Distributed Control of Gait for a Humanoid Robot

Gordon Wyeth and Damien Kee

School of Information Technology and Electrical Engineering
University of Queensland, St. Lucia, Queensland, 4072, Australia
{wyeth,damien}@itee.uq.edu.au

Abstract. This paper describes a walking gait for a humanoid robot with a distributed control system. The motion for the robot is calculated in real time on a central controller, and sent over CAN bus to the distributed control system. The distributed control system loosely follows the motion patterns from the central controller, while also acting to maintain stability and balance. There is no global feedback control system; the system maintains its balance by the interaction between central gait and "soft" control of the actuators. The paper illustrates a straight line walking gait and shows the interaction between gait generation and the control system. The analysis of the data shows that successful walking can be achieved without maintaining strict local joint control, and without explicit global balance coordination.

1 Introduction

Humanoid robots typically require coordinated control of a large number of joints. In most existing implementations of humanoid robots, coordination is achieved by the use of a central control computer that interfaces to all sensors and actuators providing local control of joint positions and torques as well as global control of balance and posture. This paper describes a distributed approach to control and coordination that provides local control of position and torque at each joint in a fashion that maintains global balance and posture.

1.1 Paper Overview

After a brief description of related work, the paper describes the GuRoo robot that forms the basis for the later experiments. Details of the architecture and design of the robot are followed by a description of the computing system that supports the distributed control system. The paper then describes the approach to distributed control and provides details of gait generation, including results gathered from a straight line walk.

2 Related Work

The OpenPino project [Yamasaki, 2000], utilizes a very centralized approach to humanoid control. An onboard SH2 micro-controller is responsible for taking high level commands from a PC, such as walk forward, stop etc, and converts them into position information. These positions are converted into PWM signals by a single CLPD, responsible for controlling all 26 degrees of freedom.

The University of Waseda's humanoid, WABIAN, employs a similar control system, with high level commands generated by the onboard Pentium 166Mhz computer[Yamaguchi, 1998]. The resulting velocity profiles are fed into one of two 16 channel D/A boards via an ISA bus, which supply the motor drivers the required signals to actuate the motors. This system is physically centralized with all computational equipment and motor drivers located in the torso.

Similarly, H6 from the University of Tokyo makes use of an onboard PIII 700Mhz computer to generate high level commands [Nishiwaki, 2000]. A pair of Fujitsu I/O controller, similar to WABIAN's, generates the control signals necessary for the motors. Individual motor drivers supplying the necessary power are located physically close to each motor.

2.1 The GuRoo Project

GuRoo is a 1.2 m tall, fully autonomous humanoid robot designed and built in the University of Queensland Robotics Laboratory [Wyeth, 2001]. The robot has a total mass of 34 kg, including on-board power and computation. *GuRoo* is currently capable of a number of demonstration tasks including balancing, walking, turning, crouching, shaking hands and waving. The robot has performed live demonstrations of combinations of these tasks at various robot displays.

The intended challenge task for the robot is to play a game of soccer with or against human players or other humanoid robots. To complete this challenge, the robot must be able to move freely on its two legs. Clearly, the robot must operate in a completely autonomous fashion without support harnesses or wiring tethers. The current GuRoo robot cannot withstand the impacts associated with playing soccer, but serves as an excellent platform for research into the design of balance behaviors and dynamic gait control. The location and axis of actuation of each joint can be seen in Figure 1.

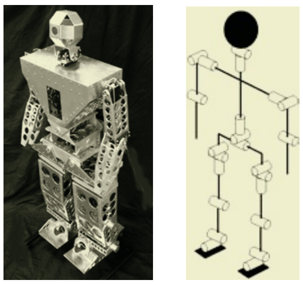


Fig. 1. The GuRoo Humanoid robot and the degrees of freedom of each joint.

2.2 Electro-Mechanical Design

The key element in driving the mechanical design has been the choice of actuator. The robot has 23 joints in total. The legs and abdomen contain 15 joints that are required to produce significant mechanical power, most generally with large torques and relatively low speeds. The other 8 joints drive the head and neck assembly, and the arms with significantly less torque and speed requirements.

The 15 high power joints all use the same motor-gearbox combination consisting of a Maxon RE 36 motor with a gearbox reduction of 156:1. The maximum continuous generated output torque is 10 Nm. Each motor is fitted with an optical encoder for position and velocity feedback. The 8 low power joints are Hi-Tec RC servo motors model HS705-MG. with rated output torque to 1.4 Nm.

The motors that drive the roll axis of the hip joints are supplemented by springs with a spring constant of 1 Nm/degree. These springs serve to counteract the natural tendency of the legs to collide, and help to generate the swaying motion that is critical to the success of the walking gait.

Power is provided by 2 × 1.5Ah 42V NiCd packs for the high power motors, and 2 x 3Ah 7.2 V NiCd battery packs for computing and servo operation. The packs are chosen to give 20 minutes of continuous operation.

2.3 Sensing

The position feedback from the encoders on the high power joints provides 867 encoder counts per degree of joint motion. In addition, each DSP can measure the current to each motor. Provision has also been made for inertial and balance sensors, as well as contact switches in the feet and in the joints.

3 Distributed Control Network

A distributed control network controls the robot, with a central computing hub that sets the goals for the robot, processes the sensor information, and provides coordination targets for the joints. The joints have their own control processors that act in groups to maintain global stability, while also operating individually to provide local motor control. The distributed system is connected by a CAN network.

3.1 Central Control

The central control of the robot derives the joint velocities required to perform the walking gait. A PIII 1.1GHz laptop currently calculates velocities for all 23 degrees of freedom in real time. The velocities are passed along a serial link to a distribution board which serves as a bridge between the serial bus and Control Area Network (CAN) Provision has been made to port this control to a Compaq IPAQ mounted on the robot to enable true autonomy, free of any tethers.

3.2 Joint Controllers

All joint controllers are implemented using a TMS320F243 Digital Signal Processor from Texas Instruments, a 16 bit DSP designed for motor control. The availability of the CAN module in this series, along with bootloader programmable internal Flash memory makes the device particularly attractive for this application. Five controller boards control the 15 high power motors, each board controlling three motors. A sixth controller board controls the eight RC servo motors. Figure 2 outlines the interaction between the various node on the control network.

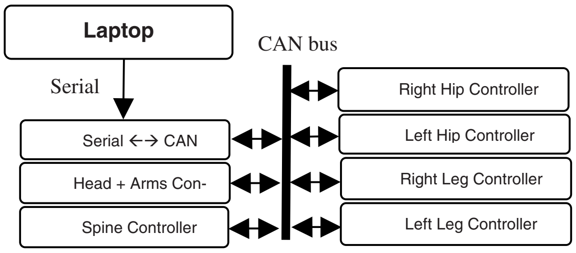


Fig. 2. Block diagram of the distributed control system.

4 Software

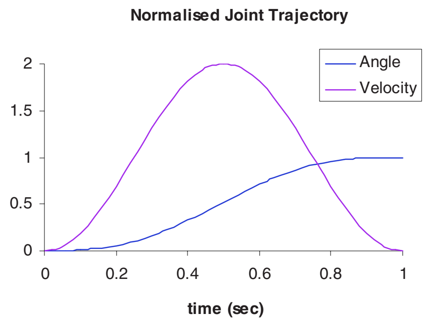
The software consists of four main entities: the global movement generation code, the local motor control, the low-level code of the robot, and the simulator. The software is organized to provide a standard interface to both the low-level code on the robot and the simulator. This means that the software developed in simulation can be simply re-compiled to operate on the real robot. Consequently, the robot employs a number of standard interface calls that are used for both the robot and the simulator including reading the encoders and setting PWM values as well as the transfer of CAN packets.

4.1 Gait Generation

The gait generation module is responsible for producing realizable trajectories for the 23 joints so that the robot can perform basic behaviors, such as standing on one leg, crouching and walking. The most important properties of the trajectories are that they are smooth and that they can be linked together in a smooth fashion. Smoothness of motion implies less disturbance to the control of other joints. Based on these criteria, a normalized joint movement as shown in Figure 3 is applied to all motor trajectories.

The trajectories are generated from a parameterized sinusoidal curve where ω is the desired joint velocity, θ is the total joint angle to move and T is the period of that movement. The trajectory contrasts with typical trajectories generated for robotic manipulators, which typically focus on smoothness of the end effector motion rather than smoothness at the joint.

Trajectories for each of the motors may be coordinated by using the same beginning time for the motion and specifying the same period for the trajectory. Trajectories may be naturally linked as the velocities all reach zero at the beginning and the end of a motion. Section 5 will illustrate how trajectories may be coordinated and linked to perform a walking operation.



$$\omega = \frac{\theta}{T}\left(1 - \cos\left(\frac{2\pi t}{T}\right)\right)$$

Fig. 3. The trajectory used for a total joint movement of 1 radian over a period of 1 second.

4.2 Joint Controller Software

There are two types of joint controller boards used in the robot – five controller boards control the fifteen high power motors and one controller controls the eight low power motors. The controller software for the low power motors is a single interrupt routine that is triggered by the arrival of a CAN packet addressed to the controller's mailbox. The routine reads the CAN mailbox for the change in position sent by the gait generation routine. The PWM duty cycle that controls the position of the RC servos is varied accordingly.

The control loop for the high power controllers has two interrupt routines. As for the low power controller, an interrupt is executed upon receipt of trajectory data in the CAN mailbox. The data is used to set the velocity setpoints for the motor control routine. There is also a periodic interrupt every 500 μs to run the motor control software. The motor control routine compares the error between velocity setpoint and the encoder reading and generates a PWM value for the motor based on a Proportional-Integral control law. The routine also checks the motor current against the current limits, and adjusts the PWM value to prevent over-current situations.

The PI control law on each joint has been hand tuned to provide both good trajectory following for typical velocity input profiles, and spring-damper model impedance to torque disturbances from gravity and the cross-coupling torques from other joints. The "soft" response of this control law to disturbance prevents torques being transmitted throughout the robot and helps to maintain global stability. The disadvantage is that the controller suffers from position error that must be accounted in the gait generation software. Section 5 illustrates how this potential liability is turned to an asset in the generation of a dynamically stable walk.

4.3 Low-Level Code

The lowest level of code on the robot provides direct access to the sensors and communication system. The level of abstraction provided by function calls at this level aids in the cross development of code between the simulator and the real robot.

4.4 Simulator

The simulator is based on the *DynaMechs* project [McMillan, 1995], with additions to simulate specific features of the robot such as the DC motors and motor drives, the RC servos, the sensors, the heterogeneous processing environment and the CAN network. These additions provide the same interface for the dynamic graphical simulation as for the joint controller and gait generation code. The parameters for the simulator are derived from the CAD models and the data sheets from known components. These parameters include the modified Denavit-Hartenberg parameters that describe the robot topology, the tensor matrices of the links and the various motor and gearbox characteristics associated with each joint. The surface data from the CAD model is also imported to the simulator for the graphical display.

For the high power DC motor joints, the simulator provides the programmer with readings from the encoders and the current sensors, based on the velocities and torques from the dynamic equations. In the case of the RC servos, the simulator updates the position of the joints based on a PD model with a limited slew rate. The programmer must supply the simulator with PWM values for the motors to provide the control. The simulator provides fake interrupts to simulate the real events that are the basis of the control software.

The simulator uses an integration step size of 500μs and updates the graphical display every 5ms of simulated time. When running on 1.5 GHz Pentium 4 under Windows 2000, the simulation updates all 23 joints at a very useable 40% of real time speed.

5 Walking

The robot can walk with a step rate of 1 Hz using a step length of 100 mm. The walk is open-loop; there is no feedback from the joint controllers to the gait generation software. The lack of global feedback, combined with the absence of a global balance sensor presents a substantial challenge in walking algorithm design.

5.1 Walking Algorithm

The robot uses a simplified version of a typical human gait. In particular, it limits the swing of the legs to prevent balance disturbance as this cannot be corrected without global balance control. In order to minimize the accelerations of the torso, head and arms (which make up 1/3 of the mass of the robot), the robot maintains a constant relative position of the torso, such that the face of the torso is always normal to the direction of travel. The allowable roll of the torso is also limited. The stabilization of the torso also reduces disturbances from gravity to the control of the leg joints.

Before walking, the robot loads each motor against gravity by performing a slight squat that introduces a 6 degree ankle pitch, with the knee and hip pitch joints set to keep the torso upright. The initial loading of the joints reduces the likelihood of backlash in the gearheads.

The walking gait commences with a side-to-side sway generated from the roll axes of the ankles and hips. The sway frequency of 0.5 Hz is sympathetic with the spring mass system formed by the ankle controllers with the mass of robot. The sway sets up the pattern of weight transfer from one foot to the other necessary to swing the legs alternately to achieve walking. At each extreme of the sway, the inertia of the upper body ensures the ZMP (Zero Moment Point) of the robot lies within the support polygon formed by the support foot, even thought the centre of mass may not. This action places requires less torque from the hip and ankle roll actuators, as the motion due to gravity brings the robot away from the extreme of each sway.

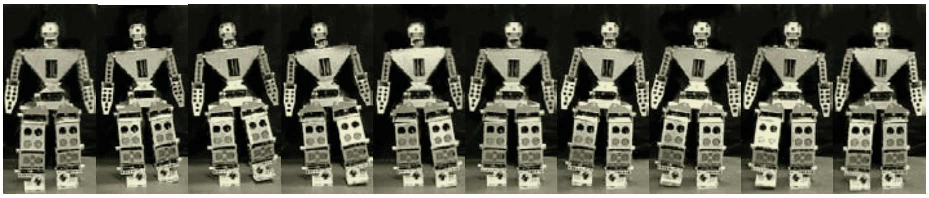


Fig. 4. Frontal view of the walking process.

Once the sway is sufficient to leave no ground reaction force on the non-supporting foot, the non-supporting leg is lifted using the pitch axes of the hip, knee and ankle. Between the lifting and lowering of the non-supporting leg, each yaw axis motor twists, so that the non-supporting leg swings forward to create the step. When the swing leg contacts the ground, the robot is dynamically stable, with the centre of mass over the supporting foot in the frontal plane, but in front of the toes in the sagittal plane. The robot then swings across to the other foot, repeating the sequence and progressing with the walk.

5.2 Analysis of Results

The motion of the robot is best analyzed by comparing the desired velocity from the gait generation module to the actual velocity at each joint. Figure 5 shows this comparison for the motion of the hip, knee and ankle in the roll, pitch and yaw axis. The graphs are initialized midway through the double support phase, with both legs in contact with the ground. The graphs comprise one second of data, describing the right leg as it moves from the double support phase, through the swing phase back to the double support phase.

At the point $t = 0.25$ s, the swing leg starts to lift and loses contact with the ground. With the hip roll axis of the swing leg no longer contributing to the support of the robot, the spring located in this axis briefly dominates the actual velocity causing the overdamped oscillation seen at hip joint at this time.

Once the foot leaves the ground the ankle roll motor switches from driving the leg from the foot, to driving the foot from the leg. This large decrease in relative inertia results in a brief increase in the magnitude of the ankle roll velocity. The foot has a relatively low inertia compared with the rest of the robot, and as such the PI controller has little trouble following the desired velocity until the foot again makes contact with the ground. The robot reaches the extreme of each sway at $t = 0.5$ s, where all motion in the roll plane ceases. The swing leg is now theoretically fully lifted, although the knee and hip pitch do not reach their desired positions until T=0.6s.

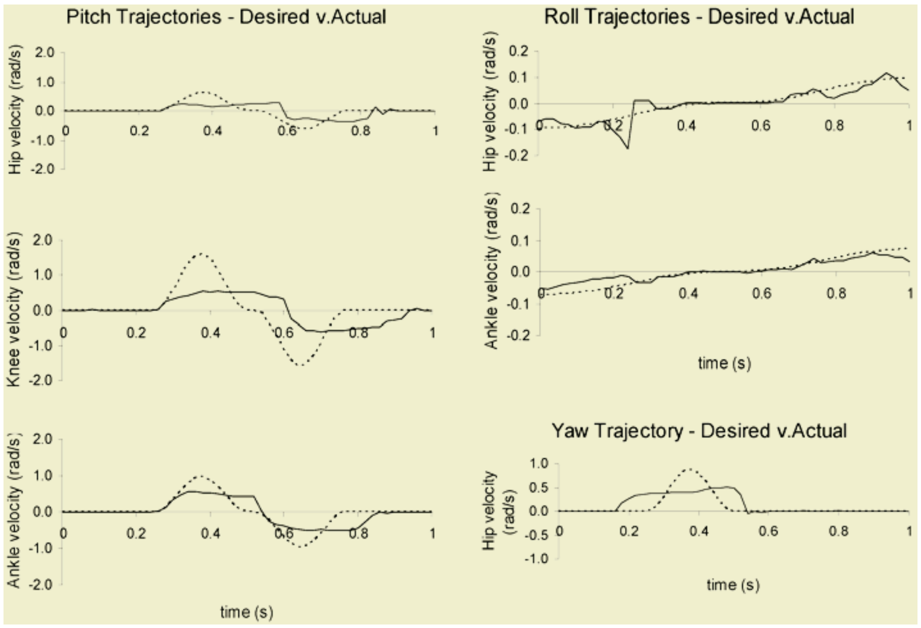


Fig. 5. Desired vs Actual joint angles for straight line walking over a 1 second period. Graphs start during the double support phase and follow the right leg through the swing phase.

The actual joint velocity profile for each pitch motor in the swing leg shows the increasing effect of gravity and leg inertia through the swing stage. The integral term in the PI controller seeks to eliminate steady state error, and as such, dominates the actual velocity, driving each pitch motor to its desired position. As the motor does not reach the maximum desired velocity, it is forced to lengthen the movement time to ensure the areas under each graph are equal. The low proportional term results in the poor tracking of the desired velocity, but enables the joint to better deal with external disturbances.

The comparison of the actual velocity with the desired velocity for each pitch motor in each leg degrades from the ankle to the knee to the hip. The ankle need only accelerate the foot, whereas the knee must accelerate the foot and lower leg. The hip pitch must accelerate the entire leg during the swing phase.

The motion of the yaw axis as the swing leg is lifted, propels the robot forward. When the yaw motion occurs on the support leg, the momentum of the robot causes the joint to overshoot its position. The swing leg is then lowered placing the robot into a double support phase. The friction of two feet against the ground and the weight of the robot on the support leg prevents the yaw axis positional error from being resolved. This provides a pre-loading of the joint that supports the motion of the next of yaw swing phase.

As the robot sways back to the other side, weight is gradually released from the supporting foot, until the torque acting on the joint overcomes the co-efficient of friction between the foot and the floor. This is not necessarily the point at which the swing leg loses contact with the ground. By time the leg has resolved this error, the joint is experiencing the yaw motion associated with the swing leg twist. As a result, the area under the curve for the hip yaw during the swing phase is greater than the desired area.

Contact with the ground is achieved at T=0.85s and once made, the robot returns to the double support phase of its gait. Both the hip and ankle of the swing leg now assist the support leg roll motors to sway the robot across to the other foot, and in the process gradually switch the roles of the support and swing leg

6 Conclusions

This paper has illustrated that a humanoid robot can walk without the need for explicit global feedback, or tightly controlled joint trajectories. By combining a group of loosely coordinated control systems that use “soft” control laws with smooth trajectory generation, the robot can use the natural dynamics of its mechanical structure to move through a gait pattern. The work in this paper shows sound walking performance that can only improve with the augmentation of global inertial sensors and feedback paths.

References

[McMillan, 1995] S. McMillan, Computational Dynamics for Robotic Systems on Land and Underwater, PhD Thesis, Ohio State University, 1995.

[Nishiwaki, 2000] K. Nishiwaki, T. Sugihara, S. Kagami, F. Kanehiro, M. Inaba and H. Inoue, Design and development of research platform for perception-action integration in humanoid robot: H6, International Conference on Intelligent Robots and Systems, IROS 2000

[Wyeth, 2001] G. Wyeth, D. Kee, M. Wagstaff, N. Brewer, J. Stirzaker, T. Cartwright, B. Bebel. Design of an Autonomous Humanoid Robot, Proceedings of the Australian Conference on Robotics and Automation (ACRA 2001), 14-15 November 2001,Sydney

[Yamaguchi, 1998] J. Yamaguchi, S. Inoue, D. Nishino and A. Takanishi, Development of a bipedal humanoid robot having antagonistic driven joints and three DOF trunk, Proceedings International Conference on Intelligent Robots and Systems, IROS 1998

[Yamasaki, 2000] F. Yamasaki, T. Matsui, T. Miyashita, and H. Kitano. PINO the Humanoid that Walk, Proceedings of First IEEE-RAS International Conf on Humanoid Robots, CDROM 2000