot defined by (1) & (2).

The gait uses a number of other static parameters to adjust the posture of the robot. To balance the robot in the frontal plane the robot's centre of mass is adjusted by applying a 'Tilt' offset to the hip pitch servo. The actual location on the hip servos can also be moved within the transverse plane by using a static offset in the kinematic grid. Feedback from the IMU is also used to add a dynamic offset to the hip pitch servo to help with balancing whilst walking. A static body offset is applied using the kinematic table to shorten or lengthen the height of body above the ground. To offset some of the lat-

eral stresses created at the hip roll servo a static camber offset is applied to the servo to splay the legs outwards. A small static offset is also applied the angle roll servo to ensure that the feet remain flat on the floor.



(a) (b) (c) (d)

Fig. 5. The tilt static offset inclines the centre of gravity to balance the robot (a). The body offset adjust the height of the robot body above the ground (b). The camber offset splays the legs to ensure the robot feet are positioned flat on the floor (c). Swing amplitude controls lateral movement of the robots centre of mass (d).

To improve the weight transfer of the robot body from one leg to another during walking a dynamic offset is applied in an equal opposing amount to the roll servos of the hip and ankle at the same period as the main sinusoid gait path. This causes the robot to sway back and forth continually shifting its centre of mass from one foot to the other. A dynamic gait graphical user interface has been developed to fine tune all the robot parameters as well as of as a means of testing the robots ability to walk on the different surface conditions.

References

1. Wolf, J., Hall, P., Robinson, P., Culverhouse, P.: Bioloid based Humanoid Soccer Robot Design. In: Proc. of the Second Workshop on Humanoid Soccer Robots. IEEE-RAS International Conference on Humanoid Robots, pp. 145-150. (2007).

2. Wolf, J., Vicente, A., Gibbons, P., Gardiner, N., Tilbury, J., Bugmann, G. and Culverhouse, P.: BunnyBot: Humanoid Platform for Research and Teaching. In: Proc. FIRA RoboWorld Congress, pp. 25-33. Springer Progress in Robotics (2009).

3. Culverhouse, P., Martin, S., Talloneau, R., Rodier, T., Hughes, N., Gibbons, P., Bugmann, G.: Vision Processing on the Bunny Robot Humanoid Robot. In: Proc. 4th Workshop on Hu-

manoid Soccer Robots. A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009), pp. 60-65. (2009)

4. Gibbons, P., Mason, M., Vicente, A., Bugmann, G., Culverhouse, P.: Optimisation of Dynamic Gait for Small Bipedal Robots. In: Proc. 4th Workshop on Humanoid Soccer Robots. A workshop of the 2009 IEEE-RAS Intl. Conf. On Humanoid Robots (Humanoids 2009), Paris (France), pp. 9-14. (2009)