The Isaac project: development of an autonomous biped robot

De Stefano Giovanni1, Faconti Davide2, Mandrile Roberto1, Mastrodomenico Lino1, Mura Andrea1, Nova Diego1, Scalabrino Luca1.

[1]Team member, [2] Project Leader (faconti_davide@yahoo.it),

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 performances. 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 chosen to give a torque larger than needed in common walking.

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 participate to the seventh edition
of RoboCup in Padua.
For this reason the robot needs to be autonomous and able to compete in a 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 take advantage of previous researches and to understand the kind of solutions that could be the best, or the pitfalls that we could find. Robot overview
Isaac is 85 cm tall and his average weight 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 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 synchronized by the upper layer that use the information given by balance sensors to create a real time trajectory on the joints angle.

1 Mechanical architecture

1.1 Design

The design of the robot is based on biomechanics data and analysis 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 kinematics 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) position: in fact the common idea is CoM should be as low as possible to help balancing, but actually this would be the worst solution, because the robot would 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. Light legs gives also less disturbance over walking patterns based on inverted pendulum model and special behaviors as ball kicking.
For this reason we use a rotation transmission system (pulley and belt) to put some motors farer from their joint: in the picture can be noticed as the ankle motor is close to the knee, while knee motor is at the top of the thigh.

Fig. 1. Hip and thigh detail, photo of first Isaac version.

2.2 Motor selection

The legs contain a total of 12 motors that are required to obtain six degrees of freedom for each leg.
Our motor selection was constrained by torque and speed requirements, as well as weight, dimensions and power consumption; for this reason the best solution seemed rotary DC brushed motors with planetary reducers.
The 4 joints those need more power (knee and hip) use the same motor-gearbox-encoder combination. The motor is a Faulhaber 3557CR,t with a nominal voltage of 24V, that 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. 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 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).
Each motor has an independent control system that performs a position/velocity control. These controls are Faulhaber MCDC2803OEM. This is a complete microcontroller and power amplifier board 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.

2 Hardware

2.1 Main computer and operative system

The robot needs a main computer to handle images from the web cam, read data from sensors, decide the commands for the motors (or, alternatively, send the data to a remote computer over the Internet and receive back the commands) and send them to the motor controllers.
The computer should be as small as possible, have low power requirements, being powered by a rechargeable battery, and have no moving parts for physical robustness (everything, including the mass storage device, must be implemented on solid state electronics).
Additionally, it must be possible to power it off at any time during its usage without damaging it and the file system on the solid state disk. This requirement imposes the use of a very strong journaling file system or, even better, read-only partitions; we have chosen the latter, not needing to modify the data on disk during normal operations.
The operating system must, of course, support all the computer peripherals, multitasking, have a TCP/IP stack, be easy to use and program, and it must support hard real time device drivers for handling various communication protocols (mainly PWM and RS-232) without the need for additional external hardware.
As a matter of practical and political[1] choice, all the software used on board must be Free Software[2]/open source.
The used hardware is a single board computer (SBC), the PC/104 module EuroTech CPU-1232.
The relevant features for this project are:

- processor: NS Geode GX1[5] processor (Pentium MMX-compatible) 267 MHz;
- memory: 128 MB DRAM;
- solid state disk: 1 x 32 pin socket, 8 MB DiskOnChip Millennium Series;
- 10/100 Mbps Fast Ethernet.
- The GX1 CPU is fast enough for our requirements and is compatible with the familiar x86 architecture.

For motors I/O and sensors inputs we use two EuroTech DAQ-1261 PC/104 digital+analog I/O modules.
The operating system used is a GNU/Linux[6] variant:

- Linux kernel, with the RTAI[7] hard real time patch from DIAPM (Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Italy);
- GNU C and C++ libraries;
- various utilities from Busybox[8].

The kernel is booted from a modified GNU GRUB[9] version burned directly on the DiskOnChip flash, instead of the proprietary BIOS.

We have written a Linux kernel module which handles all the I/Os with a single hard real time interrupt at 56.818 kHz, that handles input lines for all sensors described in the dedicated section; output lines are provided for the 6 servomotors (PWM).
Additionally the kernel module provides 12 full duplex serial lines (at 19.2 kbps) entirely emulated in software, used for communicating with the 12 motor controllers.
The Geode GX1 processor out of the box has a high real time jitter, due to the System Management Interrupt (SMI), which is normally used for emulating text mode VGA. We don't need it, so we simply disable it at boot time.
Without the SMI the worst case jitter stays always under 2 µs, a value that is perfectly acceptable for this job. The time for a single I/O operation is 1.2 µs, a rather slow but expected result because the acquisition cards use a 8 bit ISA bus. The overhead for handling an interrupt request is 3.5 µs, equivalent to a low end PC CPU at 400 MHz.
Overall the GX1 can do the job with a 56 kHz real time interrupt, but this is very close to the maximum available with this processor.

2.2 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 (useful for active compliant control) and actual 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:

- 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. Feeling the gravitational force, we can estimate torso orientation and tilt.
- 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. Gyros measures the angular velocity thus, to have an absolute angle, all the samples must be integrated (summed). An empirical algorithm is used to identify offset and exclude it.
- 1 Compass (CMPS03): it uses a PIC and 2 KMZ51 magnetic field sensors, which is sensitive enough to detect the Earth magnetic field. It has a digital PWM output with an analog filter at about 50Hz. Used to help navigation.
- 8 Force sensing resistors: we use it in a voltage partition. Their role is to feel reaction force of the ground on feet to estimate the CoM projection.

The desired torso inclination angle (on two axes) is obtained integrating gyro and accelerometer information: those, alone, couldn’t provide a reliable value.
In fact accelerometers aren’t precise during motion and normal walking, because the device feels the translation or centrifugal accelerations.

Gyros instead have problems related with the samples' integration: very small offsets can generate linearly growing error in time, and usually offset isn't a constant, being related to electric drifts (temperature primarily).
Still imagining to have a perfect gyro, the integration method is the Euler (first order) so the error between ideal and experienced curve will grow over long time periods.
Our software try to estimate the gyro offset with a weighted media considering a set of previous samples and a weight function that consider more likely as new offset, values close to the old one. This method guarantees an error growing of "only" 1 degree on 20 seconds.
We use as reference the angle obtained by gyros: that component in fact is more precise and has la larger bandwidth than accelerometer (in our implementation at least). Its error is kept very low using a close-loop control approach with the accelerometer hypnotized angle; the algorithm works in the following way:

- If the acceleration has been stable for a fixed period of time. Probably we aren't moving.
- Just in that case, correct the gyro's angle using a PI control loop (Proportional Integral).

The results of this empirical approach look generally good.

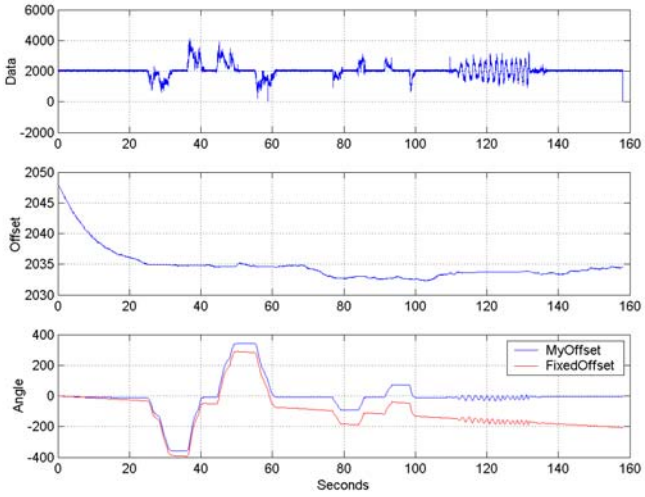

Fig.2. From top to bottom: gyros data, automatic offset identification, absolute angle computed with and without dynamic offset compensation,

3 Software

3.1 Simulation

It's very common to use computer simulation of the robot in a dynamic environment in biped locomotion research.
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 Dynamechs, 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 behavior of the robot; thus becomes impossible to "upload" results from computer to reality, as desired for example when we use self learning methods as Neural Networks or Genetic Algorithms.
We didn't care about that 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 kinematics 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 rigid bodies connected by joints and actuated with velocity-controlled motors, those can have a maximum torque or force available.

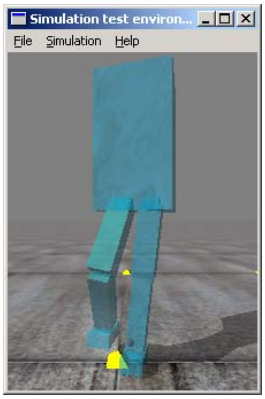

Fig.4. Screenshot of ODE simulation. The OpenGl display is a separate module.

4.2 Computer vision

The artificial vision system is based on a single web-cam connected to the main computer through the USB port and on two servos used to rotate it: the maximum rotations is 180° horizontally (left/right) and about 160° vertically (up/down).
The used hrdware is Philips ToU Cam Pro with a CCD sensor, choosed for its otimal sensibility to low illumination conditions (minimus work condition of 1 lux).
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 three areas: color labeling, noise reduction and distance calculation.
As often happen in artificial vision, the color space used is not the RGB but another one, converted to have the separation of luminance component: in fact it is in that color dimension that we have usually a large quantity of disturbance, dues to shadows and highlights. Instead of the common YUV color space, the TSL evidenced more robust behaviour: unfortunatly TSL color space has non linear relations with RGB and YUV color spaces, this means more CPU time for compute pixels. In order to save some CPU resources, we choose to store in volatile memory a pre-computed *lookup tables* to convert pixels in a very fast way.
The program is initialized with bounded areas in color space for labeling and has a multi-pass algorithm that removes noise: white compensation before color segmentation and then removing of isolated pixels, through erored/delay algorithms. Finally it separates the objects in front of the camera, using
To detect ball's distance we used the characteristics of the web-cam focal angle and the size of the object: using geometrical transformation and vectors projection is in fact possible to have a precise idea of the ball and the goal position, at condition that imposed angle on neck servos is actually applied on the camera.

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 pivot on the ground (that correspond to a Zero Moment Point control).

The ZMP is the standard dynamic balancing indicator in legged locomotion and it can be seen as the evolution of the intuitive balancing approach that consider balanced the robot when the vertical projection of the CoM lay inside the area of the feet touching the ground. The ZMP instead is the projection of CoM in the direction of the force vector, sum of all the forces applied to it, both gravitational, extern and inertial.

The step pattern is completely described by few parameters and some fixed constraints, as keep the torso vertical or inhibit discontinuity in velocity profile of the robot limbs. In every step three independent movements of the single support phase must be coordinated: the body progress in the front direction, the swinging leg motion and side oscillation for lateral balancing.

In joint space the corresponding angles are obtained with a simple inverse kinematics algorithm: the six parameters in foot space (position and orientation relative to the pelvis) are converted in six motor positions.

The basic model of an inverted pendulum considers all the mass concentrated in the hip. The pendulum pivot is defined "virtual" because it isn't a mechanical constraint but a point coincident where we fixed our desired ZMP: the resulting trajectory, in fact, is the same.

The one-dimensional differential equation is linearized for small angles and its solutions in time domain are:

$$\ddot{x} = \frac{x}{z} g$$

$$x = Ae^{kt} + B^{-kt}$$

$$\dot{x} = k\left(Ae^{kt} - B^{-kt}\right)$$

$$k = \sqrt{g/z}$$

Where x and z are the horizontal and vertical coordinates of the CoM relative to ZMP, g is the gravitation acceleration and A and B are obtained solving the velocity and position at (t = 0).

The second formula is used to create the hip trajectory during single support phase; other necessary patterns are interpolated trough a bezier curve.

Setting the velocity of the hip at beginning of single phase, we decide automatically the time required for a step, because the robot behave as a freefall pendulum; for this reason the desired velocity must be eventually restored during the double support phase, where the support area is larger.

Locally small errors are detected using pressure sensors on feet and tilt sensors, then corrected using manually tuned PID control.

4 Conclusions

The first prototype version of a humanoid robot has been presented.

The preliminary tests evidence a good architecture from both electrical and mechanical point of view.

Future works include:

- To design a new tilt sensor with a larger bandwidth.
- More precise control over each motor position.
- New legs, with stronger pulley-belt system and less angular moment to improve the inverted pendulum approximation.
- Better force sensing devices on the feet.

References

1. Shuuji Kajita: Dynamic walking control of a biped robot along a potential energy conserving orbit, IEEE Trans. Robot Atomat. Vol. 8 no. 4 (1992).
2. M. Hardt,O. von Stryk, D. Wollherr, M. Buss: Design of an autonomuos fast-walking humanoid robot.
3. Yasutaka Fujimoto, Satoshi Obata Atsuo Kawamura: Robust biped walking with active interaction control between foot and ground, Proc. of the 1998 IEEE Int. Conference On Robotics and Automation.
4. Pierre Brice Wieber: Constrained dynamics and parameterized control in biped walking.
5. Jerry E. Pratt, Gill A.Pratt: Exploiting Natural dynamics in the control of a planar bipedal walking robot, Proc. Of the 36° Annual Conference on Communication, Control and Computing.
6. Qiang Huang, Kazuhito Yokoy, Shuuji Kajita, Kenji Kaneko, Hirohiko Arai, Nohiyo Koyachi, Kazuo Tanie: Planning walking pattern for a biped robot, IEEE Transactions on Robotics and Automation, vol. 17, no.3 (2001)
7. Filippe M. Silva, Tenereiro Machado: Energy analysis during biped walking, Proc. of the 1999 IEEE Int. Conference On Robotics and Automation.
8. Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi, Hirohisa Hirukawa: A real-time pattern generator for biped walking, Proc. of the 2002 IEEE Int. Conference on Robotics and Automation.
9. R. Caballero, M.Armada, V.Sanchez: extending zero moment point to a segment using reduced order biped Model
10. Tomomochi Sugihara, Yoshihiko Nakamura, Hirochika Inoue: realtime humanoid motion generation through ZMP manipulation based on inverted pendulum control, Proc. of the 2002 IEEE Int. Conference on Robotics and Automation.
11. http://www.gnu.org/software/grub/ - GNU GRUB boot loader.
12. http://www.q12.org : ODE Open Dynamic Engine bu Russel Smith.
13. http://www.gnu.org/philosophy/why-free.html – why Free Software.
14. http://www.pc104.org/ - PC/104 embedded PC modules;
15. http://www.eurotech.it/ - EuroTech;
16. http://www.national.com/pf/GX/GX1.html - NS Geode GX1 processor;
17. http://rtai.org/ - real time Linux extension;
18. http://www.busybox.net/ - Busybox;
19. http://www.elet.polimi.it/upload/bonarini/Research/Vision/OmniRobocup.html: omnidirectional vision using convex mirror.