Manipulation et locomotion en robotique humanoïde avec optimisation temps réel des pas

Duong Dang

Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Systèmes automatiques

Présentée et soutenue par :
Duong Ngoc DANG

le : lundi 30 janvier 2012

Titre :

Humanoid manipulation and locomotion
with real-time footstep optimization

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
LAAS-CNRS

Directeur(s) de Thèse :

Dr. Jean-Paul LAUMOND

Rapporteurs :

Prof. Phillipe FRAISSE
Dr. Pierre-Yves OUDEYER

Membre(s) du jury :

Dr. Florent LAMIRAUX
Dr. Jean-Paul LAUMOND
Prof. Phillipe FRAISSE
Dr. Pierre-Yves OUDEYER

Acknowledgment

First and foremost, I would like to thank Dr. Jean-Paul Laumond for his guidance over the years. I could not have asked for a better Ph.D. advisor, inspirational, supportive and patient. I would also like to thank Dr. Florent Lamiraux for his advice and help both in scientific and technical issues. My experiments on the robot HRP-2 would not have been possible without support from Dr. Oliver Stasse and Dr. Nicolas Mansard. I also thank Dr. Michel Taïx and Dr. Phillipe Soueres who have been always ready to answer my questions about research, administrations or any other issues.

I am lucky to have a great research team at GEPETTO with a very nice ambiance among Ph.D. students. I appreciate the help from the veterans: Alireza, Anh, David, Francisco, Manish, Minh, Oussama, Sebastian when I first arrived in Toulouse, especially in finding the best restaurants in town. I also thanks Andreas, Antonio, Layale, Oscar, Olivier, Sovan, Thomard for both scientific collaborations and the enjoyable hours spending together outside lab. Thanks to Wassim and Wassima who also converted secretly to Gepettistes.

I will never be able to thank enough my family. Their unconditional love, confidence, and support are the reasons behind all the success. Special thanks to Ha for being in my life.

Contents

Chapter 1

Introduction

1.1 A brief history of robotics

1.1.1 Terminology and early concepts

Robotics as the field of design, construction and use of machines (robots) to perform tasks done traditionally by human beings[1] has only been around for a short period of time. In fact, the term *robot* appeared for the first time only in 1920 in the play "R.U.R (Rossum's Universal Robots) by Karel Capek [Cap20]. The word *robota* in Czech means tedious labor. Real autonomous robots only became widely available since the 60s starting with the manufacturing industry.

(a) Da Vinci's Knight Robot

(b) Jaquet-Doz's automata at Musée d'Art et d'Histoire of Neuchâtel, in Switzerland

Figure 1.1: Early automata

The field of robotics as we define and know is relatively young. The

[1]Definition from Encyclopædia Britannica

concept of automated machines, however, can be traced back to the ancient times. Archytas of Tarentum is usually reputed to have built mechanical steam-operated birds called "The Pigeon". Several texts in ancient China also accounted for automated machines. More recently, among other inventions, Leonardo da Vinci designed a Knight shaped machine (Figure 1.1a) around 1495, referred to as a humanoid automation [Ros06]. Later on, several types of automata were developed in Italy, Germany, Switzerland and Japan, notably the three automata *The Writer, The Draughtsman and The Musician* created by Pierre Jaquet-Doz (Figure 1.1b). Clearly, the idea of automated systems capable of performing human tasks existed long before the creation of the terminology *robot* itself.

Thanks to the brothers Capek, the (then purely futuristic) field of robotics gained gradually general public interest. This is particular pronounced after the publication of the works by Russian author Isaac Asimov, whose novels are still subject to numerous cinematographic adaptations. Asimov was perhaps most famous for the creation of *the three laws of robotics*

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.

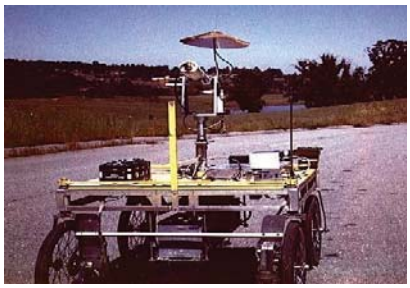
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Whether or not Asimov's Laws will one day go past the boundaries of science fiction and be incorporated into our common laws is still an open question. The enthusiasm they brought to the robotics field clearly made a positive impact.

1.1.2 Industrial and mobile robots

Ever since robotics caught general public interest, curiosity and enthusiasm, robots have been widely depicted in movies, most frequently in humanoid forms. Science fiction became, in part, a reality at the end of the 50s of the last century when General Motors introduced the Unimate robot in their factories. The incomparable to an ordinary worker, the repeatability of these machines made them more and more popular. Japan followed closely during the 60s with Kawasaki who licensed hydraulic robot design from Unimation. During the 70s, more robots were released, notably the PUMA arm, that later on went out the factories' perimeters to become a popular arm in research laboratories around the world. The 80s witnessed a rapid growth of industrial robotics. A number of new companies and robots entered the market: Kuka, ABB, Adept, CRS, etc. to name a few.

(a) Shakey the robot



(b) The Stanford cart

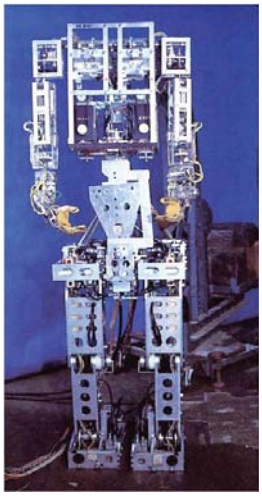
Figure 1.2: Early mobile robots

During the same period, various mobile platforms were born in research laboratories, such as *Shakey the robot* (SRI, Figure 1.2a), the *Stanford Cart* (Stanford, Figure 1.2b), and the NASA's Viking program robots. These robots with embedded vision systems opened new perspectives in term of applications. However computing power limitation at the time made any real-time manipulation virtually impossible (The Stanford Cart took around 15 minutes to analyze a view point and 1m every 10 to 15 minutes [Mor83]). Thanks to the increase in computational power as predicted by Moore's Law, computers became faster and faster and allowed the develop of a number of mobile robots: PR2 [WBdLJ08], Justin [BWS+09] etc. .

1.1.3 Humanoid robots

If human-like machines are the first images of *robots* both in the earlier concepts, automata and in Asimov's science fiction, in reality, humanoids are the last kind of robots that came into existence. The pioneers in humanoid robots are the WABOT (Figure 1.3) developed by Ichiro Kato et al. [Kat74,KOS+87].

WABOT-1 was essentially built from 2 arms mounted atop of a WL-5 (Waseda Leg 5) which was a continuous effort from the same group since 1967. WABOT-1 inherited all the advances in locomotion of WL-5 including balancing and walking. The first walking robot was still at the primary development with only quasi-static stepping which results in a 45-second long footstep. The robot also featured vision sensor, tactile sensors and a speech synthesizing system and was reported to have "the mental faculty of a one-and-haft-year-old child". WABOT-2 introduced in 1984 was even more capable manipulation-wise and really impressed the public by playing a piano during the Tsukuba Science Expo in 1985. In parallel, the WL series which serve as



(a) WABOT-1 (1973)

(b) WABOT-2 (1984)

Figure 1.3: First humanoid robots developed at the Waseda University

legs for WABOTs continued to developed. The concept of the mathematical tool introduced by Yugoslavian scientist Vukobratovic was also Incorporated into the system to achieve a dynamic walking with WL-12R in 1991 [LTK91].

It was Honda that captured a great deal of interest from the general public with the demonstration of P2 in 1996 [HHHT98]. With all components mounted on the robot (batteries, computers), P2 was arguably the first really autonomous humanoid that can walk stably. Honda followed through with P3 in 1997 and ASIMO in 2000.

Figure 1.4: Evolution of HONDA humanoid robots

The latest version of Asimo (5th generation, introduced in 2011) measured

130 cm and 48 kg with 57 degrees of freedom and can run up to 9km/h.

After Honda, other new humanoids have been released, notably the joint effort between Kawada and the National Institute of Advanced Industrial Science and Technology in Tsukuba (AIST) : the HRP series. With the active participation of AIST, there was a tighter link between the research community and the industry. More industrial quality and robust machines have been made available to universities and laboratories. HRPs robots have dimension and weight close to human's (Table 1.1).

	HRP-1	HRP-2P	HRP-2	HRP-3P	HRP-3	HRP-4C	HRP-4
Weight	130 kg	58 kg	58 kg	65 kg	68 kg	43 kg	39 kg
Height	160 cm	160 cm	154 cm	154 cm	160cm	158cm	151 cm
Dof	28	30	30	36	42	42	34

Table 1.1: Size of the HRP series

HRP-3P has the capability of functioning under high humidity condition and is still a rare water-proof humanoid to date. HRP-4C has been introduced as a robot for the entertainment industry and features numerous music and fashion shows in recent years.

Other notable humanoids robots made by the industry include the Sony's QRIO, the Toyota Partner Robots which were capable of playing musical instruments, the Aldebaran's NAO, the Kokoro's actroid etc.

Along side with industrial made robot, research laboratories around the world continue to develop and build humanoid prototypes. The availability of these humanoids, coming from different sources, provides more opportunity humanoid robotics research development.

Figure 1.5: Some of the HRP family, from left to right: HRP-4, HRP-2, HRP-3P and HRP-4c

Other robots There exists some other robots worth mentioning that do not share all the properties of previously presented humanoids. In the long run, this group of robot provides the same set of service: helping human in day-to-day tasks. They are modern mobile robot. The German Space Laboratory (DLR) is known for the development of the Justin robot which can perform cooking tasks with its two hands. The PAL Robotics' REEM is a robot with the same features: two arms, a body, a head with cameras are mounted on a mobile base instead of legs. The Willow Garage's PR2 gains in popularity recently due to its highly integrated ROS ecosystem which allows a relatively easy integration of multiple complex components ranging from planning, vision, SLAM to robot control.

Without legged locomotion, mobile robot presents a clearly advantage in term of robustness (no humanoid security system required [i.e. the lifter], since the robot cannot easily "fall"), in term development time since the controller has no longer to deal with the combination of feet commands and manipulation commands. The drawback is that theoretically mobile robot cannot reach *every where* the people go, i.e. climbing a stair or a ladder would be impossible. However, one might argue that people who need assistant from robots might have the accessibility to all area. A well adapted human environnement which accomodates wheeled chairs for example is also fit for mobile robots.

1.2 Whole-body manipulation

Many reserch work has been done in the theme of whole-body manipulation. Humanoids' high degree of redundancy creates a real challenge both in term of planning and control.

Planning tasks involving complicated situation which requires some time to plan and the result is to be executed in open-loop. Kuffner et al. used an RRT-Connect approach to plan grasping movement on the H6 humanoid robot [KKN$^+$02b]. The robot had to grasp an object underneath a table while avoiding all collisions. This approach tried to search in the space of statically-stable configurations for a solution that also lied on the collision-free configuration space. The resulting sequence of collision-free, statically-stable configurations was then smoothed and and filtered and fed to the robot.

[YPL$^+$10] used a pivoting method to manipulate a bulky object without having to lift it. This movement featured nice specialties of humanoid robots such as the coordination of the upperbody and the feet.

Escande et al. [EKM06] presented a manipulation task with a contact-support. This planner had the originality of allowing any possible contact between a robot link and the environment. Nakhaei and Lamiraux [NL08] used the computer vision system on HRP-2 to continuously update the envi-

ronment and modified the road-map accordingly while planning for a object manipulation task.

Other authors presented interesting results on whole-body coorerative motions on a humanoid robot for heavy works such as pushing a wall and twisting a valve (in simulation) [HKU03], pushing a heavy table on HRP-2 [TIS+04, HKK+07], carrying a heavy objects [HKS+05].

Grasping Tasks involving grasping and manipulating small objects have been studied under different angles. From the motion planning perspective, this problematic consists in finding a collision free path from the starting point to the predefined grasp pose. This leads directly to posture-based approaches [RMVJ01, KAAT03]. From the machine learning perspective, the grasping problem lies in finding the "best" grasp pose for a given object, namely an approriate posture that a human operator *would* choose. This can be done by modeling the target as a set of primitives and use a set of rules to choose, test, and evaluate the grasp candidates [MKCA03, GALP07]. Ranking techniques such as defining a score on the grasp according to the environnement around the object and the robot kinematics [BDN+07] also proved its effectiveness on humanoid robots. Other authors use vision or human demonstration to grasp novel objects [EK04, KFE96, SWN08]. If a lot of research efforts have been carried out at the low level, both in planning and learning aspects, little has been done in the context of legged robots where the manipulation cannot be seperated from the locomotion. This is one motivation of the work described later on in this thesis. If we can formulate a fusion framework that is flexible enough, all the techniques described above will become compatible and make the grasping with locomotion more robust.

Control There exists tasks in which the difficulty lies in the adaptation of the controller to the environment and/or involves aspects that cannot be addressed by traditional motion planning techniques. Task-space has long been a favorite pick when it comes to building a controller for a robotic manipulation. Since the early work by Khatib [Kha87] twenty-five years ago on operational space control, numerous other controllers follows.Sentis et al. expanded Khatib's original framework to humanoid robots [SK06,KBC+04] with prioritized tasks. Concurrently, Mansard et al. [MC07a,MC07b] proposed a different approach to resolve efficiently a stack prioritized tasks in position control and torque control [SRM+11] robots. These powerful controllers produced impressive experiments on humanoid robots. There lacks however a full integration with a higher level planner, such as footstep planning. In this thesis, task-based control will be heavily employed in a close connection with the planning modules.

1.3 Humanoid locomotion

Studies on biped locomotion can be traced back to the 70s, well before the introduction of the first humanoid robots. [VFJ70] presented the stability study on a simple biped with 2 masses supported by two legs which are position controlled. The robot was supposed to make point contact with the floor. Three kinds of stability were consider: *Body stability*, *Body path stability* and *Gait stability*, where the last item is specific to bipedal robot which corresponds roughly to the repeatability of the system in the leg coordinates . From this model and analysis, the author obtained a criteria for the dynamic equilibrium of the biped locomotion problem in question.

The same authors introduced later on [VS72] the concept of the Zero-Momentum-Point (ZMP). This time, not point contact but flat foot contact was considered. The ZMP provided with a simple necessary condition for all dynamic walking condition: its position must lie inside the support polygon. Since the ZMP is computable from state variables of the robot, either through a full 3D model or a simplified one such as the inverted pendulum or the cart-table, a stable walking sequence can be relatively easily computed.

[Wie02] developed a mathematical model to address the stability conditions of any walking system. The concept of viability was introduced to illustrate the set of possible robot states that allow the robot to perform a given movement without getting inside the unrecoverable zone. The union of all states satisfying such condition is the *viability kernel*. This analysis was very interesting in the sens that it describes mathematically the stability conditions of legged locomotion on arbitrary ground (as opposed to the ZMP condition which only valid for flat grounds). However, the drawbacks is that this method was more complicated and less computation-friendly that previous method relying on the ZMP.

Using the ZMP concept and a simplified model, [KKK+01] developed one of the first pattern generators for the HRP robots. This pattern generator used preview control to compute an optimal movement given a reference ZMP trajectory. [MHK+06] presented a method that computes ZMP and the center of mass's trajectories using polynomial function. Thanks to the effectiveness of polynomial computations, Morisawa et al. achieved real-time generation of walking movements as well as the possibility of modifying the next footstep during walking.

A number of more pattern generators have been developed and tested on the HRP-2 robot. [HDW+10] utilized linear and angular velocity of the robot as inputs for the pattern generator, hence remove the usual layer from the scope of the user interaction. Indeed, in the user perspective, the footsteps became completely transparent, velocities sufficed to get the robot going.

Concurrent to the work done on the HRP-2, Nishiwaki et al. [NKK+02] also proposed fast pattern generation method on humanoid robots. Again, the ZMP concept was used together with the simplified inverted pendulum model

of the robot.

Conditions on the ZMP contributes largely to the stability of a walking movement. Once we obtain and follow closely a reference ZMP trajectory, other parts of the body can be used to achieved other tasks. [VSYV06] demonstrated this principal with experiments where the robot had to step over large objects. A second pass, however, of the ZMP calculation (multi-body model) was necessary to enhance the stability of the movement since the cart-table model is no longer valid for a large amplitude movements of the feet.

Throughout the 40 year history of the ZMP, numerous studies on humanoid locomotion have been carried out using this concept. Recently, a new concept has been introduced to the community: *The capture point.* [PCDG06] defined the capture point as the point on the ground where the robot can step to in order to bring itself to a complete stop. The walking strategies were defined by the intersestion between the capture region and the base of support. Further simulations and experiments on the DLR-biped robot [EOR+11] showed that the capture point is a reliable alternative to the ZMP method

1.4 Adaptive locomotion

Adaptive methods to deal with a changing environment were common for numerous problems. [QK93, KJCL97] proposed using elastic bands for non-holonomic car-like robot. The gap between planning and control was closed by considering an elastic deformable collision-free path. During experiments, changes in the environment were detected by sensors and act as changes in external virtual forces on the elastic band. The connectivity between the robot and the goal was preserved and the found solution is locally optimized.

[LL98] used an iterative method to effectively adapt the trajectories for the Hilare robot with a trailer. The changes in the environment were recorded and corrected gradually at each control loop. This strategy resulted in a stable and smooth control scheme on the given robot. [GPFZ06] presented a decentralized approach to adapt trajectories of a group of mobile robots.

On humanoid robots, [UNN+11] proposed the singular LQ preview regulation for the problem of online foot placement. As before, the inverted pendulum model was used. The regulator problem resulted in the conditions of the target ZMP trajectory that will not make the CoM trajecotry diverge and a fast generation method of the CoM.

[KKN+03] used a $A^\star$ search strategy to plan footsteps. These quantities are computed independently from the rest of the robot. The cost function on the possible footstep candidates were defined from its amplitudes, whether or not it involves turning or stepping backwards. This principal were then applied to the HRP-2 robot [CNKK07] and ASIMO robot [CLC+05] with the robot position tracked by a motion capture system and 2D obstacles observed by external fixed cameras.

Perrin et al. [PSLY11, PSB+12] developed a pattern generator using half-steps and swept-volume approximations. Half-steps reduced the dimension and allowed doing offline approximation of the stepping domains in reasonable time. The swept-volume approximations provided an efficient method to check collision for a given footstep. This combination allowed a fast replanning algorithm that can effectively work on a changing environment.

1.5 Bio-inspired locomotion

Up to this point, we have been talking about methods that were designed for a specific humanoid robot with its advantages and drawbacks. On the one hand, with robots with precise manufacturing and a available CAD model , the kinematics, dynamics of the robot are known (almost) exactly. On the other, these fixed designs impose limits. The mentioned methods have to take into account the constraints of the mechanical design, e.g. the shape (usually flat) of the feet, the lack of a passive heel. In this section, we will look into the robots that are closer to human beings and have the potential of being stable, a inherent property coming from its structure. Boston Dynamics' BigDog [RBN+08] and PETMAN can adapt to the terrain and external perturbation. Design-wise, the BigDog has a spring in each of its feet which helps absorbing impacts. The BigDog also uses hydraulic actuators and is equipped with numerous sensors. The PETMAN, whose detailed design is not yet published, features passive heels which allow the robot to land on the back of its feet. This way of walking is much similar to how human walks (as opposed to flat foot landing in previous section).

In the research community, bio-inspired locomotion has also become an interesting topic. If the body of the better known robots consists of conventional joints and links, other scientists inspire from the animals' *vertebral column* to build their robot. Ijspeert et al. proposed a Salamander robot [ICRC07] which took advantage of the vertebral column to perform both swimming and walking locomotion. In this robot, a central pattern generator (CPG) was implemented to mimic biological systems. By choosing the right parameters for the CPG, the relatively complicated system in the conventional point of view (lots of degrees of freedom in the vertebral column) can be controlled effectively to achieve a desired locomotion.

An other trend in bio-inspired locomotion research consists in reducing to the maximum the number of actuators. To the extreme case, [McG90, CWR01, Kuo99] presented dynamically walking robots which did not use any energy at all. Such robots could walk down a slope with the gait properties similar to human's. The 3 passive robots developed at Cornell, Delft and MIT [CRTW05] could even walk on level ground by using an actuator to feed energy to the system hence play the role of gravity in previous design. Typically, the design of the feet in those robots were completely different for those in fully actuated

robot (HRP-2, ASIMO, etc.). Instead of being flat, they took a curved shape which allowed landing on the heel and minimizing the impact energy.

[LLO11] proposed using the vertebral column principle and passive walking on a humanoid robot: Acroban. Indeed, the vertebral column increased greatly the stability of the robot as it acts as a semi-passive inverse pendulum. Compliance was also a key feature of Acroban, both at control level where the maximal torque can be ajusted in realtime as well as design level with the robot's soles covered with compliant materials. These considerations made the robot robust to external perturbations. Dynamic walking was triggered by a CPG which did not involve realtime computed dynamics or ZMP-based concept as seen earlier but followed a scheme of self-stabilizing passive robots. The system was therefore stable and adaptable for changes in the environment by definition as it made no or little assumption about the external world but adapt itself accordingly.

Inspiring from biological systems also means inheritting from the nature is complexity. The dynamics of the robots such as the ETH's salamander or the Acroban are most likely more difficult to modelized than HRP2's. The control algorithms have to therefore adapt with these unknowns. Atkeson et al. [AMS97a, AMS97b] provided a throughout survey on lazy learning methods in autonomous adaptive control and their applications in robotics. [Tay04] proposed aaptive iterative learning control techniques to track trajectories on robot arms without knowing the parameters.

Both the bio-inspired design and adaptive control techniques are very interesting. The future of humanoid robots are probably more flexible, more human-like generations of domestic robots. For this generations, the assumption of robots as a chain of rigid body with fixed, perfectly measurable dynamics will be no longer valid. This is where the research axe described in this section will play a crucial role, in making the robots controllable, useful, reliable and safe.

Although the techniques described further in this thesis are designed for the "first" generation of robots with rigid bodies and known dynamics. In the future, techniques in adaptive control, learning should also be combined to enhence furthermore the performance of achieved tasks.

1.6 Approach and contribution

This thesis deals directly with theory and experiments on humanoid platforms whose dynamical properties are particularly well modelised. The rigid structure of these robots, while provide a high degree of precision in control, hardly allows exploring paths such as machine learning, bio-inspired locomotion. The control, adaptation strategy relies uniquely on algorithms. The framework here is developped on and existing techniques to provide a adaptive strategy for manipulation with locomotion in a changing environment. It merges the

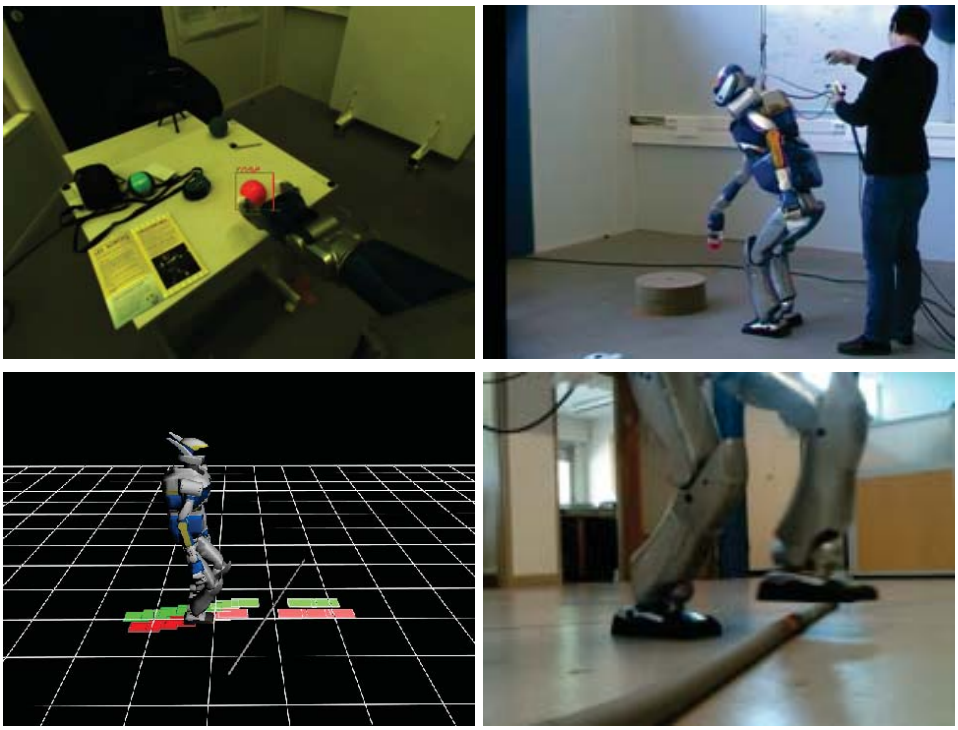


Figure 1.6: Experiments on HRP-2 using the real-time footstep optimization

traditionally slow process (motion generation) and a reactive process (sensor-based control) with perception.

If numerous works have been carried out in both manipulation and locomotion, the two aspects are usually treated as independent problems. Whole-body tasks are often considered completely separate from the footsteps. In this thesis, we tackle the problem of complex locomotion and manipulation by considering footsteps parts of the robot's kinematic chain. The optimizer is written with complete abstraction of the locomotion and works as if it only deals with a conventional robot. Designed to work on the current generation of humanoid robots which are yet to be equiped with more adaptive features found in biological systems, our framework inherits its adaption capability from its optimization scheme. Indeed, footstep optimization is put into a closed loop with the perception modules to create a framework that allows the robot to adapt reactively to changes in the environment. The goals of this framework are:

- Seamlessly integrate locomotion with whole body movement. Footsteps are considered as part of the robot and are dictated by the task applied to the augmented robot.
- Build a reactive scheme that helps the robot achieve the task even if the

environment is changed during execution.

- Resolve the foot placement by optimization so that it preserves the optimality, hence, the feasibility of the movement.
- Integrate with on-robot stereo vision to make the movement the most robust and portable possible.

Moreover, combined with a prior motion planning step, this method is less subject to local minima than classical numerical optimization approaches. The laid out framework is demonstrated in a number of different experimental situations (Figure 1.6).

Contribution

The contributions of this thesis are on the following aspects

- Design of the framework (chapter 2.1, chapter 4) [DLL11].
- Implementation of a state-of-the-art stereo vision module on HRP-2 robot, with an automated calibration procedures for cameras.
- Generic deformations of footsteps (chapter 3). The representation of footsteps as the robot's extra degrees of freedom can be used to calculate the initial footsteps as well as to adapt these footsteps on the fly during the experiment. The framework is flexible enough to take as input any initial footsteps sequence and adapt them in real-time.
- Evaluation of the framework on the HRP-2 robot with two classes of experiments [DLL12]. The first class of experiments consists in grasping an object at various height level (chapter 6). This highly redundant manipulation is carried out in harmony with an appropriate locomotion sequence. The second class of experiments illustrates the real-time footstep adjustment scheme in a typical context of humanoid robotics, i.e. stepping over objects (chapter 5).

Associated papers with the work presented in this thesis:

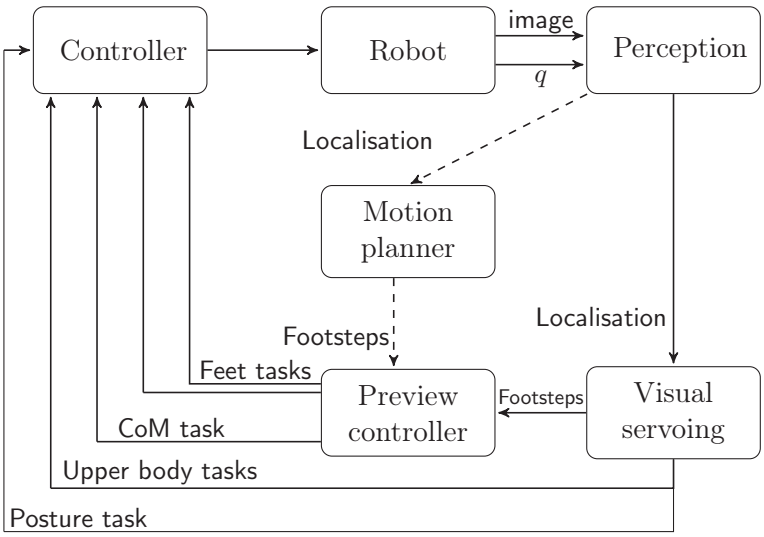
- D. Dang, F. Lamiraux et J-P. Laumond. Experiments on whole-body manipulation and locomotion with footstep real-time optimization. In *EEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2012)*, Osaka, Japan, november 2012.
- D. Dang, F. Lamiraux et J-P. Laumond. A framework for manipulation and locomotion with realtime footstep replanning. In *IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2011)*, Bled, Slovenia, october 2011.

- D. Dang. Réalisation de tâches avec locomotion sur un robot humanoïde. In *Journées Nationales de la Robotique Humanoïde*, Toulouse, France, april 2011.

Chapter 2

Framework

2.1 Global architecture



COM: Center of Mass

Figure 2.1: Global architecture

Figure 2.1 depicts the global architecture of the framework. The planner plays the role of a "visual servo" for footsteps. It optimizes the stepping sequence in real-time and in closed-loop with the vision system. The controller takes as inputs the information from the visual servo and resolves the prioritized hierarchy of the corresponding primary tasks. It then sends command to the robot in real-time at required frequency. The perception system includes an automatic calibration process which improves precision and allows the framework to perform precise tasks such as grasping. This

process is performed offline before the experiments.

2.2 Forward kinematic

2.2.1 Robot description

A robot consists of a chain of links (possibly a tree-like chain) starting from a base. Typically the base can be fixed for a manipulator (e.g. WAM [SELT91], Kuka [HBOS05]...), or can be mobile (e.g. PR2 [WBdLJ08], Justin [BWS+09], PAL's REEM [TFG+08]...) or the waist link for humanoid robots (e.g. HRP series, ASIMO, Romeo ...).

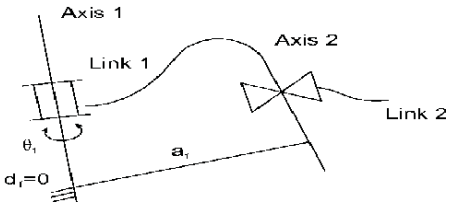


Figure 2.2: Revolute first joint

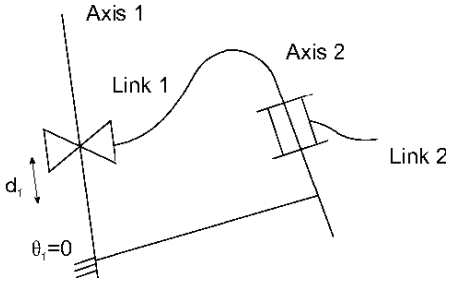


Figure 2.3: Prismatic first joint

The links are connected by joints which are usually divided into two groups: *revolute* et *prismatic* joints (Figure 2.2, Figure 2.3). Most robots are built exclusively with revolute joints (except for a very few cases, for example, the Stanford arm [DS73] and the elevator joint for mobile robots such as PR2 and cie.). Figure 2.4 and Figure 2.5 depict a simple humanoid robot with 30 revolute joints and its corresponding kinematic tree.

2.2.2 Forward kinematics formulation

Now that the transformations between subsequent joints can be established from the description of the joint and the state of the robot (see Appendix A) we can work our way from the base to any given frame:

$$ {}^{0}_{N}T = {}^{0}_{1}T {}^{1}_{2}T ... {}^{N-1}_{N}T \tag{2.1} $$

As depicted in Figure 2.6, the forward kinematics can be thought of as a injective mapping from joint space to the operational space. As detailed further in subsection 2.3.3, most of useful tasks can be expressed as the position and orientation of a frame, e.g. position of the center of mass (CoM), of an end effector. A large set of tasks can be used form these primitives.

Example For 6 degrees of freedom the branch between from the base to the end-effector is written as

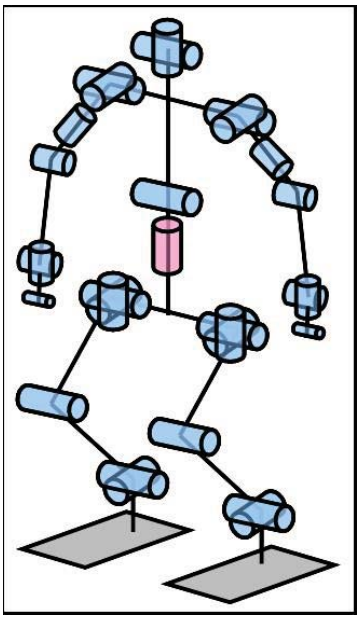
$$ {}^0_6T = \left[\begin{array}{ccc|c} & {}^0_6R & & {}^0_6p \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.2} $$

where

$$ {}^0_6R = \left[\begin{array}{c|c|c} \begin{array}{r} c_1c_{23}(c_4c_5c_6 - s_4s_6) \\ -c_1s_{23}s_5c_6 \end{array} & \begin{array}{r} c_1c_{23}(-c_4c_5s_6 - s_4c_6) \\ +c_1s_{23}s_5s_6 \end{array} & \begin{array}{r} c_1c_{23}c_4s_5 \\ +c_1s_{23}c_5 \end{array} \\ \hline \begin{array}{r} s_1c_{23}(c_4c_5c_6 - s_4s_6) \\ -s_1s_{23}s_5c_6 \\ +c_1s_4c_5c_6 \end{array} & \begin{array}{r} s_1c_{23}(-c_4c_5s_6 - s_4c_6) \\ -s_1s_{23}s_5s_6 \\ -c_1(s_4s_5c_6 + c_4c_6) \end{array} & \begin{array}{r} s_1c_{23}c_4s_5 \\ +s_1s_{23}c_5 \\ +c_1s_4s_5 \end{array} \\ \hline \begin{array}{r} -s_{23}(c_4c_5c_6 - s_4s_6) \\ -c_{23}s_5c_6 \end{array} & \begin{array}{r} s_{23}(c_4c_5s_6 + s_4c_6) \\ +c_{23}s_5s_6 \end{array} & -s_{23}c_4s_5 + c_{23}c_5 \end{array}\right] \tag{2.3} $$

$$ {}^0_6p = \begin{bmatrix} L_6c_1c_{23}c_4s_5 - L_3c_1s_{23} + L_2c_1c_2 + L_1s_1 \\ L_6s_1c_{23}c_4s_5 + L_3s_1s_{23} + L_6c_1s_4s_5 + L_2c_1c_2 + L_1s_1 \\ -L_6s_{23}c_4s_5 - L_3c_{23} + L_6c_1s_4s_5 + L_2c_1c_2 \end{bmatrix} \tag{2.4} $$

(a) HRP-2 robot



(b) HRP-2 joints

Figure 2.4: The HRP-2 robot and its joints

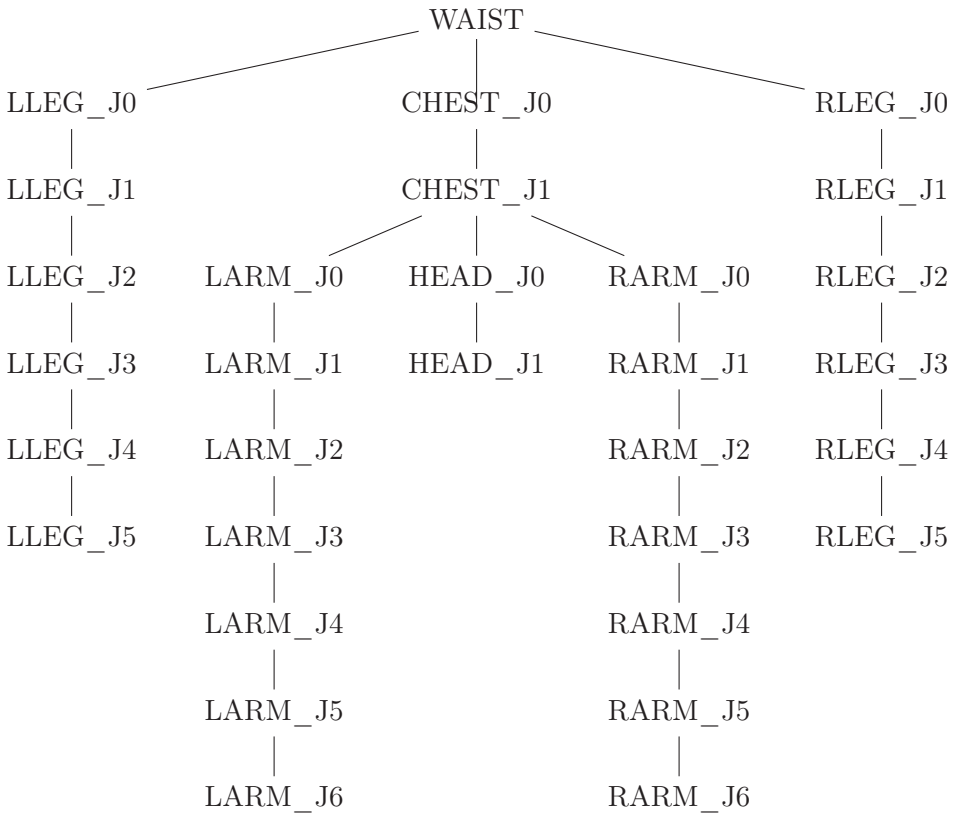


Figure 2.5: Kinematic tree of the HRP-2 robot

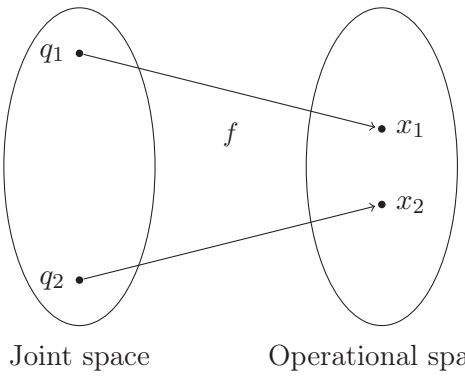


Figure 2.6: Mapping from joint space to operational space

where s and c are shorts for sin and $cosin$, the index indicates the θ to be apply to, e.g. $s_1 = \sin\theta_1, quadc_{23} = \cos(\theta_2 + \theta_3)$

2.3 Prioritized hierarchy of tasks

2.3.1 Joint space control

As described in the previous section, a robot can be thought as a sequence of actuators connected by bodies, in the most common situations, rigid bodies. The role of the controller is to, at the lowest level, given a task/set of tasks, produce a command to these actuators, i.e.

- current for electrical motors.
- air pressure for pneumatic actuator.
- hydraulic pressure for hydraulic actuators.

The next stage in the control stack is using directly torques as input, by mean of a transform function which translates torques into primitive control signals. Now, the torques can be written in the dynamic equation of the corresponding joints and allow us to design the most straight forward controller: a joint-space controller. As suggested by its name, this controller operates on the configuration vector $\boldsymbol{q}$. The goal of the process is given a desired vector $\boldsymbol{q}_d$, produce torques that allows the robot to achieve as close as possible to the desired position. This controller can involve PD control, PID control, Pyapunov-base control [SHV06] with the feedback coming directly from joint state measurements.

Tasks, however, are difficult to model in the joint space. The most common tasks involve operational points (end effector, center of mass (CoM), etc.), hence the kinematics of the robot. The process of computing a desired $\boldsymbol{q}_d$ corresponding to a desired position $\boldsymbol{x}_d$ in the operational space is called the inverse kinematics. A new box has to be added to the control schema. (Figure 2.7) The inverse kinematics box raises two difficulties. On the one

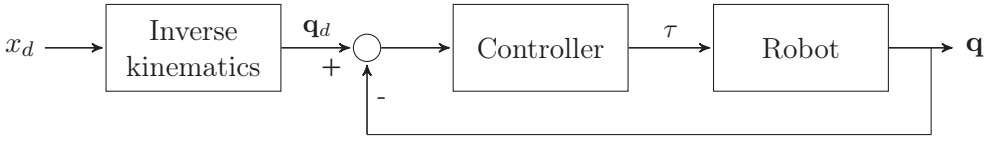


Figure 2.7: Joint space controller

hand, the analytical solution is hard to find. The forward kinematics is usually non-linear function of $\boldsymbol{q}$ consisting of trigonometric functions. Even for non redundant 6 degree of freedom robot such as the PUMA 560 arm [Bee79], a complicated results are found after a lengthy resolution involving specific tricks

that cannot be generalized.On the other hand, for redundant robots (greater than 6 degrees of freedom), there are always, for non singular case, a infinite number of solutions.

Machine learning provides an interesting tool to enhance the calculation of the the inverse kinematics when the exact model of the robot is not known [DVS01]. However, the drawback is that data are to be collected and the precision of this method depends on data quality.

2.3.2 Task representation

In addition to the difficulty of the inverse kinematics problem, the physical sense of the movement is lost when reasoning in joint space. A simple straight line might end up in a very complicated curve in joint space. This becomes more even problematic when one wants to combine two distinct task in task space in a certain order of priority. This issue is addressed by considering tasks in operational space.

In this representation, a task $\mathcal{T}$ is a characterized by a vector $\boldsymbol{x} \in \mathcal{R}^m$. This vector is often related to the Cartesian position and orientation of an end-effector. $\boldsymbol{x}$ can also be any mathematical quantity that can be expressed as a smooth function ($\mathcal{C}^1$ for the control purpose) of the configuration vector $\boldsymbol{q} \in \boldsymbol{R}^n$. For instance, if the robot is to clap its hands, $\boldsymbol{x}$ will be the euclidean distance between the too hands. For a gazing task, $\boldsymbol{x}$ is the distance in the image space between the target and the center of the image.

Let us define the Jacobian matrix $J(\boldsymbol{q}) \in \mathcal{R}^{m \times n}$ the matrix that transforms the joint velocity ($\boldsymbol{q} \in \mathcal{R}.$) to the operational space velocity ($\boldsymbol{x} \in \mathcal{R}^m$):

$$\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.5}$$

which leads to:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial x_1}{\partial q_1} & \cdots & \dfrac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial x_m}{\partial q_1} & \cdots & \dfrac{\partial x_m}{\partial q_n} \end{bmatrix} \tag{2.6}$$

The error function $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$ measures the different between the current state and the desired state of the task. We have

$$\dot{\boldsymbol{e}} = \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\dot{\boldsymbol{q}} = J\dot{\boldsymbol{q}} \tag{2.7}$$

Suppose that the robot is not in singular state, J is full rank. The resolution of $\dot{\boldsymbol{q}}$ can then be obtain by the pseudo-inverse matrix of J [BIC63].

$$\dot{\boldsymbol{q}} = J^{+}\dot{\boldsymbol{e}}^{*} + P\boldsymbol{z} \tag{2.8}$$

Where $\dot{\boldsymbol{e}}^*$ is the desired motion in the operation space. P is the projector into the null space of J ($P = I - J^+J$). P represent the redundancy of the robot with respect to a given task. With an arbitrary vector $\boldsymbol{z}$ we always have $JP\boldsymbol{z} = 0$ which guaranties that the Pz part won't disturb the given task. This redundancy provide some flexibility for the robot to perform additional tasks which is the basis of the prioritized hierarchy of tasks.

2.3.3 Task hierarchy

Mansard and Chaumette [MC07a] proposed a stack of tasks to resolve cases where the robot has to perform a ordered set of tasks. First, the solution for the first task $\mathcal{T}_1$ is simply

$$\dot{\boldsymbol{q}}_1 = J^+\dot{\boldsymbol{e}}_1^* + P_1\boldsymbol{z}_1 \tag{2.9}$$

As discussed earlier, the term $\boldsymbol{z}_1$ provides us with some flexibility to perform additional task at lower priority than $\mathcal{T}_1$. Indeed, when a task $\mathcal{T}_2$ is added, one can choose $\boldsymbol{z} = J_2^+\dot{\boldsymbol{e}}_2^* + P_2\boldsymbol{z}_2$. The new command input becomes

$$\dot{\boldsymbol{q}}_{12} = J_1^+\dot{\boldsymbol{e}}_1^* + P_1(J_2^+\dot{\boldsymbol{e}}_2^* + P_2\boldsymbol{z}_2) \tag{2.10}$$

Due to the definition of the projection operator P_1, the effects of $\dot{\boldsymbol{q}}_{12}$ on the first task $\mathcal{T}_1$ is

$$\begin{aligned}\dot{\boldsymbol{e}}_{12} &= J_1\dot{\boldsymbol{q}}_{12} \\ &= J_1J_1^+\dot{\boldsymbol{e}}_1^* + J_1P_1(J_2^+\dot{\boldsymbol{e}}_2^* + P_2\boldsymbol{z}_2) \\ &= \dot{\boldsymbol{e}}_1^*\end{aligned} \tag{2.11}$$

which means that the priority is preserved. The new control input $\dot{\boldsymbol{q}}_{12}$ produces on $\mathcal{T}_1$ exactly the same effect as if only task 1 has been treated. Recursively, subsequent task $\mathcal{T}_i$ added to the stack result in the choice of the "free" term $\boldsymbol{z_{i-1}}$. At last, a stack of tasks with decreasing priority $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_n$, produce the following command input

$$\dot{\boldsymbol{q}} = J_1^+\dot{\boldsymbol{e}}_1^* + P_1J_2^+\dot{\boldsymbol{e}}_2^* + P_1P_2J_3^+\dot{\boldsymbol{e}}_3^* + \cdots + \left(\prod_{i=1}^{n-1} P_i\right) J_n^+\dot{\boldsymbol{e}}_n^* \tag{2.12}$$

This formulation however does not guarantee the priority property of the stack. Indeed, since matrix multiplication is not per mutable we cannot assure that $P_1P_2J_3^+\dot{\boldsymbol{e}}_3^*$ respect the task $\mathcal{T}_2$, i.e.

$$J_2(P_1P_2J_3^+\dot{\boldsymbol{e}}_3^*) \neq J_2P_2(P_1J3^+\dot{\boldsymbol{e}}_3^*) = 0 \tag{2.13}$$

To preserve the priority specification, the projector P_i has to project z_i into the null space of all previous tasks. In other words, $P_i = I - N_{1..}$, where

$N_{1..}$ is the intersection of all the individual null spaces, or the null space of the following subspace

$$J_i^A = \begin{bmatrix} J_1 \\ \vdots \\ J_i \end{bmatrix} \tag{2.14}$$

$$P_i^A = I - J_i^{A+} J_i^A \tag{2.15}$$

Replace this new projector into the previous formula we obtain

$$\dot{\boldsymbol{q}} = J_1^+ \dot{\boldsymbol{e}}_1^* + P_1^A J_2^+ \dot{\boldsymbol{e}}_2^* + P_2^A J_3^+ \dot{\boldsymbol{e}}_3^* + \cdots + P_{n-1}^A J_n^+ \dot{\boldsymbol{e}}_n^* \tag{2.16}$$

In equation (2.10), z_1 was chosen only as if the term $J_1^+ \dot{\boldsymbol{e}}_1^*$ does not effect task $\mathcal{T}_2$. To take this into account $\dot{\boldsymbol{e}}_2^*$ should be replaced by $\dot{\boldsymbol{e}}_2^* - J_2 J_1^+ \dot{\boldsymbol{e}}_1^*$ where the second term compensate the effect of the previous stage into the current task.

Finally, the formulation for n tasks $\mathcal{T}_1, \mathcal{T}_2, \ldots \mathcal{T}_n$, with the desired behavior $\dot{\boldsymbol{e}}_1^*, \dot{\boldsymbol{e}}_2^*, \ldots, \dot{\boldsymbol{e}}_n^*$ is written in the following recursive form [SS91]

$$\dot{\boldsymbol{q}}_0 = 0 \tag{2.17}$$

$$\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{q}}_{i-1} + J_i P_{i-1}^A \widetilde{\dot{\boldsymbol{e}}_i^*} \quad \forall i = 1 \ldots n \tag{2.18}$$

$$\widetilde{\dot{\boldsymbol{e}}_i^*} = \dot{\boldsymbol{e}}_i^* - J_i \dot{\boldsymbol{q}}_{i-1} \tag{2.19}$$

2.3.4 Control law

In subsection 2.3.2, a task is defined as a mapping from joint space $\boldsymbol{q}$ to operational space $\boldsymbol{x}$. In many cases, this representation involves a desired task space configuration $\boldsymbol{x}_d$. In subsection 2.3.3, the control input $\dot{\boldsymbol{q}}$ is computed from the desired derivatives of the error vectors $\dot{\boldsymbol{e}}_i$. The missing part is a control law that produces $\dot{\boldsymbol{e}}_i$ from $\boldsymbol{q}_d^i$ and the current state.

First order differential equation This control law is expressed in the form

$$\dot{\boldsymbol{e}}^* = -\lambda \boldsymbol{e} \tag{2.20}$$

where λ is a positive gain.

This leads to a control law for any given task and its Jacobian J

$$\dot{\boldsymbol{q}} = -\lambda J^+ \dot{\boldsymbol{e}} \tag{2.21}$$

With a decent low level control, we can expect that the real derivative of the task error is close to the desired value $\dot{\boldsymbol{e}} = \dot{\boldsymbol{e}}^*$. We then obtain

$$\frac{\partial \boldsymbol{e}}{\partial t} = -\dot{\boldsymbol{e}} = \lambda \boldsymbol{e} \tag{2.22}$$

$$\frac{\partial \boldsymbol{x}}{\partial t} = -\lambda(\boldsymbol{x} - \boldsymbol{x}_d) \tag{2.23}$$

$$\boldsymbol{x} = \boldsymbol{x}_d + (\boldsymbol{x}_0 - \boldsymbol{x}_d)e^{-\lambda t} \tag{2.24}$$

The robot converges exponentially to the desired position $\boldsymbol{x}_d$ However, it is usually desired to control the characteristic time of the exponential function which define λ. (2.20) provides no control over the initial velocity $\dot{\boldsymbol{x}}_0$. Discontinuity in the movement will appear at the beginning of each task. Moreover, (2.20) does not guarantee any boundary for $\dot{\boldsymbol{e}}$ which may lead to a infeasible control law $\dot{\boldsymbol{q}}$ (a required movement too fast that the robot is not capable of).

Second order differential equation [SCD02] proposed the following evolution of the error vector

$$\ddot{\boldsymbol{q}} + \alpha \dot{\boldsymbol{e}} + \beta \boldsymbol{e} = 0 \tag{2.25}$$

This control law allows more flexibility in the initial condition $\mathbf{0}$ and $\dot{\boldsymbol{e}}$ which were not possible with the simple first order differential equation. However, tuning α and β is a tedious task.

Non homogeneous first order differential equation This control law, proposed by [MC07a], is govern by the following equation

$$\dot{\boldsymbol{e}}_i^* = -\lambda e + \rho(t) \tag{2.26}$$

where

$$\rho(t) = e^{-\mu t}(\dot{\boldsymbol{e}}_A(0) + \lambda \boldsymbol{e}_B(0)) \tag{2.27}$$

This scheme is most widely used in the multi-task controller since it has enough flexibility on initial condition and is easier to tune the parameters. Two characteristic times appear here. The first one $\tau_1 = 1/\lambda$ corresponds to the exponential convergence in the first control law. The second $\tau_2 = 1/\mu$ corresponds to the transient period at the beginning. At $t \gg \tau_2$ the controller exactly like the original control law. Moreover

$$\begin{aligned} \dot{\boldsymbol{e}}^*\big|_{t=0} &= \lambda \boldsymbol{e}\big|_{t=0} + e^{-\mu.0}(\dot{\boldsymbol{e}}_A(0) + \lambda \boldsymbol{e}_B(0)) \\ &= -\lambda \boldsymbol{e}_B(0) + (\dot{\boldsymbol{e}}_A(0) + \lambda \boldsymbol{e}_B(0)) \\ &= \dot{\boldsymbol{e}}_A(0) \end{aligned} \tag{2.28}$$

The transition from task A to task B preserve the continuity of the movement. Apply this principal into the task hierarchy in subsection 2.3.3, i.e.

$$\dot{\boldsymbol{q}} = \begin{bmatrix} J_1^1 & P_1^A J_2^+ & \dots & P_{n-1}^A J_n^+ \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{e}}_1^* \\ \dot{\boldsymbol{e}}_2^* \\ \vdots \\ \dot{\boldsymbol{e}}_n^* \end{bmatrix} \tag{2.29}$$

we have, τ being the last instant where the stack of tasks is modified:

$$\dot{\boldsymbol{q}} = J_1^+ \dot{\boldsymbol{e}}_1^* + P_1^A J_2^+ \dot{\boldsymbol{e}}_2^* + P_2^A J_3^+ \dot{\boldsymbol{e}}_3^* + \cdots + P_{n-1}^A J_n^+ \dot{\boldsymbol{e}}_n^* + e^{-\mu(t-\tau)}(\dot{\boldsymbol{e}}(\tau)) + \Lambda \boldsymbol{e}(\tau)) \tag{2.30}$$

The recursive form will then becomes:

$$\dot{\boldsymbol{q}}_0 = 0 \tag{2.31}$$

$$\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{q}}_{i-1} + J_i P_{i-1}^A \widetilde{\dot{\boldsymbol{e}}_i^*} \quad i = 1 \dots n \tag{2.32}$$

$$\widetilde{\dot{\boldsymbol{e}}_i^*} = \dot{\boldsymbol{e}}_i^* - J_i \dot{\boldsymbol{q}}_{i-1} \tag{2.33}$$

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_n + e^{-\mu(t-\tau)}(\dot{\boldsymbol{e}}(\tau)) + \Lambda \boldsymbol{e}(\tau)) \tag{2.34}$$

2.4 Locomotion

2.4.1 Dynamic equations

Let us now consider the locomotion of humanoid robots. Humanoids are usually represented as a kinematic tree starting from a *free flyer*, commonly called the waist. Mobile robots also have this degree of freedoms that are directly actuated by the base wheels. For a humanoid robot however, there is no physical joints that control the free flyer. The robot moves around thanks to the movement of its feet. For this reason, it is important to distinguish two phases during a movement

- Single support phase: only one foot of the robot touch the ground (support foot). The other foot (flying foot) is in the air moving to the new position.
- Double support phase: both feet of the robot are on the ground. During this phase, the center of mass of the robot is shifted foot to prepare for the next step.

The third possible phase is when both feet leave the ground (jumping) and out of the scope of this framework. When the robot is standing straight and stable, its center of mass lies inside its support polygon. In fact, if the motion is slow enough so that we can neglect the accelerations, we can prove that the robot is stable if its center of mass lies inside the support polygon. How

about walking? We can of course generate a sequence of poses that is statically stable. By moving from one pose to one another at a sufficiently slow pace we can displace the robot from one corner of the room to another.

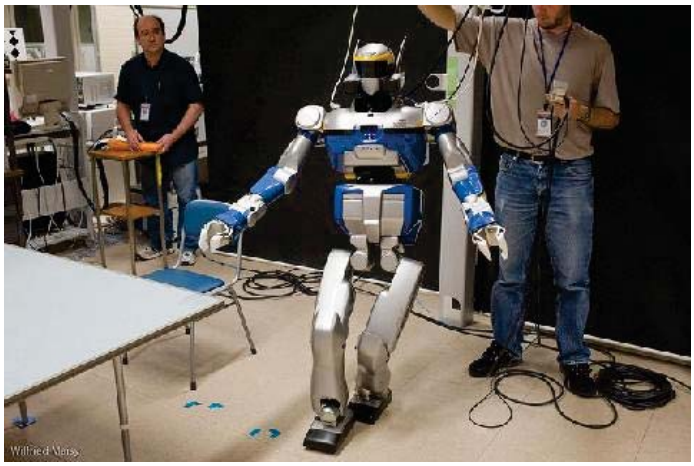


Figure 2.8: Quasi-static walking. The robot moves between stable poses during the experiment [EK09]

.

Figure 2.8 depicts a *quasi-static walking* sequence which looks very different from the way we walk. The reason behind that is the fact that our walk is not static. During the single support phase, the center of mass can jump outside of the support polygon. We do not fall because in the next phase, the flying foot hit the ground and 'recover' the center of mass inside its support polygon. In fact, during a dynamic walk, the projection of the center of mass bounces back and forth between the two feet.

Let us consider a humanoid robot modeled in Figure 2.9. We denote the following quantities on the body j:

- $\boldsymbol{x}_j$ the position of center of mass
- m_j the mass
- I_k the inertia matrix
- R_k the rotation matrix
- $\boldsymbol{\omega_k}$ the angular velocity

By definition the total mass and the position of the center of mass are

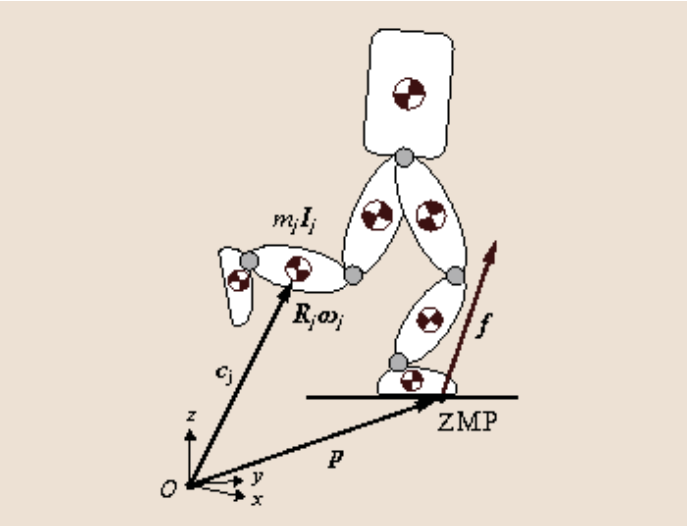


Figure 2.9: Humanoid robot model

$$M = \sum_{j=1}^{N} m_j, \tag{2.35}$$

$$\boldsymbol{c} = \sum_{j=1}^{N} m_j \boldsymbol{x}_j / M \tag{2.36}$$

The dynamic wrench of the system can be written as

$$\begin{bmatrix} \dot{\boldsymbol{P}} \\ \dot{\boldsymbol{L}} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N} \ddot{\boldsymbol{x}}_i \\ \sum_{i=0}^{N} \left[m_i \boldsymbol{x}_i \times m_i \ddot{\boldsymbol{x}}_i + R_i I_i \dot{\boldsymbol{\omega}}_i - R_i \big((I_i \times \boldsymbol{\omega}_i) \times \boldsymbol{\omega}_i \big) \right] \end{bmatrix} \tag{2.37}$$

$\boldsymbol{P}$ and $\boldsymbol{L}$ are respectively linear and angular momentum of the system. External generalized forces acted on the system are

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{N} m_i \boldsymbol{g} + \sum_{i=0}^{M} \boldsymbol{f_i} \\ \sum_{i=0}^{N} m_i \boldsymbol{x}_k \times \boldsymbol{g} + \sum_{i=0}^{M} \boldsymbol{p}_i \times \boldsymbol{f_i} \end{bmatrix} \tag{2.38}$$

where $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots \boldsymbol{f}_M$ are contact forces and $\boldsymbol{p}_i$ there position.
The Newton's laws dictate that

$$\begin{bmatrix} \dot{\boldsymbol{P}} \\ \dot{\boldsymbol{L}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} \tag{2.39}$$

2.4.2 Zero Moment Point

‘

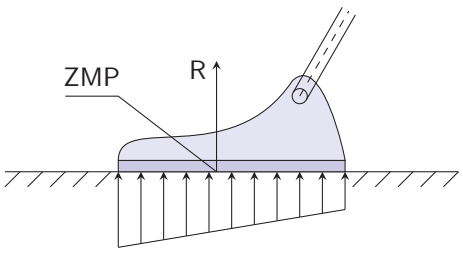


Figure 2.10: Zero-moment point

The Zero Moment Point (ZMP) (Figure 2.10), introduced by Vukobratovic and Stepanenko [VS72], is defined as the point on the surface of the foot where the resultant force R. Let $\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_M$, be the contact forces from the floor. The position of the ZMP can be calculate as

$$\boldsymbol{p} = \frac{\sum_{i=1}^{M} \boldsymbol{p}_i f_{iz}}{\sum_{i=1}^{M} f_{iz}} \tag{2.40}$$

By definition, the position of the ZMP is the barycenter of all the contact points, weighted by the component z of the contact forces. It lies therefore in the convex hull of these contacts point, hence, inside the support polygon. The total torque applied by all $\boldsymbol{f}_i$ around the ZMP can be written as

$$\boldsymbol{\tau}^{ZMP} = \sum_{i=1}^{M} (\boldsymbol{p}_i - \boldsymbol{p}) \times \boldsymbol{f}_i \tag{2.41}$$

$$\boldsymbol{\tau}_x^{ZMP} = \sum_{i=1}^{M} ((p_{iy} - p_y) f_{iz} - (p_{iz} - p_z) f_{iy}) \tag{2.42}$$

$$\boldsymbol{\tau}_y^{ZMP} = \sum_{i=1}^{M} ((p_{iz} - p_z) f_{ix} - (p_{ix} - p_x) f_{iz}) \tag{2.43}$$

$$\boldsymbol{\tau}_z^{ZMP} = \sum_{i=1}^{M} ((p_{ix} - p_x) f_{iy} - (p_{iy} - p_y) f_{ix}) \tag{2.44}$$

When the floor is flat $p_{iz} = p_z$ for $\forall i = 1 \ldots N$. We have

$$(p_{iz} - p_z) f_{iy} = (p_{iz} - p_z) f_{ix} = 0 \quad , \forall i = 1 \ldots N \tag{2.45}$$

Replace (2.40) into (2.42) and (2.43) we obtain

$$\boldsymbol{\tau}_x^{ZMP} = \boldsymbol{\tau}_y^{ZMP} = 0 \tag{2.46}$$

which explains why this point is calleddcontrol frame the zero-moment point. We going to use this property to find the position of the ZMP as a function of dynamic parameters. The moment created by the external forces on the ZMP is

Now, using the definition (2.41) and results obtained in (2.37), (2.38) and (2.39) we have

$$\begin{aligned}\boldsymbol{\tau}^{ZMP} &= \sum_{i=0}^{M} \boldsymbol{p}_i \times \boldsymbol{f}_i - \boldsymbol{p} \times \sum_{i=0}^{M} \boldsymbol{f}_i \\ &= (\dot{\boldsymbol{L}} - \sum_{i=0}^{N} m_i \boldsymbol{x}_k \times \boldsymbol{g}) - \boldsymbol{p} \times \boldsymbol{f} \\ &= \dot{\boldsymbol{L}} - M\boldsymbol{c} \times \boldsymbol{g} - \boldsymbol{p} \times \boldsymbol{f} \\ &= \dot{\boldsymbol{L}} - M\boldsymbol{c} \times \boldsymbol{g} - \boldsymbol{p} \times (\dot{\boldsymbol{P}} - M\boldsymbol{g})\end{aligned} \tag{2.47}$$

Project $\boldsymbol{\tau}^{ZMP}$ on the x and y axis and using the fact that these components are null we finally obtain the position for the ZMP

$$p_x = \frac{Mgx + p_z \dot{\boldsymbol{P}}_x - \dot{\boldsymbol{P}}_y}{Mg + \dot{\boldsymbol{P}}_z} \tag{2.48}$$

$$p_y = \frac{Mgy + p_z \dot{\boldsymbol{P}}_y + \dot{\boldsymbol{P}}_x}{Mg + \dot{\boldsymbol{P}}_z} \tag{2.49}$$

2.4.3 Cart-table model

Kajita et al. propose a simplified model of a walking humanoid robot as shown in Figure 2.11. This simplified model proves extremely effective in the context of walking pattern generation. The robot is represented by a 2-D cart-table of the same mass and at the height of the center of mass z_c. The equation of motion is written as

$$\boldsymbol{\tau} = -Mg(x - p) + M\ddot{x}z_c \tag{2.50}$$

With the result obtain in (2.46) we can write

$$p = x - \frac{z_c}{g}\ddot{x} \tag{2.51}$$

When the cart-table is static (or as per Newton's relativity, moving uniformly) $\ddot{x}$ is simply 0. We then deduce $p = x$ which means that the ZMP is at the same position as the center of mass. In this case, the static equilibrium is achieved when the center of mass, hence the ZMP lies inside the support polygon.

As $\ddot{x}$ increases, the ZMP moves further away from the center of mass.

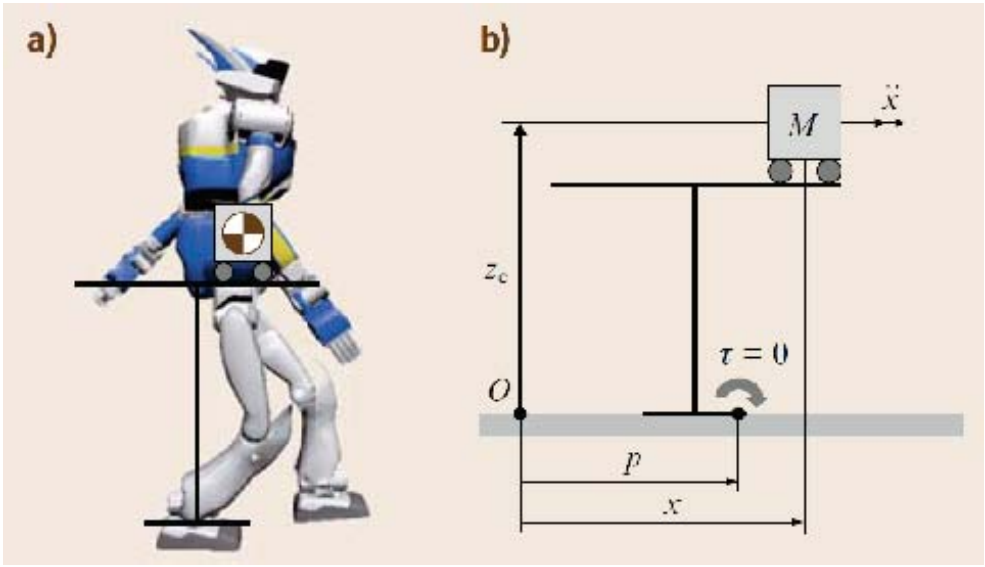


Figure 2.11: Cart table model

Above a certain level, the ZMP in equation (2.51) will lie outside of the support polygon. Since the physical ZMP by definition al way falls inside the support polygon. This condition means that the robot, or the cart-table for that matters, cannot hold a plan contact with the floor. The system is unstable.

2.4.4 Walking pattern generation

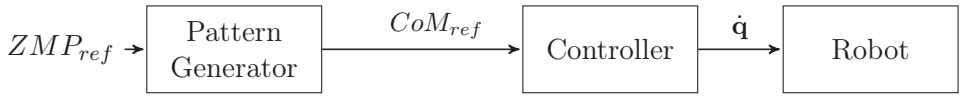


Figure 2.12: Pattern Generator

A ZMP-based walking pattern generation consists in finding feasible $\boldsymbol{q}(t)$ that satisfies a reference ZMP trajectory $p_x^d(t), p_y^d(t)$. This desired ZMP trajectory, in turn, is defined by the desired characteristic of the motion, e.g. footstep placements [MHK$^+$06], walking velocity [HDW$^+$10] etc. . In light of the control framework in subsection 2.3.3, trajectories of operational points $\boldsymbol{x}_d$ can be fed to the controller. Modern pattern generator typically produces theses intermediate operational point trajectories, notably the center of mass.

Early works such as Takanishi et al. [ToLTK90] used a Fast Fourier Transformation to obtain and resolve the ZMP desired in the frequency domain. Kagami et al. tackled this problem by writing the the dynamic equations in

the discrete time domain [KKN+02a]. The desired ZMP can then be resolved in linear time with respect to the size of data.

In this section, two other methods that are most frequently used in the framework are presented.

Pattern generation with preview control

Kajita et al. propose, for this pattern generation method, considering the jerk of the center of mass

$$\frac{d}{dt}\ddot{\boldsymbol{x}} = u_x \tag{2.52}$$

With the introduction of this new quantity, the equation of movement becomes (only the resulting equations for x direction is written, the analogy for y direction is completely straight forward.)

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\\ \ddot{\boldsymbol{x}}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\\ \ddot{\boldsymbol{x}}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} u_x \tag{2.53}$$

$$p_x = \begin{bmatrix}1 & 0 & -z_c/g\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\\ \ddot{\boldsymbol{x}}\end{bmatrix} \tag{2.54}$$

The second equation is obtained from previous results in the cart-table model of the ZMP. This is a classic dynamical system with p_x as known constant (predefined desired ZMP).

Discretize this equation for a time period T we obtain

$$\begin{aligned}\boldsymbol{x}(k+1) &= A\boldsymbol{x}(k) + Bu(k)\\ p(k) &= C\boldsymbol{x}(k)\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{x}(k) &\equiv \begin{bmatrix}x(kT) & \dot{x}(kT) & \ddot{x}(kT)\end{bmatrix}^T\\ u(k) &\equiv u_k(kT)\\ p(k) &\equiv p_x(kT)\\ A &\equiv \begin{bmatrix}1 & T & T^2/2\\ 0 & 1 & T\\ 0 & 0 & 1\end{bmatrix}\\ B &\equiv \begin{bmatrix}T^3/6\\ T^2/2\\ T\end{bmatrix}\\ C &\equiv \begin{bmatrix}1 & 0 & -z_c/g\end{bmatrix}\end{aligned}$$

The cost function is defined as

$$J = \sum_{i=k}^{\infty} \left(Q_e e(i)^2 + \Delta \boldsymbol{x}^T(i) Q_x \Delta \boldsymbol{x}(i) + R \Delta u^2(i) \right) \tag{2.55}$$

where $e(i)$ denote the tracking error $e(i) = p(i) - p^{ref}(i)$, $Q_e, R > 0$, Q_x is a 3×3 symmetric non-negative definite matrix. $\Delta \boldsymbol{x}(k), \Delta u(k)$ are the incremental state error and input.

$$\begin{aligned} \Delta u(k) &= u_x(k) - u_x(k-1) \\ \Delta x(k) &= x_x(k) - x_x(k-1) \\ \Delta e(k) &= p_x(k) - p_x^{ref}(k=1) \end{aligned}$$

The cost function penalizes the accumulated sum of the tracking error and the derivatives of system states. In other words, it optimizes over a long run a sum of the tracking error and system's efforts.

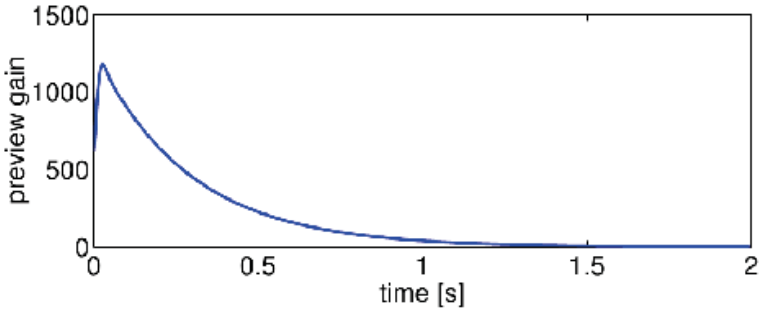


Figure 2.13: Preview control gain G_p for $T = 5ms, z_c = 0.814m, Q_e = 1.0, R = 1.0e-6$

In practice, the cost function (2.55) cannot use the infinite sum. With a previewed window be N_L, the solution for (2.55) is

$$u(k) = -G_i \sum_{i=1}^{k} (e(k)) - G_x \boldsymbol{x}(k) - \sum_{i=1}^{N_L} G_p(j) p^{ref}(k+j) \tag{2.56}$$

where G_i, G_x and $G_p(j)$ are gains calculated from Q_e, Q_x, R and system's parameters. Note that as $u(k)$ depends on the reference after instant k. The evolution of G_p indicates how the predicted future affect the present control signal. Figure 2.13 depicts this evolution. With the tested setup, the resulting signal depends on the foreseeable future up to $2s$.

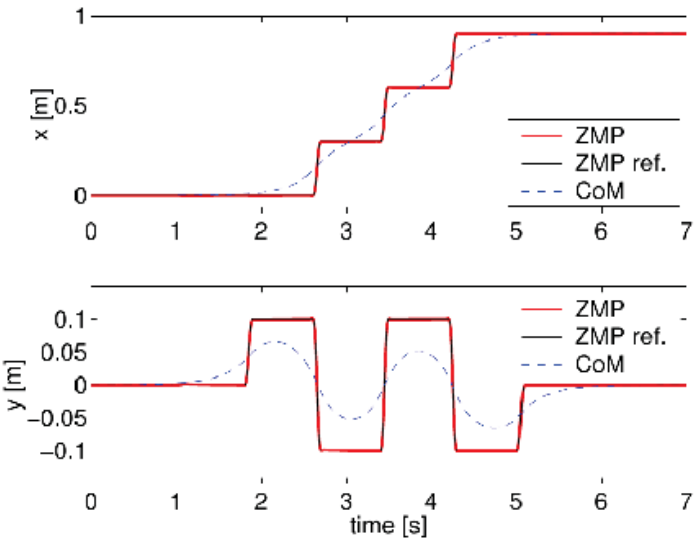


Figure 2.14: Tracking performance for preview windows 1.6s

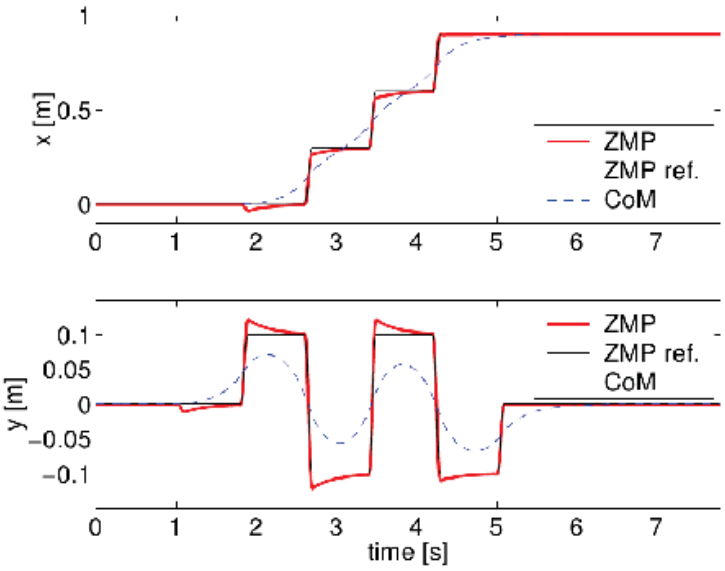


Figure 2.15: Tracking performance for preview windows 0.8s

Figure 2.14 and Figure 2.15 depict the tracking performance with two different values of preview windows. This method performs well when we take into account the future within the next 2 steps (stepping period is $0.8s$). However if this window is too short, the resulting ZMP starts to leave the reference curve. Depends on the amplitude of this error, this may lead to failure when the ZMP falls outside or close too the edge of the support polygon. Figure 2.14 and Figure 2.15 also show the stair-like shape of the ZMP which "jumps" from the one single support to another while the center of mass moves continually. Figure 2.16 shows clearly that the walk is dynamic. For a large part of the walking period, the center of mass lies outside the support polygon.

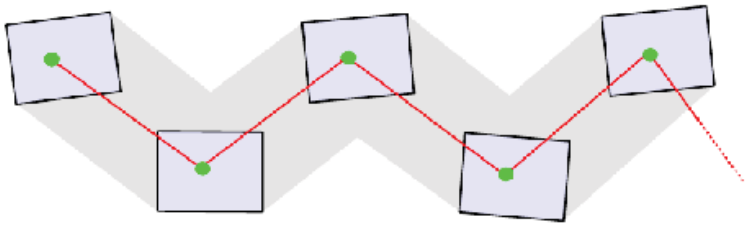


Figure 2.16: The ZMP (green zone) and the CoM (red line) during a walk

Pattern generation with polynomials

Morisawa et. al. [MHK+06] proposes a method to generate walking pattern using polynomials. In this method, the reference trajectories of ZMP and the CoM are computed simultaneously from the desired characteristic of the walk, in particular, foot placements. Let us recall the two phases of a walking movement (subsection 2.4.1). A complete walk is then divided into intervals of *single support* and *double support*.

Let $\boldsymbol{x}_G^{(j)} = [x^{(j)} \quad y^{(j)} \quad z^{(j)}]^T$ and $\boldsymbol{p}^{(j)} = [p_x^{(j)} \quad p_y^{(j)} \quad p_z^{(j)}]^T$ the position of the center of mass and the ZMP in the interval (j).

As with (2.51) but with slightly different notation (z_c was the height of the center mass *with respect to* the ZMP) we have

$$x^{(j)} = p_x^{(j)} + \ddot{x}^{(j)} \frac{z^{(j)} - p_z^{(j)}}{g} \tag{2.57}$$

This is where the polynomials come in. Suppose that the ZMP is piece-wise polynomial function of time, namely

$$p_x^{(j)}(t) = \sum_{i=0}^{N_j} a_i^{(j)} (\Delta t_j)^i \tag{2.58}$$

$$\Delta t_j = t - T_{j-1} \tag{2.59}$$

Δt_j being the elapsed time since the beginning of the segment T_{j-1}. Replace into (2.57) we obtain

$$\begin{aligned}
x^{(j)} &= V^{(j)} c_j + W^{(j)} s_j + \sum_{i=0}^{N_j} A_i^{(j)} (\Delta t_j)^i \\
A_i^{(j)} &= \left\{ \begin{matrix} a_i^{(j)} + \sum_{k=1}^{(N_j - i)/2} b_{i+2k}^{(j)} a_{i+2k}^{(j)} & , i = 0, \ldots, N_j - 2 \\ a_i^{(j)} & , i = N_j - 1, N_j \end{matrix} \right\} \\
b_{i+2k}^{(j)} &= \prod_{l-1}^{k} \frac{(i+2l)(i+2l-1)}{\omega_j^2} \\
c_j &\equiv \cosh(\omega_j \Delta t_j) \\
s_j &\equiv \sinh(\omega_j \Delta t_j) \\
\omega_j &\equiv \sqrt{g/(z^{(j)} - p_z^{(j)})}
\end{aligned} \tag{2.60}$$

where $V^{(j)}$ and $W^{(j)}$ are scalar constants. The missing coefficients, namely $V^{(j)}$'s, $W^{(j)}$'s and $A_k^{(j)}$'s, have to satisfy all the boundary conditions:

(a) Initial position and velocity of the center of mass:

$$x^{(1)}(T_0) = V^{(1)} + A_0^{(1)} \tag{2.61}$$

$$\dot{x}^{(1)}(T_0) = W^{(1)} + A_1^{(1)} \tag{2.62}$$

(b) Position and velocity of the center of mass between neighbor segments:

$$V^{(j)} \cosh(w_j \Delta T_j) + W^{(j)} \sinh(w_j \Delta T_j) + \sum_{i=0}^{N_j} A_i^{(j)} (\Delta T_j)^i = V^{(j+1)} + A_0^{(j+1)} \tag{2.63}$$

$$V^{(j)} \omega_j \sinh(w_j \Delta T_j) + W^{(j)} \omega_j \cosh(w_j \Delta T_j) + \sum_{i=0}^{N_j} i A_i^{(j)} (\Delta T_j)^i = V^{(j+1)} \omega_j + A_1^{(j+1)} \tag{2.64}$$

$$\forall j = 1, \ldots, m-1$$

(c) Terminal position and velocity of the center of mass:

$$V^{(m)}\cosh(w_j\Delta T_j) + W^{(m)}\sinh(w_j\Delta T_j) + \sum_{i=0}^{N_j} A_i^{(m)}(\Delta T_j)^i = x^{(m)}(T_m) \tag{2.65}$$

$$V^{(m)}\omega_j\sinh(w_j\Delta T_j) + W^{(m)}\omega_j\cosh(w_j\Delta T_j) + \sum_{i=0}^{N_j} iA_i^{(m)}(\Delta T_j)^i = \dot{x}^{(m)}(T_m) \tag{2.66}$$

(d) Initial position and velocity of the ZMP at each section:

$$p^{(j)}(T_{j-1}) = A_0^{(j)} - \frac{1}{\omega_j^2}A_2^{(j)} \tag{2.67}$$

$$\dot{p}^{(j)}(T_{j-1}) = A_1^{(j)} - \frac{6}{\omega_j^2}A_3^{(j)} \tag{2.68}$$

$$\forall j = 1, \ldots, m$$

(e) Terminal position and velocity of the ZMP at each section:

$$p^{(j)}(T_j) = \sum_{i=0}^{N_j}\left\{\left(A_i^{(j)} - \frac{(i+1)(i+2)}{\omega_j^2}iA_{i+2}\right)(\Delta T_j)^i\right\} \tag{2.69}$$

$$\dot{p}^{(j)}(T_j) = \sum_{i=0}^{N_j}\left\{i\left(A_i^{(j)} - \frac{(i+1)(i+2)}{\omega_j^2}\text{ß}A_{i+2}\right)(\Delta T_j)^{i-1}\right\} \tag{2.70}$$

$$\forall j = 1, \ldots, m$$

ΔT_j is the duration of segment $j/$

Let us define the following quantities

$$\begin{aligned}
\boldsymbol{y} = \; & \begin{bmatrix} V^{(1)} & W^{(1)} & A_0^{(1)} & \ldots & A_{N_1}^{(1)} & \ldots \\ V^{(m)} & W^{(m)} & A_0^{(m)} & \ldots & A_{N_m}^{(m)} \end{bmatrix}^T \\
\boldsymbol{\omega} = \; & \begin{bmatrix} x^{(1)}(T_o) & \dot{x}^{(1)}(T_o) & p^{(1)}(T_o) & \dot{p}^{(1)}(T_o) & & \\ p^{(1)}(T_q) & \dot{p}^{(1)}(T_1) & 0 & 0 & p^{(2)}(T_1) & \dot{p}^{(2)}(T_1) & \ldots \\ x^{(m)}(T_o) & \dot{x}^{(m)}(T_o) & p^{(m)}(T_o) & \dot{p}^{(m)}(T_o) & & \end{bmatrix}^T
\end{aligned} \tag{2.71}$$

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{z}_{1,1} & 0 & \dots & 0 & \dots & 0 \\ \boldsymbol{Z}_{2,1} & \boldsymbol{Z}_{1,2} & 0 & & & \vdots \\ 0 & \ddots & \ddots & \ddots & & 0 \\ \vdots & 0 & \boldsymbol{Z}_{2,j} & \boldsymbol{Z}_{1,j+1} & \vdots & 0 \\ 0 & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & 0 & \boldsymbol{Z}_{2,m-1} & \boldsymbol{Z}_{1,m} \\ 0 & \dots & 0 & \dots & 0 & \boldsymbol{z}_{2,m} \end{bmatrix} \tag{2.72}$$

and

$$\boldsymbol{Z}_{1,j} = \begin{bmatrix} \boldsymbol{0}_{2\times N_j} \\ \boldsymbol{z}_{1,j} \end{bmatrix} \quad , \boldsymbol{Z}_{12j} = \begin{bmatrix} -\boldsymbol{z}_{2,_j} \\ \boldsymbol{0}_{2\times N_j} \end{bmatrix}$$

$$\boldsymbol{z}_{1,j} = \begin{bmatrix} 1 & 0 & 1 & 0 & \frac{2}{\omega_j^2} & 0 & 0 & \dots & 0 \\ 0 & \omega_j & 0 & 1 & 0 & \frac{6}{\omega_j^2} & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

$$\boldsymbol{z}_{2,j} = \begin{bmatrix} 1 & 0 & 1 & \Delta T_j & (\Delta T_j)^2 & \dots & (\Delta T_j)^i & \dots & (\Delta T_j)^{N_j} \\ 0 & 0 & 0 & 1 & \Delta T_j & \dots & i(\Delta T_j)^{i-1} & \dots & N_j(\Delta T_j)^{N_j-1} \\ c_j & s_j & f_0^{(j)} & f_1^{(j)} & f_2^{(j)} & \dots & f_i^{(j)} & \dots & f_{N_j}^{(j)} \\ w_j s_j & w_j c_j & 0 & g_1^{(j)} & g_2^{(j)} & \dots & g_i^{(j)} & \dots & g_{N_j}^{(j)} \end{bmatrix}$$

$$f^{(j)} = \begin{cases} 1 & (i=0) \\ \Delta T_j & (i=1) \\ (\Delta T_j)^i - \dfrac{i(i-1)}{\omega_j^2}(\Delta T_j)^{i-2} & (i=2,\dots,N_j-2) \\ (\Delta T_j)^i & (i=N_j-1,N_j) \end{cases}$$

$$jg = \begin{cases} 1 & (i=1) \\ 2\Delta T_j & (i=2) \\ i(\Delta T_j)^{i-1} - \dfrac{i(i-1)(i-2)}{\omega_j^2}(\Delta T_j)^{i-3} & (i=3,\dots,N_j-2) \\ i(\Delta T_j)^{i-1} & (i=N_j-1,N_j) \end{cases} \tag{2.73}$$

Under matrix form these boundary conditions

$$\boldsymbol{w} = \boldsymbol{Z}\boldsymbol{y} \tag{2.74}$$

which leads to the classic solution

$$\boldsymbol{y} = \boldsymbol{Z}^{+}\boldsymbol{w} \tag{2.75}$$

The matrix $\boldsymbol{Z}$ has to be full rank to assure a solution. The necessary

condition on the N_j is

$$2m + \sum_{j=1}^{m}(N_j + 1) \geq 6m + 2 \tag{2.76}$$

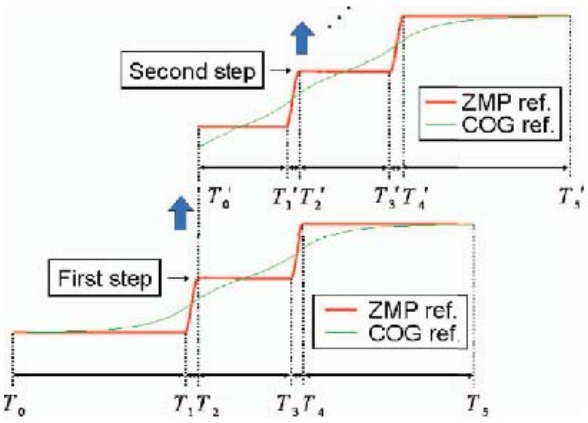


Figure 2.17: Step scheduling with analytical CoM strategy

In practice, the above analytical solution is fast enough to plan two steps within one control cycle of $5ms$. During the first control cycle, the first two steps is planned (lower part of Figure 2.17). ZMP and CoM trajectory are computed in the piece-wise polynomial form with the boundary being $T_o, \ldots, T_5$ At the end of the first step, (T_2) the robot plans the next two steps. This process goes on continuously to achieve a real-time pattern generation. It is also shown that by adjusting the duration of the current executing phase, this method is capable of changing even the next footstep in the stack. This capability proves handy to the adaptive walking scheme presented in the next chapter.

2.5 Motion generation

Let us go back to the hierarchy of tasks presented in subsection 2.3.3. This time, rather than control the robot following task-space desired derivatives $\boldsymbol{e}_d$, we try to find the pose satisfying this task hierarchy. In other words, we will be interested in particular in the robot position at the end of a (set of) task(s). In addition, inequalities will also be considered to be members in out stack of tasks. In the second part of this section, this "pose" will be used as a high level plan to be used in the controller.

2.5.1 Prioritized linear and inequality systems

Let $x \in \mathcal{R}^n$ be an vector lying inside a convex set Ω. With $b \in \mathcal{R}^m, d \in \mathcal{R}^k, A \in \mathcal{R}^{m \times n}, C \in \mathcal{R}^{k \times n}$ Let us consider a system of linear equalities/inequalities:

$$Ax = b \tag{2.77}$$

$$\text{or} \qquad Cx \leq d \tag{2.78}$$

Under the optimization form, the solution of (2.77) can be written as

$$\arg\min_{x \in \Omega} ||Ax - b||^2 \tag{2.79}$$

The resolution of (2.78) is a bit trickier. We have to introduce a slack variable w and add the corresponding constraint on it.

$$\begin{aligned} &\arg\min_{x \in \Omega} ||w||^2 \\ &\text{subject to} \quad w \geq Cx - d \end{aligned} \tag{2.80}$$

The equivalence of (2.77) and (2.78) can be found in your favorite convex optimization book.

Now, if we are to solve (2.77) and (2.78) together, the solution is then written in the combined problem

$$\begin{aligned} \%labeleq : 5 \qquad &\arg\min_{x \in \Omega, w \in \Re^n} ||Ax - b||^2 + ||w||^2 \\ &\text{subject to} \quad w \geq Cx - d \end{aligned} \tag{2.81}$$

Now that the resolution can be written by the same form, let us consider a prioritized system where each stage is either (2.77) or (2.78). Kanoun et al. [KLW+09, KLY10] proposed to resolve this system stage by stage. Indeed, in each step, the convex set Ω is the solution found in the previous stage. The solutions S_i at stage i are written under the recursive form:

$$\begin{aligned} &S_0 = \Re^n \\ &S_i = \arg\left\{ \min_{x \in S_{i-1}} ||A_i x - b_i||^2 \right\} \text{ for equality tasks} \\ &S_i = \arg\left\{ \min_{x \in S_{i-1}} ||w||^2 \qquad \text{s.t.} \quad C_i x - d \leq w \right\} \\ &\quad \text{for inequality tasks} \end{aligned}$$

In robotic manipulation, tasks are typically written in equality form, such as reaching task, gaze task, or any operational point task for that matter.

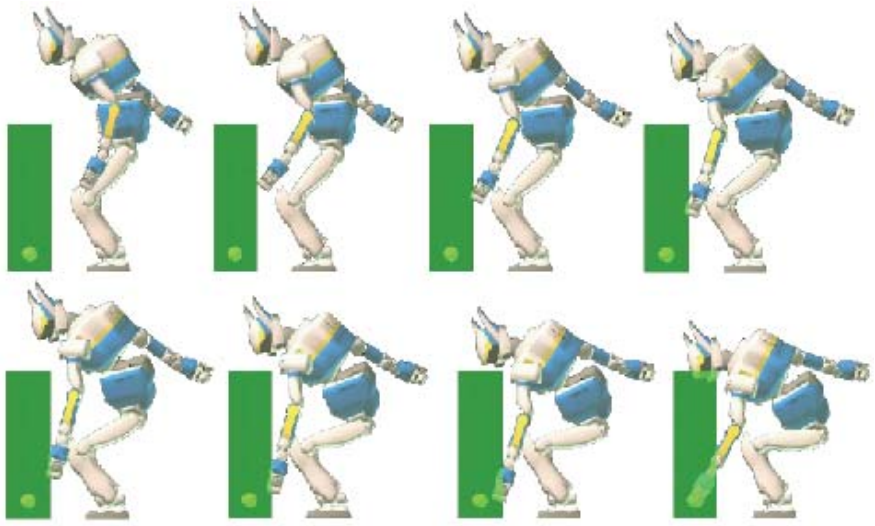


Figure 2.18: An example of inequality task at lower priority than an equality task [KLW+09]

Constraints such as collision avoidance, joint limits are in general in inequality form. Other than that, a task can also exist in inequality form as demonstrated in [KLW+09].

In the example presented in Figure 2.18, the robot has to reach to a goal and at a lower priority keep the hand outside of the vision field of the camera. The constraints and tasks, both in equality (eq) and inequality (in) form, applied on the robot in descending order of priority are:

- (in) Joint limits
- (in) Self collision avoidance
- (in) CoM constraints (must lie inside the polygon support)
- (eq) Hand reaching task
- (in) Vision task
- (eq) Reference posture task

We notice that the the robot keep the hand outside the vision field as long as possible. The vision task is violated at the end of the movement when it becomes necessary for the hand to reach the goal.

2.5.2 Application to footstep planning

In the previous section, a hierarchy of tasks have been resolved by writing a prioritized system of optimization problems. In this section, the same approach

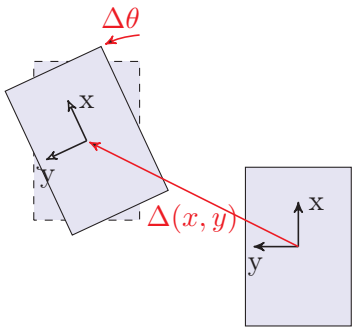


Figure 2.19: Representation of one step

is applied for locomotion by expanding the kinematic chains with virtual parts corresponding to the footsteps. Let us consider a single footstep in Figure 2.19 represented by the displacements in x and y axes $\Delta x, \Delta y$ and the rotation angle $\Delta\theta$. For now we are only dealing with a flat surface, z is constant everywhere the robot places its feet. Providing that the walk is stable and depending on the physical characteristic of the robot and the the pattern generator in service, there are boundaries on the step variable $\Delta x, \Delta y, \Delta\theta$

$$\begin{aligned} \Delta x_{\min} \leq \Delta x \leq \Delta x_{\max} \\ \Delta y_{\min} \leq \Delta y \leq \Delta y_{\max} \\ \Delta\theta_{\min} \leq \Delta\theta \leq \Delta\theta_{\max} \end{aligned} \tag{2.82}$$

If the robot is to step one step and stay in equilibrium at the new position, the CoM must stay in the new support polygon. Suppose that in this new position, the robot has to perform an addition step, say a reaching step, we can now build a new prioritized system of equality and inequality task. The additional constraints compare to to the example in Figure 2.18 are

- (in) Step limit as in (2.82)
- (in) Self collision constraints of the two step.
- (in) CoM constraint (must lies inside the *new* support polygon)

The new variable of this optimization problem is

$$\bar{q} = [q_1 \quad q_2 \quad \dots \quad q_n \quad \Delta x \quad \Delta y \quad \Delta\theta]^T \tag{2.83}$$

General case of a movement involving k steps Building the an optimization problem with the new variables as in (2.83) was presented in of [KLY10]. The main idea of this approach is to add a virtual joint (Figure 2.20) with three degree of freedoms for each step. $\Delta x, \Delta y, \Delta\theta$ now become the configuration of the new virtual joints on the augmented robot. Suppose that

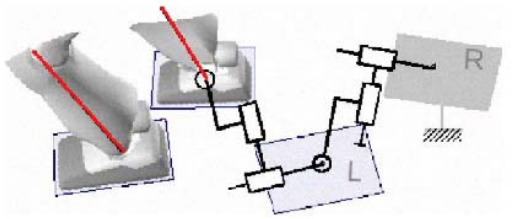


Figure 2.20: Virtual kinematic chain

the robot has n degrees of freedom and is to execute k steps, the resulting virtual kinematic chain adds $3k$ degrees of freedom to the real robot to form a $n + 3k$ d.o.f. kinematic chain.

Constraints on how far the robot can physically step (2.82) or turn become joint limits for theses new joints. The constraints that the robot should not step one foot on one another becomes auto-collision avoidance. On this new robot with this set of constraints in addition, a set of appropriate tasks can then be applied, namely:

- inequality constraints, in order, joint limits, projection of the center of mass, self-collision avoidance of robot, self-collision avoidance for the virtual manipulator, position and orientation of the supporting foot
- object manipulation task. e.g. grasp, reach for an object with 2 hands, etc.
- parallel task for the upper body during walking,
- gaze task (keep objects of interest in the vision field).

Figure 2.21 depicts how the upper body task "attracts" the augmented robot hence initiate footsteps. The last n degrees of freedoms in the result represents the final posture, and the final position of the virtual kinematic chain represents footsteps. The complete motion in configuration space can then be found by passing the footsteps to a preview controller which output trajectories for left foot, right foot and the center of mass. These trajectories in turn, in addition with the upper body task at the final step can be given as equality tasks to the prioritized stack. The resulting motion is therefore never computed before-hand. It is instead the result of the control architecture.

Determine the number of steps

In the previous section, it was assumed that the number of steps k was known before hand. However, the target detected by the vision system can be found in a wide range of distance. Therefore, it is difficult to guess in advance how

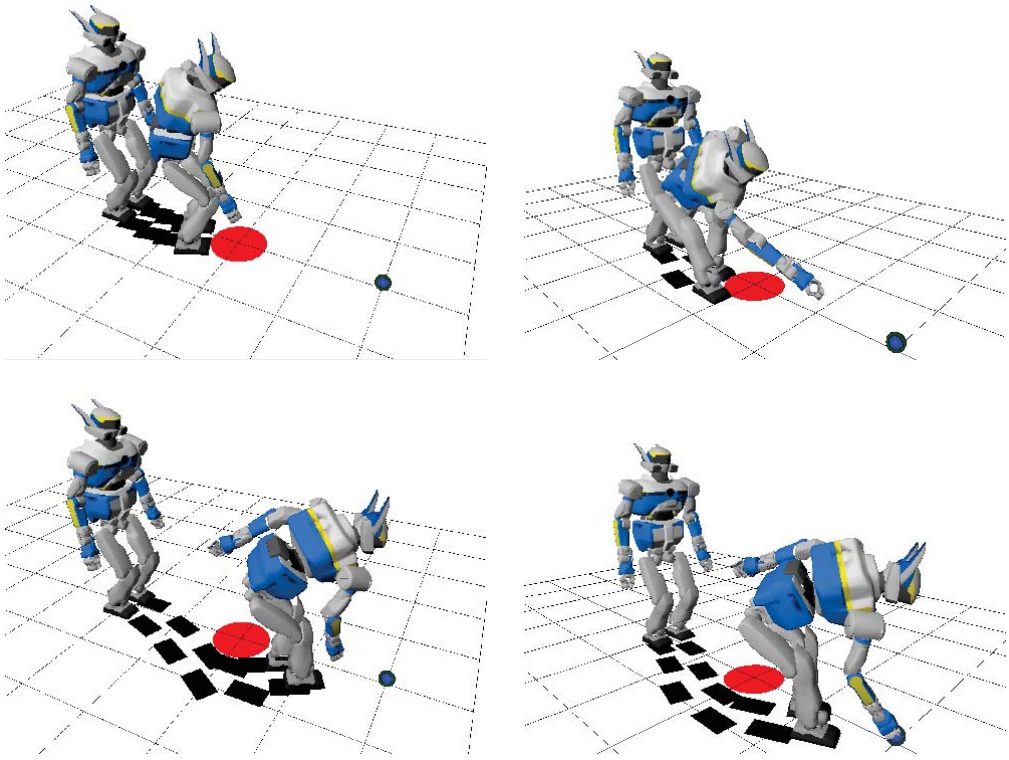


Figure 2.21: Deployment of virtual kinematic chain

many steps the robot should make. One important point in this section and also the first modification made on this paper to the original method is the fact this parameter k can be found automatically by the optimizer. The algorithm is summarized as follows:

Algorithm 1 Footsteps planning with variable number of steps

Require: initial position.
Require: manipulation task.
Require: obstacle positions.
Ensure: footprints and final posture Initialize solver with 0 virtual link.
  repeat
    Solve the optimization problem.
    if Reach local minimum **then**
      Add new virtual link
    end if
    Check manipulation task error
  until goal reached

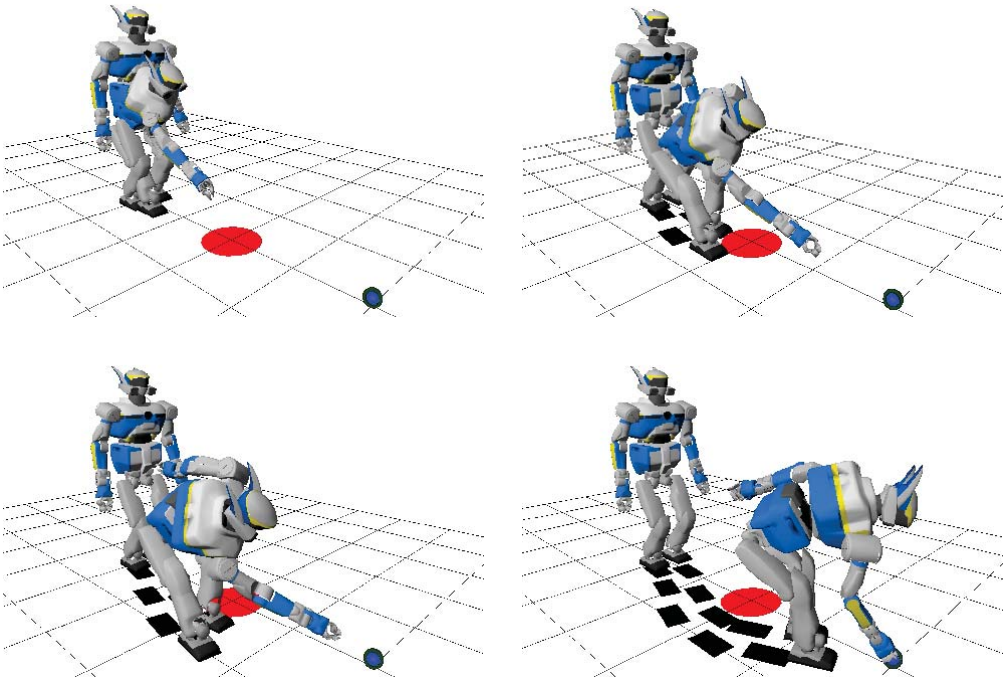


Figure 2.22: Find the number of steps automatically by the optimizer

Figure 2.22. depicts the exploration of the virtual kinematic chain in space.

2.6 Integration on the HRP-2 robot

Previous sections in this chapter detailed a global view of different components in the framework. This section provides more details on how these components are deployed on a real platform, such as the HRP-2 robot.

2.6.1 Controller

The choice of the controler described in section 2.3 is motivated by current design of humanoid robots. Such robots, including the HRP series, are precise machines which provide high-quality measurements, e.g. the errors for joint encoders are negligable. As a result, the robot state, and therefore the Jacobians can be calculated with almost perfect precision. In practice, a good calibration on our HRP-2 robot can be achieved with less then $10^{-3} rad$ of error for the joint encoders. This allows, for example, a precise reaching task only by position feedback (position-based servoing).

The formalism of control described here can be extensible to the cases where the robot's kinematics and dynamics are not or cannot be modeled

precisely. Some enhancements, however, are needed to those cases. First, the robot has to learn [Atk89, AMS97a] its kinematics and dynamics to correct the errors in the model and improve the analytically calculated task Jacobians. Second, closed-loop manipulation with perception has to be more intensively used. A representation of tasks in the image space is a possible remedy to imprecision in the dynamical model of the robot. Both image-based servoing and position-based have their avantanges and disadvantages [Cha98]. The perfect knowledge of the robot kinematics and dynamics here motivates the design of the perception system detailed in chapter 4.

2.6.2 The Stack Of Tasks

The graph of entities

Mansard et. al. [MSEK09] put forwards the pricinples presented in section 2.3 the StackOfTasks software architechture. The flow of information is designed as signals being transfered in a graph of entities, similar to a SimuLink graph. Mathematical forumulations, such as the calculation of Jacobian, are represented by entities, requires input signals (input information), and provides output signals (results). For example, the Jacobian J of a given task is the output signal of the entity *dynamic* which needs robot states q as the input signal. It is worth noting that at this point, no middleware is involved, all the entities are created within a same process. Each type of entities is organized into a C++ class that can be loaded using a dynamic library. This architecture of plugins allows growing the graph rapidly.

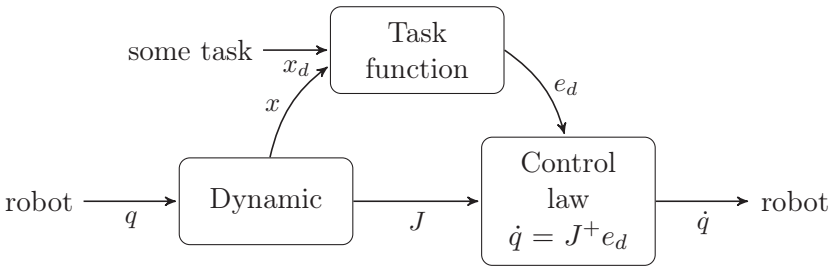


Figure 2.23: Computation of motor commands from system states in the Stack O fTasks

Figure 2.23 shows a minimal graph in which the control law $\dot{q}$ is computed for a given task x_d. At each control loop, the robot requests for $\dot{q}$ which triggers the calculation of all signals that $\dot{q}$ which, in turn, depends on: J, e_d, x etc. A real-life graph is similar to Figure 2.24. The graph now is much more complicated due to the size of the stack of tasks and their computation. The fundamental stays unchanged, at each control loop, the requested control input triggers all the depending computation, hence triggers the information flow in the whole graph.

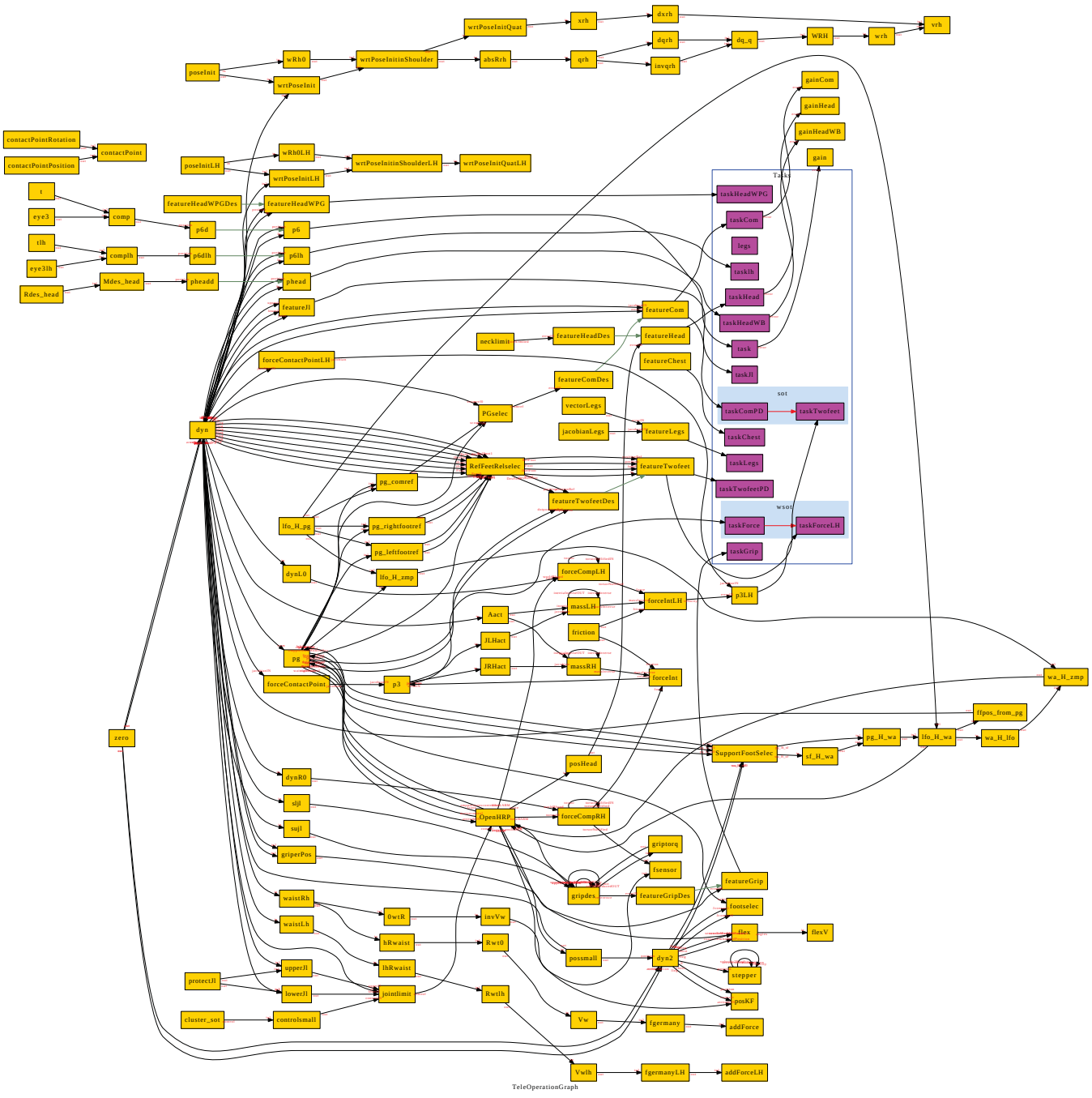


Figure 2.24: A graph of entities in action [MSEK09]

Walking control using the StackOfTasks

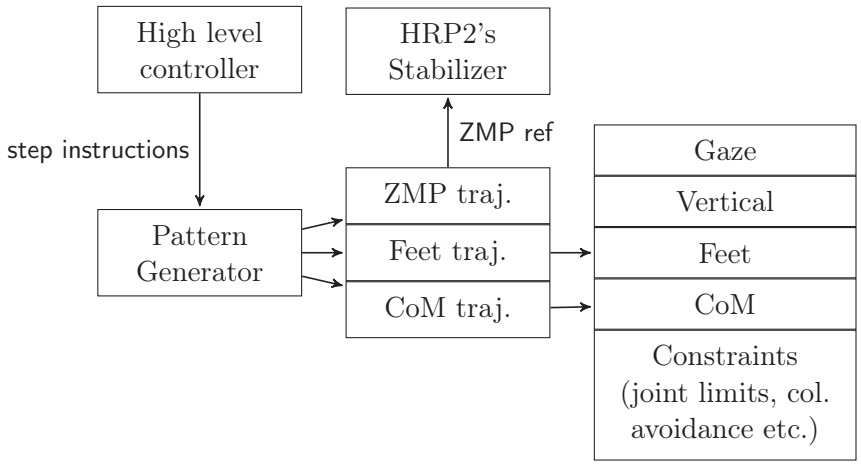


Figure 2.25: Integration of the pattern generator inside the StackOfTasks

As presented earlier, the pattern generator can output CoM and ZMP trajectories. These trajectories satisfy the stability condition imposed by the cart-model model. The controller presented in subsection 2.3.3 can now perform the tracking task on the desired CoM trajectory. The last piece of the puzzle is now assuring that the feet take off and land at the desired positions. Again, a polynomial function can be used to calculate desired foot trajectories. These trajectories are fed to the controller to obtain the full walking movement.

The two pattern generators presented in the previous section and other algorithms are integrated into the StackOfTasks [MSEK09] thanks to a modular architecture [SVW+08] which allows the quick and efficient implementation of different type of pattern generators. Figure 2.25 details the information flows during a walking experiment. A high-level controller send "commands" such as foot placements, walking speed to a pattern generator. This entity produces CoM and foo trajectories to feed into the StackOfTasks as position task in the Cartesian space. Figure 2.26 depicts these trajectories during an experiment. The desired ZMP is fed to the stabilizer which uses this position and the force sensors to assure the stability of the planned movement.

2.6.3 Footstep planner - controller connection

The previous section described the design of the Stack Of Tasks, how the internal entities were created and interacted with each other. This section details the connection between the Stack Of Tasks and the outside world. These outside modules usually involve 'slow' components vision which talk to the controller in a asynchronous manner. In practice with the robot HRP-2,

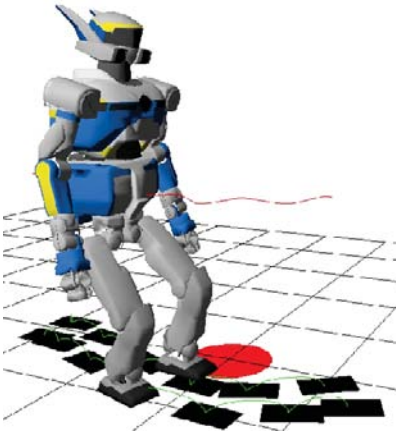


Figure 2.26: CoM (red line) and Foot trajectories (blue lines) tracking during a walking experiment

planning, optimization and vision run on a standard Linux distribution while the controller runs on a RT-Linux kernel.

Figure 2.27 depicts the additional items that has been added to the existing Stack Of Tasks to connect it to an external footstep planner/optimizer (commonly called planner in the figure). All request and response between the two PC use CORBA [Pop98], The *Corba Server* entity plays the role of the bridge between normal entity of the Stack Of Tasks and external modules. The the role of *footstep manager* is to stock the stepping plan given by the planner and update that to the pattern generator in a timely manner.

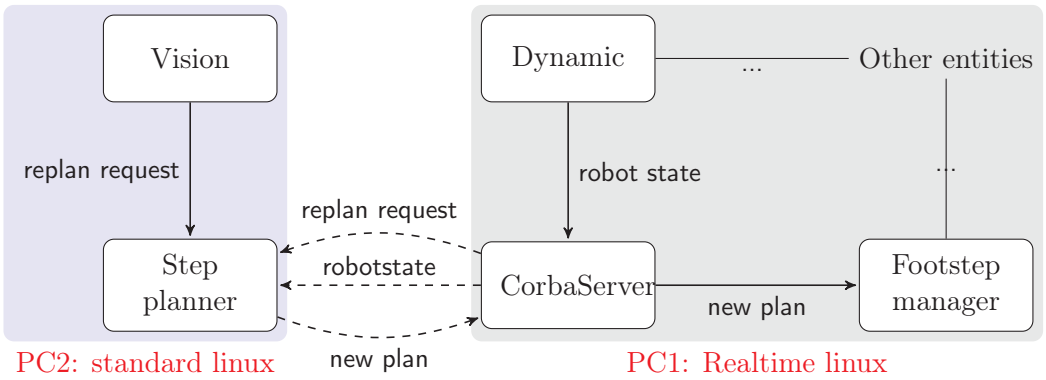


Figure 2.27: Controller-planner connection. Dashed arrows represent CORBA connection

Chapter 3

Reactive locomotion by deformable footsteps

3.1 Reactive walking

In the previous approach, Kanoun et al. used the resolution of the inequality/equality systems twice to obtain a complete movement. The footsteps obtained by the sliding robot in section 2.4 were fed to a pattern generator to obtain a feasible CoM reference trajectory. This trajectory, in turn, was fed to a second solver along with appropriate upper body tasks (e.g. well timed hand and gaze tasks). The whole process took around one minute on a single core CPU clocked at 3GHz. Since a complete movement was needed before any movement on the robot, all the computation was performed offline.

Other than the computation time, the strategy of evaluating everything once offline (top and middle part of Figure 3.1) has an additional drawback. That is, it is virtually impossible to apply on the real robot. The first reason is that the sensors are usually not good enough to observe the target just once, compute, then execute the full motion successfully; especially in the case of using the robot's camera as the unique perception sensors. The error with a stereo system is typically high when the target is far away and small when the target is close. Without observing the target as the robot approaches it, the robot cannot successfully grasp the object.

What if another kind of sensor is used in lieu of the cameras, e.g. a motion capture system ? Even if this new sensor system has negligible errors, say a few millimeters for grasping purpose, there are still other factors that complicate our movement. Indeed, since the free flyer (the waist in the case of HRP-2) is not actively controlled but moves in space via the feet, a humanoid robot typically drifts while working. On the HRP-2 robots, this is particularly true for long movements and/or when the robot has to turn sharply. Without any correction, even with perfect sensors, a sequence involving walking and precise manipulation is doomed to fail.

To remedy this problem, a reactive approach should be used. We will observe the environment continuously to correct both the perception errors and the drift of the robot. Only the first stage of the resolution of the inequality and equality system is carried out. We take the intermediate results (i.e. the sliding robot) and put it in a closed loop with the perception system (the last group in Figure 3.1).

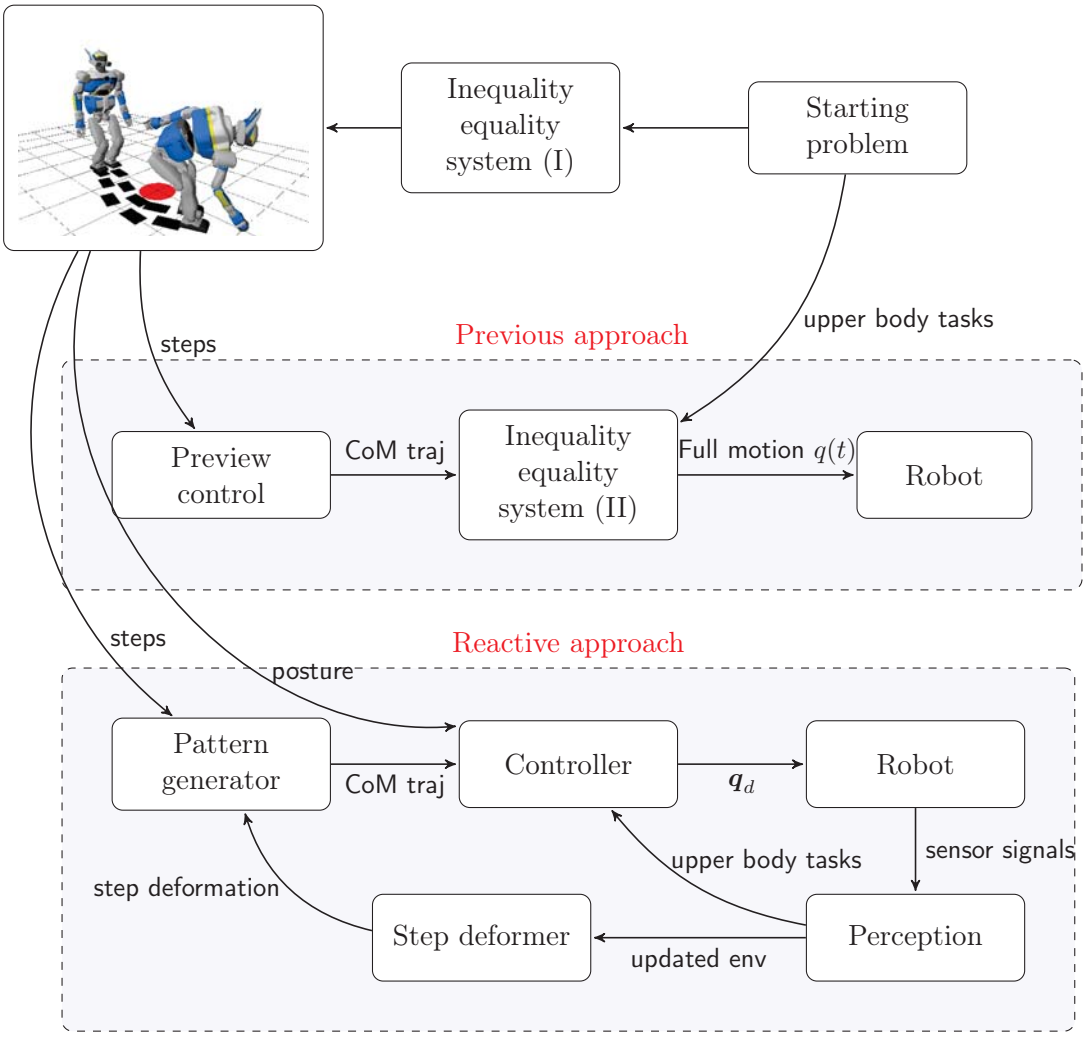


Figure 3.1: Reactive walking vs. offline computed walking

3.2 Floor representation

In section 2.5 of the last chapter we talked about how to represent footsteps as parts of the robot kinematic chain. What about the interaction between

these footsteps with the environment?

Let us consider a configuration of the floor as in Figure 3.2, where the robot has to navigate around circle obstacles.

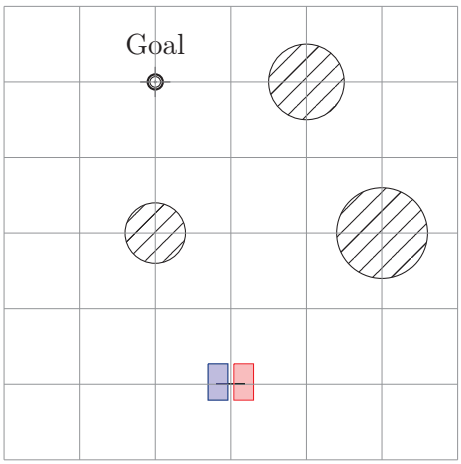
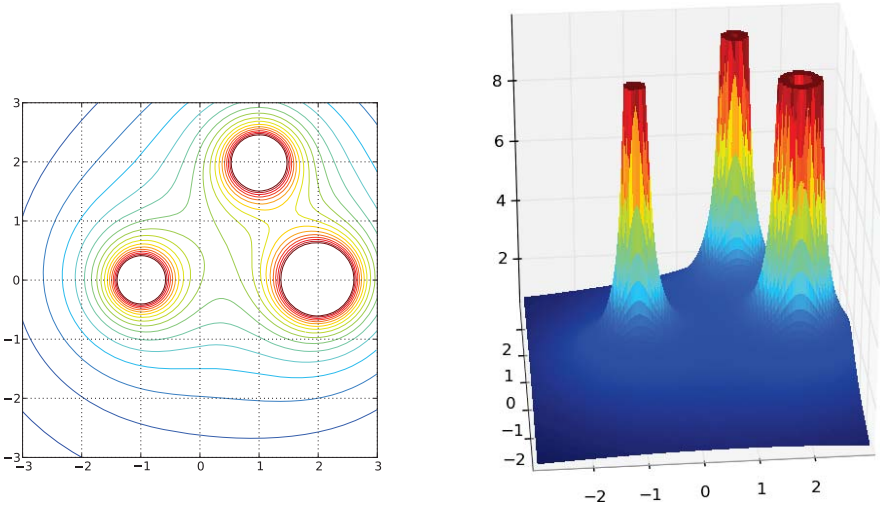


Figure 3.2: A simple configuration of the floor

3.2.1 Continuous potential field



(a) Contour lines (b) Potential values

Figure 3.3: Potential Field

To prevent the robot from stepping into obstacles, a potential field $f(x)$ can be created around obstacles. Figure 3.3a and Figure 3.3b depict the contour

and the shape of such potential field of the form

$$f(x_i) = \sum_j \frac{C_i}{d_{ij}} \tag{3.1}$$

where C_j are a constants, and d_{ij} is the distance from the footstep i to the obstacle j. for illustration purpose, the infinity values of the potential around obstacles are maxed-out at a displayable value.

The optimization problem is written on the footsteps with this potential added to the objective function. At stage k which corresponds to the equality task $Ax = b$, i.ethe final footstep has to reach a predefined goal, we solve the following optimization problem

$$\begin{aligned} \arg\min_x ||Ax - b|| + \sum_i f(x_i) \\ \text{subject to } x \in \mathcal{S}_{k-1} \end{aligned} \tag{3.2}$$

If we go ahead and solve (3.2), the resulting footsteps can appear "unnatural" with portions in which the robot seems to on a line. The reason behind this is the fact that the optimizer converges with the footsteps at lowest potential possible. They therefore fall when possible to the "valleys" in the potential field in Figure 3.3b. To fix this issue, we can either add to the cost function the terms relating a footstep configuration to the "preferred" configuration (the robot standing straight, two feet parallel at a predefined distance [0.19m for HRP-2], co-linear with the walking direction, at equal distance to the waist).

3.2.2 Obstacle representation by inequalities

Using previous representation, we will have to choose the potential function and tune

- the shape of the potential function (order),
- the constants for the potential function,
- the cost function for the step configuration.

Furthermore, any slight modification in the obstacle's position results in a modification of the potential field hence the results. This modification might happen on an obstacle far away and has physically little effect on our stepping sequence.

Instead of the potential field, we can again use the inequality representation to exclude the regions around obstacles. Obstacle avoidance for footsteps become an inequality constraint on the augmented robot. Figure 3.4 depicts such a representation. The prohibited regions are obstacles plus a buffer zone to assure some security.

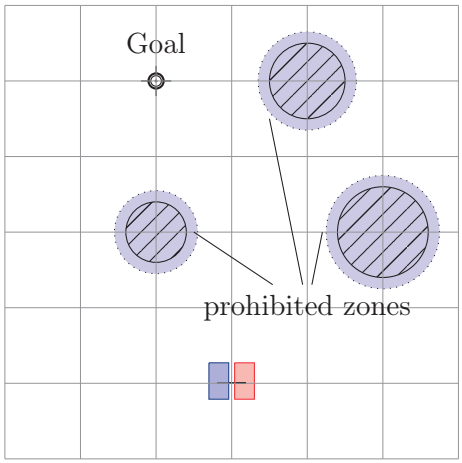


Figure 3.4: Constraints on foot placements

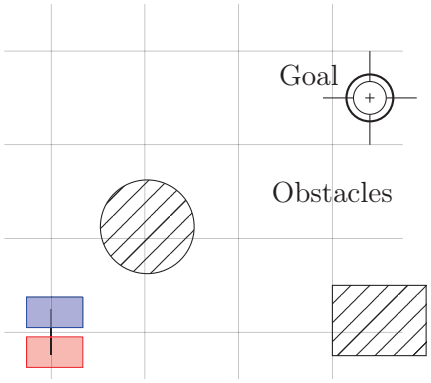
Back to our previous previous optimization problem in chapter 2, the system of equalities (eq) and inequalities (in) written on the augmented robot (robot + footsteps) become:

- (in) Joint limits
- (in) Self-collision avoidance
- (in) (new) Collision avoidance between footsteps and obstacles
- (in) CoM constraints (whose projection must lie inside the polygon support)
- (eq) Upper body manipulation task
- (eq) Reference posture task

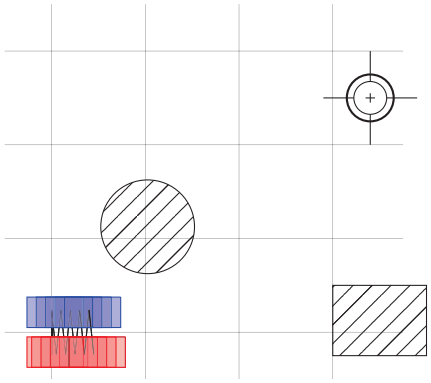
For the rest of this chapter, this representation by inequalities is chosen over potential field approach since it represents more rigorously obstacles and requires less parameter tuning.

3.3 Environment adapted local footstep deformation

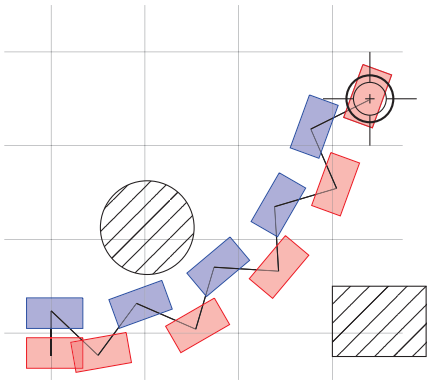
For step adaptation purposes, we can omit the degree of freedom corresponding to the robot and write directly our problem on the vector $\boldsymbol{q} \in \mathcal{R}^{3k}$ with k being



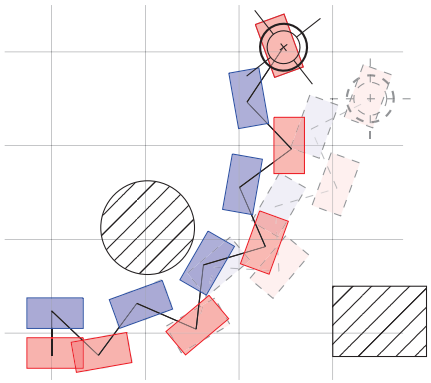
(a) Initial position, obstacles and goal



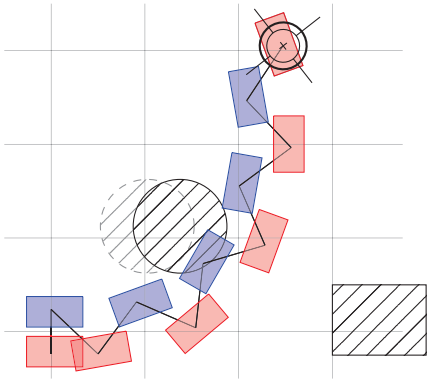
(b) Initial footstep sequence



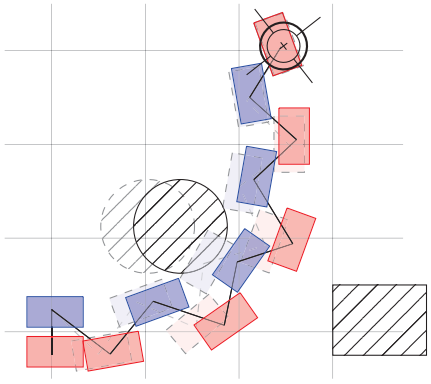
(c) Convergence of the sliding robot to goal



(d) Deformation of footsteps due to change in goal location



(e) Change in position of one obstacle



(f) Deformation of footsteps due to change in obstacle location

Figure 3.5: Step deformation by local action on footsteps

the number of steps.

$$\boldsymbol{q} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta \theta_1 \\ \vdots \\ \Delta x_k \\ \Delta x_k \\ \Delta \theta_k \end{bmatrix} \tag{3.3}$$

The set of equality and inequality tasks on the footsteps contains

- (in) $3k$ footstep limit constraints on $\Delta x, \Delta y, \Delta \theta$.
- (in) $k-1$ self-collision constraints for successive footsteps.
- (in) kn constraints with each step and each n prohibited zone (obstacles)
- (eq) position task for designated step (last step for Figure 3.5)

By construction, a modification in the goal results in a deformation , so does the displacement of one of the obstacles. Figure 3.5a to Figure 3.5f depict this sort of step deformation. We change the movement by acting directly on the footsteps. The upper body of the humanoid robot becomes completely transparent to the solver.

3.4 Task-driven deformation

In previous section the upper body is transparent to the footstep deformation calculation. In this section, at the outermost layer, footsteps ,in turn,become transparent. Using the concept presented in section 2.5, a footstep is just another joint on a robot. Instead of working on a robot of n degrees of freedom, the solver now deals with $n+3k$ degrees of freedom altogether. The deployment of the sliding robot is the direct consequence of the task applied on an operation point of the augmented robot. In other words, footsteps are not computed explicitly, but are rather a *side effect* of the manipulation task.

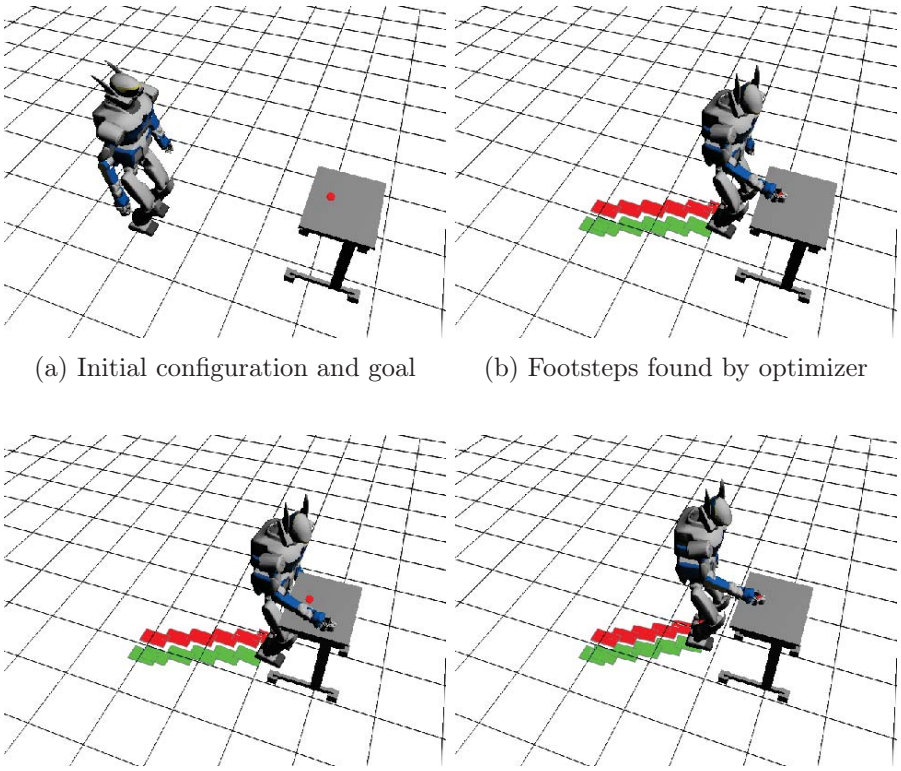
In the same spirit, the step deformation is a direct consequence of the modification in the task. Algorithm 2 shows a pseudo code of the adaptation scheme by task. The optimization is written on the states q^a of the augmented robot. Whenever the scene changes and affect the manipulation task, a new optimization is written. Note that if the changes are local (see chapter 6, the new optimization problem is done in little time, hence allow the robot to react quickly with those changes. Each time the solution on the augmented robot states q^a is modified, the stepping sequence in the controller (the footstep manager in subsection 2.6.3) hence adapt the footsteps.

Algorithm 2 Task-driven footstep adaptation

Require: Initial conditions $q_0 \in \mathcal{R}^n$
Require: Number of steps to make k
Require: Task $\mathcal{T}$
Ensure: Achieve task $\mathcal{T}$
Build the augmented robot
Build the optimization problem on the e augmented robot
Find the initial sequence $q_0^a \in \mathcal{R}^{n+3k}$
Update the desired footstep to the controller
current step number = 0
while current step number $< k$ (there are pending steps) **do**
  check position of target
  check current step number k_1
  if Goal position change **then**
    Build the new optimization problem on the augmented robot
    Freeze the first $3(k_1 + 1)$ dof in q^a
    Find the *new* sequence $q_0^a \in \mathcal{R}^{n+3k}$
    Update the desired footstep to the controller
  end if
end while
Stop locomotion and execute manipulation task

Figure 3.6 depicts a situation where the goal is moved during the movement and the corresponding footstep deformation

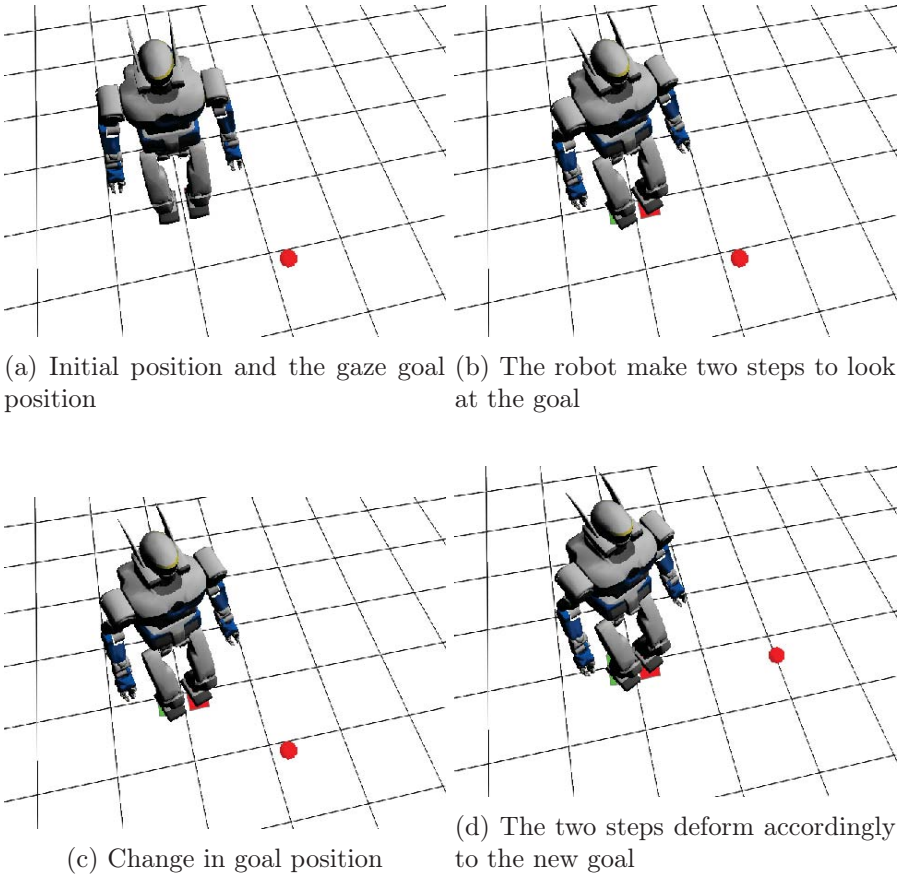
Figure 3.7 depict locomotion generated by a gaze task and step modification when this gaze task is modified.



(a) Initial configuration and goal (b) Footsteps found by optimizer

(c) Change in goal position (d) Deformation due to task change

Figure 3.6: Step deformation by task modification



(a) Initial position and the gaze goal position

(b) The robot make two steps to look at the goal

(c) Change in goal position

(d) The two steps deform accordingly to the new goal

Figure 3.7: Step deformation by gaze task modification

Chapter 4

Perception with stereo cameras

4.1 Overview of modules and middleware

In this section we are interested in the sensor-based manipulation infrastructure. We will visit different modules and talk about their roles in the system and the connections between them.

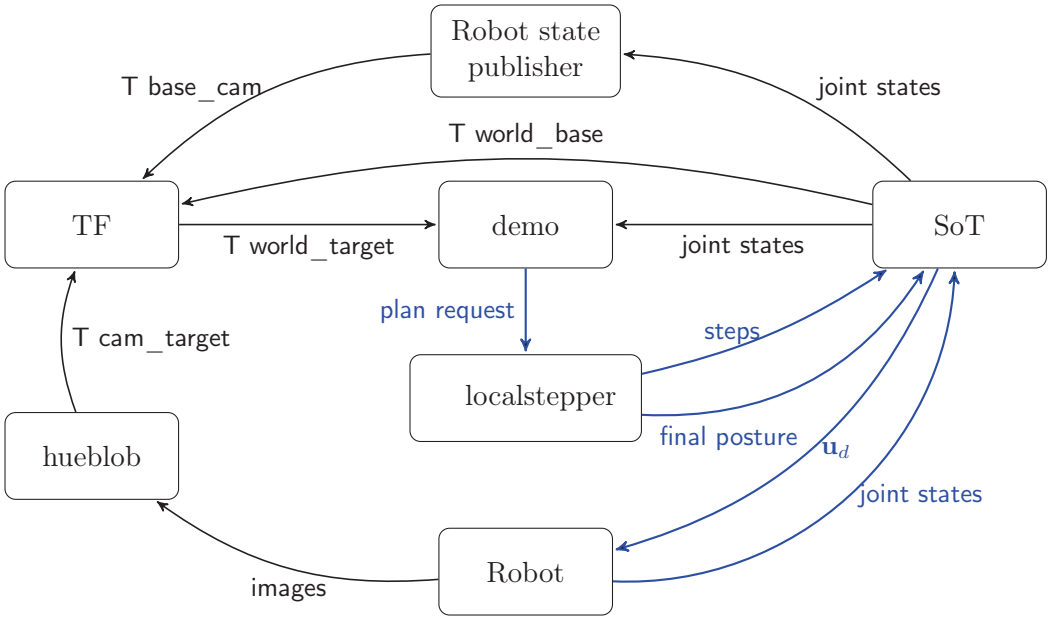


Figure 4.1: Communication between modules
Color code: black connection for ROS, blue connection for CORBA

Interactive demontrations on the HRP-2 robot such as those presented in chapter 5 and chapter 6 typically consist of the following components:

- The controller (the StackOfTasks or Sot): the most critical part, sends

low-level commands, runs at high frequency ($200Hz$ for HRP2).

- The perception module (hueblob): uses stereo cameras mounted on the robot's head, provides tracked objects' positions.
- Planning module (localstepper): usually most expensive computations, responsible for the generation of either full joint-space movements or high level action plan, such as the case of localstepper.
- The supervisor (Demo): The finite state machine which is responsible to ensure the logic of the demonstrations.

4.2 Perception module

Along with the adaptation aspects presented in chapter 3, this framework has a pronounced vision components. This section focuses on the details of the perception module (hueblob).

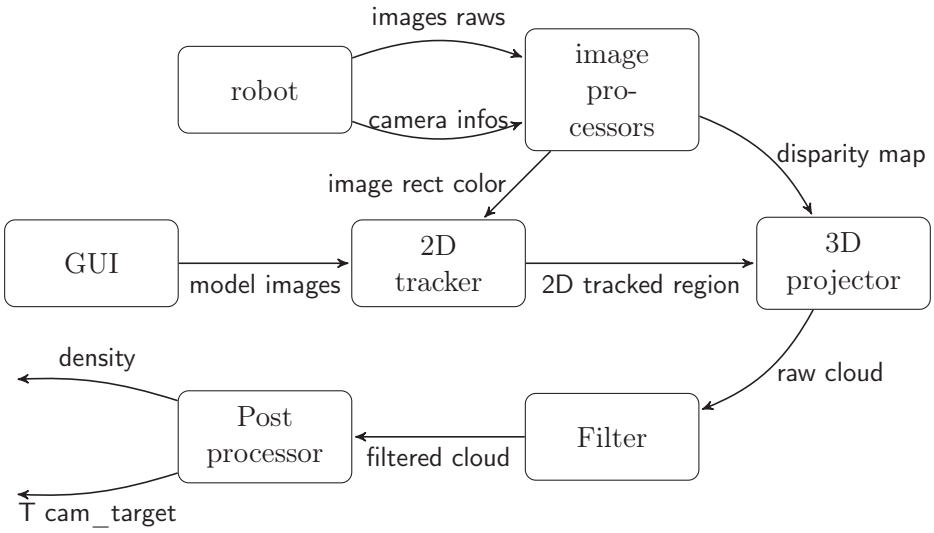
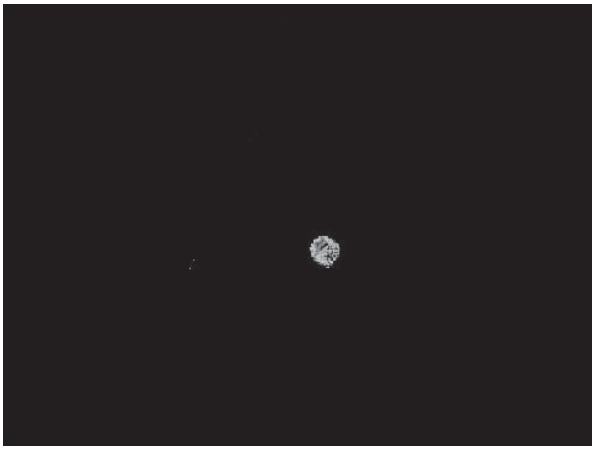


Figure 4.2: 3D Object tracking using CAMShift algorithm

Figure 4.2 depicts the detailed message flows through hueblob. At first, the target object is tracked using the CAMSHIFT algorithm [Bra98] on a single rectified image (HSV scale). Red Green Blue (RGB) color system is very sensitive to lightning condition. A dark scene reduces all three components of an object which makes the tracking difficult in RGB space. On the other hand, the Hue Saturation Value (HSV) color work much better. The hue value (color) of the object is separated from the saturation (color concentration) and brightness. This component is therefore less affected by light conditions and can faithfully represent out objects.

(a) Model

(b) Sample image



(c) Back projection to model histogram

Figure 4.3: Back projection on model's hue histogram

At first, the histogram is calculated from the model(s) (Figure 4.3a. The rectified image from the robot (in HSV) is then back projected into the model's histogram. In clear, for each pixel in the test image, the hue value is looked up in the corresponding bin of the histogram and written to the back projected image. This value can be thought as how close the pixel in question matches the model, it represents the probability that a pixel in a test Image belongs to an object area. Figure 4.3c depicts such a back projection where the lightest area corresponds to the most probable ob jet area.

Next, the tracking algorithm consists in finding a window

- whose center coincides with the center of the probability value (back projected value) inside the window

- whose size adapts to the total of the probabilities in the window.

More concisely, let $I(x, y)$ be the back projection value at (x, y), in each search window, let us define the zeroth moment:

$$M_{00} = \sum_x \sum_y I(x, y) \tag{4.1}$$

the mean location (centroid of the probabilities)

$$\begin{aligned} x_c &= \frac{M_{10}}{M_{oo}} = \frac{\sum_x \sum_y xI(x, y)}{\sum_x \sum_y I(x, y)} \\ y_c &= \frac{M_{01}}{M_{oo}} = \frac{\sum_x \sum_y yI(x, y)}{\sum_x \sum_y I(x, y)} \end{aligned} \tag{4.2}$$

The pseudo-code for the CAMShift algorithm which derives from the makeshift algorithm [Che95] is:

1. Choose a location for the search window
2. Makeshift
   - Choose window search size
   - Compute the mean location for the search window ((4.2))
   - Center the window the the mean location
3. Adapt the search window size to the zeroth momentum M_{00}
4. Repeat step 2,3 until convergence

Modify model on the fly

Generally speaking, the CAMShift method is robust with respect to light condition, as long as the hue of the object does not vary significantly. This means that the scene can be lighter or dimmer but if the main color of the light source is unchanged, then the initial model works without any problem. In practice, these are cases where this assumption is not true. In hueblob, the monitor user interface (Figure 4.4) allows a modification of the model during experiments. This possibility proves extremely helpful for example when the experiment setup is moved from one scene to another. In 2012, the robustness of the software has been tested when the whole framework is tested in a different place then the usual location in our laboratory. Despite the fact that the light color is completely different, hueblob tracks the object robustly thanks to the new model added on the fly.

Once the object is tracked in the 2D-image, hueblob utilizes the disparity image to extract the 3D points. After filtering out outliers, one obtains a point cloud of the tracked object. The final outputs are

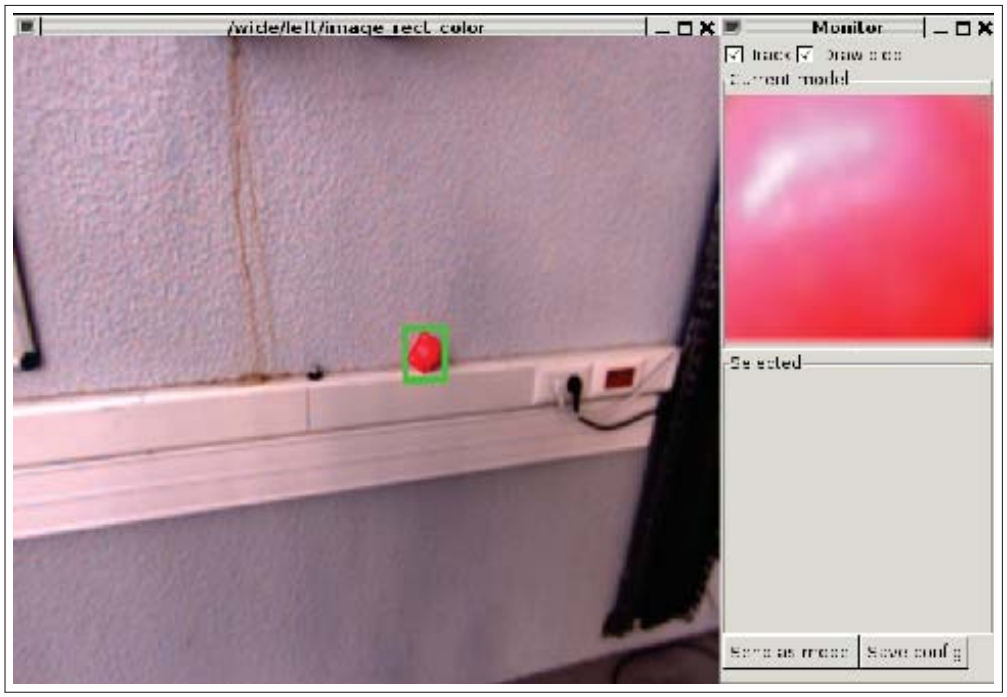


Figure 4.4: Hueblob interface

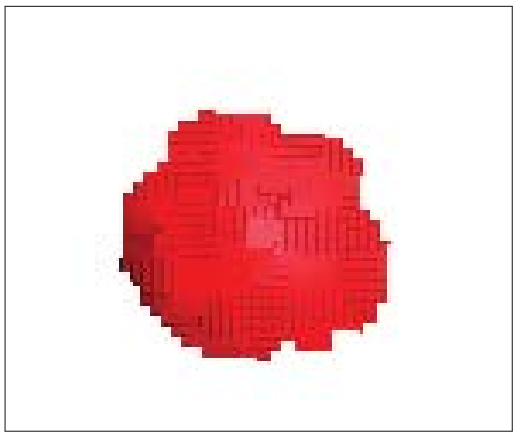


Figure 4.5: 3D points on the tracked object

- the position of this point cloud (Figure 4.5)
- and this density of object, i.e. the ratio of the number of points in the tracking window that has a valid disparity and window size.

4.3 Automated extrinsic calibration for humanoid robots

As shown in Figure 4.1, hueblob only computes the transformation from the camera to the target. To obtain the target position in the world coordinates, the position T_{bc} of the cameras (c) with respect to the robot base b is needed.

$$T_{bc} = T_{bh}T_{hc} \tag{4.3}$$

The transformation T_{bh} from the base to the last head joint h can be computed precisely given that the joint encoders are precise. The missing piece here is the position of the camera in the head of the robot T_{hc}. In the first attempts, T_{hc} was deduced from the CAD model of the HRP-2 robot and did not produce a satisfactory precision for tasks such as grasping.

To deal with this imprecision, a ROS package has been created to automatically calibrate these extrinsic parameters for a humanoid robot. A chessboard is fixed to a robot hand (Figure 4.6). This chessboard is moved around in the vision field of the camera so that a set of chessboard position - robot configuration is recorded.

Let us place ourselves in the chessboard frame, which now becomes a fixed chessboard. We find the familiar "eye-on-hand" setup and the standard hand-eye calibration (Figure 4.7, the head joint has been renamed to G).

Denote:

- G_i: the frame attached to the head at time i
- C_i: the frame attached to the camera at time i
- CW: the frame attached to the chessboard (fixed, at all time)
- H_{Gi}: transformation from G_i to CW
- H_{Ci}: transformation from CW to C_i
- H_{cg}: transformation from the camera to the head (constant at all time)

The data acquisition process is given in Figure 4.8. At each pose i, the transformation from the chessboard to the camera and to the head is computed by a chessboard detector and the forward kinematics of the robot.

For each pair (i, j), the transformations from C_i to C_j and from G_i to G_i are

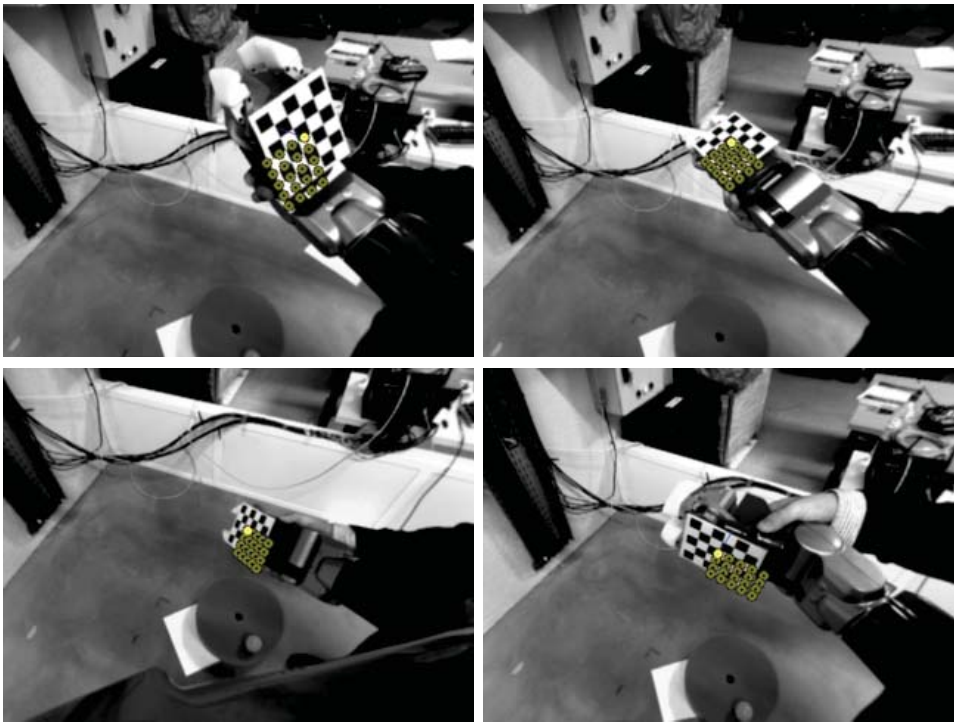


Figure 4.6: Hand-eye calibration process

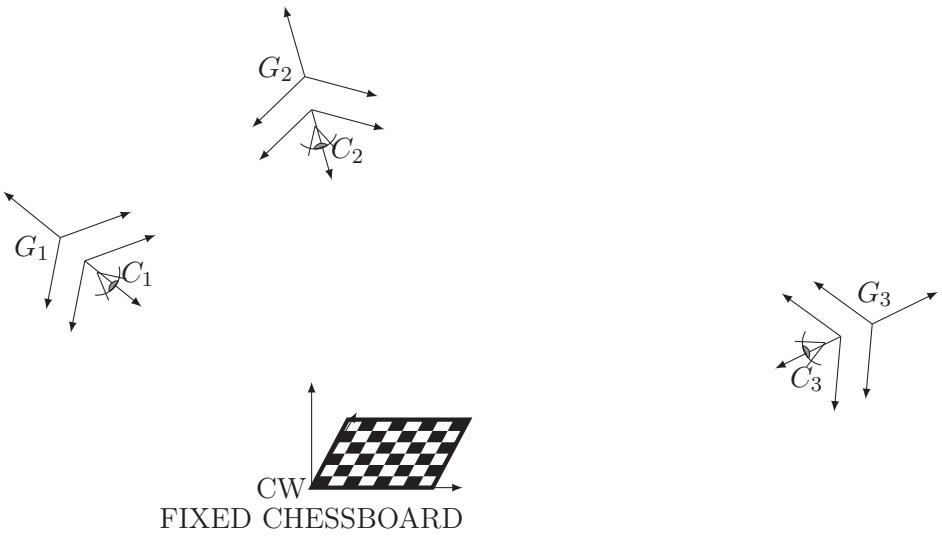


Figure 4.7: Movement of the camera in the chessboard perspective

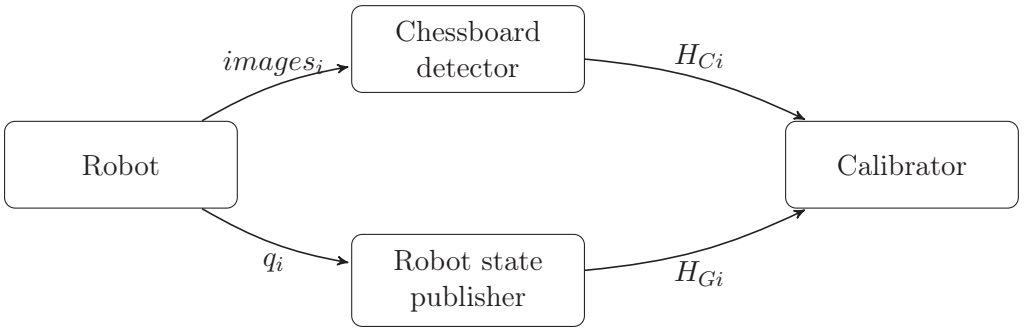


Figure 4.8: Data acquisition for extrinsic parameter calibration

$$H_{Cij} = H_{CiCW} H_{CWCj} = H_{Ci}^{-1} H_{Cj} \tag{4.4}$$

$$H_{Gij} = H_{GiCW} H_{CWGj} = H_{Ci} H_{Cj}^{-1} \tag{4.5}$$

On the other hand, we also have

$$H_{Cij} = H_{CiGi} H_{GiGj} = H_{cg} H_{Gij} \tag{4.6}$$

In (4.6), H_{Cij} and H_{Gij} are measured quantities (or more precisely computed directly from measure quantities). The role of the calibrator is to run an optimization method on thease measurements. [TL89] is chosen for this task for its efficiency and robustness.

Table 4.1 shows the calibrated extrinsic parameters for all the cameras on HRP-2.

camera	No poses	x	y	z	rx	ry	rz
narrow left	111	0.074 1	0.026 6	0.141 8	0.046 2	0.273 0	0.034 2
narrow right	111	0.072 4	−0.033 3	0.135 4	−0.008 0	0.220 1	0.005 9
wide left	121	0.081 2	0.070 0	0.057 0	−0.008 0	0.220 1	0.005 9
wide right	121	0.076 6	−0.075 9	0.055 3	0.011 0	0.221 9	−0.009 5

Table 4.1: Extrinsic parameter calibration on HRP-2. Lengths are in (m)

Chapter 5

Stepping over

5.1 Problem statement

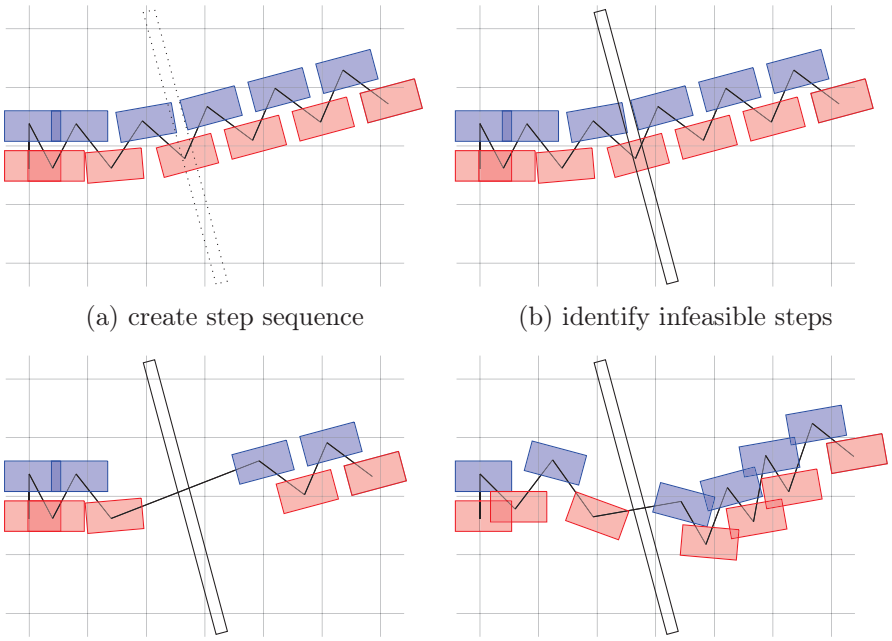
In this chapter, the method presented in the following chapter is applied on the humanoid robot HRP-2 to step over an object. The assigned task is to overcome a long cylindrical bar. The bar is long enough and its unknown characteristics make it impossible for the robot to step on. This example illustrates one of the key specialty of legged locomotion as opposed to wheeled robot.

The problem by itself has been studied in the past under different angles. Chestnutt et. al. [CLC+05] presented $A*$ search as a method to plan footsteps on the ASIMO robot. This method used external cameras to track 2D obstacles and planned in real time to avoid these regions on the ground.

Guan et al. [GYT05] used optimization techniques to overcome 3D obstacles in a quasi-static fashion. Stasse et. al. [VSYV06] proposed different strategy to dynamically step over objects. Perrin et. al. [PSB+12, BPM+11] used planning technique to achieve this same task with a changing environment, perceived by a motion capture system.

In this chapter the robot steps over the obstacle using a different approach. Footsteps are optimized by the system of task and constraints which includes in particular inequality constraints presented in chapter 2. The obstacle's position is estimated by the stereo vision system mounted on the robot. As any other vision system, the precision of the estimated position gets better when the robot gets closer to the tracked object (bar). Moreover, the bar is also intentionally moved by a human during the experiment. As a result, either to take into account the updated perceived position or a real displacement of the object, there is a need of reactive footstep adjustment.

The particularity of the task presented in this chapter is that it uses local optimization techniques to perform footstep deformations. The resulting deformed footsteps preserve the optimality characteristic which allows the robot to easily perform the task. Integrated with the stereo cameras mounted



(a) create step sequence (b) identify infeasible steps

(c) remove infeasible steps (d) optimize with localstepper

Figure 5.1: Initial stepping over step sequence with localstepper

on the robot, this method is more portable compared to external perception system. Finally, the framework is flexible enough be able to adapt any initial walking strategy to deal with the changing environment.

5.2 Computation of initial stepping sequence

5.2.1 Initial sequence with localstepper

Let Figure 5.1a be an an arbitrary and feasible stepping sequence. This sequence allows the robot to get from the initial position to the designated zone. The initial footsteps can be computed by, among other methods, localstepper itself by adding a position task to the last footprints of the centipede presented in chapter 3.

Now, obviously, with the bar added to scene, the initial stepping sequence is not feasible anymore as stepping on the bar is not allowed (Figure 5.1b). These infeasible steps can now be removed (Figure 5.1c). (If the number of removed steps is odd, one additional step is removed from the sequence to preserve the left-right sequence.

The newly created stepping sequence will likely be infeasible due to the large distance of the footprints at either side of the bar. This infeasible sequence is then (re)fed to localstepper which optimizes these footsteps with

constraints described in chapter 3 (including maximum distance constraint between subsequent steps) plus the collision constraints between the bar and all footsteps. By solving the optimization problem, we obtain a feasible step sequence in Figure 5.1c.

3d obstacles

Until now, we assume that the obstacles are flat. With this assumption, the resulting footstep placement obtained in Figure 5.3 guarantees us a collision free movement. To take into account possible collisions, one has to consider not only the foot placements at either side of the obstacles but also the transition phases around these steps.

Suppose that the operation point to control each feet is located at d_1 from the heel and d_2 from the toe (Figure 5.2). The obstacle is a blue half-cylinder of diameter $2R$ (Figure 5.3). In most of the walking schemes on humanoids that include flat feet like the HRP-2 robot, the feet are always parallel to the ground. We can reasonably assume that while the foot is parallel, the only possible collision is between the sole and the obstacle.

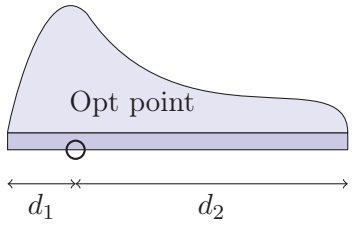


Figure 5.2: Robot foot

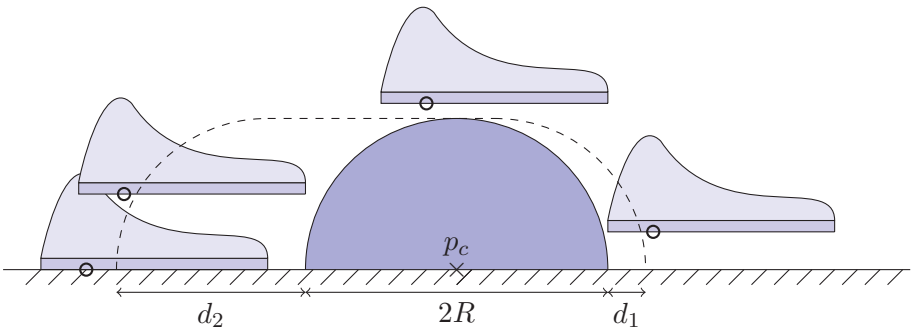


Figure 5.3: Stepping over a cylindrical bar

Let $P_o(t)$ be the trajectory of the operational point O. Let us extend the left half of the blue cylinder by d_2 and the right half by d_1, and call the resulting

region $\mathcal{C}$ The necessary and sufficient condition of a collision free path is:

. $P_o(t)$ *is collision-free* $\Leftrightarrow P_o(t) \notin \mathcal{C} \quad , \forall t$

With the obstacle in Figure 5.3 located at $p_c = [x_c \quad y_c]^T$, this condition becomes:

$$\forall t : \begin{cases} (y(t) - y_c)^2 + (x(t) + d_2 - x_c)^2 \geq R^2 & , x(t) \leq x_c - R - d_2 \\ y(t) - y_c \geq R & , x_c - R - d_2 < x(t) < x_c + R + d_1 \\ (y(t) - y_c)^2 + (x(t) - d_1 - x_c)^2 \geq R^2 & , x(t) \geq x_c + R + d_1 \end{cases} \tag{5.1}$$

We can now resolve this condition with 2D step conditions in previous section and obtain a feasible trajectory to overcome a cylinder

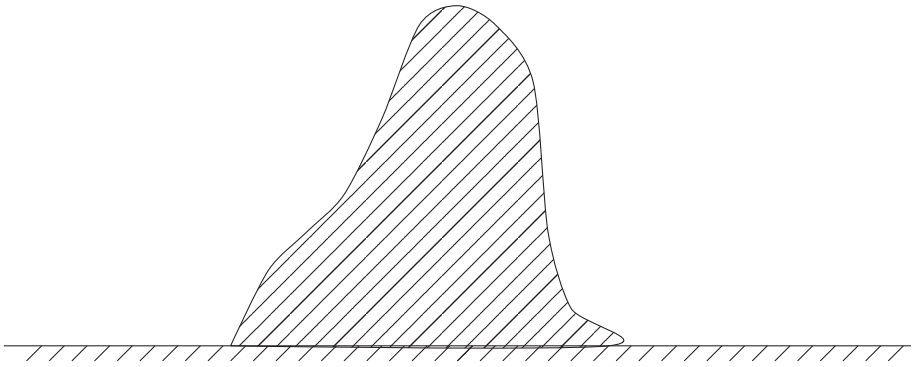


Figure 5.4: General shape obstacles for which testing collision for the robot sole suffices

Arbitrary shaped obstacles Previously, we claimed that the foot can only come into contact with the cylinder in Figure 5.3 with its sole. This is true for a class of objects $\mathcal{O}$ (Figure 5.4) that satisfy:

$$\forall (x, y, z) \in \mathcal{O}, z' < z : (x, y, z') \in \mathcal{O}, z' < z \tag{5.2}$$

These objects can be thought of as "mountain shape" objects where the higher cross sections always fall inside the lower (support) cross sections. Primitives of this class includes: semi-spheres, semi-cylinders with the flat section on the ground, cubes, boxes. For even more complicated shapes, we can create a set of bounding object that belong to $\mathcal{O}$.

5.2.2 Adapt externally planned initial sequence

The computation of crossing trajectory did not take into account the possible collision between the knee and the obstacles. (Figure 5.5).

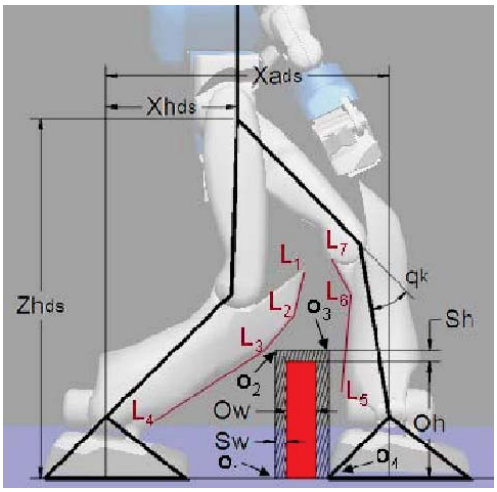


Figure 5.5: Collision avoidance for the knee

To deal with these situation, an external stepping strategy proposed by Stasse et. al can be used as the initial stepping sequence [VSYV06]. In addition to the consideration of knee position at double support phase, this method also proposed doing a second pass of ZMP calculation, this time with the multibody model. The effect of this second pass is more visible when the obstacles become too large that the cart-table model is no longer suitable.

Localstepper, as indicated by its name, is a local method and not immune to local minima. For instance, this method will not work if a wall is built between the initial position and the bar. In this case, a real motion planning method should be used to obtain the initial sequence.

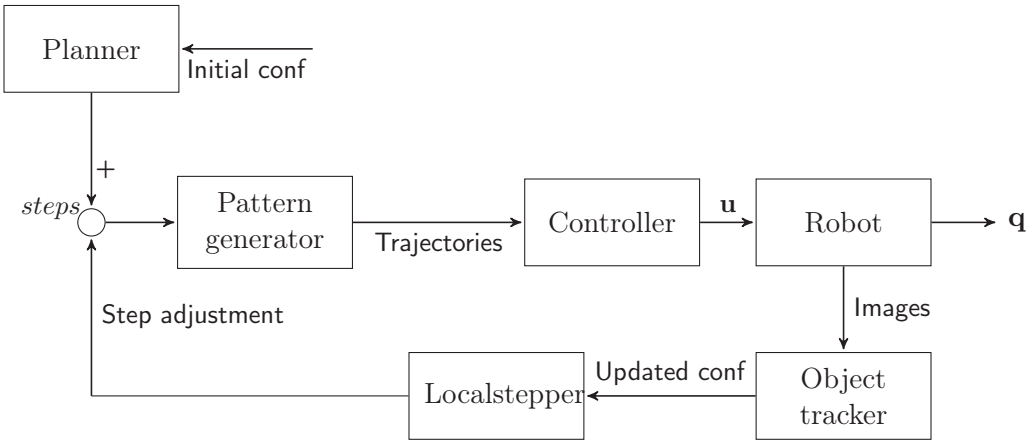


Figure 5.6: Step deformation loop

5.3 Online deformation

As the perceived position of the obstacle is continuously updated, and as the fact that the obstacle might be moved during the experiment, the footsteps have to be recomputed as fast as possible.

Provided that the shape of the obstacle is unchanged (long cylindrical bar with known diameter), the robot only needs to make sure that the two subsequent steps that cross the bar stay unchanged *with respect to the bar*. We then recover the same situation as described in section 3.3: stepping towards a moving target.

The footstep adjustment scheme for the stepping-over experiment can be written as algorithm 3, when x_0, y_0, x are three-dimensional vectors in the footprint coordinate (x, y, θ).

Algorithm 3 Footstep adjustment for stepping over experiment

Require: current plan.
Ensure: new plan
1: $x_0 \leftarrow$ initial target step
2: $y_0 \leftarrow$ initial bar position
3: **loop**
4: $y \leftarrow$ current bar position
5: new target $x \leftarrow x_0 + y - y_0$
6: recompute footsteps
7: **end loop**

5.4 Experiment

In the experiment depicted in Figure 5.7, the bar is marked by an orange band and detected by the module detailed in chapter 4.

Thanks to the online optimization scheme, the robot has no problem crossing both a fixed and a mobile bar. Figure 5.8 describes how the bar is tracked during the movement, Figure 5.9 depicts the estimated position of the bar.

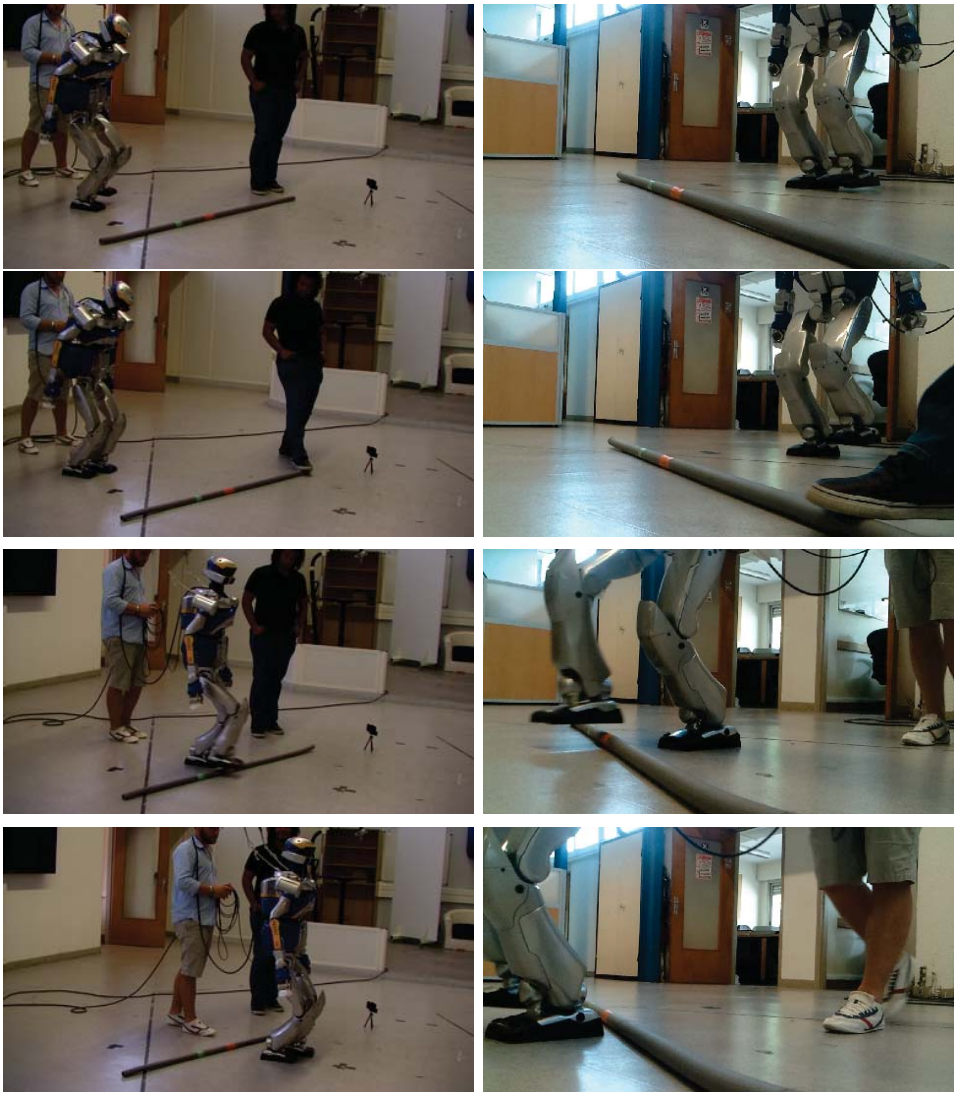


Figure 5.7: Stepping over a bar on HRP-2

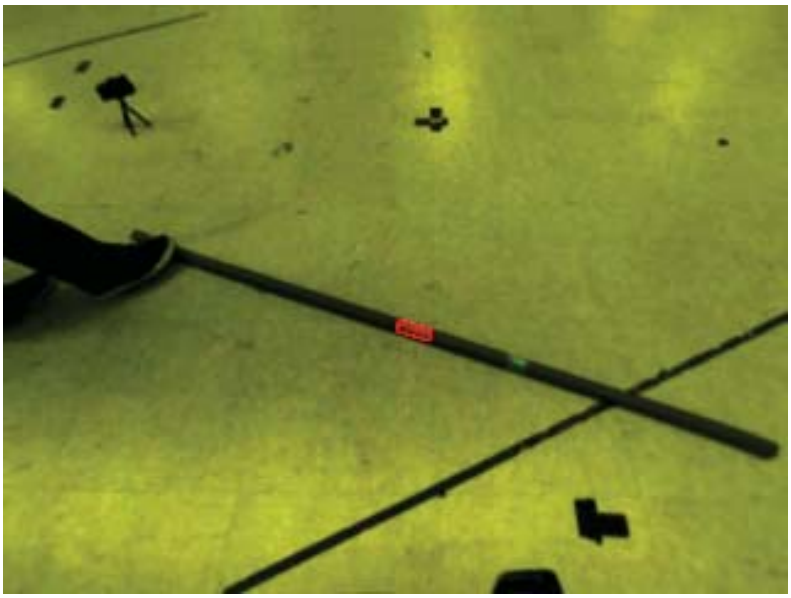


Figure 5.8: Tracked bar by the robot

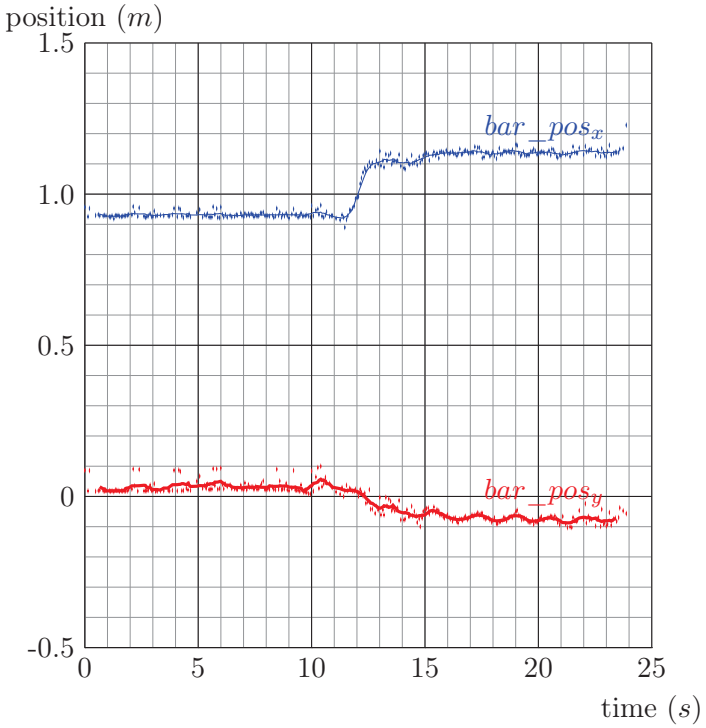
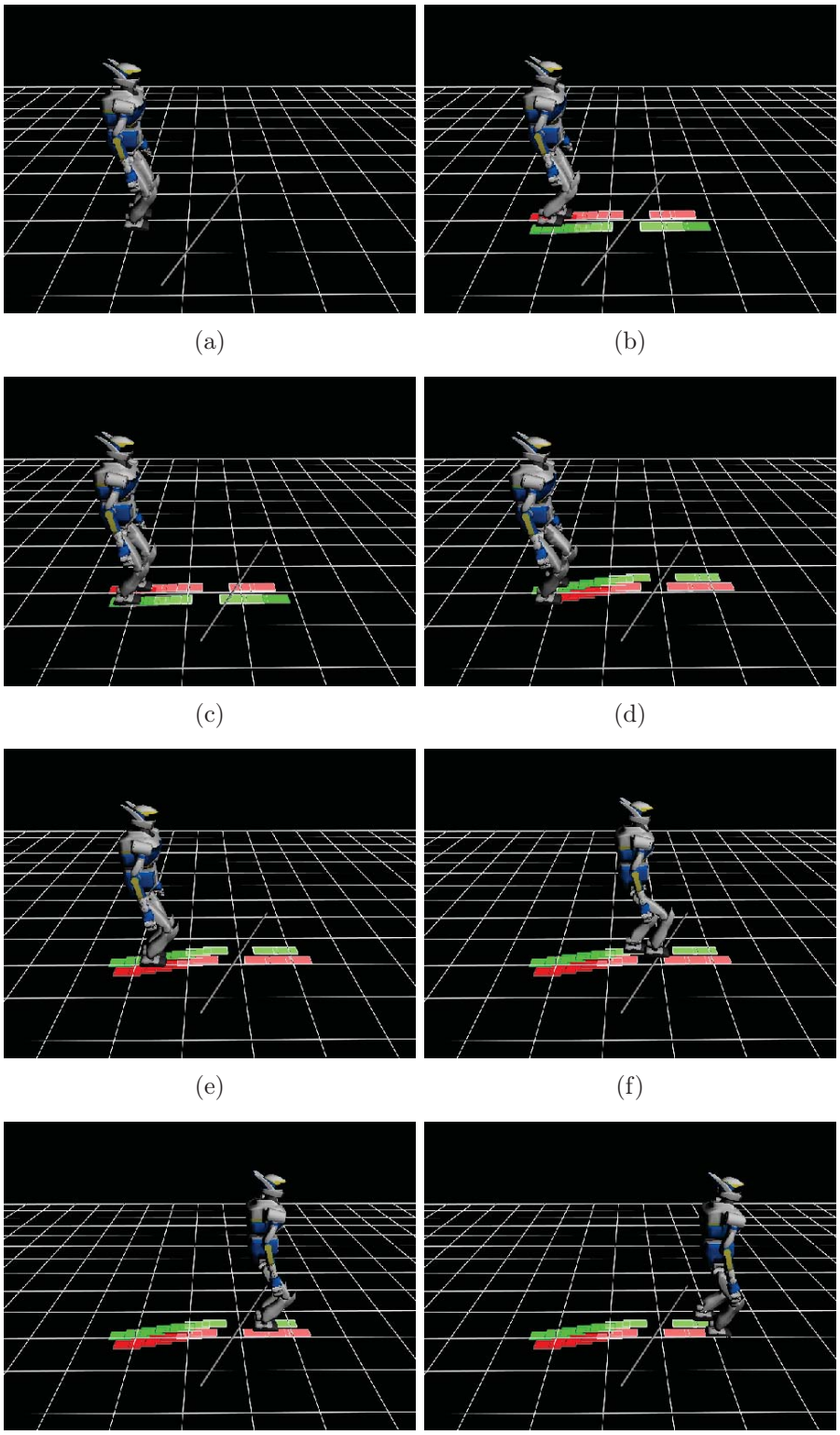


Figure 5.9: Perceived position of the bar (x and y components) which is moved while the robot is walking (around second 12).



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 5.10: Replanning for a moving bar

Chapter 6

Object grasping

In this chapter we are interested in manipulation tasks which involve both locomotion and manipulation. The task given to the robot is to locate a ball in front of it, plan appropriate steps and the grasping pose. The robot then executes the step sequence, tracks the target in real-time, replans and updates its steps. The end of the locomotion part is blended into the grasping part.

6.1 Reactive online footstep replanning

6.1.1 Planning process

As presented in chapter 2, for planning purpose, footsteps are consider as parts of the humanoid robot.

The planning process is the following

- Build constraints on humanoid robot (notably joint limits and self-collision avoidance).
- Build the augmented robot (or centipede as coined by Kanoun et. al. [KLY10]).
- Build constraints for newly added degrees of freedom. For augmented joints (footsteps), these constraints correspond to the limits imposed by the pattern generator (max length of a step, max orientation), and self-collision avoidance: the robot should not step one foot on another.
- Build constraints for obstacles avoidance in case the robot has to avoid regions on the floor (i.e. the bar in previous chapter).
- Add equilibrium condition for the standing robot at the end of the stepping process. (the CoM has to stay inside the support polygon at the end of the movement)
- Build tasks which include

- Hand task for grasping
- Gaze task to keep the target inside the vision field
- Reference posture task

All the constraints are inequalities and have higher priority than tasks. All tasks are listed in descending order of priority. Starting from the given requirement the localstepper returns a final configuration $\boldsymbol{q} \in \mathcal{R}^{n+3k}$ of the augmented robot (Figure 6.1) (Suppose the robot has n degree of freedoms executing k steps).

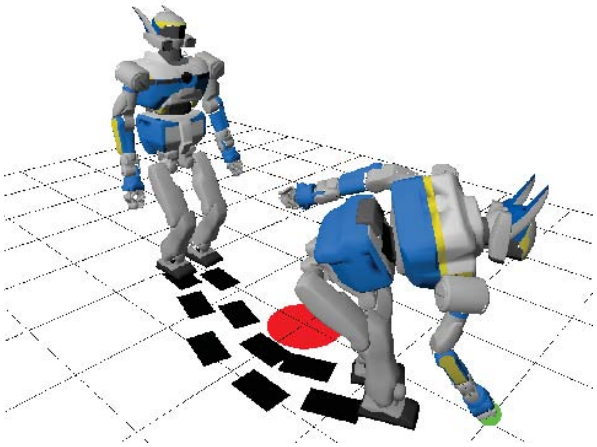


Figure 6.1: Output of localstepper

posture desire

From this resulting vector $\boldsymbol{q}$, we can extract the k steps and the final posture. The steps and the desired posture are expressed (x_i, y_i, θ_i) in local coordinates are:

$$\begin{aligned} x_i &= q_{3(i-1)+1} & i &= 1 \dots k \\ y_i &= q_{3(i-1)+2} & i &= 1 \dots k \\ \theta_i &= q_{3(i-2)+3} & i &= 1 \dots k \\ posture^d &= [q_{3k+1} \dots q_{3k+n}]^T \end{aligned} \tag{6.1}$$

6.2 Online footstep replanning

One big challenge in our context is the use of stereo-vision to detect and localize objects. Localization errors grow dramatically with the distance between the object and the sensor. Usually, this distance is maximal in the initial configuration of the robot, when motion planning is performed.

The resolution of a complete motion in joint space described in previous section typically takes up to one minute for a long motion. Yet, to be reactive,

replanning should finish at least once every stepping interval of the robot (typically under a second). It is clear that replanning the motion up to configuration space motion is not feasible in practice.

6.2.1 Footsteps and final posture replanning

As explained earlier intermediate results, i.e. final posture and footsteps, of the planning stage are sufficient for the controller. This is where replanning becomes feasible since the computation of footsteps and posture is typically from 3 to $10s$, a replanning could be well below $1s$ and hence guarantee reactivity. The augmented robot then starts at the previous state and updates its tasks according to new sensory information about landmarks. Algorithm 4 describes the stages of planning, the results are shown in Figure 6.2. The solver converges faster than planning footsteps and posture from initial position since the current configuration is already close to the goal configuration. Table 6.1 shows replanning time for a grasping task with goal as a ball on the ground at a distance around 2 meters. The modification is taken randomly in arbitrary directions.

Algorithm 4 Footprint replanning

Require: current plan.
Ensure: new plan
loop
  $replan\_needed \leftarrow false$
  check position of target in camera
  check current step number
  if Goal position changes **then**
    Update gaze and hand task
    $replan\_needed \leftarrow true$
  end if
  if current step number changes **then**
    Update virtual manipulator in solver
    $replan\_needed \leftarrow true$
  end if
  if $replan\_needed$ **then**
    Replan to new solution
  end if
end loop

The replanning process for the whole-body and footsteps depends greatly on configuration and it is not guaranteed that the computation time will be less than the stepping time ($0.8s$). Without modification, small corrections (e.g. due to drifting) can be dealt with. Otherwise, an execution strategy must be

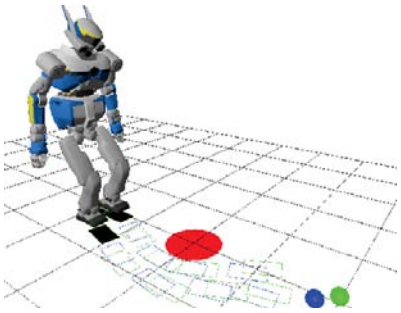


Figure 6.2: Footsteps replanning

Goal modification (m)	Max (ms)	Min(ms)	Average (ms)
0.01	2615	251	595.1
0.02	2159	275	673.2
0.05	2343	488	920.7
0.1	2498	593	1299.8
0.2	4166	608	1977.0
0.5	7123	610	3685.3

Table 6.1: CPU time for replanning of final posture and footsteps (milliseconds) for a locomotion-grasp task

applied at the planning-control bridge to deal with large errors, such as the case of stereo vision system.

Stop and go One obvious approach to address the real time issue is to stop the robot or let it step in place if the planner takes too long to response.

Interactive optimization As shown in section 2.5, the complex robot tends towards the goal during the optimization process. At early stages, i.e. when the robot starts, having the full solution with all footsteps and final pose is not necessary. In fact, even if a full solution is found, it is highly likely that this solution will change during walking motion to compensate sensory errors. The planner can keep up with the controller by only replanning at each correction to an intermediate state. Namely, line 14 in Algorithm 4 will be modified to: "replan to new solution or timeout", where the timeout chosen is one stepping period.

While stop and go strategy guarantees an optimized trajectory at each replanning stage, it produces unnatural movements on the robot. Using the interactive approach, the robot does not have to stop while correcting its paths towards the new goal. However, this method relies on the assumption that the intermediate stage found at each timeout is safe for the robot.

As it turns out, there exists a third strategy which consists in reducing the dimension of the optimization problem in replanning stage by blocking some or all joints belonging to the upper body. At each replanning step, optimized foot

Goal modification (m)	Max (ms)	Min(ms)	Average (ms)
0.01	16	8	10.7
0.02	38	7	11.4
0.05	12	9	11.0
0.1	48	9	15.5
0.2	23	11	20.0
0.5	117	15	33.6

Table 6.2: CPU time for replanning footsteps (milliseconds) for a locomotion-grasp task

placements and posture are found without compromising the time constraint. Without having to interrupt the optimization process at each timeout as the second approach, this method is also simpler to implement on the robot. This third strategy is chosen for experiment and is detailed in the following section.

6.2.2 Real-time footstep replanning

While footsteps have to be changed to correct execution drift or perception error, the posture does not. In fact, unless new obstacles are found near the goal the previous posture stays valid. For example, if the robot is to approach and open a door, even the position of the door on its reference has change, the previous posture still guarantees dynamic equilibrium. This scenario covers most cases in experiment. The most part of the last n degrees of freedom of the augmented robot can be 'frozen' while other joints are free to move. Suppose l degrees of freedom are locked. This translates into reducing the dimension of the Jacobean, hence the complexity of the problem.

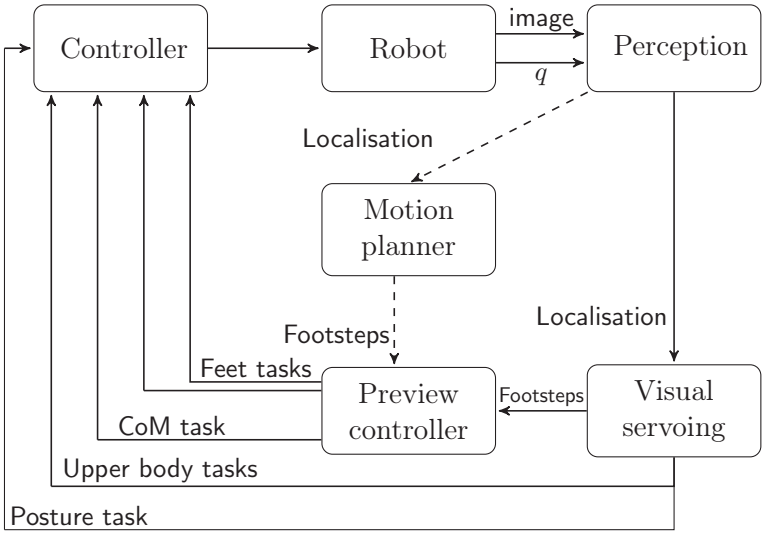
Table 6.2 shows replanning time in a grasping scenario similar to the one described in previous section with the posture of the standing robot "frozen". This is the other extreme case compared to Table 6.1 where every joint in the standing robot is free to move. In the first replanning scheme, the robot can barely correct small errors, so the robot has to make many small corrections at a time, hence a large amounts of additional steps have to be added to the locomotion stage. In the second replanning schema however, the replanning time is much lower than the time necessary for the robot to make a step. This guarantees a real-time replanning running behind the scene and updating robot footprints continuously.

6.3 Motion blending by prioritized stack of task

Let us recall how the robot adapts its movement to face changes in the environment.

The visual servo in Figure 6.3 has in fact two parts

- *localstepper* which regenerates posture and footsteps.



COM: Center of Mass

Figure 6.3: Information flow in a grasping task

- a servo which feeds directly the target into the grasping task.

The perception module returns a 3D goal position. The planner only outputs the final posture and desired footsteps. How do we use these three pieces of information in the controller? Obviously, by the use of tasks. The robot must approach the goal first, thanks to the trajectories supplied by the pattern generator. Then, at some point, it has to grasp the object with a position task on its hand. The desired posture will be used at lower priority as a hint. The feasibility of this posture, as calculated by the planner, is the necessary condition for a successful grasp.

The question is more about the chronology of those tasks. If we start the grasping task and the posture task too early, the robot has to walk with the grasping posture too early which is obviously not suitable. Moreover, with most of the pattern generators, a humanoid robot needs a more or less straight posture during walking so that it follows well the cart-table simplification. Start the posture task at step 0 is therefore unfeasible. If on the other extreme, we start the posture task too late, the robot which is not guided by the posture might end up in Figure 6.4 with the hand stuck behind the body.

One more point worth noting is that a successful grasp requires that the hand pass by the pregrasp point in the configuration described in Figure 6.5. To achieve this, piece-wise polynomial curves are used. [PT97] The conditions imposed to this curve are:

- start out from the hand's current velocity

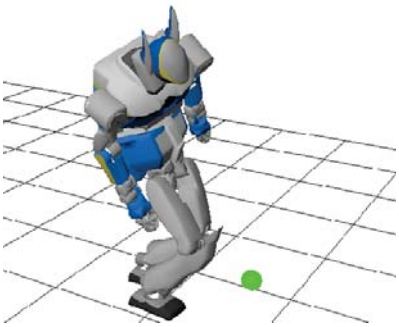


Figure 6.4: Hand stuck as the posture task was activated too late.

- pass by initial point, passage points and final goal position
- stop at zero velocity

First, the curve is presented by a pth-degree curve (Figure 6.6) Where $0 \leq u \leq 1 = t/t_o$ is the current time normalized by the total time on the curve t_o. $\boldsymbol{P}_i, \bar{u}_i, N_{i,p}$ are the control points, knots and the basis functions. $\boldsymbol{Q}_i$ are the passage points. Note $\boldsymbol{D}_o, \boldsymbol{D}_n$ initial and final velocity. The control points are found by solving the following system of $n+2$ linear equations.

$$\begin{aligned} C(\bar{u}_k) = \sum_{i=0}^{n+2} N_{i,p}(\bar{u}_k)\boldsymbol{P}_i = \boldsymbol{Q}_k \quad , i = 1 \ldots n+1 \\ -\boldsymbol{P}_0 + \boldsymbol{P}_1 = \frac{u_{p+1}}{p}\boldsymbol{D}_0 \\ -\boldsymbol{P}_{n+1} + \boldsymbol{P}_{n+2} = \frac{1 - u_{m-p-1}}{p}\boldsymbol{D}_n \end{aligned} \tag{6.2}$$

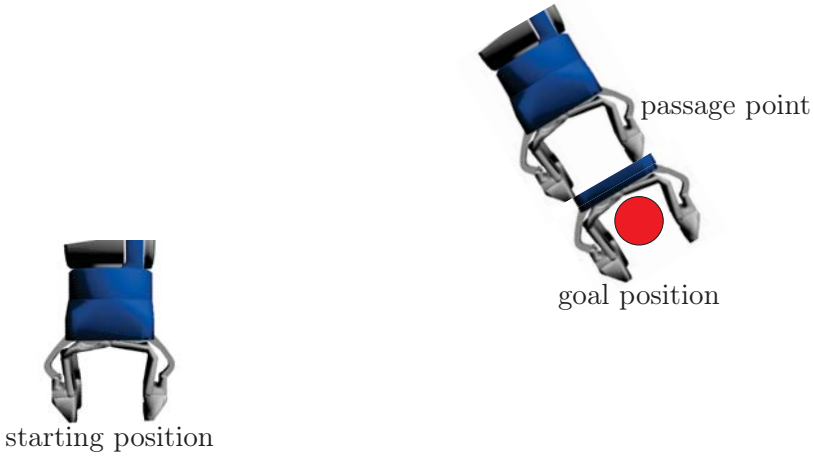


Figure 6.5: Mandatory passage point for grasping

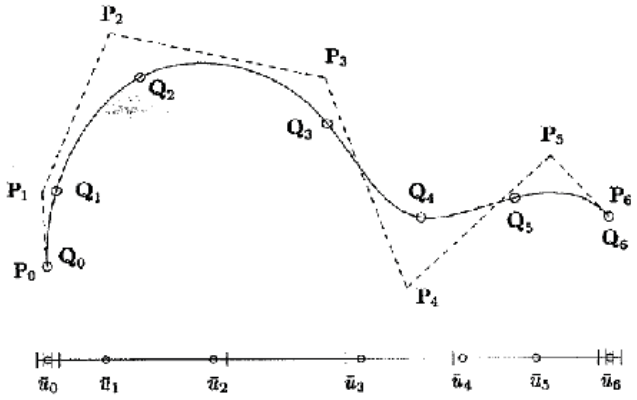


Figure 6.6: Cubic spline

6.4 Experiments

This section present a series of experiments validating our proposed framework. (Figure 6.7, 6.8, 6.9, 6.10, 6.11, 6.12) In these experiments, the goal is to grasp an object outside of the reach of the robot so that stepping will be involved.

Goals are also intentionally moved during the movement to illustrate the adaptation scheme applied on the footsteps. Keep in mind though that the perceived position of the goal changes, so this adaptation happens all the way till the end. For the robot, the goal always moves, only its amplitude become smaller when the goal is near as the vosion become more precise.

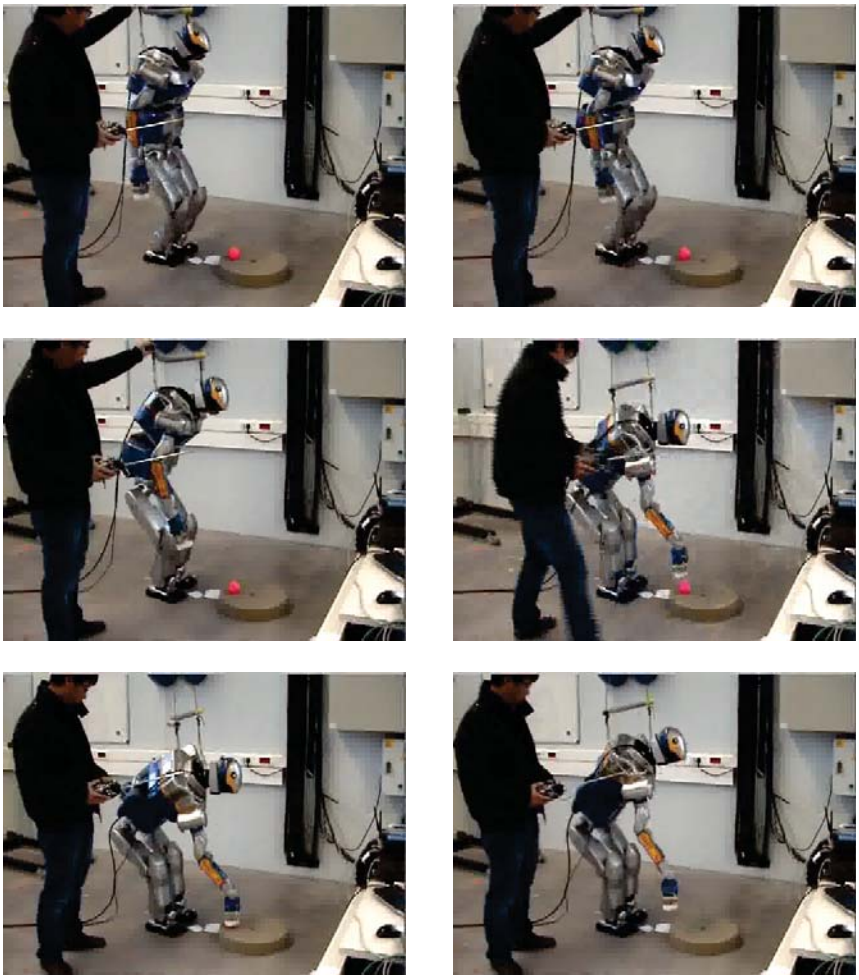


Figure 6.7: Grasping an object at close distance. This first experiment is intended to test the grasping accuracy and the stereo vision system. Since the target is at such close range. The steps that the robot made are only to prepare the feet position to well support the planned posture.

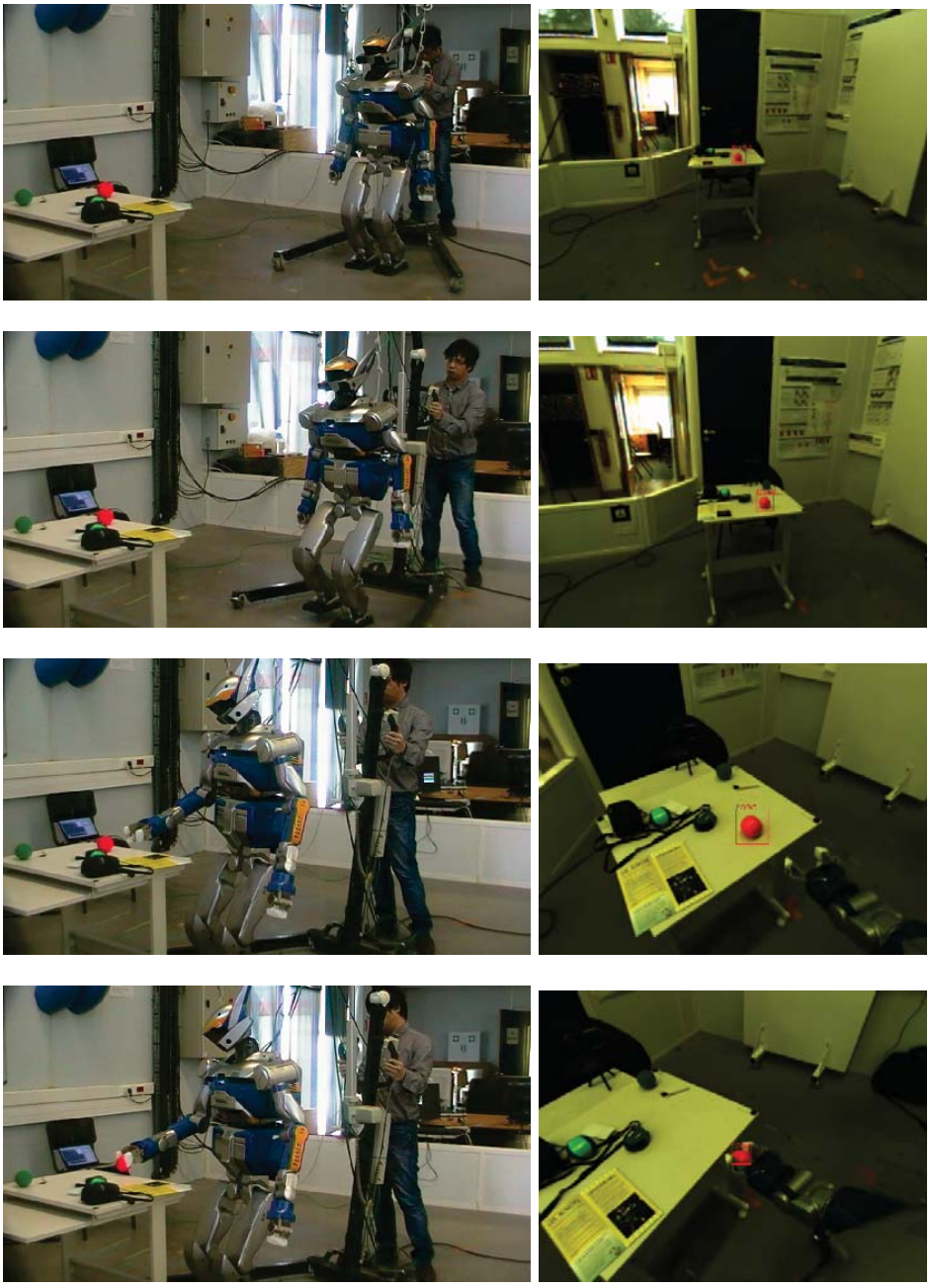
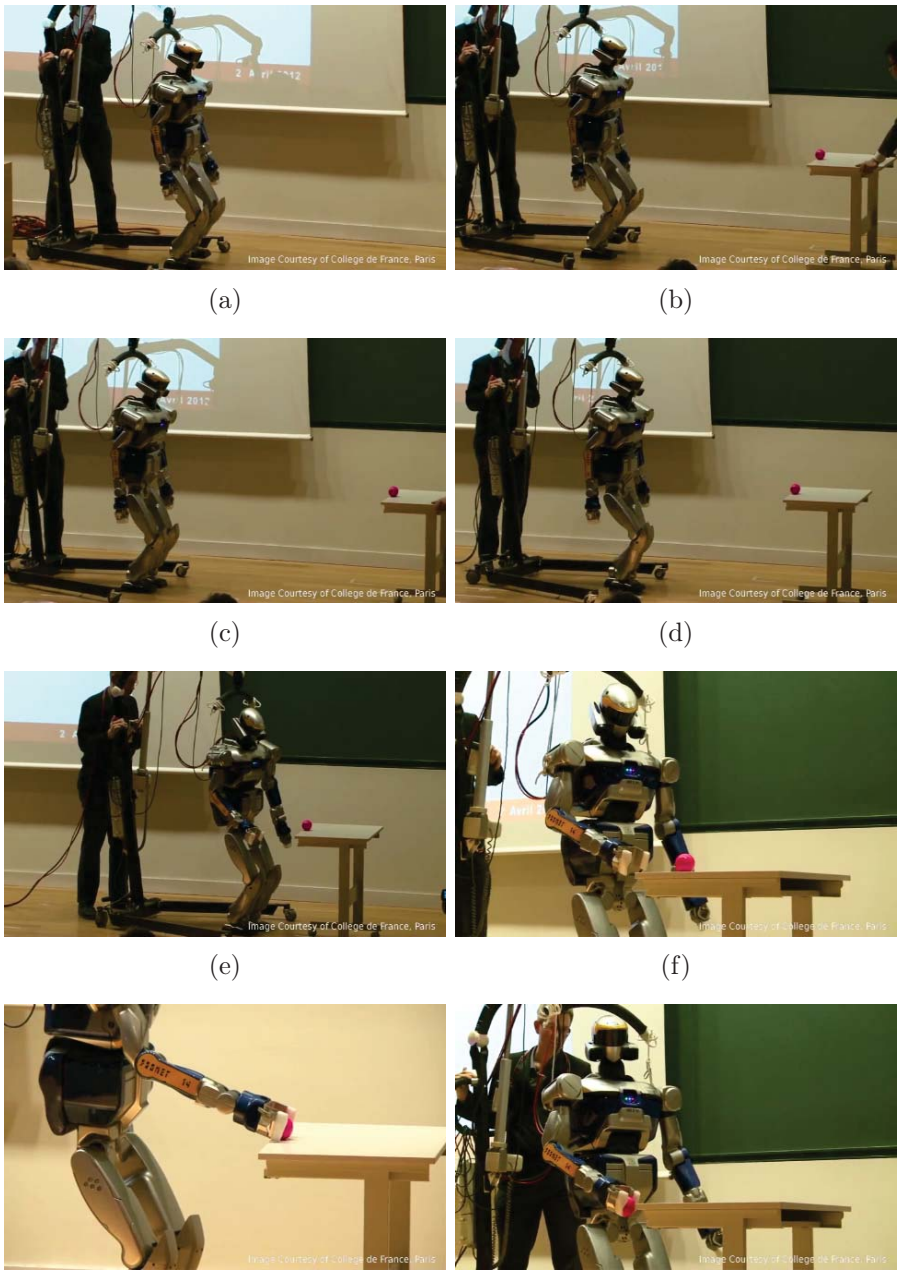


Figure 6.8: Grasping an fixed object at distance, at table level. The robot searches, locates the target, plans the first stepping sequence and the grasp postures. The target continues to be tracked during the whole experiments (43s). 97 footstep re-computations has been carried out during the walking sequence (lasted 14s).



(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figure 6.9: Grasping object on the table which is moved during the movement. This experiment has been presented in public at the College de France on April 2, 2012 with a very short preparation time. The light on the stage changed subsequently the hue of the learned target. The perception module was capable of taking a new model thanks to the design in chapter 4 and adapt to the new light condition. The table was intentionally moved during the movement to illustrate the fact that the robot replanned its footsteps as it received updated information about the environnement.

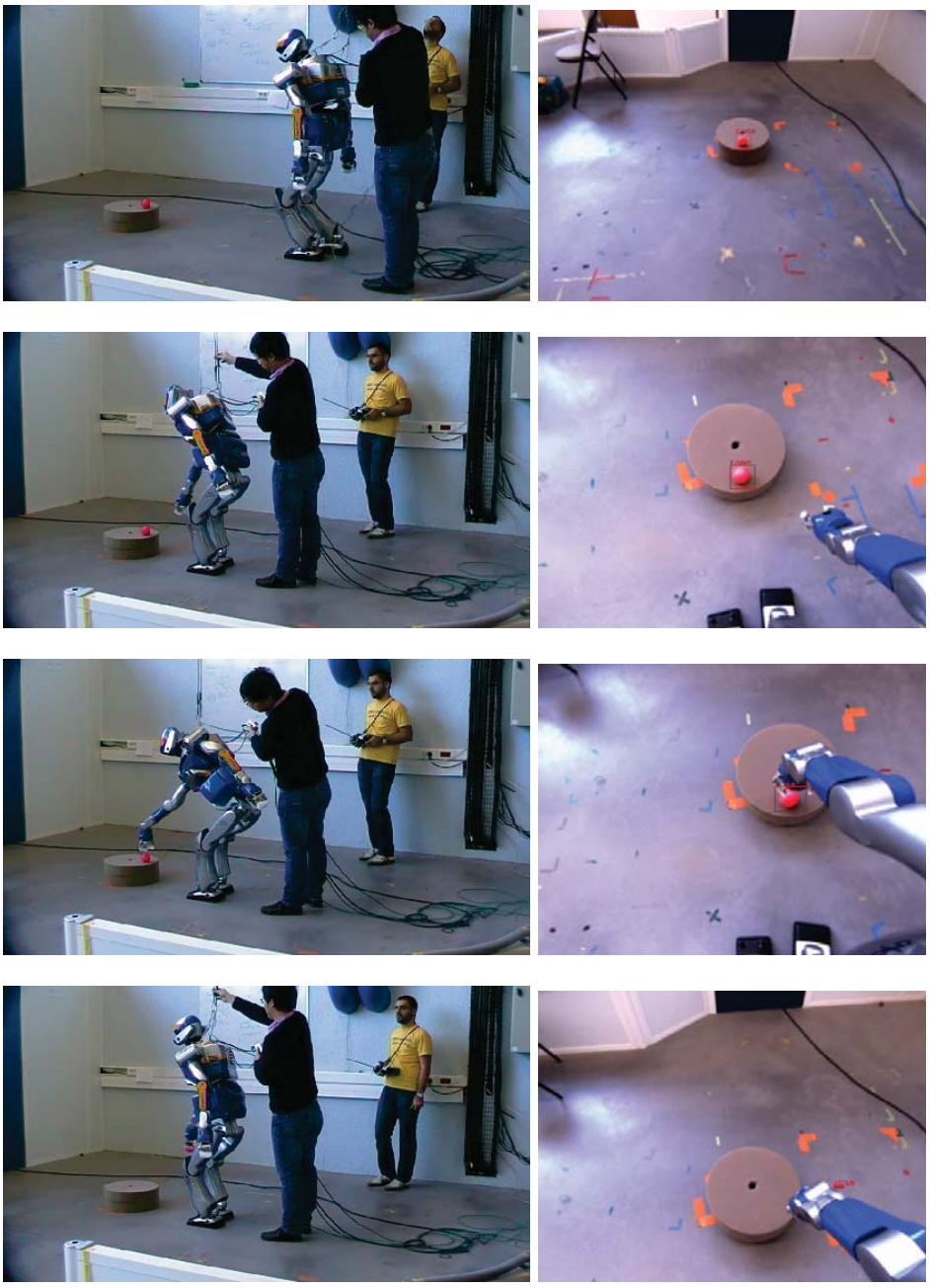


Figure 6.10: Grasping object at ground level. The foam was placed to protect the hand in case of emmergency

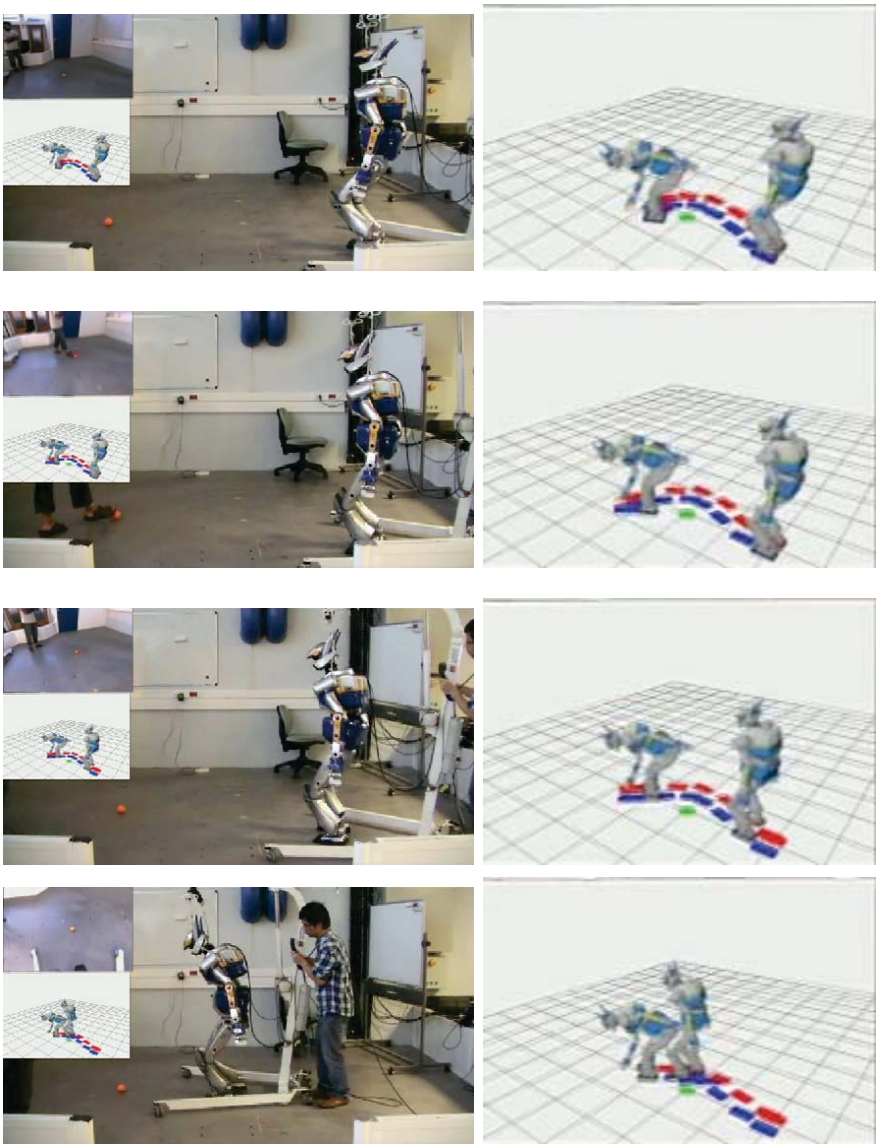


Figure 6.11: Foot step adaptation during a grasping movement This experiment illustrates how the robot adapt its footsteps (right figures). The target is pushed while the robot walks.

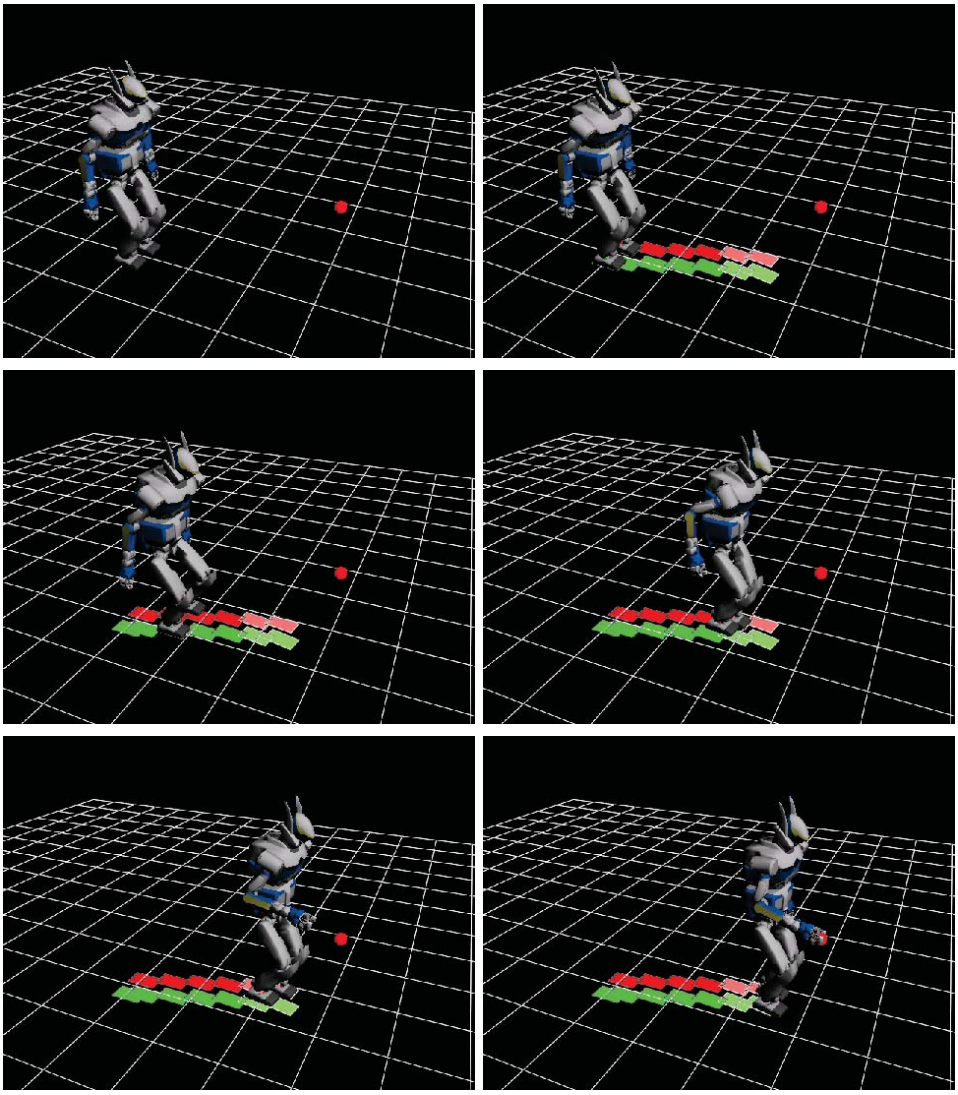


Figure 6.12: Grasping without stopping. This experiment is the only one that has not been tested on the robot. Since the whole grasping process hapends while the robot is walking, the swinging movement of the chest brings it dangerously close to the grasping arm in simulation. More safety measures are to be designed before applying this movement on the robot.

Chapter 7

Conclusions

7.1 Manipulation-locomotion fusion with closed-loop vision

The framework detailed in this thesis tackles some interesting problems in humanoid robotics.

First, it considers the dual-properties of redundancy and underactuation. On the one hand, with a large number of actuated joints (30 for the HRP-2), the robot can easily achieve a given task (which usually "costs" a few degrees of freedom) while preserving a certain flexibility. In fact, the construction of a prioritized take hierarchy based on null-space projection has been studied for years in robotic manipulation, including the *StackOfTasks* framework used in this thesis. On the other hand, the 6 degrees of freedom in the robot representation corresponding to the position and orientation of the base (the waist for humanoid robots) are not directly controlled. The robot relies on its feet and their interaction with the ground to propel itself forwards. This aspect is the source of numerous works on locomotion which were partly presented in this thesis. Up to now, research in humanoid robotics either had focused on one of these two aspects; locomotion and manipulation of humanoid robots had been usually considered as separate problems.

The work presented in this thesis attempts to fuse the two aspects. With footsteps represented as parts of the humanoid robots, locomotion and manipulation are considered in the same problem. Furthermore, with the use of a virtual link attached to each footstep, the robot can make a complete abstraction of stepping. The choice of foot placements is transparent to the optimizer which consider them the same as the real and actuated joints on the robot. Locomotion becomes an effect induced and guided by manipulation tasks.

Second, the proposed framework deals with another interesting question which is the combination of optimization and closed-loop control. Indeed, in robotic manipulation, it is often necessary to trade capability for reactivity and

vice versa. On the one hand, the robot is capable of executing a complicated task if enough computation time is given to it. In this case, the whole motion is computed offline first, then executed in open-loop on the robot. For example, one can use optimization techniques to generate a complete motion when the robot needs to make a few steps before grasping an object. The costly process takes up to several minutes to provide a complete motion. On the other hand, the robot can be reactive and responsive to the changes in environment if it uses sensors to detect the changes and feed them to the controller in a closed-loop framework. A complete visual servoing for example needs the task to be "simple". Stepping with grasping in the example mentioned earlier, which traditionally takes minutes to compute, is not suitable. The method used in this thesis combines an optimizer, which is slow, a task-based controller and a perception system. The trick here is to separate the two stages in the optimization process and take only the intermediate results, i.e. the necessary footprints and the final posture, as the inputs of the controller. With this approach, the robot is able to accomplish complicated tasks involving both locomotion and manipulation while adapting itself reactively to the environment.

In summary, this framework can be thought of as the fusion of locomotion and manipulation. The perception-decision-action loop integrates one stage of the optimization process, i.e. footstep optimization, which allows an adaptation of the whole movement to the changes in the environment. The validity of this framework is demonstrated in two types of experiments on the robot: stepping over an object and grasping with walking. This is, to our knowledge, the first time such fusion of locomotion and manipulation in a reactive manner is demonstrated on a humanoid robot.

7.2 Application scope

The perception-decision-action loop proposed in this thesis makes use of the following components.

- Footstep and posture optimizer.
- Stereo vision with color object tracking.
- Task-based controller.

This loop starts out from an initial sequence composed of a feasible, collision-free footprints sequence and a final posture. As explained in chapter 5, an external method can be used to step over a large object.

Similarly, *any* initial sequence, such as those obtained by motion planning technique as in Figure 7.1, can be used as input in the presented loop. With this input, the robot observes the environment and adjusts its footsteps locally to deal with changes in the environment, such as, the initial error in perceived

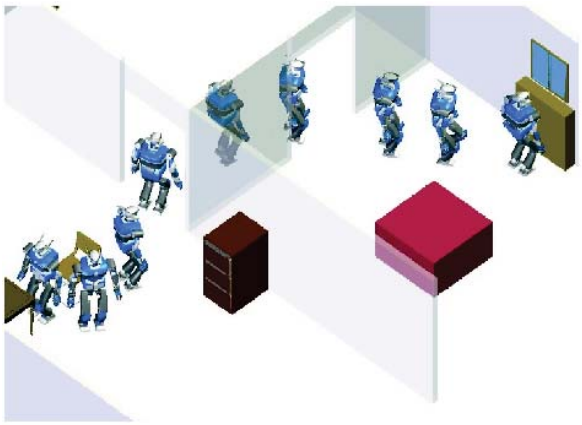


Figure 7.1: A possible initial sequence found by motion planning techniques. [EKTL+11]

position of the door, positions of the furniture. The current version of the object tracking model relies on color objects. It suffices that the landmarks be easily recognizable by their color as in Figure 7.1.

7.3 Limitations

If the validation of the proposed framework has been shown with experiments on a real robot and that extensions to other scenarios are possible, there is still room for improvement. The three main limitations of the approach are the local optimality of the formalism, its dependency on the perfect knowledge of the model and a limited perception module.

7.3.1 A local approach

The adaptation scheme works by resolving an optimization problem on footsteps and postures. It assumes that the *changes in the environment happen incrementally*. This is true for changes in the perceived position of a fixed object or, for example, an object moved by human. When the target is tracked continuously, these changes are small and the optimizer always works in the neighborhood of the previous solution, therefore it converges in little time. Another assumption is that the *structure of the environment stays unchanged*. This condition also ensures that the local optimization converges in reasonable time.

Two possible pitfalls directly related to the local approach are

- lost of tracking of a landmark. When this landmark reappears, the change is no longer local. The optimizer fails to converge in small time.

- the structure of the scene changes, e.g. the robot has to grasp an object on the floor. During its movement, the object is pushed under a table, the change is no longer local and real-time adaptation is not possible.

7.3.2 Platform limitation

The presented framework is designed to work on HRP-2 and robots of similar properties, such as NAO, Asimo, etc. It relies on a perfect knowledge of the robot kinematics and dynamics. This condition allows us to build a precise and capable controller, e.g. , to grasp an object based only on its position. More specifically, in HRP-2 whose mobility is in parts limited by its flat foot design, the pattern generator employed in the framework works best on solid and level ground. The robot won't be able, for example, to walk reliably on a deformable foam or on ice.

7.3.3 Perception limitation

The perception module has been designed following criteria on robustness, simplicity and maintainability. It relies uniquely on color blobs to detect and track objects. The complexity of the scene in which landmarks and targets can be tracked is limited. The framework does not work, for example, on a unordered scene with a target containing no dominant colors or which is not distinguishable from neighbor objects.

Object detection is also out of the scope of this study. For example, the perception module is not designed to recognize a mug and detects similar object in a new scene. Such capability, however, is to be added to the perception module in the future using state-of-the-art algorithms such as [VJ01, LM02] etc.

7.4 Perspectives

Extension New scenarios can be developed using exactly the same principals, such as Figure 7.1. The robot takes a sequence of footsteps and a task posture from any planning or optimization method as input then adapts it to the real environment. It can be imagined that a dynamic task requiring precise foot placements can be enhanced by the adaptation scheme presented here. For instance, if the robot was to make steps before kicking a football, the last step before the shoot should be well positioned with respect to the ball. Using this framework, the robot can continuously track the ball and adapt its footsteps depending on what it sees.

Robustness Our experiments are carried out in controlled environment. The target is well known and easily recognizable by color. The grasping strategy is predetermined by perfect knowledge of the target.

More *robust vision components* can be incorporated into the system with another state-of-the-art object detection. *Machine learning* can also be used to determine the optimal grasping point. This, on one hand, improves the robustness, and on the other, allows grasping unknown objects that had never been seen before but share features with trained objects.

With the existing design of the HRP-2 robot, the *locomotion* on non-flat floor is a challenging problem. Walking on a slope experiment has been carried out and provided promising results. Other research efforts are being made in using control to make walking on unknown and uneven ground possible. New generations of humanoid robots will all hopefully include passive heels. With new robot design and new walking strategy, a new definition of a feasible step will have to be made and fed to the footstep optimizer as constraints.

Future robots In the future, chances are that the humanoid robots will resemble more and more to humans and/or other living organism. For example, one can imagine that the robot will have soft hands with haptic sensors. The control paradigm might also be different [PB06]. A future robot with McKibben pneumatic actuators will have to be controlled differently. The controller, as a well defined mathematical formulation with Jacobians, task space also have to be rethought. Again, machine learning can play an important role when the precise dynamical properties is not known or too difficult to model [Oud10]. However, if the underling controller evolves and the top level interface stays the same, the adaptation scheme presented here will still be applicable as is.

In this work, the conditions imposed on the robot are designed based only on its limits. In the future, to make the robot able to work in human environment, with people around it, safety conditions will have to applied to the robot. These conditions will have to be incorporated in the optimization process, at a higher priority than existing constraints: step feasibility, robot self-collision, obstacle avoidance. The framework based on the perception-decision-action loop presented here will be complementary to the intrinsic adaptation capability of future robots.

Appendix A

Joint description

A joint is characterized by its type (prismatic or revolute) and its relative position and orientation with respect to the parent joint. A complete description of a robot can therefore be obtained by combining the hierarchy tree depicted in Figure 2.5 and the description of each joint. Two popular joint representations are

Direct frame relation between joints For each joint, we record:

- x, y, z: position of the new frame in its parent frame
- r_x, r_y, r_z (raw-pitch-yaw representation), or
  x, y, z, w (quaternion): rotation with respect to parent frame
- type: prismatic or revolute
- axis: (x, y or z)

translation and orientation, usually in quaternion or roll-pitch-yaw form are stored along with the the joint's type and local axis. This representation is used, for example in the URDF format of the PR2 robot or the HRP2's VRML.

Devanit-Haternberg parameters This is a minimal representation proposed by [HD64] describes the relative position of consecutive joint frames in Devanit and Haternberg do that by defining 4 quantities (Figure A.1):

- d: offset along previous z to the common normal
- θ: angle about previous z, from old x to new x
- r: length of the common normal
- α: angle about common normal, from old z axis to new z axis

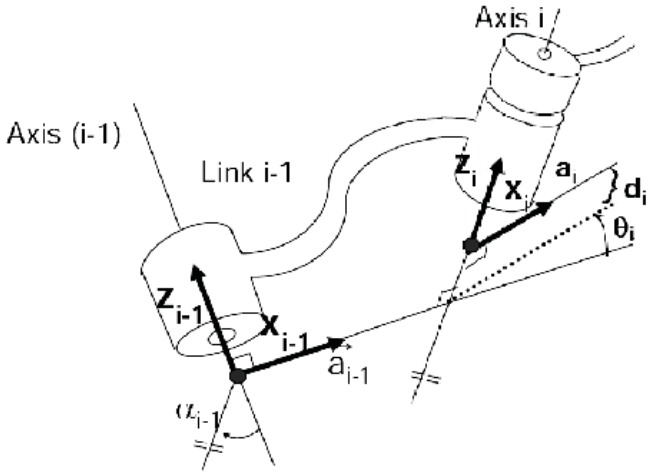


Figure A.1: DH parameter assignments

For all joints (revolute and prismatic), 3 out of 4 quantities are fixed, the fourth term corresponds to the degree of freedom of the joint. This variable is θ for revolute joint and d for prismatic joints. The convention states that all joints revolute or translate around its z axis.

With either representation one should be able to write the transformation matrix between joint $i-1$ and its child (joint i), e.g.

For revolute joint around z axis with direct representation and raw-pitch-yaw representation, with cx, sx, cy ... are abbreviations of $\sin r_x$,$\cos r_x$,$\cos r_y$ ...

$$
\begin{aligned}
{}^{i-1}_{i}T = &\begin{bmatrix} czcx & -czsxcy + szsy & czsxsy + szcy & x \\ sx & cxcy & -cxsy & y \\ -szcx & szsxcy + czsy & -szsxsy + czcy & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\times \begin{bmatrix} c\theta & s\theta & 0 & 0 \\ -s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{A.1}
$$

And for DH parameters, $c\theta$, $s\theta$ ... is the abbreviation for $\cos\theta$, $\sin\theta$ ...

$$
{}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i.c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}.d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i.s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}.d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.2}
$$

Bibliography

[AMS97a] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997. 10.1023/A:1006559212014.

[AMS97b] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113, 1997. 10.1023/A:1006511328852.

[Atk89] C.G. Atkeson. Learning arm kinematics and dynamics. *Annual review of neuroscience*, 12(1):157–183, 1989.

[BDN+07] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner. Grasp planning in complex scenes. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 42 –48, 29 2007-dec. 1 2007.

[Bee79] RC Beecher. Puma: Programmable universal machine for assembly. *Computer vision and sensor-based robots*, pages 64–66, 1979.

[BIC63] A. Ben-Israel and A. Charnes. Contributions to the theory of generalized inverses. *Journal of the Society for Industrial and Applied Mathematics*, 11(3):667–699, 1963.

[BPM+11] L. Baudouin, N. Perrin, T. Moulard, F. Lamiraux, O. Stasse, and E. Yoshida. Real-time replanning using 3d environment for humanoid robot. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 584 –589, oct. 2011.

[Bra98] G. R. Bradski. Conputer vision face tracking for use in a perceptual user interface. *Proc. of Intel Technology Journal, 1998*, 1998.

[BWS+09] Christoph Borst, Thomas Wimbock, Florian Schmidt, Matthias Fuchs, Bernhard Brunner, Franziska Zacharias, Paolo Robuffo Giordano, Rainer Konietschke, Wolfgang Sepp, Stefan Fuchs,

Christian Rink, Alin Albu-Schaffer, and Gerd Hirzinger. Rollin' justin - mobile platform with variable base. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 1597 –1598, may 2009.

[Cap20] K. Capek. Rossum's universal robots. *Prague, CZ*, 1920.

[Cha98] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. *The confluence of vision and control*, pages 66–78, 1998.

[Che95] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790 –799, aug 1995.

[CLC+05] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 629 – 634, April 2005.

[CNKK07] J. Chestnutt, K. Nishiwaki, J. Kuffner, and S. Kagami. An adaptive action model for legged navigation planning. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 196 –202, 29 2007-dec. 1 2007.

[CRTW05] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[CWR01] S.H. Collins, M. Wisse, and A. Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

[DLL11] D. Dang, F. Lamiraux, and J.-P. Laumond. A framework for manipulation and locomotion with realtime footstep replanning. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 676 –681, oct. 2011.

[DLL12] D. Dang, J.P. Laumond, and F. Lamiraux. Experiments on whole-body manipulation and locomotion with footstep real-time optimization. In *Humanoid Robots (Humanoids), 2012 11th IEEE-RAS International Conference on*. IEEE, 2012.

[DS73] Boris M. Dobrotin and Victor D. Scheinman. Design of a computer controlled manipulator for robot research. In *Proceedings of the 3rd international joint conference on Artificial intelligence*,

IJCAI'73, pages 291–297, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[DVS01] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 298 –303 vol.1, 2001.

[EK04] S. Ekvall and D. Kragic. Interactive grasp learning based on human demonstration. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3519 – 3524 Vol.4, 26-may 1, 2004.

[EK09] A. Escande and A. Kheddar. Contact planning for acyclic motion with tasks constraints. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 435–440. IEEE, 2009.

[EKM06] A. Escande, A. Kheddar, and S. Miossec. Planning support contact-points for humanoid robots and experiments on hrp-2. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2974 –2979, oct. 2006.

[EKTL+11] A. El Khoury, M. Taix, F. Lamiraux, et al. Path optimization for humanoid walk planning: an efficient approach. In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics*, pages 179–184, 2011.

[EOR+11] J. Englsberger, C. Ott, M.A. Roa, A. AlbuSchaffer, and G. Hirzinger. Bipedal walking control based on capture point dynamics. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4420–4427. IEEE, 2011.

[GALP07] C. Goldfeder, P.K. Allen, C. Lackner, and R. Pelossof. Grasp planning via decomposition trees. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4679 –4684, april 2007.

[GPFZ06] A. Gil-Pinto, P. Fraisse, and R. Zapata. A decentralized algorithm to adaptive trajectory planning for a group of nonholonomic mobile robots. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 404 –417, oct. 2006.

[GYT05] Yisheng Guan, K. Yokoi, and K. Tanie. Feasibility: Can humanoid robots overcome given obstacles? In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1054 – 1059, apr 2005.

[HBOS05] G. Hirzinger, J. Bals, M. Otter, and J. Stelter. The dlr-kuka success story: robotics research improves industrial robots. *Robotics Automation Magazine, IEEE*, 12(3):16 – 23, sept. 2005.

[HD64] R.S. Hartenberg and J. Denavit. *Kinematic synthesis of linkages*. McGraw-Hill New York, 1964.

[HDW+10] A. Herdt, H. Diedam, P.B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl. Online walking motion generation with automatic footstep placement. *Advanced Robotics, 24*, 5(6):719–737, 2010.

[HHHT98] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1321–1326. Ieee, 1998.

[HKK+07] Kensuke Harada, Shuuji Kajita, Fumio Kanehiro, Kiyoshi Fujiwara, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. Real-time planning of humanoid robot's gait for force-controlled manipulation. *Mechatronics, IEEE/ASME Transactions on*, 12(1):53 –62, feb. 2007.

[HKS+05] K. Harada, S. Kajita, H. Saito, M. Morisawa, F. Kanehiro, K. Fujiwara, K. Kaneko, and H. Hirukawa. A humanoid robot carrying a heavy object. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1712 – 1717, april 2005.

[HKU03] Yoonkwon Hwang, A. Konno, and M. Uchiyama. Whole body cooperative tasks and static stability evaluations for a humanoid robot. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1901 – 1906 vol.2, oct. 2003.

[ICRC07] A.J. Ijspeert, A. Crespi, D. Ryczko, and J.M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

[KAAT03] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. In *Computer Graphics Forum*, volume 22, pages 313–322. Wiley Online Library, 2003.

[Kat74] I. Kato. Information-power machine with senses and limbs (wabot 1). In *First CISM-IFToMM Symp. on Theory and Practice of Robots and Manipulators*, volume 1, pages 11–24. Springer-Verlag, 1974.

[KBC+04] Oussama Khatib, Oliver Brock, Kyong-Sok Chang, Diego Ruspini, Luis Sentis, and Sriram Viji. Human-centered robotics and interactive haptic simulation. *The International Journal of Robotics Research*, 23(2):167–178, 2004.

[KFE96] I. Kamon, T. Flash, and S. Edelman. Learning to grasp using visual information. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2470 –2476 vol.3, apr 1996.

[Kha87] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43 –53, February 1987.

[KJCL97] M. Khatib, H. Jaouni, R. Chatila, and J.P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 2920 –2925 vol.4, apr 1997.

[KKK+01] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. The 3d linear inverted pendulum mode: a simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239 –246 vol.1, 2001.

[KKN+02a] Satoshi Kagami, Tomonobu Kitagawa, Koichi Nishiwaki, Tomomichi Sugihara, Masayuki Inaba, and Hirochika Inoue. A fast dynamically equilibrated walking trajectory generation method of humanoid robot. *Autonomous Robots*, 12:71–82, 2002. 10.1023/A:1013210909840.

[KKN+02b] J.J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[KKN+03] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Online footstep planning for humanoid robots. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 932 – 937 vol.1, September 2003.

[KLW+09] Oussama Kanoun, Florent Lamiraux, Pierre-Brice Wieber, Fumio Kanehiro, Eiichi Yoshida, and Jean-Paul Laumond. Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pages 724–729, Piscataway, NJ, USA, 2009. IEEE Press.

[KLY10] Oussama Kanoun, Jean-Paul Laumond, and Eiichi Yoshida. Planning foot placements for a humanoid robot: A problem of inverse kinematics. *The International Journal of Robotics Research*, 2010.

[KOS+87] I. Kato, S. Ohteru, K. Shirai, T. Matsushima, S. Narita, S. Sugano, T. Kobayashi, and E. Fujisawa. The robot musician wabot2 (waseda robot2). *Robotics*, 3(2):143–155, 1987.

[Kuo99] A.D. Kuo. Stabilization of lateral motion in passive dynamic walking. *The International Journal of Robotics Research*, 18(9):917–930, 1999.

[LL98] F. Lamiraux and J.P. Laumond. A practical approach to feedback control for a mobile robot with trailer. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3291 –3296 vol.4, may 1998.

[LLO11] Olivier Ly, Matthieu Lapeyre, and Pierre-Yves Oudeyer. Bio-inspired vertebral column, compliance and semi-passive dynamics in a lightweight robot. In *IEEE IROS*, San Francisco, United States, 2011.

[LM02] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900 – I–903 vol.1, 2002.

[LTK91] Q. Li, A. Takanishi, and I. Kato. A biped walking robot having a zmp measurement system using universal force-moment sensors. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1568–1573. IEEE, 1991.

[MC07a] N. Mansard and F. Chaumette. Task sequencing for high-level sensor-based control. *Robotics, IEEE Transactions on*, 23(1):60 –72, February 2007.

[MC07b] N. Mansard and F. Chaumette. Task sequencing for high-level sensor-based control. *Robotics, IEEE Transactions on*, 23(1):60 –72, feb. 2007.

[McG90] T. McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–82, 1990.

[MHK+06] M. Morisawa, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, S. Nakaoka, and H. Hirukawa. A biped pattern

generation allowing immediate modification of foot placement in real-time. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 581 –586, dec. 2006.

[MKCA03] A.T. Miller, S. Knoop, H.I. Christensen, and P.K. Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1824 – 1829 vol.2, sept. 2003.

[Mor83] H.P. Moravec. The stanford cart and the cmu rover. *Proceedings of the IEEE*, 71(7):872 – 884, july 1983.

[MSEK09] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1 –6, June 2009.

[NKK+02] K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2684 – 2689 vol.3, 2002.

[NL08] A. Nakhaei and F. Lamiraux. Motion planning for humanoid robots in environments modeled by vision. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 197–204. IEEE, 2008.

[Oud10] P.Y. Oudeyer. On the impact of robotics in behavioral and cognitive sciences: from insect navigation to human cognitive development. *Autonomous Mental Development, IEEE Transactions on*, 2(1):2–16, 2010.

[PB06] R. Pfeifer and J.C. Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[PCDG06] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200 –207, dec. 2006.

[Pop98] A.L.M. Pope. *The CORBA reference guide: understanding the common object request broker architecture*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[PSB+12] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida. Fast humanoid robot collisionfree footstep planning using swept volume approximations. *Robotics, IEEE Transactions on*, 28(2):427 –439, apr 2012.

[PSLY11] N. Perrin, O. Stasse, F. Lamiraux, and E. Yoshida. A biped walking pattern generator based on half-steps , for dimensionality reduction. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1270 –1275, may 2011.

[PT97] L.A. Piegl and W. Tiller. *The NURBS book*. Springer Verlag, 1997.

[QK93] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802 –807 vol.2, May 1993.

[RBN+08] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, et al. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the 17th World Congress*, pages 10823–10825, 2008.

[RMVJ01] D.A. Rosenbaum, R.J. Meulenbroek, J. Vaughan, and C. Jansen. Posture-based motion planning: Applications to grasping. *Psychological Review*, 108(4):709, 2001.

[Ros06] M.E. Rosheim. *Leonardo's lost robots*. Springer Verlag, 2006.

[SCD02] P. Soueres, V. Cadenat, and M. Djeddou. Dynamical sequence of multi-sensor based tasks for mobile robots navigation. In *7th Symp. on Robot Control (SYROCO'03)*, volume 2, pages 423–428, 2002.

[SELT91] K. Salisbury, B. Eberman, M. Levin, and W. Townsend. The design and control of an experimental whole-arm manipulator. In *The fifth international symposium on Robotics research*, pages 233–241. MIT Press, 1991.

[SHV06] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. John Wiley & Sons New York, NY, USA:, 2006.

[SK06] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2641 –2648, May 2006.

[SRM+11] L. Saab, O. Ramos, N. Mansard, P. Soueres, and JY Fourquet. Generic dynamic motion generation with multiple unilateral constraints. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4127–4133. IEEE, 2011.

[SS91] B. Siciliano and J.-J.E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1211 –1216 vol.2, June 1991.

[SVW+08] O. Stasse, B. Verrelst, P.B. Wieber, B. Vanderborght, P. Evrard, A. Kheddar, and K. Yokoi. Modular architecture for humanoid walking pattern prototyping and experiments. *Advanced Robotics, 22*, 6(7):589–611, 2008.

[SWN08] A. Saxena, L. Wong, and A.Y. Ng. Learning grasp strategies with partial shape information. AAAI, 2008.

[Tay04] Abdelhamid Tayebi. Adaptive iterative learning control for robot manipulators. *Automatica*, 40(7):1195 – 1203, 2004.

[TFG+08] R. Tellez, F. Ferro, S. Garcia, E. Gomez, E. Jorge, D. Mora, D. Pinyol, J. Oliver, O. Torres, J. Velazquez, and D. Faconti. Reem-b: An autonomous lightweight human-size humanoid robot. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 462 –468, dec. 2008.

[TIS+04] T. Takubo, K. Inoue, K. Sakata, Y. Mae, and T. Arai. Mobile manipulation of humanoid robots - control method for com position with external force. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1180 – 1185 vol.2, sept.-2 oct. 2004.

[TL89] R.Y. Tsai and R.K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *Robotics and Automation, IEEE Transactions on*, 5(3):345 –358, jun 1989.

[ToLTK90] A. Takanishi, Hun ok Lim, M. Tsuda, and I. Kato. Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 323 –330 vol.1, jul 1990.

[UNN+11] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba. Online decision of foot placement using singular lq preview regulation. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 13 –18, oct. 2011.

[VFJ70] M. Vukobratovic, A. A. Frank, and D. Juricic. On the stability of biped locomotion. *Biomedical Engineering, IEEE Transactions on*, BME-17(1):25 –36, jan. 1970.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001.

[VS72] M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15(1-2):1–37, 1972.

[VSYV06] B. Verrelst, O. Stasse, K. Yokoi, and B. Vanderborght. Dynamically stepping over obstacles by the humanoid robot hrp-2. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 117 –123, dec. 2006.

[WBdLJ08] Keenan Wyrobek, Eric Berger, H.F. Machiel Van der Loos, and J. Kenneth Salisbury Jr. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In *Proc. International Conference on Robotics and Automation (ICRA)*, 2008.

[Wie02] Pierre-Brice Wieber. On the stability of walking systems. In *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba Japon, 2002.

[YPL+10] Eiichi Yoshida, Mathieu Poirier, Jean-Paul Laumond, Oussama Kanoun, Florent Lamiraux, Rachid Alami, and Kazuhito Yokoi. Pivoting based manipulation by humanoid robot. *Autonomous Robots*, 28:77–88, 2010. 10.1007/s10514-009-9143-x.

List of Figures

List of Tables

Abstract

This thesis focuses on realization of tasks with locomotion on humanoid robots. Thanks to their numerous degrees of freedom, humanoid robots possess a very high level of redundancy. On the other hand, humanoids are underactuated in the sense that the position and orientation of the base are not directly controlled by any motor. These two aspects, usually studied separately in manipulation and locomotion research, are unified in a same framework in this thesis and are resolved as one unique problem. Moreover, the generation of a complex movement involving both tasks and footsteps is also improved becomes reactive. By dividing the optimization process into appropriate stages and by feeding directly the intermediate result to a task-based controller, footsteps can be calculated and adapted in real-time to deal with changes in the environment. A perception module is also developed to build a closed perception-decision-action loop. This architecture combining planning and reactivity validated on the HRP-2 robot. Two classes of experiments are carried out. In one case the robot has to grasp an object far away at different height level. In the other, the robot has to step over an object on the floor. In both cases, the execution conditions are updated in real-time to deal with the dynamics of the environment: changes in position of the target to be caught or of the obstacle to be stepped over.

Keywords: manipulation, locomotion, footstep optimization, real-time, adaptation, computer vision, visual servoing, reactivity

Résumé

Cette thèse porte sur la réalisation des tâches avec la locomotion sur des robots humanoïdes. Grâce à leurs nombreux degrés de liberté, ces robots possèdent un très haut niveau de redondance. D'autre part, les humanoïdes sont sous-actionnés dans le sens où la position et l'orientation ne sont pas directement contrôlées par un moteur. Ces deux aspects, le plus souvent étudiés séparément dans la littérature, sont envisagés ici dans un même cadre. En outre, la génération d'un mouvement complexe impliquant à la fois des tâches de manipulation et de locomotion, étudiée habituellement sous l'angle de la planification de mouvement, est abordée ici dans sa composante réactivité temps réel. En divisant le processus d'optimisation en deux étapes, un contrôleur basé sur la notion de pile de tâches permet l'adaptation temps réel des empreintes de pas planifiées dans la première étape. Un module de perception est également conçu pour créer une boucle fermée de perception-décision-action. Cette architecture combinant planification et réactivité est validée sur le robot HRP-2. Deux classes d'expériences sont menées. Dans un cas, le robot doit saisir un objet éloigné, posé sur une table ou sur le sol. Dans l'autre, le robot doit franchir un obstacle. Dans les deux cas, les condition d'exécution sont mises à jour en temps réel pour faire face à la dynamique de l'environnement : changement de position de l'objet à saisir ou de l'obstacle à franchir.

Mots-clés: manipulation, locomotion, optimisation de pas, temps réel, adaptation, vision par ordinateur, asservissement visuel, réactivité