The Isaac project: development of an autonomous biped robot

teacher: Menga Giuseppe[1]

students: De Stefano Giovanni[1], Faconti Davide[1], Mandrile Roberto[1], Mastrodomenico Lino[1], Mura Andrea[1], Nova Diego[1], Scalabrino Luca[1].

[1]Politecnico di Torino, Corso Duca degli Abruzzi 24
10129 Torino, Italia

Abstract. This paper describes the design of a humanoid robot built in Politecnico di Torino by seven students. Mechanical architecture and control algorithms are based on a preliminary research on human biped locomotion, adapted and tested using computer simulation; the result is an unusual design that should maximize the performance. We designed dedicated electronic components and the software runs under a real time operative system to use the 100% of the on-board computer. The motors and the power supply are choosen to give a torque larger than the one needed in common walking, studied with simulation.

1 Introduction

The Isaac project had from the beginning two main goals: to build a platform to test potentialities of biped locomotion on mobile robots, and to partecipate to the seventh edition of RoboCup in Padua.
For the second reason the robot needs to be autonomous and able to compete in the football match using computer vision to recognize objects and artificial intelligence to decide how to behave.
All the team members had a different role in the development, and the paper will try to summarize their results after one year of work. Many researches on humanoid robots are going on all around the world, especially in America and Japan, so we tried to summarize the results of other groups and to understand the kind of solutions that could be the best, or the pitfalls that we could find.

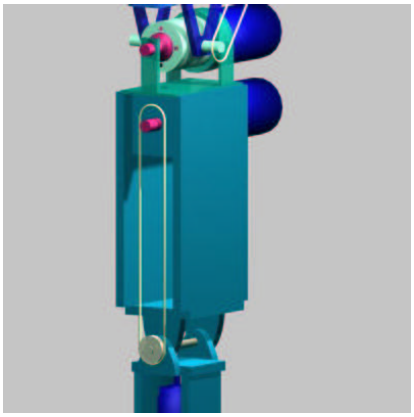
1.1 Robot overview

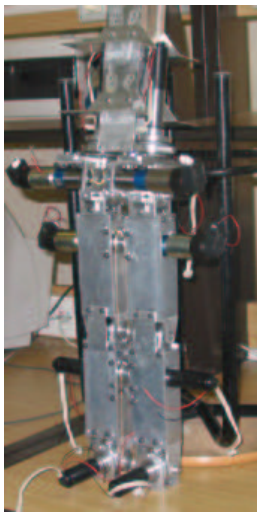
Isaac is 85 cm tall and his average weigth is 15 Kg. It has 6 degrees of freedom on each leg: this is considered the standard to provide full three-dimensional movements; the neck has two motors to rotate the camera, making possible the ball tracking in real time, the arms instead have just one degree of freedom, at least in this prototype version of the robot.
On the main computer we have mainly three software modules: artificial vision and sensor processing, game planning, and walking.
The control of the movement has two layers: at the lower level we have just a position or velocity control of the single motor; all the motor are sinchronized by the upper layer that use the information given by balance sensors to create a real time trajectory on the joints angle.

2 Mechanical architecture

2.1 Design

The design of the robot is based on biomechanic data and analisys on human being.
The optimal solution seems to keep the proportion of the human legs on the robot, in particular to have the same thigh and tibia length, and a small distance (in the frontal plane) between hip joints.
For an efficient walking, both from kinematic and energetic point of view, we will use heel and toe rolling on the ground: this oblige to have smaller feet, or at least not longer than human one.
But the most important building criteria is based on Center of Mass (CoM) consideration: in fact there is the common idea that it has to be as low as possible to help balancing, but actually this wuold be the worst solution, because the robot wuold require larger oscillations of the upper body in the frontal plane, and smaller steps in the sagittal plane.
If the CoM is high instead, it's possible to have natural dynamic walking; the disadvantage is that little variations (errors) in the hip position or torso orientation have more influence on balancing, but this can be compensated with an active control system that uses a tilt sensor and pressure sensors on the feet.
The ideal position of the CoM is the pelvis, thus the legs must be light and their mass distribution have to be as close as possible to the hip. For this reason we use a rotation trasmission system to put some motors farer than their own joint: in the picture can be noticed as the ankle motor is close to the knee, the knee one is at the top of the thigh.





2.2 Motor selection

The legs contain a total of 12 motors that are required to obtain six degree of freedom for each leg. The torque and speed requirements but also weight, space and power consumption limited our choice of actuators to rotary DC motors.
The 4 high power joints all use the same motor-gearbox-encoder combination.
The motor is a Faulhaber 3557CR that with a nominal voltage of 24V can provide 80mNm of torque continuously with a matching current consumption of almost 2 A and a maximum permissible speed of 6000 RPM. The maximum continuous torque in output from the planetary gearbox (efficiency 60%) on the shaft (considering a rotation ratio of 134:1) it's about 7 Nm.The total mass of the motor-gearbox-encoder unit is 0.49Kg.
The 8 low power joints use all the same motor-gearbox-encoder combination.
The motor is a Faulhaber 2657CR that with a nominal voltage of 24V can provide 44mNm of torque continuously with a matching current consumption of 1.5A and a maximum permissible speed of 6000 RPM. The maximum continuous torque in output from the planetary gearbox (efficiency 60%) is up to 4 Nm for this type of motor. The total mass of the motor-gearbox-encoder unit is 0.32Kg.
Due to the planetary gearbox reduction ratio the maximum output speed is 45 RPM (or 4.7 rad/s) for all motors. Each motor has an independent control system that perform a position/velocity control. These controls are Faulhaber MCDC2803OEM. This is a complete control system that in a small space (5 cm x 3 cm) has a core with various function mode (Stepper, PWM Velocity, PWM Position, Analog Velocity, Analog Position and RS232). In Our project we adopt the RS232 method for a mixed position/velocity control.

3 Hardware and software

3.1 Computer and software environment

The robot is controlled by a single board computer, the EuroTech CPU-1232 PC/104 module.
The relevant features for this project are:

1. 1 processor: NS Geode GX1 processor (Pentium MMX-compatible) 267 MHz;
2. memory: 128 MB DRAM;
3. Solid State Disk: 1 x 32 pin socket, 8 MB DiskOnChip Millennium Series;
4. 10/100 Mbps Fast Ethernet.

The Geode GX1 processor has a low real time jitter after disabling the System Management Interrupt (SMI), which is normally used for emulating text mode VGA.
For motors' I/O and sensors' input we use two EuroTech DAQ-1261 PC/104 digital+analog I/O modules.

The best solution for the software environment and the operating system seems to use a GNU/Linux variant:

1. Linux kernel, with the RTAI hard real time patch from DIAPM (Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Italy); more informations at http://rtai.org/;
2. GNU C and C++ libraries;
3. various utilities from Busybox.

We have written a Linux kernel module which handles all the I/Os with a single hard real time interrupt at 29102 Hz. It handles input lines for all sensors described in the dedicated section; output lines are provided for up to 6 servomotors (PWM). Additionally the kernel module provides 12 full duplex serial lines (at 9600 bps) entirely emulated in software, used for communicating with the 12 Faulhaber MCDC2803OEM motor controllers.

3.2 Sensors

The robot sensors must provide information to the balancing and walking software module, in particular the orientation of the upper body, the center of pressure on the feet. We can also have feedback from the motors receiving current consumption (usefull for compliant control) and current the current position.
In this way we can have a complete information about the three dimensional position of the robot in the space, making possible to correct in real time deviation from desired path or movements.

We used the following sensors:

1. 3 axis accelerometer (ADXL202): they have a measurement range of +/-2g (static and dynamic acceleration), and a digital PWM output with an analog filter at about 50Hz.
2. 2 axis gyroscope (CRS04): has a measurement range of +-150°/s. It has an analog output between 0 and 5V and a bandwidth of 85Hz.
3. 1 compass (CMPS03): it uses a PIC and 2 KMZ51 magnetic field sensor, which is sensitive enough to detect the Earth magnetic field. It has a digital PWM output with an analog filter at about 50Hz.
4. 8 force sensing resistors: we use it in a voltage partition.

The electronic system uses 4 boards witch contains the sensors and the serial interface.

4 Software

4.1 Simulation

It's very common to use computer simulation of the robot in a dynamic environment.
We decided to use a free C++ library that looks the easier to configure and modify, called ODE, and provided by Russel Smith. It's only disadvantage on other package as Dynamech, it's the integration system, that it's only first order.
This means that the movements the simulator shows will diverge in time from the real behaviour of the robot, making impossible to transport results from computer to reality, as desired for example when we use self learning methods as Neural Networks or Genetic Algorithms.
The team decided to not care about this point for mainly a reason: a simulation could be close to reality only with an appropriate knowledge of the physical structure (mass distribution, axis friction) of the robot and a precise model of the motors and their control system.
Not having all these information, we decided to used ODE only to debug the software architecture before the prototype building: those tests on kinematic and simple dynamic algorithms made us save a lot of time, improving parallel working of the team members.
ODE is open source and it can create tree-structures of rigit bodies connected by joints and actuated with velocity controlled motors, those can have a maximum torque or force available.

4.2 Computer vision

This software part has been developed from sketch without use any external library, with the goal to maximize velocity and performance in the RoboCup environment.

We focused on mainly two area: color labelling, and distance calculation.
As often happen in artificial vision, the color space used is not the RGB but another one, linearly converted to have the separation of luminance component.
The program is initialized with bounded areas in color space for labelling and has a multi-pass algorithm that removes noise: white compensation before color segmentation and removing of isolated pixels after. Finally it separate the objects in front of the camera, giving them a shape.
Knowing the characteristics of the web-cam focal angle and the size of the object, using geometrical transformation and vectors projection is possible to have a precise idea of the ball and the goal position.

4.3 Walking

In this phase of the project many things still have to be implemented.
To have dynamic and stable walking we use an inverted pendulum model of the robot with a moving virtual pernos on the ground (that correspond to a Zero Moment Point control).
During the step, that is completely described by few parameters there are three independent movements: body progress in the front direction, swinging leg during the single support phase and side oscillation for lateral balancing.
We use the position of hip and ankles to determine the legs position, and an inverse kinematic function translate it into joint angles on every motor shaft.
The trajectory are created with exponential curves, the solution of the differential equation of the linearized inverte pendulum on small angles, or bezier curve interpolation during the transition phases, as middle-stance (CoM projection on the middle of the foot or double support phase), when we have to inizialize the velocity and position for the next free falling of the rising movement of the pendulum.
Locally small error are detected using pressure sensors on feet and correct using a PID controll.

References

1. http://www.minimotor.ch/ Faulhaber motors and controllers
2. http://rtai.org/ Real Time linux extension
3. http://www.q-12.org ODE Open Dynamic Engine
4. http://www.eurotech.it/ PC104 Series
5. http://www.fiord.com/download/pc104/eurotech/daq_1261.pdf I/O System