DESIGN AND IMPLEMENTATION OF AN AUTONOMOUS CLIMBING ROBOT

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ruixiang Zhang
January 2014

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Jean-Claude Latombe) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Oussama Khatib)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Kenneth Salisbury)

Approved for the University Committee on Graduate Studies

Abstract

This dissertation describes the design, implementation, and experimentation of an autonomous free-climbing robot, Capuchin. The objective of our project is to create a multi-limbed robot capable of climbing vertical terrain autonomously using techniques similar to those used by human free climbers. When a "free" climber climbs a steep crag or an artificial climbing wall, she uses nothing else but her hands and feet to make contact with terrain features such as holes, cracks, ledges or protrusions. Unlike "aid" climbing, which uses special equipments, tools, or engineered features, free climbing only relies on friction at the contacts between the climber and the terrain. In order to make a multi-limbed robot climb in a similar way, four fundamental challenges must be addressed: robot design, sensing, motion planning and motion control. Our work focuses on robot design (including sensors) and motion control. However, our robot, Capuchin, is an integrated system including a simplified sensing system and a pre-existing motion planner running off-line.

A good robot design can increase the inherent ability of the robot to climb complex terrain. It may also lead to better performance and make other issues easier, such as motion planning and control. A four-limb structure was chosen after consideration of the robot's capability and complexity. Simulation was used during the design process to optimize performance, in particular to maximize the workspace reachable by the end-effectors (fingers).

Sensors have been selected to allow the robot to both acquire information about the terrain and control its motion. Each finger is equipped with a camera. Vision feedback allows the robot controller to accurately dock the finger on a terrain feature at a location computed by the planner. It also allows modifying a planned trajectory

in real-time, when the terrain differs slightly from the model that had been used by the planner or when other small errors occur (for instance, if the robot slips slightly at a contact). Each finger is additionally equipped with a force sensor that gives the magnitude and orientation of the reaction force at a contact. The four force sensors are used by the robot controller to maintain the robot in quasi-static equilibrium, by adjusting the robot posture and the joint torques when needed, so that the reaction forces at the contact point continuously within their Coulomb friction cones.

A two-stage motion planner previously developed by Bretl and Hauser for free-climbing and other multi-limbed robots navigating on challenging and irregular terrain is used in this work. This planner decomposes a climbing motion into a sequence of moves, each performed with a fixed set of robot-terrain contacts (this set is called a "stance"). The transition at the end of each move consists of either breaking a contact or making a new one. The planner first computes a sequence of stances. Next it computes a trajectory for the move to be performed at each stance. If it fails to find a move at one step, it considers another sequence of stances.

The core part of our research has been the design of the motion controller. The main problem we had to solve is a multi-contact force control problem. One of our most important findings has been the following: for quasi-static climbing, it is not necessary, even not desirable, to continuously control the forces exerted by the robot at the contact points. Instead, it is preferable to continuously monitor these forces and perform joint torque adjustments only when some reaction forces get too close to the boundaries of the friction cones or to their maximal magnitude. This strategy was not obvious when we started our research. In fact, we first developed a motion controller that continuously adjusted joint torques to keep measured reaction forces as close as possible to the terrain normals at the contact points. But computing these adjustments is rather time consuming. Moreover, this approach leads the robot to perform delicate adjustments frequently. As a result, robot motion was neither as smooth, nor as reliable as we would have liked. Instead, our new approach, which we call "lazy" force control, leads to a faster servo rate and much smoother motion. Our experiments show that on average adjustments only amount for a small percentage (less than 10%) of the total time spent climbing. They also demonstrate that

Capuchin can reliably climb vertical artificial climbing walls autonomously and can handle small errors in the terrain model used by the planner.

Acknowledgements

I am deeply grateful to my advisor Professor Jean-Claude Latombe for his guidance and support during my Ph.D. study at Stanford. I would not have enjoyed my life so much at Stanford without his help and patience.

I would also like to thank my readers, Professor Oussama Khatib and Professor Kenneth Salisbury, and other committee members, Professor Steve Rock and Professor Scott Delp.

It is a pleasure to thank our group members, Peggy Yao, Ankur Dhanik, Kris Hauser, Phil Fong, Tim Bretl and Liangjun Zhang, who made the years in our lab, Clark S245, memorable.

I'm thankful to my parents and my beloved Rui for their love and support.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project and my Ph.D. study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Goal and motivation

Compared to wheeled or track robots, legged robots are potentially more capable of traveling over challenging (e.g., steep and irregular) terrain. So, it is not surprising that much research has been carried out in recent years to design such robots, for instance humanoid robots [30], quadruped robots [57], and other multi-limbed robots [71]. Figure 1.1 shows a few of these robots. Various forms of locomotion have been successively achieved, but mostly on flat and horizontal terrain, or on regular and piecewise flat terrain (e.g., stairs). Some motion planners have also been specifically developed to allow legged robots to navigate without collision among obstacles [38], but with no specific consideration for equilibrium. A relatively small body of research has focused on dynamic locomotion on challenging terrain. Big Dog [57], a quadruped robot, provides an impressive example of such research. Other projects have focused on learning navigation skills on uneven, but only slightly sloped terrain, mostly with quadruped robots [40]. To our knowledge, there has not been any major research project aimed at developing an integrated autonomous multi-limbed robot equipped with appropriate sensing, planning, and control capabilities to "free-climb" quasi-vertical and irregular terrain to reach user-specified goal locations. The aim of our research has been to create such a robot, Capuchin.

Figure 1.1: A few examples of legged robots: (left) Asimo is a humanoid robot developed by Honda; (center) Big Dog is a quadruped robot developed by Boston Dynamics for fast navigation on various types of terrains; (right) Athlete is a six-legged robot developed by JPL for planet exploration.

(a) A terrain feature (small ledge) on a rock wall.

(b) Fingers at a contact hold on this feature.

Figure 1.2: Terrain feature and hold

Unlike *aid* climbing that takes advantage of special equipment, tools and/or engineered terrain features, *free* climbing only relies on friction at the contacts between the climber and the rigid terrain. So, a human free climber moves on a steep rock crag or an artificial climbing wall using nothing else but her body (mostly her hands

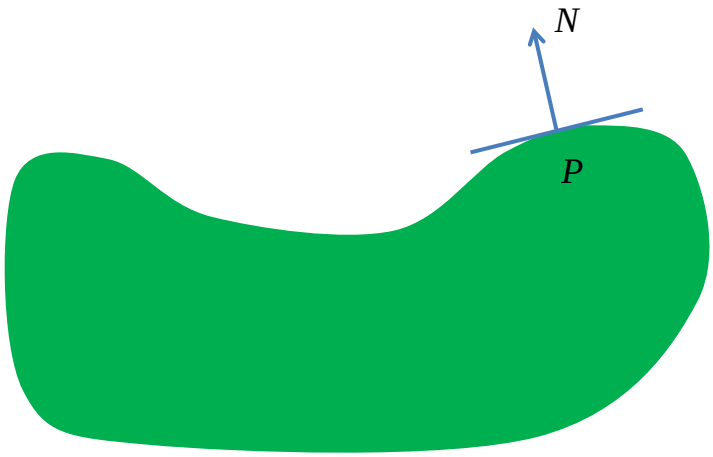


Figure 1.3: A hold (P, N) on a terrain feature. The climbing wall of Capuchin has several such features, as will be shown in Figure 2.12.

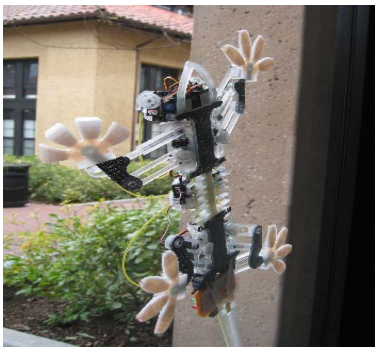
and feet) to make contacts with irregularly distributed terrain features, such as protrusions, holes, ledges, and cracks (Figure 1.2(a)). We will call the small portion of terrain surface where the climber makes a contact a *hold* (Figure 1.2(b)). In our work, we will model each hold by a point P defined by its coordinates and a normal vector N, the normal to the terrain at P, as shown in Figure 1.3.

The climber alternatively breaks contacts and creates new ones. Due to the irregularity of the terrain, each move is unique. While moving, the climber adjusts her body posture (hence, the position of her center of mass) and exerts appropriate forces at the holds in order to remain in equilibrium. The overall motion of a good human climber usually seems relatively slow, but smooth and fluid. This motion is the result of both deliberate planning and careful control using visual and tactile feedback. Vision is needed to identify candidate holds and tactile sensing is used to perceive their detailed shape and optimize contact locations. Although holds with normals pointing upward are often promising, they may not always exist and holds with other orientations can be used as well. Equilibrium only requires that any force exerted by the body and gravitational force be balanced by other exerted forces. Because adopting adequate body postures, performing precise motion, and applying appropriate forces

involve strategic thinking, free climbing is often considered by human climbers to be both a problem-solving and a physical activity. Mere strength is not enough.

Creating an autonomous free-climbing robot is obviously a challenging project. From a research viewpoint the goal is interesting by itself, as it can lead to making progress in a number of important technical areas of robotics, such as complex planning, multi-contact control, equilibrium maintenance, and delicate use of sensor feedback. In this context, a special motivation is that "cheating" is almost impossible: any significant imperfection in the design of the system, and the robot will fall frequently. Conversely, if the robot operates successfully, it means that key issues must have been addressed satisfactorily. So, research on free-climbing robots may eventually lead to developing effective methods that will benefit all multi-limbed robots, even robots navigating on less challenging terrain. But, in addition, such research may also contribute to making progress in a number of specific applications, like search-and-rescue in disaster environments, e.g., in rubbles formed by collapsed buildings after an earthquake, and planetary exploration, e.g., to survey steep craters and cliffs on the Moon and Mars.

The work presented in this dissertation builds upon the previous work done at Stanford by Tim Bretl [12, 13, 14, 15, 16], Kris Hauser [26, 27, 28], and Teresa Miller [47, 48]. The design of our robot, Capuchin, benefited from the experiments conducted by Bretl on a previous climbing robot, Lemur, designed and built by NASA-JPL. The planner used by Capuchin is the planner initially developed by Bretl and later improved by Hauser. Capuchin's motion controller benefited from our analysis of the limitations of Miller's early controller.



Stickybot (Stanford)

Spinybot (Boston Dynamics)

Figure 1.4: Stickybot and Spinybot. Stickybot climbs flat surfaces, while Spinybot climbs surfaces with tiny texture.

1.2 Related work

1.2.1 Climbing robots for specific uses

Various climbing robots have been developed over the past 10-15 years. Most of them rely on special robot hardware, equipment or tools, or on engineered features of the terrain. For example, robots for engineered environments [1, 4, 6, 9, 10, 58, 73, 75] have end-effectors, such as pegs, hooks, and special grippers, that match engineered features of the environment, like handrails, wire fences, bars, and poles. Robots for inspection of pipes and ducts [51, 59, 77] rely on frictional contacts with surfaces (as do free climbers), but take advantage of geometric regularity to perform precomputed cyclic gaited motion. Adhesive robots [3, 7, 17, 18, 20, 22, 35] stick to flat or smoothly curved surfaces using devices like suction cups or magnets. Consequently, they are limited to environments consisting of glass, metal, or other smooth surfaces. Recently, bio-inspired robot feet have been developed to create robots that can climb on building walls, tiles, and other smooth surfaces. Among them, Stickybot [37] uses a rubber-like material with tiny polymer hairs made from a micro-scale mold to mimic gecko's feet, while Spinybot [5] has feet equipped with many tiny claws. See Figure 1.4. None of these robots could free-climb vertical terrain with both small and large irregular features.

1.2.2 Lemur: A first attempt to build a free-climbing robot

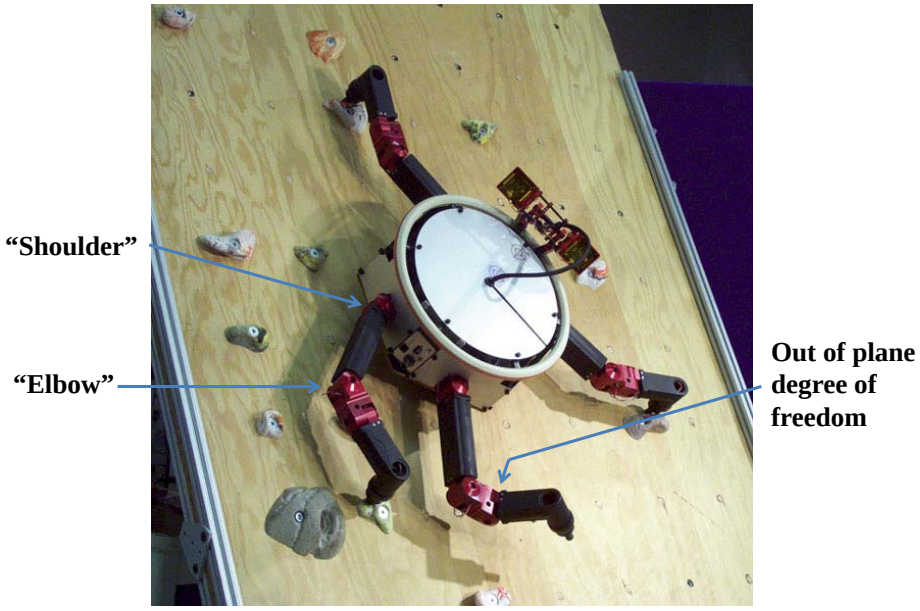


Figure 1.5: The Lemur robot designed and built by the Jet Propulsion Laboratory. To our knowledge, this is the first free-climbing robot ever built.

Lemur IIb (Figure 1.5) is a four-limbed climbing robot created by the Jet Propulsion Laboratory (NASA) [82]. To our knowledge, this is the first attempt to build a free-climbing robot. Lemur consists of four identical limbs mounted on a circular chassis with equal spacing between them. The robot has a total mass of 7 kg. Each limb has three revolute joints, providing two in-plane ("shoulder" and "elbow") and one out-of-plane degrees of freedom. All joints are highly geared and have the same drive-trains, capable of a maximum continuous torque of 5.0 Nm and a maximum speed of 45 deg/s. Each end-effector is a single peg wrapped with high-friction rubber (similar to the rubber covering human climbing shoes). Lemur's elbow/knee joint can bend in both directions, but with a mechanical stop at 90 degrees. This 90-degree limit restricts significantly the reachable workspace of each Lemur's finger. It prevents Lemur from performing useful motions often performed by human climbers (see Figure 1.6) which require complete folding of the knee or elbow. It also makes

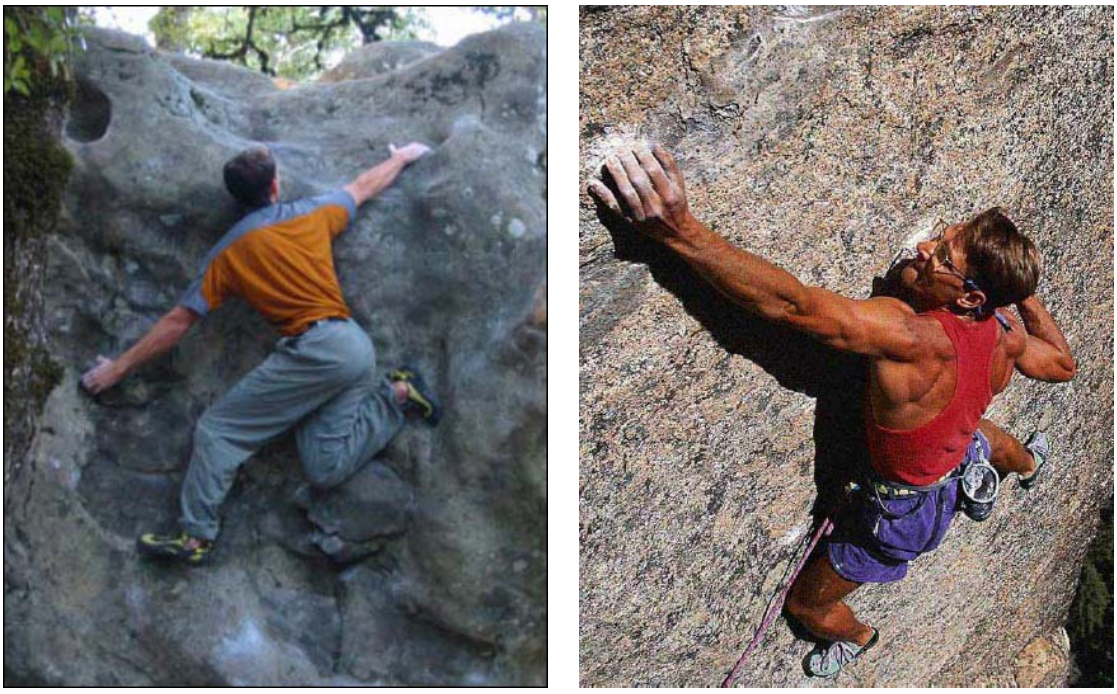


Figure 1.6: Human climbers bending their knee (left) or elbow (right) beyond 90 degrees.

control more complicated due to the non-unique solution of the inverse kinematics of each limb. These lessons were taken into account in the design of Capuchin, which will be discussed in Chapter 2.

Using Lemur, Bretl developed an offline multi-step planner [15]. This planner takes a model of the terrain as input, which lists all the possible holds (P, N) where the robot can make contact. The planning algorithm will be introduced in Section 3.4. For Capuchin, we are using an extended version of this planner developed and adapted to our robot by Hauser [26].

An open-loop position controller was implemented on Lemur to perform the trajectories computed by the planner. Despite the fact that this controller does not use force or vision feedback, Lemur was able to execute free-climbing motions computed by Bretl's planner at very slow velocity to reduce positioning errors and slipping risks. As one would expect from an open-loop climbing robot, the resulting system was not

reliable and Lemur felt frequently. Lack of reliability was mainly caused by both errors in positioning fingers at contact locations and application of improper contact forces. Vision and force feedback is needed to achieve accurate positioning of fingers at holds and exert adequate contact forces. Even if contacts were achieved accurately, force control would still be necessary to maintain balance (see Chapter 4). Designing vision and force feedback control system to achieve reliable free-climbing motion has been the primary objective of our research with Capuchin.

1.2.3 Tenzing

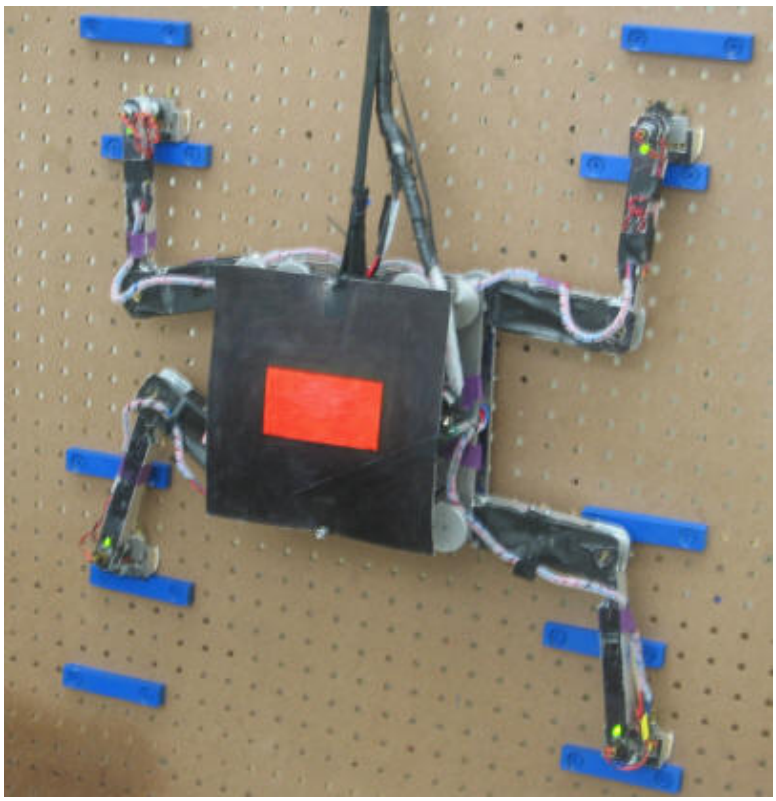


Figure 1.7: The Tenzing robot designed and built at Dartmouth College

Tenzing (Figure 1.7) is a free-climbing robot built at Dartmouth College [42, 81]. Like Lemur, it is a four-limb planar robot with two revolute joints in each limb. A hobbyist servo motor is used on each elbow and shoulder joint. Tenzing is designed

to climb a planar wall with artificial terrain features. A force sensor is mounted on each limb endpoint to control the magnitude of the contact force along the vertical direction. The body has a tilt sensor used to keep the body upright. An embedded controller performs low-level coordination. The controller communicates with a Java-based server running on a PC using a RS232 serial port. A camera, not mounted on the robot and located at some distance away from the wall, is used to determine the position of the robot and locate terrain features on the wall. It is reported in [42, 81] that the robot can climb in an interactive mode and in an automatic mode. In the interactive mode, a human user enters a sequence of holds on a graphic interface. In the automatic mode, a program automatically plans a path up the wall. The details of the control and planning algorithms are not available. In addition, there are some obvious limitations in the design of Tenzing. Hobbyist servo motors are inexpensive, but they have low joint angle precision. In addition, the belt used to drive the elbow joint increases backlash greatly. Therefore, precise position control cannot be achieved. Even more important, the force sensors only measure the vertical components of the contact forces. In general, this information is insufficient to maintain the robot in static equilibrium. For this reason, the robot, which is constrained to keep its body upright, can only climb walls equipped with relatively large terrain features having an horizontal contact surface pointing upward (in blue in Figure 1.7). The videos (available from [81]) show that Tenzing motions are not very smooth. Capuchin does not have these limitations. Unlike Tenzing, it is equipped with vision sensors to precisely dock its fingers against the holds selected by the planner. The normals of the holds do not have to point upward. Capuchin's force sensors measure the magnitudes and orientations of the reaction forces. Both its planner and controller are more sophisticated in order to exploit the greater capabilities of the robot and its sensors.

1.3 Project Overview

As we have already mentioned, free-climbing is challenging because it only relies on friction at the contacts between the robot and selected holds. For a robot to free-climb autonomously, four fundamental challenges must be considered: robot design, sensing, planning and control. We briefly review each of them below.

1.3.1 Robot design

A good robot design is important to achieve our objective: climbing like a human climber. It can increase the inherent ability of the robot to climb complex terrain. It may also lead to better performance and make other issues easier, like control and planning. Thanks to its many degrees of freedom the human body can achieve complicated and difficult motions. But it would be extremely difficult, at best, to design and build a robot capable to fully emulate the human body. Moreover, the control of such a robot would also be prohibitively complex. So, while our robot should have the same basic structure and abilities that the human body has, it should also be simple enough to be actually built and relatively easy to control. In other words, our design must strike a good balance between functionality and complexity. Our general design guideline has been to achieve the functions needed for free-climbing in a way that is as simple as possible.

We chose a four-limb structure after careful consideration of the robot's capability and complexity. Fewer limbs would not allow the robot to maintain quasi-static equilibrium in the presence of arbitrarily oriented holds. More than four limbs would lead to greater complexity in robot design, self-collision avoidance, and planning, as well as increased cost and weight. Simulation was used during the design process to optimize performance, in particular to maximize the workspace reachable by the endpoints (fingers) of the limbs. In Chapter 2, we will discuss the design and building of our robot, Capuchin.

1.3.2 Sensing and feedback

Humans have multiple sensors such as eyes, ears and hands to see, listen to and feel the world. Among them vision and touch are critical for climbing. However, the location of the human eyes is far from optimal for climbing. Indeed, eyes are too close to the climbing terrain and, so, holds are often occluded. But humans have good tactile sensing in fingertips. By moving a finger into contact with a visually occluded region of the terrain, a human climber can perceive the shape of a feature (including its roughness) and use this information to adjust the position of the finger appropriately. Unfortunately, despite recent progress [43], technology does not exist yet to create a sensitive and robust (that is, time resistant) tactile sensor for a climbing robot. Therefore, instead, we chose to equip each finger of Capuchin with a camera to precisely determine its position relative to a hold when it makes a new contact. Forces exerted at contacts are also important information that a human climber feels in order to maintain balance and avoid slipping. So, we decided to equip each fingertip of Capuchin with a three-component force sensor. Such a sensor gives both the magnitude and orientation of a reaction force at a hold.

1.3.3 Motion planning

The choice of holds is critical for a free-climbing robot. Only a limited number of potential holds are available in any climbing terrain and only a much smaller number are actually useful. Hold sparsity led Bretl to design a two-stage motion planner that first selects the sequence of holds where the robot should place its fingers, and next plan the continuous motion to achieve these contacts [15]. Hauser later extended this planner in functionality and performance [26]. In our system, we use this planner, which was adapted to Capuchin by Hauser. More detail will be given in Chapter 3.

1.3.4 Motion control

Following the design and construction of Capuchin, the most important part of our work has been the development of Capuchin's motion controller. The motion plans computed by the planner are theoretically correct, assuming that the model of the

terrain is accurate. But even if this assumption could be rigorously achieved, the robot would still have to apply adequate forces at the holds during the execution in order to remain in equilibrium. The role of the motion controller is to concurrently perform the motion computed by the planner, handle small discrepancies between the planner's model and the real world, and make sure that reaction forces keep the robot in quasi-static equilibrium.

Capuchin's controller uses the visual feedback provided by the cameras mounted on the fingers to accurately dock the fingers at the holds selected by the planner. It uses the data provided by the force sensors to monitor and adjust the contact forces so that they keep satisfying the static-equilibrium constraints. An important finding of this project is that, for quasi-static climbing, it is not necessary, or even desirable, to perform continuous force control. Instead, it is preferable to monitor contact forces continuously and perform occasional force adjustments when they are needed. This "lazy" force control approach was not obvious at the beginning of our work. Our tests have shown that it achieves both reliability and reasonable performance. The experiments reported in Chapter 5 will demonstrate the key role played by vision and force feedback.

1.4 Research contributions

The main contributions of our work are the following:

- Design and construction of a four-limb free-climbing robot, Capuchin

  A four-limbed free-climbing robot Capuchin was designed and built after careful consideration of the robot's capability and complexity. "Achieve the necessary functions in the simplest possible way" was our guideline. Analogies and simulations were also used during the early phases of the design. The design of the elbow joints, the motor-to-joint connections, and the mounting of the fingers directly on the shafts of the force sensors are novel design ideas that together play an important role in Capuchin's performance.

- Integration of vision and force sensors

Vision and force sensors allow Capuchin to sense the terrain and control its motion. Each finger is equipped with a camera to allow the robot to accurately dock the finger on a hold or modify a planned trajectory in real-time, when there are small errors in the terrain model or when a finger slips slightly. Each finger is also equipped with a force sensor that measures the magnitude and orientation of the force exerted at a contact. Appropriate signal processing removes noise from the data returned by the sensor. The four force sensors are used by the motion controller to adjust contact forces and keep the robot in quasi-static equilibrium.

- Lazy control algorithm for executing Capuchin's motions

  Instead of continuously controlling forces, our lazy controller monitors forces continuously and adjusts them only when the quasi-static equilibrium constraints are close to be violated. This approach, which reduces the amount of on-line computation, allows smoother motion. It also reduces the number of delicate force adjustments. Implementation and tests demonstrate that it achieves reliability and good performance (in terms of speed and motion smoothness). In our experiments force adjustments account for a small fraction of the total execution time of a climbing motion (typically, less than 10%).

- Integrated implementation

  The overall Capuchin system integrates our lazy force control algorithm, sensor feedback, and the motion planner provided by Hauser. Its implementation and tests have demonstrated that Capuchin is able to autonomously and reliably free-climb a vertical artificial climbing wall with various distributions of holds. Experiments and data collected during those experiments validate our technical choices.

Chapter 2

Robot design

Robot design determines the inherent capabilities of a robot. It is therefore a fundamental step in building a new robot system. A good design can increase the performance of the robot and may often make other issues like control and planning easier.

Robot design requires making tradeoffs among many factors, such as functional capability and complexity, weight and strength of mechanical parts, weight and power of actuators, cost and performance of sensors, and so on. It is a complicated process that has not a single optimal solution. There is no standard method to comparatively evaluate the end result. The high-level guideline that we used throughout our work is to *achieve the functions needed for free-climbing with the simplest possible design.*

2.1 Kinematic design

The kinematic design of Capuchin required us to consider several important issues, in particular: number of limbs, number of degrees of freedom (DOFs) in each limb and body, distribution of these DOFs over the robot structure, and detailed specification of each link of the limbs and body. After resolving these issues, our work focused on the detailed design of each part. During the initial phase of the process, we loosely used the body structure of humans and animals that are good at climbing as a source of inspiration. We then ran more formal and quantitative simulations to evaluate

key capabilities, like workspace reachability. We also took advantage of the lessons learned during the experiments performed by Bretl with the Lemur robot (see Section 1.2.2), especially in the design of the elbow/knee joint.

2.1.1 Overall design

Early on we made the important decision (consistent with the high-level design guideline mentioned above) that each of the robot's end-effectors would be a rigid cylindrical peg, called a finger throughout this thesis. We briefly considered simple grippers, but to be actually useful they would have had to be able to exert large squeezing forces, for instance to pinch small terrain protrusions. Hence, they would have considerably increased the weight and complexity of the robot. They would also have led to more complicated contact models in both the planning and control software. Although human climbers sometimes use several fingers to squeeze terrain features or their fists/feet to achieve secure jams in narrow cracks, most of their moves relies on simple contacts. When multiple fingers are used together, it is mostly to achieve greater combined strength. Another important early decision was to limit Capuchin to quasi-static climbing. Under this decision, the robot must be in static equilibrium at any one time. This may sound as a major restriction, but in fact most good climbers avoid dynamic moves, as relying on momentum to perform difficult moves is risky and potentially tiring. So, the restriction is less important as it may seem.

With only rigid cylindrical fingers to make contact with the terrain, the robot needs at least three simultaneous contacts in order to maintain static equilibrium. Hence, at least four limbs are required, to allow one limb to move to a new hold, while the other three maintain balance. More limbs, like in the ATHLETE robot (see Figure 1.1), would allow more simultaneous contacts, hence could make it easier to achieve equilibrium. But this choice would have increased weight and complexity, as more DOFs would have had to be coordinated to avoid self-collision among limbs. It would also have required the robot to find more good holds on the climbing wall, while such holds are often sparse. Again following our high-level design guideline, we decided that Capuchin will be a four-limb robot, with two links on each limb.

In general, a human climber tries to keep her body as close as possible to the terrain, in order to reduce the magnitude of the forces that must be exerted at the contacts to maintain balance. The same is also true for a climbing robot. Therefore, our design aims at keeping the center of mass of the robot as close to the climbing terrain as possible.

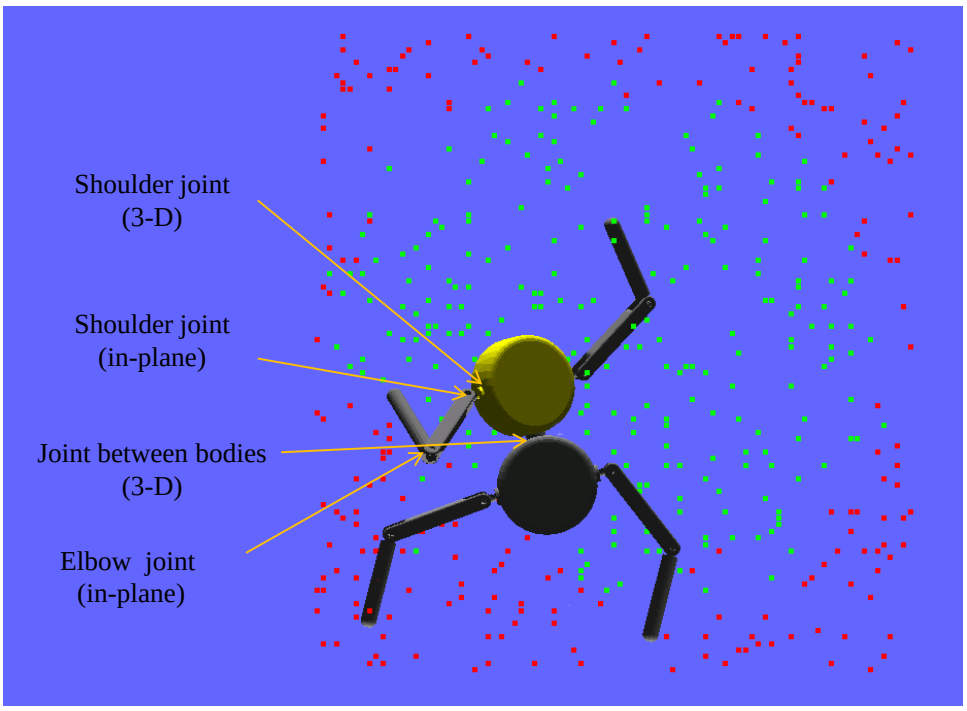
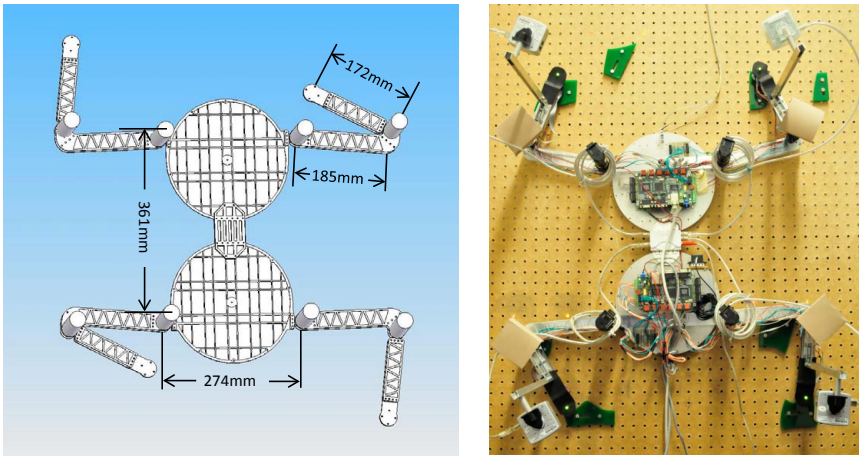


Figure 2.1: Capuchin-3D has a revolute DOF between its upper and lower bodies, two revolute DOFs in each shoulder/hip, and one revolute DOF in each elbow/knee. Simulation was used to optimize kinematic reachability. Here, the green points are kinematically reachable by the top-right finger, while keeping the other three fingers fixed. The red points are not reachable.

Our initial goal was to design a robot (hereafter called Capuchin-3D) that would be capable to climb arbitrarily curved surfaces with protrusions and depressions. This goal led to a preliminary design depicted in Figure 2.1. In contrast to the Lemur robot, which has only one body, Capuchin-3D has an upper and a lower body connected by a revolute joint. Two limbs are attached to each of the two bodies, and each limb has three DOFs: two in the shoulder/hip between the body and the first link of the limb,



(a) Planar design of Capuchin (b) Capuchin on a climbing wall

Figure 2.2: The planar version of the Capuchin design (a) and its implementation (b). The flat green plastic objects mounted on the climbing wall in (b) are the features on which all holds are located. These features have some small thickness and the holds (candidate points where the robot may make contact) are located on their contours. Holes in the supporting panel are not used by Capuchin. They are only used to fix features and easily change their distribution.

one in the elbow/knee between the first and the second link of the limb. In principle, the combination of the revolute DOF between the upper and lower bodies and the two DOFs in each shoulder/hip would have allowed Capuchin-3D to climb complex curved surfaces by twisting the DOF between the upper and lower bodies (as shown in Figure 2.1).

However, we quickly found out that this design was too complicated and expensive to implement. It would also have required too much time to make it operational. So, we decided to scale it down by restricting the climbing abilities of Capuchin to planar surfaces. We retained most of the initial design, but we eliminated the revolute DOF between the upper and lower bodies and we kept only one DOF in each shoulder/hip. The resulting design and its final implementation are shown in Figure 2.2. In this simplified design each limb has two revolute joints, providing two in-plane DOFs. Note that this design has a slight drawback in addition to the limitation to planar terrain. The terrain features (the flat green plastic objects in Figure 2.2(b)) must have some thickness so that their contours can be used for robot-terrain contacts.

Because Capuchin cannot lift its fingers, it must move them around the features to avoid collision. This is a constraint that the planner must take into account.

For simplification, throughout the rest of this thesis the terms shoulder and elbow will be used respectively to refer to the first and second joints in both the upper and lower limbs, instead of shoulder/hip and elbow/knee.

2.1.2 Geometric dimensions

Some capabilities, such as the workspace reachable by each finger, are very important for the robot to climb. We chose several parameters of Capuchin with the help of simulation in order to maximize its capabilities. For example, we used simulation to determine the optimal length of a limb relative to the size of the body and the relative length of the two links of a limb. For a given set of relative dimensions, we generated a large distribution of contact points at random, picked three of them as the positions of three of the robot fingers and determined the reachability of all other points by the fourth finger using inverse kinematics. To illustrate Figure 2.1 shows the workspace of the upper right finger of Capuchin-3D when the other three fingers have fixed positions. The green dots are reachable, while the red ones are not. Such simulation, as well as weight and actuation considerations, led to the dimensions of Capuchin given at the end of this chapter.

2.1.3 Shoulder joints

Each Capuchin's shoulder joint has an angular range of 0 to 225 degrees. This range is slightly larger than that of Lemur's shoulder, which from 0 to 180 degrees.

2.1.4 Elbow joints

Lemur's elbow angle can vary between -90 and $+90$ degrees. There are two limitations in this choice. First, Lemur's elbow cannot bend beyond 90 degrees (either way) and as a consequence the workspace reachable by a finger is limited to a relatively narrow region shown in Figure 2.3. As mentioned in Section 1.2.2, this limitation

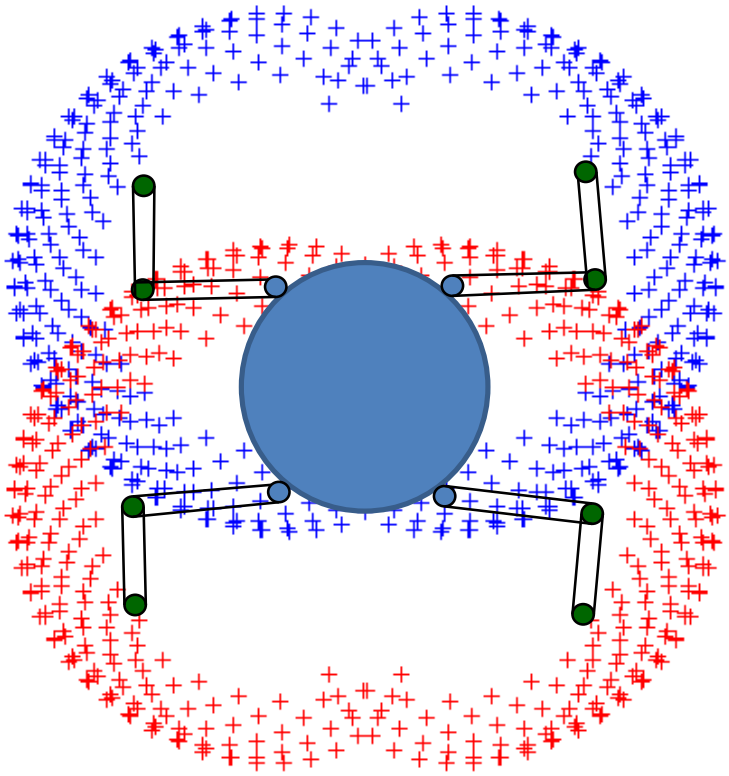


Figure 2.3: Workspace of Lemur's fingers for a fixed position of the body: blue for the upper limbs and red for the lower limbs. The [-90dg,+90dg] range of the elbow joints prevents Lemur from performing important climbing moves (see Figure 1.6).

prevents Lemur from achieving some important climbing postures often used by human climbers, which require folding a limb almost completely (Figure 1.6). Second, since the joint is able to bend to both sides relative to its straight configuration, each limb has two inverse kinematics solutions, making control potentially more complicated. Instead, Capuchin's elbow angle can vary between 10 and 180 degrees. This choice increases the size of the workspace region reachable by a finger (Figure 2.4). Comparing this figure with Figure 2.3, one can see that large regions above and below the robot body are not reachable by Lemur's fingers, while they are reachable by Capuchin's fingers. Reachability of these regions is needed to achieve postures similar to those shown in Figure 1.6. There are two small regions reachable by Lemur, on both sides of its body, but not by Capuchin, but this area has relatively low utility

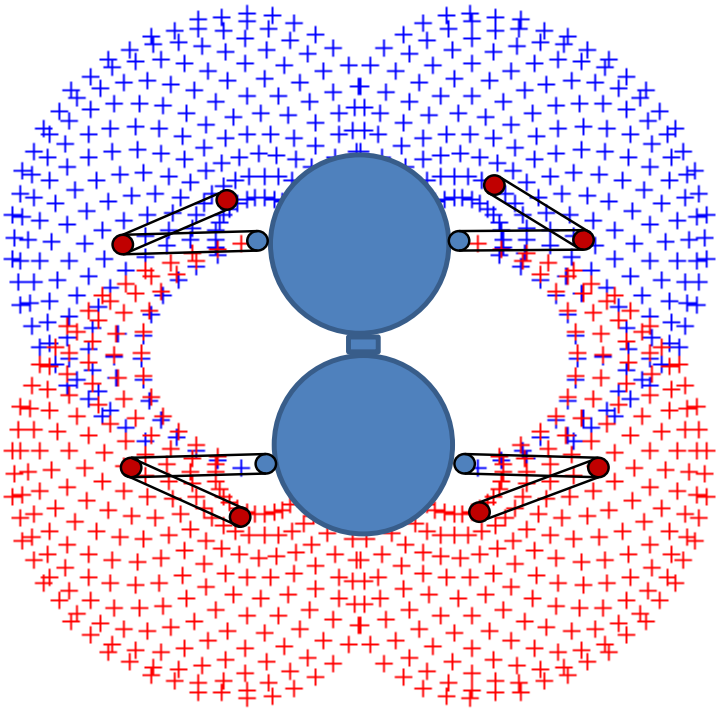


Figure 2.4: Workspace of Capuchin's fingers for a fixed position of the body: blue for the upper limbs and red for the lower limbs.

as it is one where the upper and lower limbs may collide with one another.

While we wanted to create an elbow joint that can achieve a small folding angle, another goal was to keep both links of a limb in the same plane in order to keep the robot's center of mass as close as possible to the climbing wall. To achieve this twofold goal, we designed the shapes of the two links as shown in Figure 2.5.

2.1.5 Fingers

As we indicated before, we decided at a very early stage of the design process that the end-effectors of the limbs would be cylindrical fingers. Any other choice would have greatly complicated the mechanical design of the robot and its actuators, sensors, planner, and controller. As we will see in Chapter 5, this choice still allows Capuchin to climb interesting terrains.

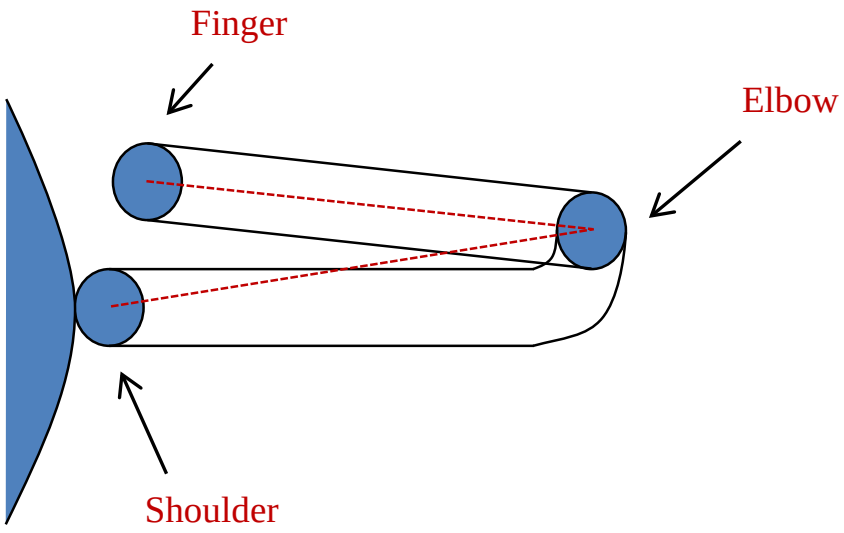


Figure 2.5: Shapes of the two links of a limb at the elbow joint.

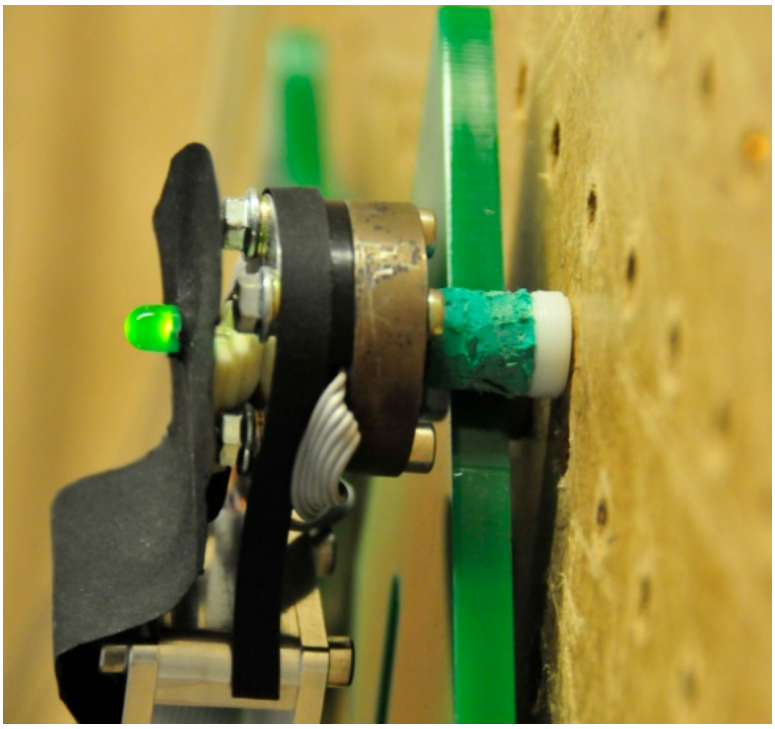


Figure 2.6: Finger design

Figure 2.6 shows a Capuchin finger in contact with a terrain feature. The finger is wrapped with rubber similar to the one used to cover climbing shoes used by human climbers in order to increase friction at contact points. The finger is actually the shaft of a force sensor in order to reduce sensing errors (see Section 2.3).

2.2 Actuation

Power, weight, complexity, and backdrivability are the most important features to be considered when we choose and design the actuation system. A backdrivable joint has two main advantages. First, it has less friction, therefore it is more responsive to changes in torque and may facilitate force feedback control. Second, in case the robot falls down from the climbing wall, the actuation system and robot limbs are less likely to be damaged if joints are backdrivable.

We considered several solutions for our actuation system: motor and gearhead drive, motor and pulley drive, hydraulic drive and pneumatic drive.

Hydraulic and pneumatic drives are powerful and have applications in large robot systems with relatively simple kinematic structures. They do not seem suitable for our robot, which is comparatively much smaller with more DOFs. The pump engine needed to provide power for hydraulic or pneumatic actuation would be a major problem.

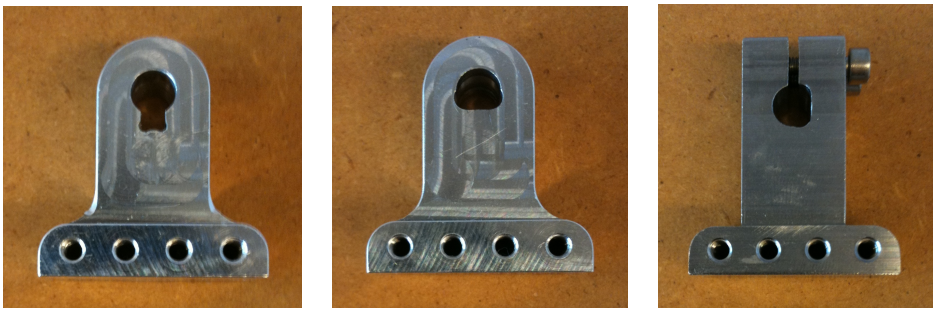
Motor-pulley drive has been used widely in robot design [69]. However, in our design a one-level pulley system could not provide enough torques at the joints, while a two-level pulley system would have been too complicated and would have occupied too much space on the robot.

Motor-gearhead drives can satisfy most of our criteria and, so, we chose this solution for our actuation system. The choice of gearhead requires making tradeoffs between reduction ratio and backdrivability. Our tests showed that in order to be backdrivable, the gearhead could not have a ratio larger than 200:1. Planetary gearhead Maxon GP 32C 166948 has a reduction ratio of 190:1 and is able to connect to a powerful DC brushed motor Maxon RE25 118746. This combination of motor and gearhead set is backdrivable (but with relatively high friction) and provides a

Figure 2.7: Actuator set for each joint. From left to right: connector, gearhead, motor, and encoder.

maximal torque of 7Nm. We chose this combination (Figure 2.7) for the actuation of all joints of Capuchin.

2.2.1 Joint connection design



(a) Connector version I (b) Connector version II (c) Connector version III

Figure 2.8: Three successive versions of the limb-motor connector of the shoulder joint. The third version(c), which minimizes backlash, is the final one implemented on Capuchin.

Backlash is a common problem for robot joints. Precise control cannot be achieved

when backlash is large. So, the mounting of an actuator on the robot and its connection to the corresponding joint is an important issue. For each shoulder joint, we mounted the actuator on the first link of the corresponding limb. We designed a connector, one side of which is fixed on the robot body and the other side on the actuator's shaft. Our first version of the connector was made of aluminum alloy and its design is shown in Figure 2.8(a). But tests revealed a strong backlash between the connector and the shaft. This led us to design a new version of the connector, still made of aluminum alloy, but with a D-shaped hole in which the actuator's shaft is inserted (Figure 2.8(b)). This version of the connector worked well initially, but backlash appeared after some experiments. We observed that the flat contact surface in the hole had been deformed. This led us to design a third version of the connector (Figure 2.8(c)). We decided to use steel, instead of aluminum alloy, although this increases the weight of the connector. A small opening and fastening skews allow us to keep the connection tight. This third version works well on Capuchin. Almost no backlash has been detected even after many experiments. Figure 2.7 shows the connector mounted on the actuator's shaft.

2.3 Control system and sensors

Controller board Figure 2.9 shows the customized 6-channel high-speed controller board that we have selected for our robot. This board is capable of position control, speed control and current control at a frequency of 6kHz. It also has six ports to read the encoders on the motors, as well as A/D ports to read sensors with analog outputs. It can communicate with a PC through serial port and high-speed Ethernet. Since Capuchin has 8 motors in total, two 6-channel controller boards are used. The extra 2 channels on each board could be useful in the future to add other joints (e.g., to lift fingers out of plane to cross over terrain features).

Force sensing hardware We chose a strain gauge force sensor (Figure 2.10(a)) to equip each finger of Capuchin. It is a durable steel force sensor with a small size (28mm in diameter) and light weight (44 grams). One sensor is mounted at the

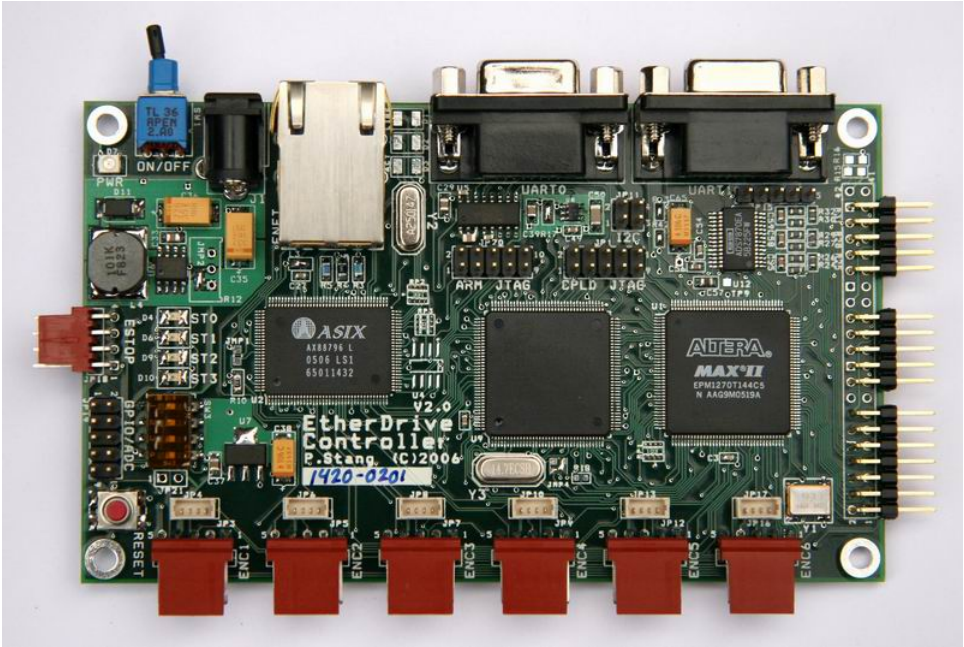
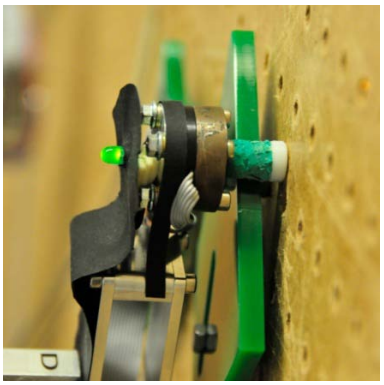


Figure 2.9: Six-port controller board of Capuchin

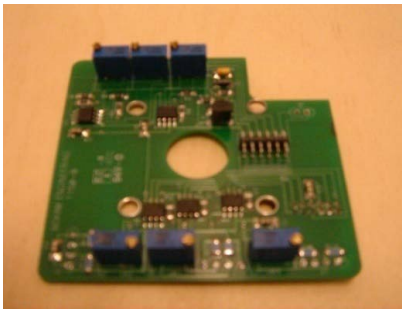
extremity of each limb of Capuchin and, as mentioned before, the shaft of the sensor is used as the limb's finger (Figure 2.10(b)). The sensor measures force components along three orthogonal x, y and z directions up to a maximal net force of 50N. The output analog signals are linear in the measured force and have almost no drift with time or temperature. However, they must be amplified (Figure 2.10(c)) before they are input into the A/D ports of the controller board. Since the amplified signals have high-frequency noise, we built an analog low-pass filter (Figure 2.10(d)) to remove this noise. Our RC filter, which uses a resistor of 10kΩ and capacitor of 0.01μf, has a cutoff frequency of 1.6kHz.

Cameras A fire-wire camera (Figure 2.11(a)) is mounted on the terminal link of each limb, 28cm above the finger, as shown in Figure 2.11(b). As mentioned before, this camera is used by the controller to localize the relative position of a finger and a terrain feature in order to accurately dock the finger at a hold selected by the planner. In the future, it might also be used to detect small slippage of a finger on a feature. Figure 2.11(c) shows an image acquired by a camera above the finger. This image

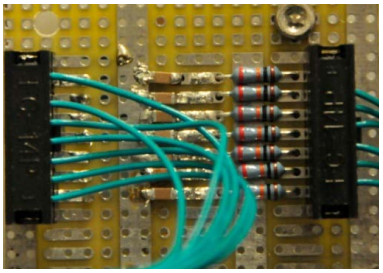
(a) Strain gauge force sensor used in Capuchin.



(b) The shaft of the force sensor is the finger.



(c) Amplifier of the sensor's output signals.

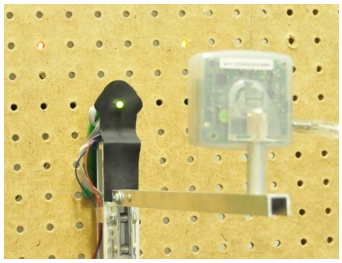


(d) Low-pass filter used to eliminate high-frequency noise.

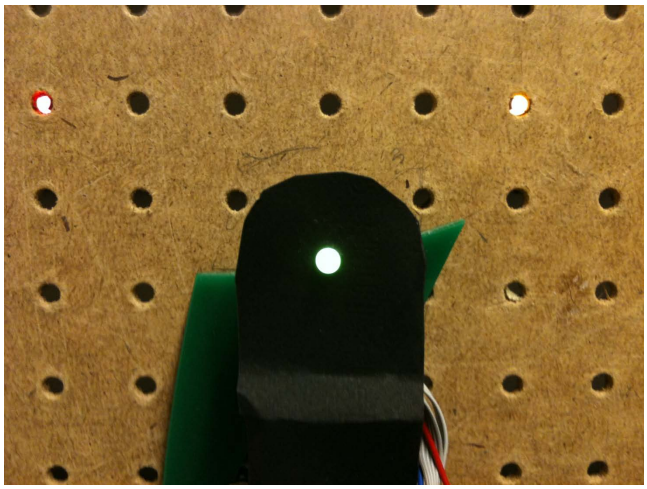
Figure 2.10: Force sensing hardware system implemented on Capuchin.

covers an area of 18cm x 13.5cm of the climbing wall. The fire-wire camera sends image data to the PC directly through fire-wire card on PC. LEDs are mounted on the climbing wall around each feature to simplify the treatment of the images returned by each camera.

(a) Fire-wire camera used by Capuchin.



(b) Mounting of camera above a finger.



(c) A camera returns the image of an 18cm x 13.5cm area of the climbing wall.

Figure 2.11: Camera mounted on each of Capuchin's fingers.

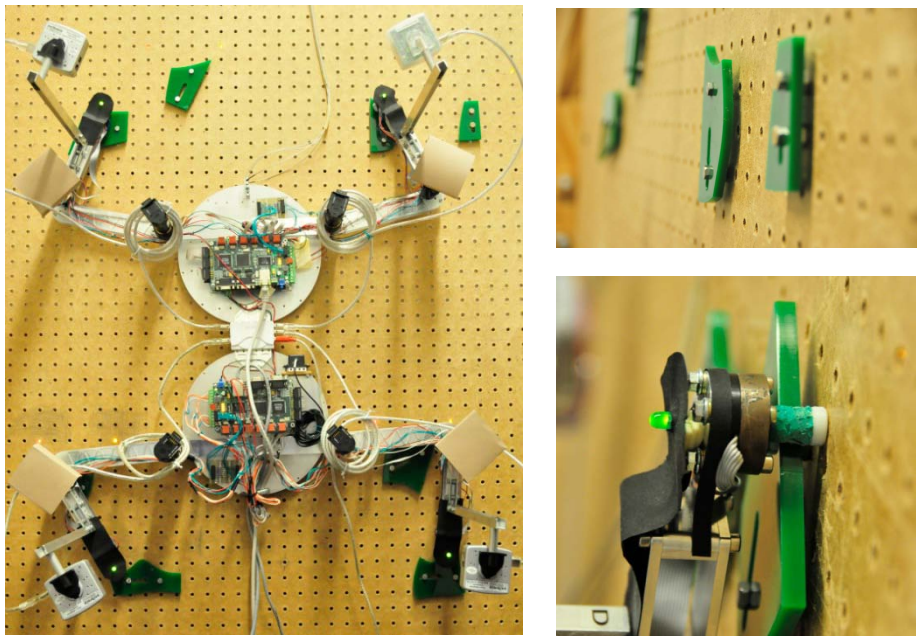


Figure 2.12: Capuchin on a climbing wall: the terrain features are variously shaped plastic plates (green) and each hold is a point on the contour of a feature.

2.4 Climbing wall

The climbing wall used in our experiments is a vertical planar board with artificial terrain features mounted on it (Figure 2.12). Each feature is a relatively small (green) plastic plate with an irregular contour that can easily be mounted anywhere on the board with any orientation. By selecting and distributing such features differently on the climbing wall, terrains with various levels of difficulty can be created to perform climbing experiments (Chapter 5). The entire structure of Capuchin, except its fingertips, is parallel to the wall at a small distance from the features. Therefore, the only possible collisions/contacts between Capuchin and the wall are between the fingers and the features. Collisions must be avoided, except to make contact at selected holds on feature contours.

2.5 Summary of Capuchin characteristics

- Diameter of each of the two circular parts of the body: 250mm
- Distance between the two circular parts of the body: 50mm
- Length of the first link of a limb: 185mm
- Length of the second link of a limb: 172mm
- Weight of each limb: 1.12kg
- Total weight of the robot: 7.5kg
- Maximum torque at a shoulder joint: 7Nm
- Maximum torque at an elbow joint: 7Nm
- Angular range of a shoulder joint: 0 to 225dg
- Angular range of an elbow joint: 10 to 180dg

Chapter 3

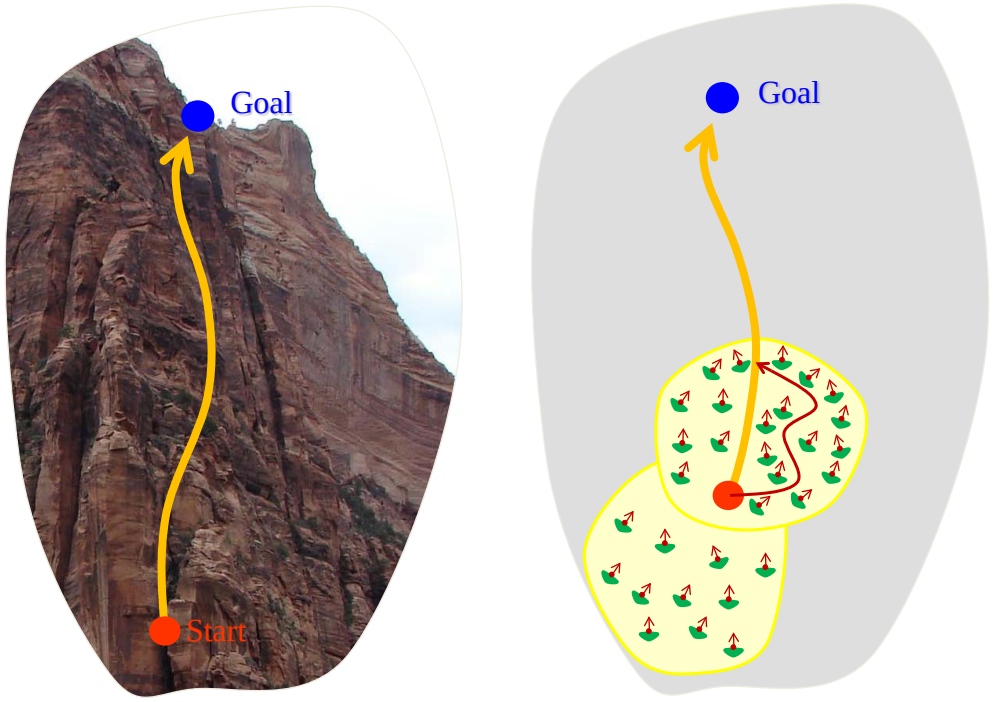
Climbing and planning

3.1 General scenario of free-climbing

To ascend a steep rock wall, a human climber typically starts by looking at the entire wall to acquire a low-resolution map of the major terrain features, such as ledges, cracks, protrusions, and pockets. Based on this approximate map, she plans a coarse navigation path, as illustrated in Figure 3.1(a). This path only gives high-level direction, since detailed information about the small features of the terrain is still lacking. To start the ascent, the climber also acquires a detailed map of the terrain features around her current position. She identifies candidate holds that can be used for contacts in order to move up. She usually plans a few moves to reach an intermediate waypoint along the coarse path and executes them. At the beginning and end of each move either a new contact with the terrain is achieved or a contact is broken. So, each move is performed at a fixed set of contacts, usually 3 or 4. While moving, the climber adjusts both the forces exerted at the contacts and the position of her center of mass to remain in equilibrium. As she progresses, she acquires additional detailed information about the terrain, plans new moves to proceed further, etc... Occasionally, she may revise the coarse navigation path.

So, a general scenario of free-climbing includes the following steps:

1. Global sensing of the terrain



(a) Coarse route planning. (b) Local sensing, planning and control.

Figure 3.1: General scenario of free-climbing: (a) the climber first acquires a low-resolution map of the climbing wall and plans a coarse plan; (b) while climbing she iteratively acquires a detailed map of the local terrain, identifies candidate holds, and plans new moves to proceed further.

2. Planning of coarse navigation path

3. Local sensing and detection of candidate holds

4. Detailed motion planning

5. Motion execution

Steps 3, 4, and 5 are repeated until the climber reaches the top of the wall.

This thesis considers only a subset of this scenario and focuses on steps 4 and 5. We assume that a detailed map of the terrain and the candidate holds is already available to the robot. The methods described in this chapter and the next are used

by Capuchin to plan and execute the moves that allow it to climb from an initial set of contacts to a final set of contacts.

3.2 Climbing moves and stances

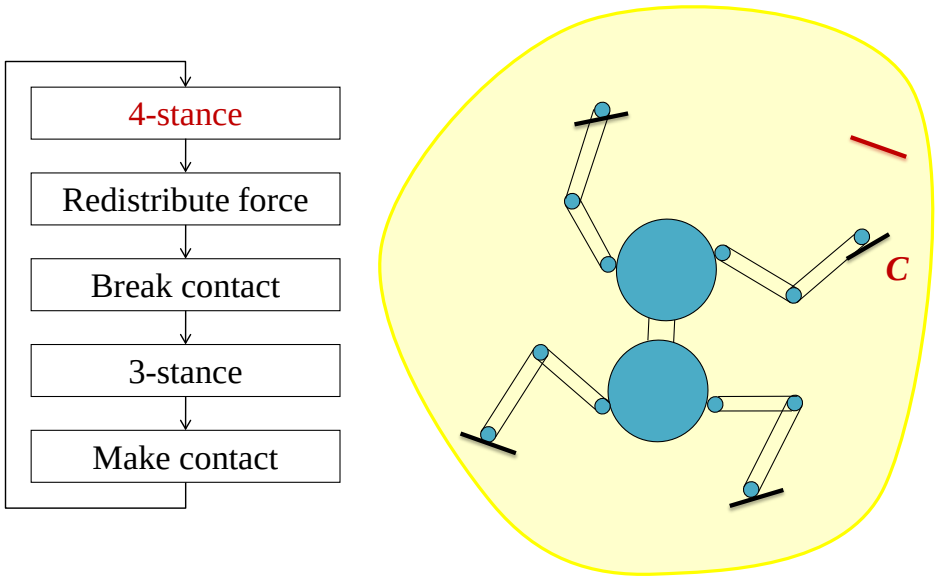


Figure 3.2: Capuchin at a 4-stance. A stance is a fixed set of three or four contacts (here, four).

As mentioned in Chapter 2, Capuchin can only contact the terrain with its fingers. Consider a situation at some time T_0 where each of its four fingers are positioned at contact holds on the terrain, as illustrated in Figure 3.2. Suppose that the short-term goal of the robot is to move its top-right finger, currently at a hold C, to a new hold (shown red). To break the contact, it must first adjust its posture to redistribute the contact forces over the three other fingers, so that the contact force applied on hold C by the top-right finger becomes 0. See Figures 3.3 and 3.4. The force redistribution corresponds to a move M_1 at a fixed set of four contacts. We call this set a *4-stance*. Once the contact force applied on hold C becomes 0, the robot breaks this contact at time T_1 and moves at a *3-stance* (a fixed set of three contacts) to dock its top-right

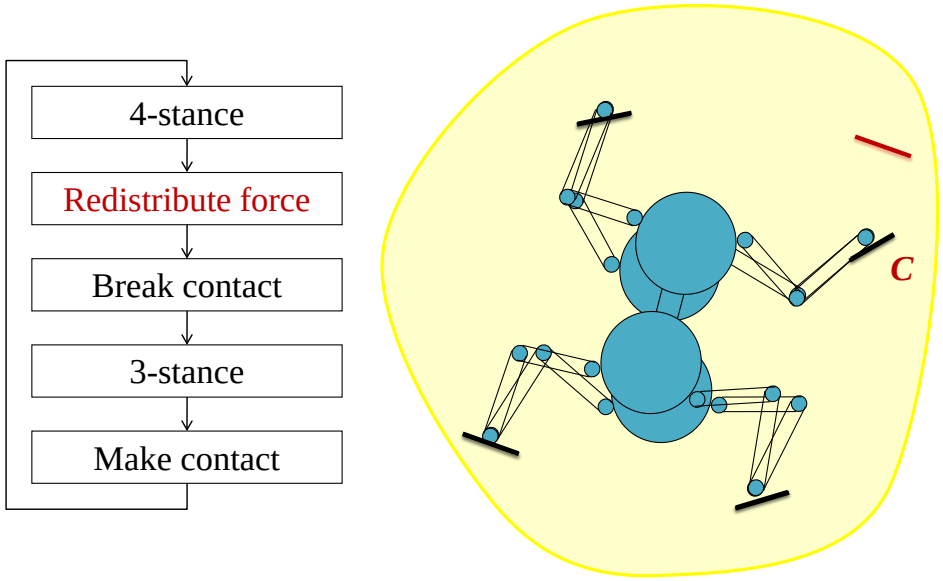


Figure 3.3: To break the contact of the top-right finger at hold C, Capuchin must first adjust its posture to bring the contact force exerted at C to zero.

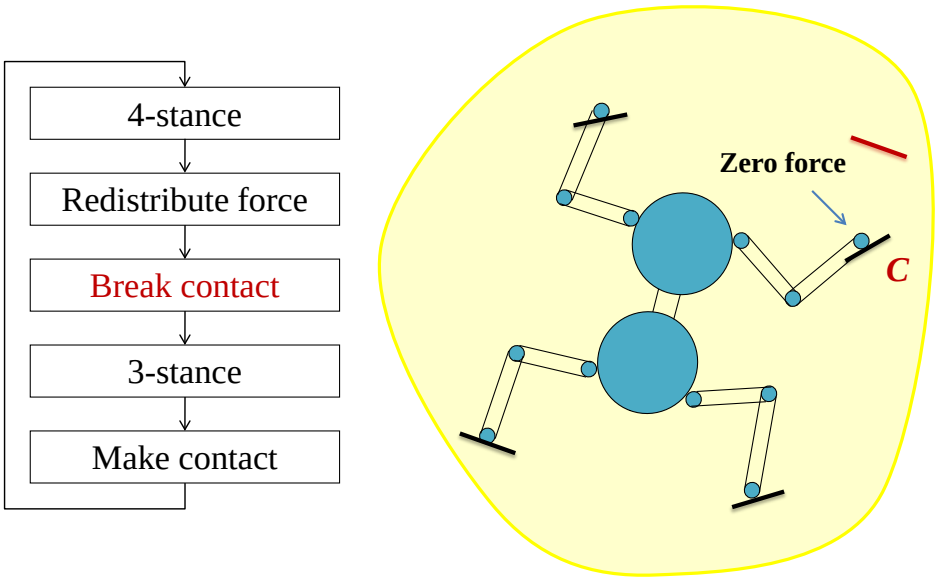


Figure 3.4: When the contact force at hold C is zero, Capuchin relies only on the other three contacts to maintain equilibrium. It can then break the contact at C.

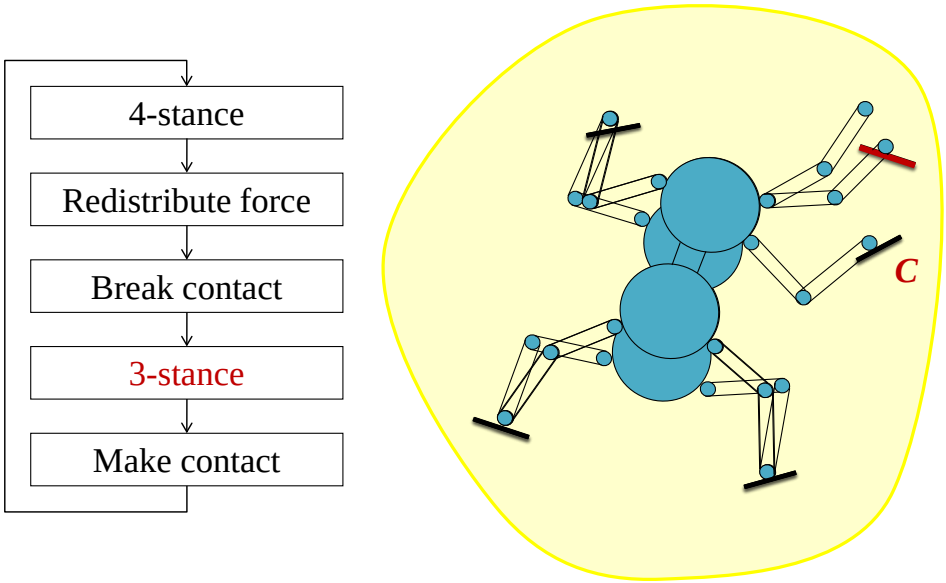


Figure 3.5: While bringing its top-right finger to the new hold (shown red) Capuchin is at a 3-stance. It still needs to adjust its overall posture to remain in equilibrium.

finger at the new hold at a time T_2, as shown in Figures 3.5 and 3.6. This motion corresponds to a move M_2 at a 3-stance.

To continue climbing, the robot will then have to break contact at one of the other three fingers, reach a new hold with the corresponding finger, etc... So overall, the entire climbing motion consists of successive *steps*, each of which changes a set of four contacts into another set of four contacts differing from the former by one single contact. Each step is made of two successive *moves*, the first at a 4-stance to bring a contact force to zero, the second at a 3-stance to bring a finger to a new hold. The relation between motion steps and moves is depicted in the diagram of Figure 3.7.

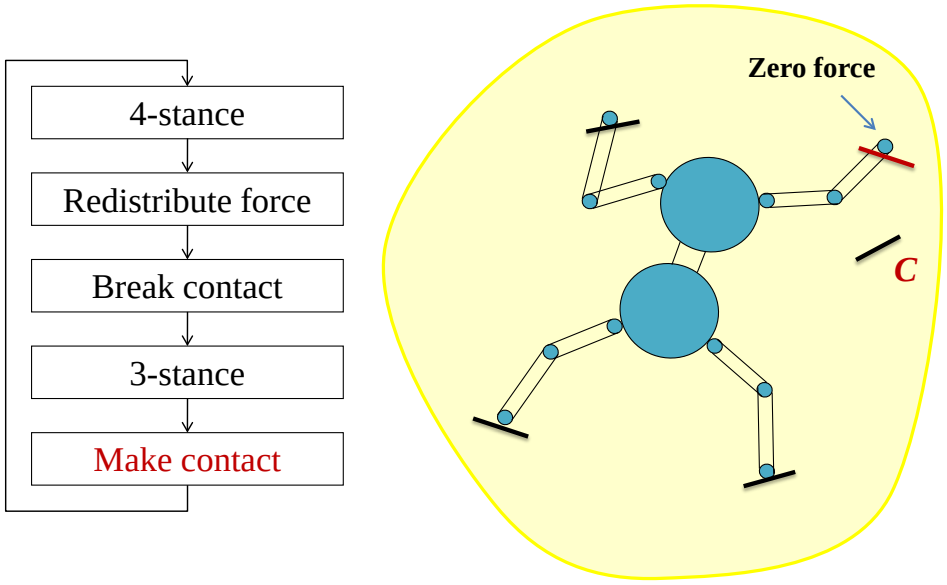


Figure 3.6: At the instant when the top-right finger makes contact with the new hold, the exerted force at this contact is zero. During the next move, the robot will adjust its posture to exert a non-zero force at this contact in order to bring to zero the force at another contact.

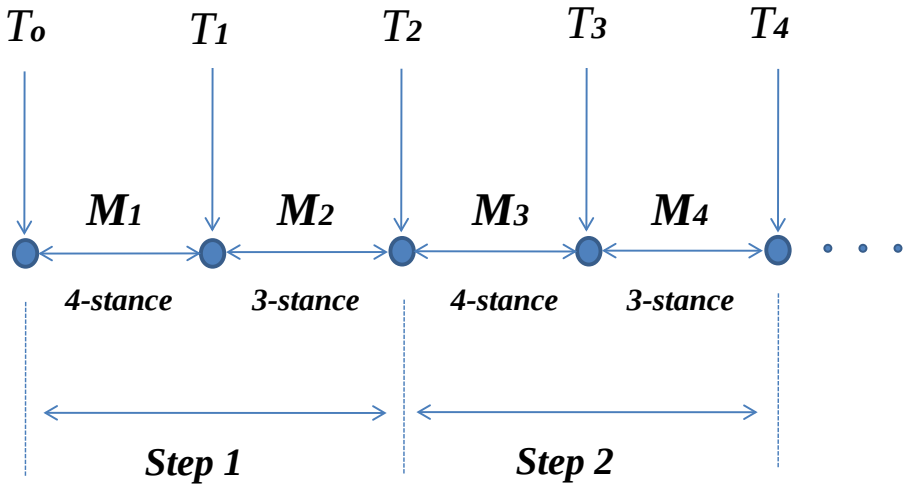


Figure 3.7: Climbing motion diagram: steps, moves, and stances.

3.3 Static equilibrium constraints

As mentioned in Chapter 1, we require that the robot remains in quasi-static equilibrium throughout the climbing motion. There are two main reasons for this choice:

- Quasi-static motion is sufficient to ascend challenging climbing walls.
- Dynamic motions make the planning and control much more complicated and climbing much less reliable.

In fact, good human climbers rarely perform fast dynamic motions. During such motions it is difficult to keep other contacts fixed and thus there is a greater risk of slipping and loosing equilibrium. Nevertheless, exploiting momentum is useful in some circumstances, for example to reach distant holds. Allowing such motions will be an interesting topic for future research. It will probably require a more elaborate contact model than the point contact model used in our work, e.g., a rolling contact model [11].

In order to achieve static equilibrium, Capuchin must apply adequate forces at the holds. At each configuration q of Capuchin (a set of 11 parameters defining the position and orientation of its bodies and the positions of its fingers in a Cartesian coordinate system), we regard the robot as a static rigid object in contact with a rigid environment and acted upon by gravity. The reaction forces at the contact holds must then satisfy the conditions given in Figure 3.8. In these conditions, $p_1, ..., p_4$ are the contact points represented by their coordinate vectors. Under the Coulomb model of friction the set of reaction forces f_k that can be applied on the robot at each hold p_k forms a friction cone FC_k. This cone is determined by the normal vector of the contact surface at p_k and the coefficient of friction μ (assumed for simplification to be the same at all holds). The angle of the cone is $\theta = 2tan^{-1}(\mu)$. In the second condition, $CM(q)$ denotes the coordinate vector of the center of mass the robot at configuration q. The first and second conditions respectively requires that the reaction forces at the holds and the gravitational force mg sum up to zero and that the torques around the origin of the Cartesian coordinate frame caused by reaction forces and the gravitational force sum up to zero. The third condition requires that all reaction forces be within their respective friction cones. In Figure 3.8, CM is positioned at the mid-point between the lower and upper bodies. In fact, CM varies around this position depending on the configuration of the limbs, hence the term $CM(q)$ in the second condition. Due to the relatively small weight of the limbs, the variations of

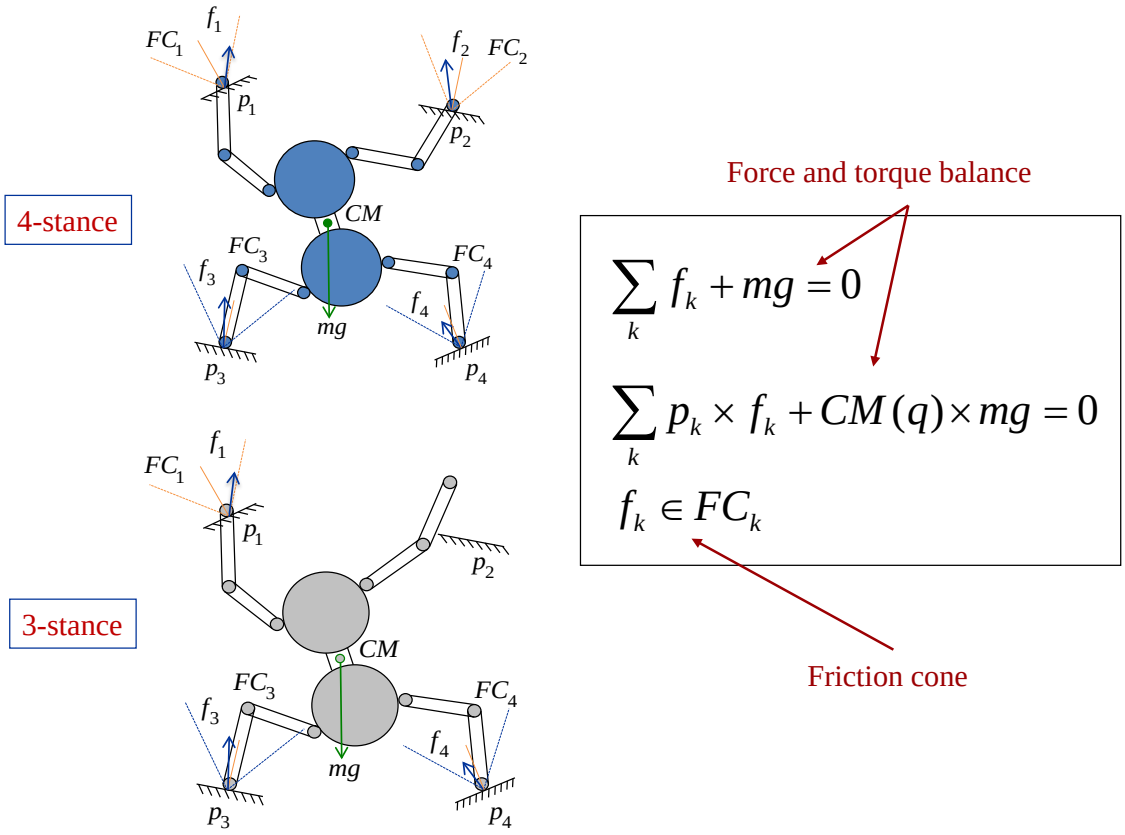


Figure 3.8: Static equilibrium conditions. They express that the reaction forces at the 3 or 4 holds and the gravitational force result in zero total force and torque on the robot. The reaction forces are also constrained to lie within the Coulomb friction cones at the holds.

CM around the robot's center are rather small.

One can remark that for a given μ the existence of a set of reaction forces f_k that meet the equilibrium conditions depends only on the positions and orientations (normals) of the holds. A stance is not feasible if there is no set of reaction forces that meet all these conditions. No feasible climbing motion can possibly include a move at such a stance.

One can also remark that if a set of reaction forces achieves equilibrium for one position of CM, the same set of forces also achieves equilibrium for any other position of CM along the same vertical line. The projection of all the valid positions of CM

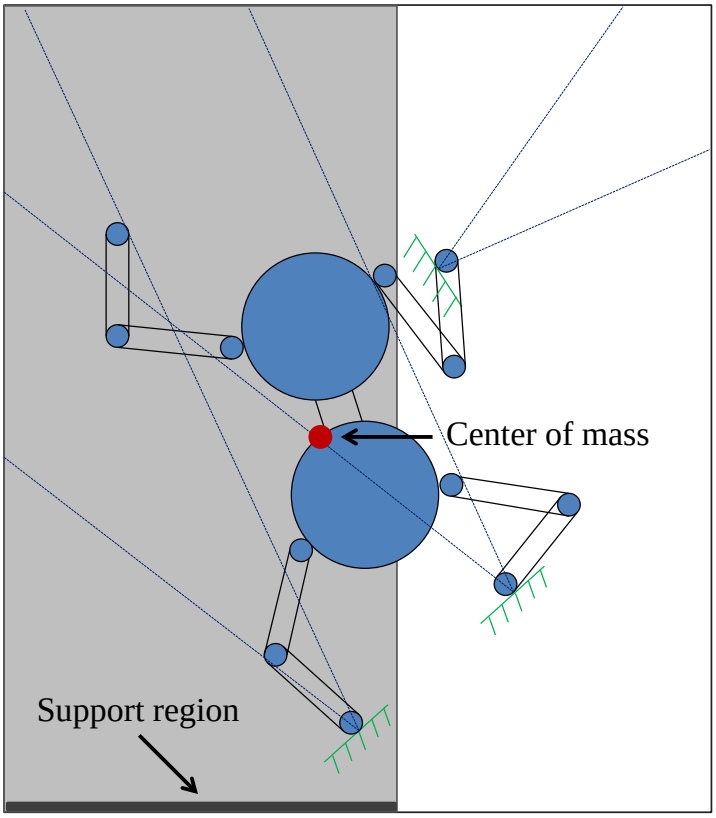


Figure 3.9: On uneven terrain, the support region is not guaranteed to lie under the holds.

into a horizontal plane is the support region. Therefore, for a given μ, the support region is entirely determined by the locations and orientations of the holds. In our work we assume that the planar climbing wall is vertical. Hence, the support region degenerates to a horizontal line segment (possibly a half-line). For a given stance, one can pre-compute this segment. Then, the existence of reaction forces satisfying the equilibrium conditions at a configuration q of the robot can be tested by simply verifying that $CM(q)$ projects vertically into this segment. As noticed in [15], this segment is not guaranteed to lie under the holds, as illustrated in Figure 3.9. For a truly three-dimensional robot like Capuchin-3D (Section 2.1.1), one can approximate the support region at a given stance by a convex polygon by first approximating the friction cones by cones with polygonal cross-sections [15]. It is then easy to test if

$CM(q)$ projects into this polygon.

3.4 Motion planning

The motion planner used by Capuchin is the planner developed by Hauser [26]. The planner is quite general, but a few of its components must be adapted to the kinematics of the robot. The adaptation to Capuchin was done by Hauser. Since we have had no direct contribution to this work, we only give a high-level presentation of the planner. We refer the reader to [15] and [26] for more information.

The input of the planner consists of the following components:

- a map of the terrain that defines the geometry, positions and orientations of the features,
- a set of possible holds located on the contours of the features,
- the initial 4-stance and the initial configuration of the robot at this stance, and
- the goal 4-stance.

In the current implementation of the system, the map of the terrain is provided by a human user, but it would be relatively easy to extract it automatically from an image of the climbing wall. The set of possible holds is also defined by a human user. Various automatic techniques have been proposed by Bretl [15] and Hauser [26]. One technique is to randomly sample points on the contours of the features and retain them as possible holds with a probability proportional to a measure of their expected utility. For instance, points with a normal pointing upward would have greater utility than points with a normal pointing downward. Points located in concave sections of feature contours would have greater utility than points on convex sections.

The output of the planner consists of a sequence of stances alternating 4- and 3-stances, along with a motion path describing the move to be performed at each stance. Such a path is a sequence of waypoints, where each waypoint is a configuration of the robot defined by a vector of 11 parameters. There are several ways of choosing these parameters. For reasons discussed in Chapter 4, we chose the two coordinates

of the midpoint between the robot's lower and upper bodies, an angle defining the orientation of these bodies, and the 8 coordinates of the fingertips. The planner does not select the robot velocity along a motion path. It only assumes that dynamic effects are negligible. It does not select the forces to be applied by the fingers at the contact holds. It only verifies that a set of forces verifying the static equilibrium conditions exist.

The planner is based on a stance-before-motion approach introduced by Bretl [15]. This approach consists of first selecting a sequence of 4- and 3-stances, such that any two consecutive 4-stances differ by a single hold, and then planning a motion path at each stance, such that each path ends at the initial configuration of the following path. The sequence of stances is computed by searching a stance graph, while each motion path at a given stance is computed using a PRM (Probabilistic RoadMap) planner [32, 36, 61, 63].

The stance graph is defined as follows. Each node is a 4-stance defined by set of 4 holds and an assignment of the 4 fingers of Capuchin to these holds. Since even a relatively small number of holds would result in a huge number of 4-stances, simple tests are performed to quickly eliminate impossible stances. For instance, all four contacts must be within a maximum distance of each other, corresponding to the maximal span of the robot limbs. Two nodes in the stance graph are connected by an edge if and only if they differ by a single hold. So, two 4-stances connected by an edge uniquely determine the 3-stance that separates them.

The stance graph is still usually very large and, so, is not pre-computed. Only the relevant part of the graph is computed while it is searched for a sequence of adjacent 4-stances connecting the initial stance to the goal one. During the search, a tree of stances representing several partial candidate sequences of adjacent stances is constructed. At each step of the search, the algorithm selects a pending stance σ in this tree and installs the adjacent 4-stances into the tree as the children of σ. The search terminates when the goal stance is added to the tree. However, this simple algorithm would often generate sequences containing non-feasible stances where the equilibrium conditions cannot be satisfied. As was observed by Bretl [15], the bottlenecks in any sequence of adjacent stances are the transitions from 4- to

3-stances and from 3- to 4-stances. At each such transition, the robot is maximally constrained kinematically, since it must be in contact with four holds, while relying on only three of them to achieve equilibrium. Between transitions, either it is in contact with only three holds, hence has more remaining DOFs to maintain balance, or it is in contact with four holds, but can use all of them to maintain balance. So, prior to inserting a new 4-stance σ'_4 as a child of a 4-stance σ_4 in the search tree, the planner checks that both the transition between σ_4 and the 3-stance σ_3 between σ_4 and σ'_4, and the transition between σ_3 and σ'_4 are feasible. It performs these checks by computing one conformation in each transition at which the equilibrium conditions can be satisfied. If no such conformation can be computed for one transition, σ'_4 is not inserted as a child of σ_4. Experience shows that a feasible motion often exists between two feasible transitions. The search algorithm also uses various heuristics to guide the search.

Once a sequence of 4-stances between the initial and the goal stance has been found, the planner uses a PRM method to compute the motion path for each successive move required by the sequence (see Figure 3.7). Each move is planned separately, taking into account the kinematic constraints to maintain contact with the holds, the equilibrium constraints, the collision avoidance constraints with the terrain features, the self-collision avoidance constraints between limbs, and the torque limit constraints at the various joints. If the PRM method fails to generate a motion path for some move, the stance graph is searched again for another sequence of stances.

Capuchin's planner takes several minutes or more to compute a climbing motion involving 12 to 18 successive moves. Although this time could probably be reduced significantly by optimizing the code, in our implementation the planner runs offline and is not called back while Capuchin is climbing.

The Capuchin system includes a graphic interface allowing a user to interact with the planner. Various geometric terrain models can be designed by positioning pre-defined features. The motion paths computed by the planner are visualized by animating the robot accordingly. The center of mass, the support region, and the magnitude and direction of contact forces satisfying the equilibrium constraints are also displayed.

Chapter 4

Motion Control

The goal of motion control is to make the robot follow the motion path computed by the planner at the successive stances. These paths were computed assuming accurate positioning of the fingers at the holds. Therefore, precisely docking a finger against a selected hold is a critical precondition for the robot to reliably execute the motion. But precise docking alone does not guarantee that appropriate forces will be exerted at the hold. So, while the controller should try to follow the planned trajectory accurately, it should also make sure that the robot applies appropriate contact forces to maintain static equilibrium. In addition, since the planner also assumes an accurate model of the terrain, it is desirable that the controller be able to modify a planned motion to accomodate small errors in the location of some terrain features.

Figure 4.1 illustrates why precise docking is a critical precondition to eventually achieve static equilibrium. The green region depicts a typical terrain feature used in Capuchin's climbing wall. Let (P, N) be a hold on the curved contour of this feature. Suppose that the motion path computed by the planner requires Capuchin to bring one of its fingers to this hold. Assume further that, instead, due to some control or prior positioning errors, this finger achieves contact at a nearby point P' (with normal N'). Due to the curved contour of the feature, the orientations of N and N', hence of the friction cones at P and P', may differ significantly. As a result, making contact at P' may not make it possible for the robot to achieve static equilibrium, that is, no reaction force at P' may allow the static equilibrium conditions stated in Figure 3.8 to

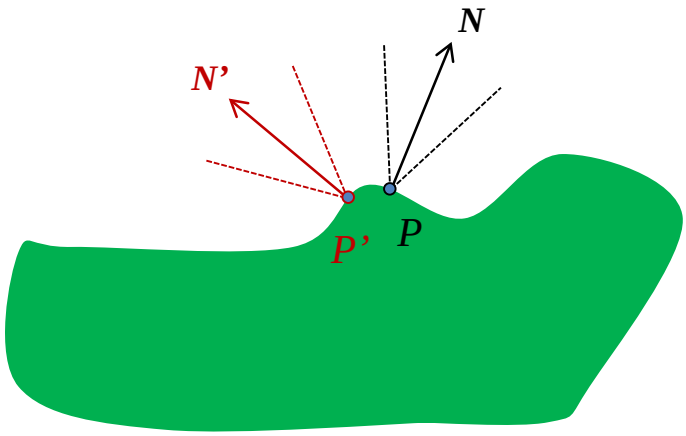


Figure 4.1: Two nearby contact points P and P' on the curved contour of a terrain feature (shown green) may have quite different normals N and N'. Hence, the friction cones at these two points provide different ranges of reaction forces.

be satisfied. Furthermore, even if equilibrium was still achievable, finger positioning errors are likely to be amplified at each climbing step, as is illustrated in Figure 4.2. In this figure the upper-right finger must dock against a new hold. However, the lower-right finger had previously made contact with a terrain feature at a point slightly off the intended hold (error e_1). This error caused a small rotation of the robot around the lower-left finger, which in turn led to a much larger position error e_2 of the upper-right finger. In Capuchin, we achieve precise finger docking using vision feedback as will be described in Section 4.1.2.

But, even with perfectly positioned fingers, Capuchin may still not exert adequate forces at the holds. To illustrate, assume that the robot climbs a chimney-like terrain as shown in Figure 4.3. Here, all the holds are located along two vertical lines. All the four fingers are in contact with holds, exactly as had been computed by the planner. Capuchin is in static equilibrium only if the four reaction forces exactly compensate gravity. However, the maximal vertical component of each reaction force allowed by the friction cone is proportional to its horizontal component. So, if the robot does not exert sufficient forces, the vertical components of the reaction forces will not be sufficient to compensate the gravitational force, and the robot will fall.

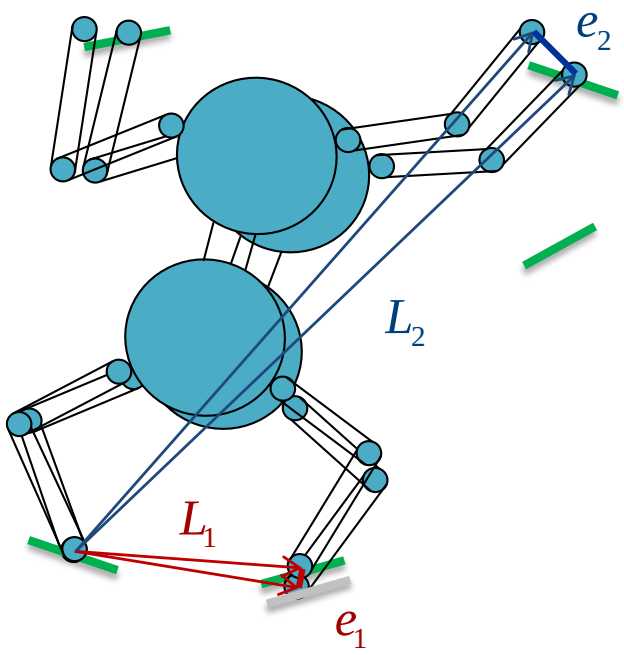


Figure 4.2: Amplification of positioning errors. Here, an initial small error (denoted by e_1) in the position of the lower-right finger later results in a much bigger positioning error (e_2) while docking the upper-right finger. Error e_1 causes a small rotation of the entire robot around the lower-left finger. This rotation results in a larger positioning error e_2 of the upper-right finger because the distance L_2 is much larger than L_1.

To be more specific, let μ be the friction coefficient at all holds and N be the magnitude of the horizontal force exerted by the robot at each hold. The maximal vertical component of the reaction force that each hold can provide is $f = \mu N$. So, in the perfectly symmetric situation of Figure 4.3, we must have: $N \geq mg/4\mu$, where mg denotes the magnitude of the gravitational force. A larger N will make the equilibrium more robust (as the reaction forces will be closer to the normals at the holds). But if N is too large, it may eventually result in damaging the robot. Our controller uses the data returned by the force sensors located at the fingers to adjust the forces exerted by the robot and keep it in static equilibrium. Our control method is novel and will be described in Section 4.2.

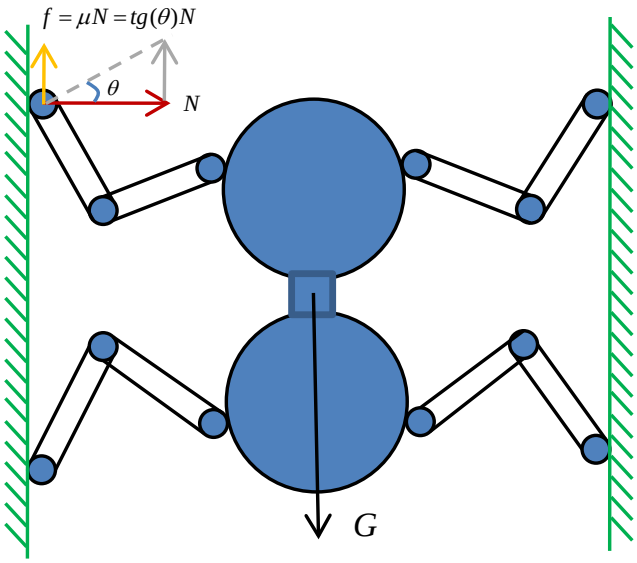


Figure 4.3: Capuchin on a chimney-like terrain. The maximal vertical component of the reaction force at each hold is $f = \mu N$, where N is the magnitude of the horizontal force exerted by the robot at this hold and μ the friction coefficient. In the symmetric situation depicted in this figure, N must be at least $mg/4\mu$ for the robot to be in static equilibrium. If it is smaller, the resultant of the four reaction forces will not be able to fully compensate for the gravitational force, and the robot will fall. If it is larger, the total reaction force will point inside the friction cone, instead of along its boundary; equilibrium will be more robust. But exerting too large forces could eventually result in damaging the robot.

4.1 Position control

4.1.1 Control in Cartesian space

Capuchin's controller should try to follow the motion path computed by the planner as closely as possible, but it should also be able to make corrections in order to compensate for reasonably small violations of the assumptions made by the planner. Since precisely docking the fingers at the selected holds and exerting adequate forces at these holds are both critical, a description of the motion trajectories by means of waypoints of the robot's fingers and body in Cartesian space makes it easier for the controller to use vision and force feedback. This is why each waypoint generated by

the planner (see Section 3.4) is defined by 11 parameters: the two coordinates of the robot's center in the plane of the climbing wall, an angle defining the orientation of its body, and the two coordinates of each finger.

A linear interpolator generates additional trajectory points between the waypoints given by the planner. By selecting the servo rate (up to some maximal rate) at which these points are targeted and the spacing between them, we can approximately tune the velocity of the robot. These two parameters have been selected empirically by trials-and-errors to obtain motions that are smooth and reliable, and nevertheless as fast as possible. In the current implementation the frequency of the control loop is set to 300Hz and the spacing between two consecutive trajectory points is such that the fingers and robot center move by less than 0.1mm, each, and the robot's body rotates by less than 0.02 degrees. Our experiments show that with this parameter setting the robot is reasonably fast, though slower than human climbers. It is much faster than Lemur.

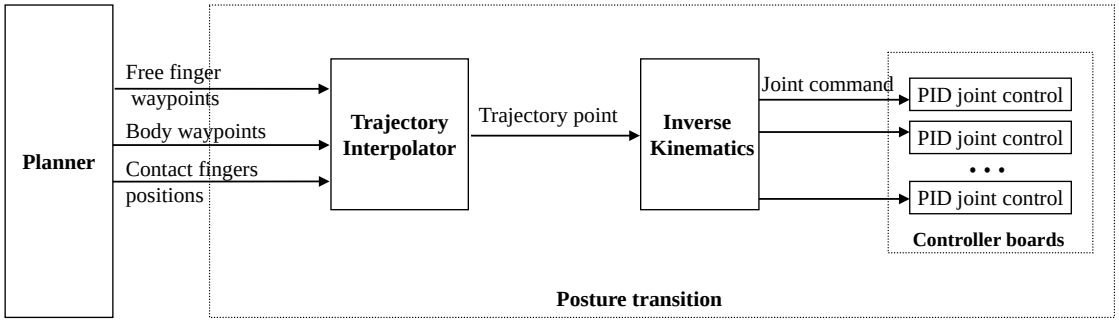


Figure 4.4: Position control diagram.

The only DOFs of Capuchin that are actuated are the eight joints in the limbs. At each control cycle, the joint angles to be achieved at the next trajectory point are computed using simple inverse kinematics from the position/orientation of the robot's body and the positions of the fingers that define that point. Each of these joint angles is fed into the corresponding PID controller of the two controller boards mounted on the robot (Section 2.3). Figure 4.4 shows the diagram of position control.

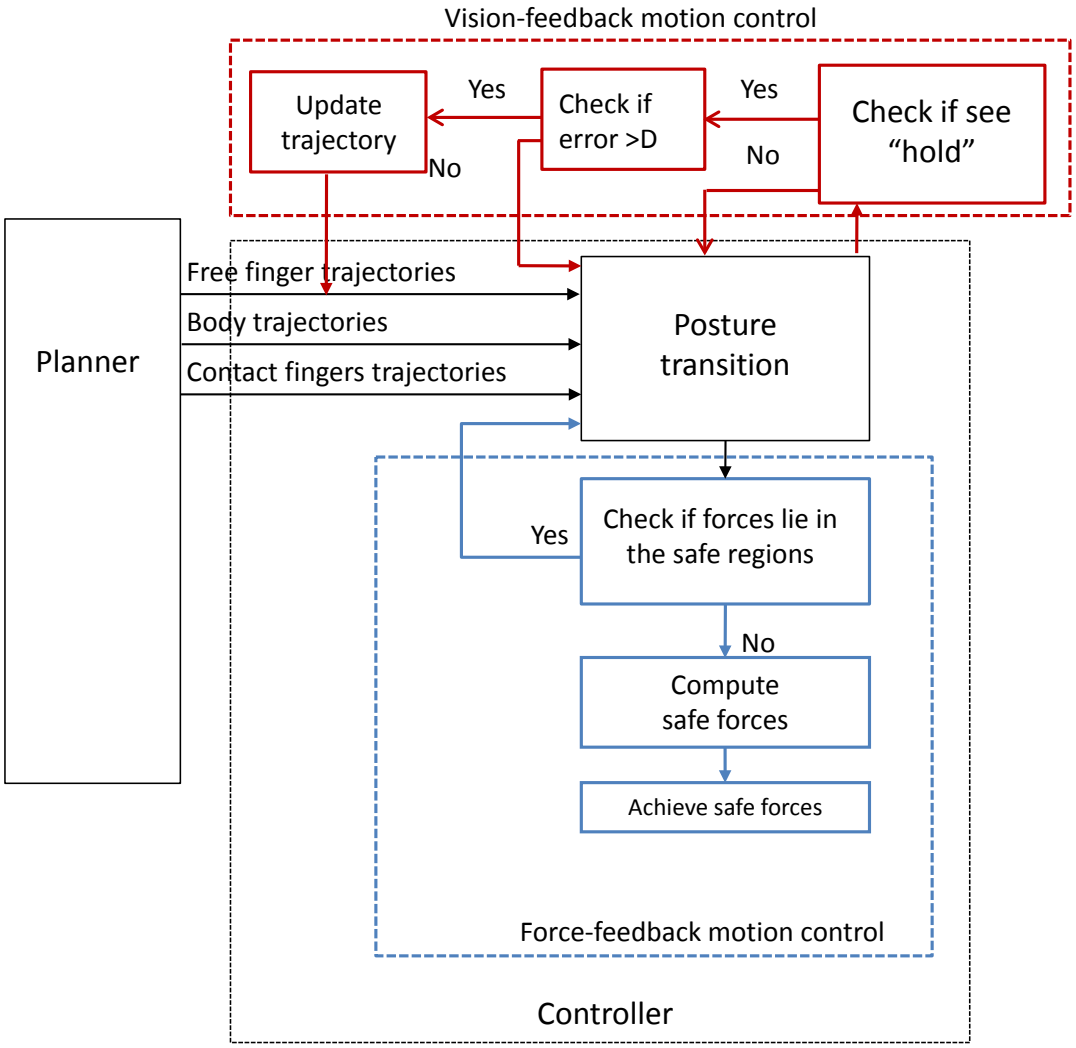


Figure 4.5: Overall motion control diagram of Capuchin with sensor feedback: the vision-feedback part for finger docking is shown in red and the force-feedback part to keep the robot in equilibrium is shown in blue. The box labeled "Posture transition" contains to the position control diagram shown in Figure 4.4.

4.1.2 Vision-feedback finger docking

Capuchin uses vision feedback to control the docking motions of its fingers, in order to precisely position them at the selected holds. Figure 4.6 shows the full controller diagram. The part corresponding to vision-feedback control is shown in red, while the

part corresponding to force-feedback control (Section 4.2) is shown in blue. The "Posture transition" box (in black) contains the position control diagram of Figure 4.4.

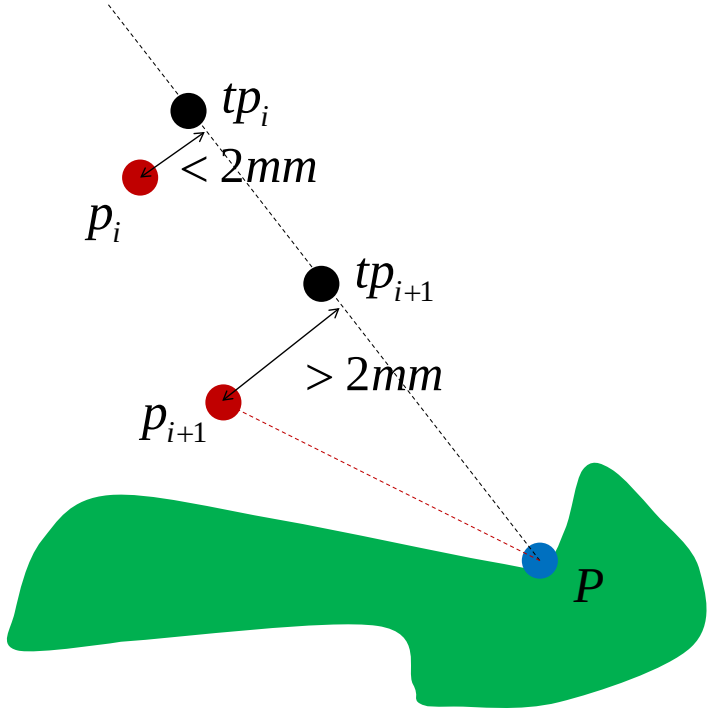


Figure 4.6: Illustration for the vision feedback control method for accurate finger docking (see main text).

When Capuchin performs a 3-stance move to bring a finger to a new hold, position control alone ("Posture transition" box) is used until the targeted hold enters the field of view of the camera mounted above the moving finger. Vision-based control is then activated to perform accurate docking. Vision-based control has been intensively studied in robotics, in particular to control robot arms grasping objects [11, 25, 41, 64]. Our method for finger docking is quite simple, but works well in practice.

We describe how our method operates using the illustration in Figure 4.6. In this illustration, Capuchin is supposed to follow a trajectory (black dash line) that leads to hold P. A sequence of trajectory points, including tp_i and tp_{i+1}, has been computed along this line. At each control cycle, the image returned by the camera is analyzed to measure the distance between the current position of the finger (red

point p_i at cycle i) and the trajectory that the finger should follow. If this distance is less than a threshold (set to 2mm in our implementation), as is the case for p_i, then no correction is being made. If it is greater than this threshold, as for p_{i+1} at the next cycle, then the trajectory is updated and replaced by a new trajectory (red dash line) connecting the current position of the finger to P. Trajectory points along this new trajectory are computed and fed into the "Posture transition" box. The change of trajectory affects only the moving finger, hence the joint angles in the corresponding limb. The other six joint angles are not affected. The change of trajectory is assumed to be sufficiently small for the overall controller to maintain the robot in equilibrium. This is a reasonable assumption since the weight of a limb is relatively small. Several trajectory corrections are possible during a docking operation, but this very rarely happened in our experiments. However, if the finger eventually deviates too much from the original trajectory or if P is no longer reachable, e.g., because a joint in the limb of the moving finger has reached a mechanical stop, then the controller interrupts the motion. In this case, it should also call back the planner to get a new trajectory, but this option has not been implemented yet in our current system.

4.2 Force control at the holds

Force-based control to maintain the robot in equilibrium is much more difficult than vision-based docking. It is also more critical. Even if Capuchin was imperfectly positioning its fingers on the terrain, it might still be able to remain in equilibrium by applying adequate forces at the contacts. In contrast, even with perfectly positioned fingers, inadequate force control will cause Capuchin to fall often. The diagram of the force-feedback part of Capuchin's controller is shown in blue in Figure 4.5. Before describing our method, let us review some previous work on force control.

4.2.1 A short review of some force control methods

The design of position/force control strategies has attracted a considerable amount of research for more than two decades (e.g., see [31, 45, 56, 70], to only cite a few

references). Below we first present basic methods that have initially been designed for controlling forces at single contacts. Then we review more recent work on multi-contact force control.

Basic methods

Three major approaches have been proposed [54]: impedance control, hybrid position/force control and linear optimal control.

- Impedance control maintains a prescribed relationship between the force exerted by a robot's end-effector and the error on its position. The end-effector should behave as if it was connected to the environment by a spring or a damper [31]. Impedance control can be regarded as a generalization of earlier research on stiffness and damping control. In stiffness control task-space position errors are related to exerted forces through a stiffness matrix [62]. This can be implemented with or without force sensing and these two implementations are called active and passive stiffness control, respectively. Passive stiffness control can be interpreted as PD control in task space directions. Active stiffness control corrects positions proportionally to the inverse stiffness of the measured force. Position control is executed by an inner servo control loop. Damping control [70] is based on a similar idea, but uses the sensed forces to modify the velocity.

- Hybrid position/force control partitions the task space into a position control subspace and a force control subspace [45]. It specifies a desired force in the task-space directions along which the end-effector motion is constrained by contacts with the environment, and a desired velocity in the other directions.

- Linear optimal control is similar to hybrid control in the sense that the desired position and force have to be specified, usually based on natural and artificial constraints [45]. However, here the task space is not partitioned into two subspaces. Instead, the adaptive controller tries to meet the control objectives by minimizing a performance index of the position and force errors. The approach is quite general, as one may select weights in the performance index specifically to achieve hybrid or impedance control.

These approaches have been broadly applied to robots, like robot arms, making one contact with their environments. However, the control of robots making multiple contacts, like our climbing robot, raises even greater challenges.

Multi-contact force control

Convex optimization has been explored for problems involving locomotion of legged robots [46] and dexterous manipulation with multi-finger hands [65]. It has also been used in the first controller of Capuchin [47]. All these problems require controlling mechanisms with redundant DOFs achieving multiple contacts with the environment. Limits on friction and forces/torques are represented as linear constraints.

McGhee and Orin [46] were the first to use mathematical programming to control redundant, over-actuated robot systems, such as legged robots. Schelgl et al. [65] used linear programming to optimize forces applied by a robot hand in order to improve its ability to perform dexterous manipulation tasks. Fujimoto and Kawamura [21] formulated the control of the forces exerted by a bipedal walking robot and the position of the robot's center of mass as a quadratic program. Miller extended this work to control the motions of a free-climbing robot [47]. Her work led to the first motion controller of Capuchin. In this controller, PD control generates desired force on the body center and convex optimization is used to select the torque to be exerted by each joint of the limbs. Miller's method was able to control the climbing motions of Capuchin with some success. However, due to the large friction in the joints, the joint torques could not be controlled precisely, resulting in jerky motions and unreliable overall performance. Moreover, when Miller tested her method on Capuchin, vision was not available for docking fingers precisely. Our research nevertheless benefited greatly from Miller's work.

Sentis and Khatib [66] proposed a multi-level hierarchical control structure to synthesize whole-body behaviors, mostly for humanoid robots. Their idea is to combine multiple behavioral primitives (concerning center of mass, hands, legs, head, ...) in a prioritized fashion. Primitives are categorized into constraints, operational tasks, and postures. In the hierarchical control structure, constraints have the highest priority and should always be guaranteed. Operational tasks should be accomplished without

violating the acting constraints. Postures should be approximated as well as possible using the residual degrees of freedom.

Park and Khatib [53] developed a multi-contact control algorithm that performs force control with the highest priority. Motion control is performed with a lower priority in the null-space of force control. A dynamic control structure is used to control each contact force and motion independently.

Control of internal forces

Internal force control has been used for both multi-finger grasping [34, 39, 68] and multi-arm manipulation systems [76, 78]. The usual goals of internal force control are to minimize internal forces [50] and to apply contact forces that satisfy friction and equilibrium constraints without exceeding specified limits [19, 24].

Williams and Khatib [72] proposed a model, called the virtual linkage, to characterize internal forces and moments for multi-grasp manipulation. The virtual linkage is a physical model designed to represent the object being manipulated, more specifically the internal forces and moments acting at the various contacts.

In a more recent work, Sentis, Park, and Khatib [67] proposed a torque-based method for the control of internal forces and moments produced during multi-contact interactions between multi-limbed robots and their environment based on the virtual linkage model. This method was integrated into the framework for whole-body prioritized control of operational tasks, postures, and internal forces [66]. It was demonstrated on the Asimo humanoid robot.

4.2.2 Continuous direct torque control

When we designed Capuchin, one of our goals was to build a robot with backdrivable joints (see Chapter 2) so that direct joint torque control could be used. But some gearing in the actuation turned out to be necessary for the robot's joints to exert torques large enough to lift the robot. As a consequence of balancing conflicting factors, Capuchin's joints ended up being backdrivable, but with large friction. Experiments by Miller [47] demonstrated that direct torque control could still be realized, but with

performance that are not fully satisfactory. Direct torque control resulted in motions that were not smooth and unable to follow the desired trajectories well for at least two reasons: (1) joint torques needed to achieve desired forces at the contacts could not be estimated and controlled with good accuracy, and (2) due to the complexity of the on-line computation needed to optimize the joint torques the servo rate was not high enough.

The analysis of the results obtained by Miller led us to a different approach. In our approach, in order to achieve desired forces at the contacts, we control the positions of the fingers using the data returned by the force sensors, instead of the joint torques. This can be regarded as a variation of impedance control. Experiments show that contact forces can be achieved with good precision. In addition, we no longer try to achieve optimal contact forces at each cycle. Instead, our controller makes force corrections only when they are needed (lazy control). We first present the lazy control approach below (Section 4.2.3). Next we will describe how force corrections are being made (Section 4.2.4).

4.2.3 Lazy force control

Continuous force control has several drawbacks. It leads to frequent computation and results into a relatively low servo rate. By trying to keep the orientations of the reaction forces as close as possible to the normals at the holds, it often requires the robot to exert greater forces. It also causes more frequent delicate adjustments. On the other hand, Capuchin does not have to exert a specific set of contact forces to maintain static equilibrium. A whole range of contact forces satisfy the equilibrium constraints. Capuchin should only make sure that none of the reaction forces gets too close from the boundary of the friction cone at the corresponding hold. Based on these remarks, we developed a lazy force control algorithm that monitors reaction forces at each cycle of position control and adjusts them only when needed.

The force control part of the Capuchin's controller is shown in blue in Figure 4.5. At each cycle, the contact forces are measured and checked against the corresponding

friction cones. If any force is too close to the boundary of its cone, the controller adjusts joint torques to reach a more adequate set of contact forces (as will be described in Section 4.2.4).

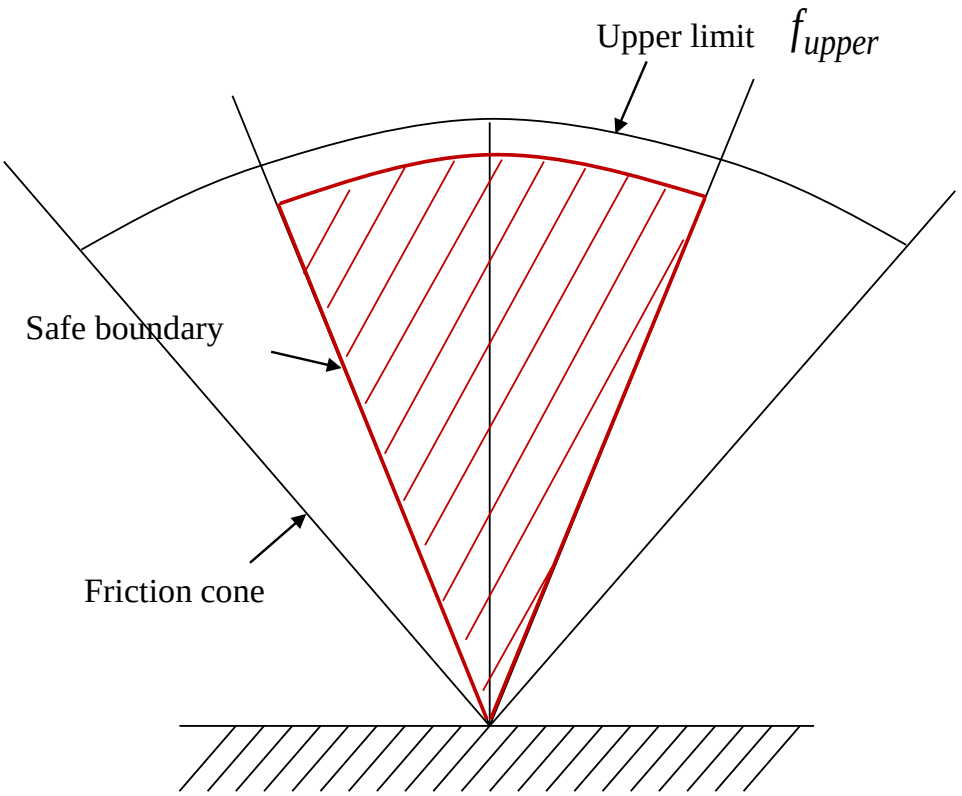


Figure 4.7: Definition of the safe region inside a friction cone.

More precisely, we define a "safe" region in each friction cone, as shown in Figure 4.7. The safe region is obtained by slightly shifting inwards the two sides of the cone, as well as the upper limit on the magnitude of the reaction force that can be measured by a sensor. The underlying idea is that if reaction forces are within their safe regions at one control cycle, then there are enough margins for the robot to move safely toward the next trajectory point during the following control cycle. The safe region is defined in the same way for all friction cones.

Our experiments show that, thanks in good part to the accuracy of vision-based

docking, reaction forces stay in their safe regions at most cycles. Hence, adjustments are not frequently needed. At most cycles, robot control simply follows the diagram shown in Figure 4.4.

4.2.4 Force adjustment

If the reaction force at a hold is measured outside its safe region, the controller redistributes the forces exerted at the holds before commanding the robot to move to the next trajectory point. Our approach to force redistribution consists of two steps: first compute a set of contact forces that, each, lie within their safe regions and, collectively, satisfy the equilibrium constraints; then adjust the joint torques to achieve these forces. We describe these two steps in more detail below.

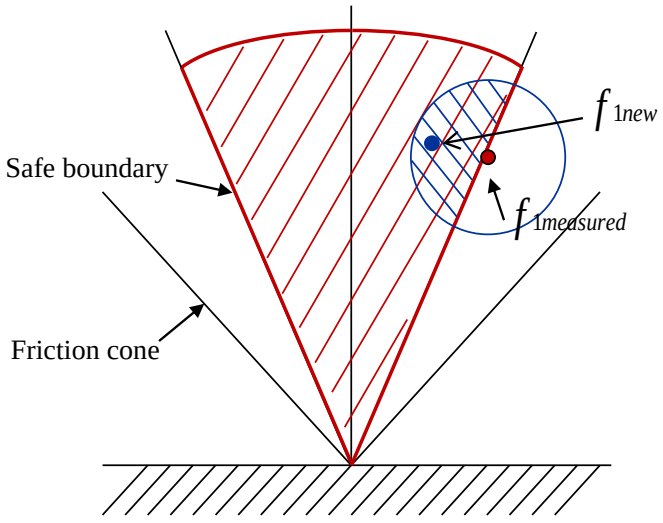


Figure 4.8: Illustration for the computation of a force in its safe region (see text).

Computation of a safe set of reaction forces. One way to compute a safe set of reaction forces would be to solve a quadratic optimization problem aimed at maximizing the sum of the squares of the distances of the forces from the boundaries of their safe regions under the static-equilibrium constraints formulated in Figure 3.8.

Instead, we use a random sampling algorithm that may not generate optimal forces, but can often find a set of valid forces more quickly. At a 4-stance, we select the two reaction forces – let us call them f_1 and f_2 – that lie the further away outside their safe regions. The new values of f_1 and f_2 are obtained by sampling uniformly at random the intersection of a neighborhood disk centered at the measured value and the safe region, as illustrated in Figure 4.8. If only one force, f_1, lies outside its safe region, we compute the new value of f_1 in the same manner; we select f_2 to be any of the other forces and we keep its value unchanged. We then compute the other two forces from the new values of f_1 and f_2 so that the first two equilibrium constraints are satisfied. At a 3-stance, we only select one force, f_1, that lies outside its safe region; we sample a new value of f_1 and we compute the other two forces in the same way as above. In both cases, if the two forces calculated from the equilibrium conditions lie in their safe regions, then a new set of safe contact forces has been generated successfully. Otherwise, the whole process is repeated until a set of safe forces is generated. The number of iterations needed is usually small, typically less than 10.

Achieving the targeted safe forces. Capuchin uses position control with force feedback to achieve the set of safe contact forces computed as described above. Each finger in contact with a hold is "pushed" in the opposite direction of the desired change in force, as illustrated in Figure 4.9. However, since the contact forces remain within their friction cones, there is no actual motion of the fingers. The only significant changes are in the joint torques. Joint angles in the limbs may also undergo tiny changes due to the deformation of the rubber covering the fingers and the flexibility of the overall mechanical structure. Actually, such flexibility is needed to adjust forces smoothly. The correction is performed at 300Hz. At each cycle, the controller sends a position command to push each finger by an increment proportional to the difference between the current reaction force measured by the sensor and the targeted safe force. This approach can be regarded as a simple variant of impedance control, where dynamics is not considered.

Since all contact forces are coupled, the change of any contact force has an impact on the other contact forces and therefore on the equilibrium of the robot. However,

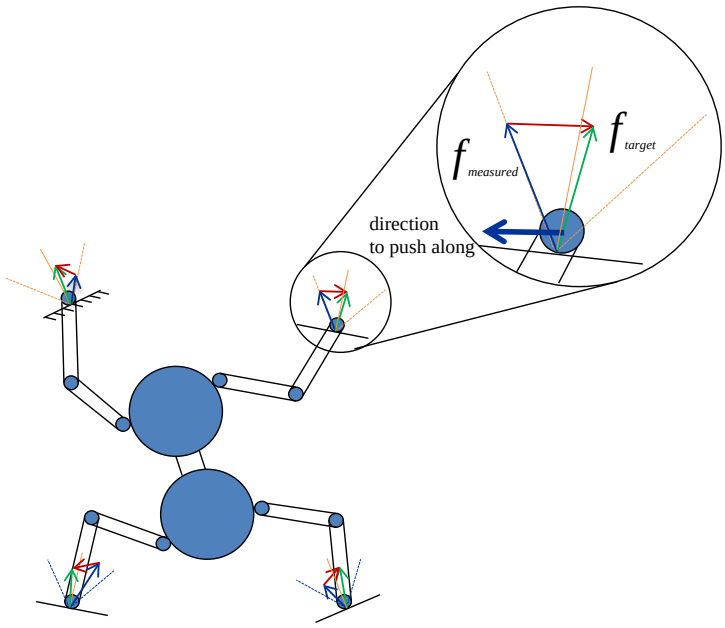


Figure 4.9: Force adjustments in the fingers. Each finger in contact with a hold is "pushed" in the opposite direction of the desired change between the measured force $f_{measured}$ and the new targeted force f_{target}. However, there is no actual motion of the fingers and the only significant changes are in the joint torques.

our experiments show that the computed set of safe contact forces can be achieved reliably by simultaneously pushing all fingers toward achieving their respective contact forces.

In our experiments, a force adjustment is usually achieved in less than 100 cycles at 300 Hz, that is, in less than 0.3 seconds. Overall, force adjustments (including the computation of the safe forces) account for less than 10% of the time needed to execute a climbing motion plan.

Special cases. In our Capuchin system, the measured orientation of a reaction force of small amplitude is often noisy. Relying on such a measurement could lead Capuchin to often perform unnecessary adjustments. In our implementation, we deal with this issue as follows. If a reaction force smaller than 2N is detected to lie outside the cone of its safe region, this violation is ignored. If the force is comprised between

2 and 20N, then its orientation should be outside the cone of the safe region for 10 consecutive cycles for the controller to perform an adjustment. If the force is greater than 20N, an adjustment is performed as soon as a violation is observed.

In general, as mentioned above, the fingers do not move during a force adjustment. However, there are two potential exceptions: (1) a finger may slide against a feature and (2) a finger may not have achieved contact at a hold. In our experiments, we observed this second case when we turned off vision-based docking while one feature was not positioned as in the terrain map given to the planner. Whenever the controller detects an unexpected motion in the limb joints during a force adjustment, it terminates the execution of the plan.

4.3 Drift

Corrections in the joint angles, mainly during finger docking (especially when there are small errors in hold locations in the terrain model) and to a lesser extent during force adjustments, may lead the robot's body to progressively drift away from the planned trajectory. This is due to the fact that all corrections are made by adjusting joint angles, while assuming that the body has a correct position and orientation. This drift may eventually result in situations where it is no longer possible to achieve equilibrium.

No significant drift was observed during our experiments. This is due to the fact that in each of our experiments the number of moves was rather small. In the general scenario outlined in Section 3.1, the climb is divided into shorter climbs (similar in length to those performed in our experiments) between waypoints selected along a coarse navigation path. The trajectory planned for each of these shorter climbs should be computed relative to the local model of the terrain. So, no significant drift should happen relative to each local trajectory.

However, on a high climbing wall the successive trajectories could eventually deviate from the initial coarse navigation path. To prevent such a deviation, either the robot would have to localize itself relative to landmarks identified before starting the ascent of a climbing wall, or would have to be equipped with sensors providing global

localization, such as remote cameras and GPS-like sensors.

Chapter 5

Experiments and results

The purpose of our experiments has been to test the general ability of Capuchin to reliably climb planar vertical walls with irregularly placed features and holds. A more specific aim was to validate its motion control algorithms, especially force-based and vision-based control. In each experiment, a map of the terrain was given to the planner, including the geometric shapes and locations of the features and the positions of the holds along the contours of the features. The planner was run offline and the computed trajectory executed. The planner was not called back during execution.

In this chapter we report on four representative experiments (called I, II, III, and IV). The first three experiments test force-based control, both the lazy control approach and the force adjustment algorithm, with accurately modeled terrains. In each case, we compare the implemented force-based controller with an open-loop position controller (with no force and vision feedback). We analyze in detail the variations of the measured reaction forces during climbing as they provide interesting insights into the behavior of the controllers. While the climbing wall in experiment I is simple enough for both open-loop position control and force-based control to work, the climbing terrains in experiments II and III are more challenging and only force-based control leads Capuchin to climb them successfully. In experiment IV we test the ability of the vision-based controller to handle small errors in the terrain model given to the planner.

In the description of the four experiments we refer to each finger by an index (1,

2, 3, and 4) and each hold by an index (1, 2, ..., and 10) as shown in Figure 5.1. The position and orientation of the features vary by sufficiently small amounts across experiments for this indexing to apply to all of them. In all four experiments we placed a single hold on each feature, so each feature has the same index as the hold placed on it. The vertical component of a reaction force at hold i is denoted by F_{iy}.

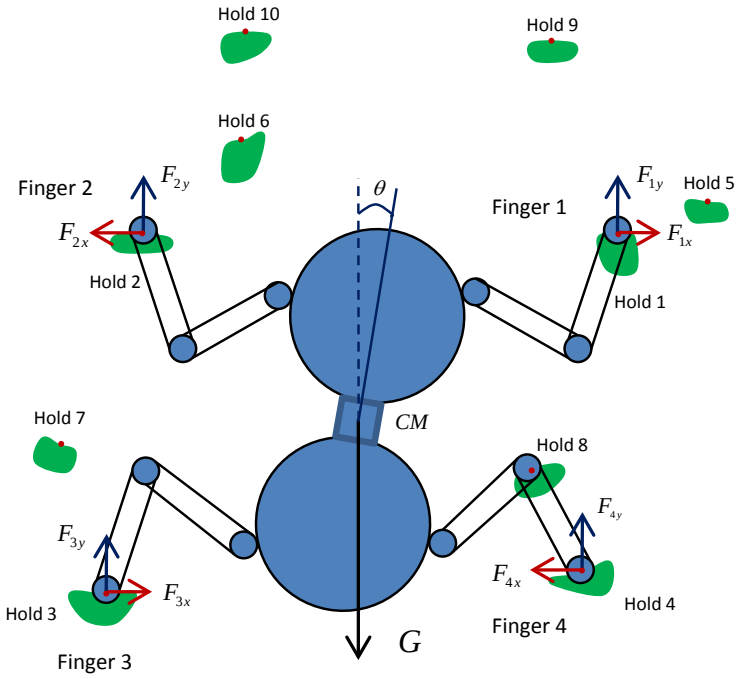


Figure 5.1: Indexes of fingers, holds, and contact forces.

5.1 Experiment I: Climbing an easy terrain

This experiment (see Figure 5.2) demonstrates that Capuchin can climb some easy terrains both with open-loop position control and lazy force control. In this experiment, the robot system was given an accurate geometric model of the terrain and a precise description of the robot's initial configuration. All ten holds are located on portions of feature contours that are almost horizontal and, so, have near vertical upward normals. This makes it quite easy for Capuchin to achieve static balance at

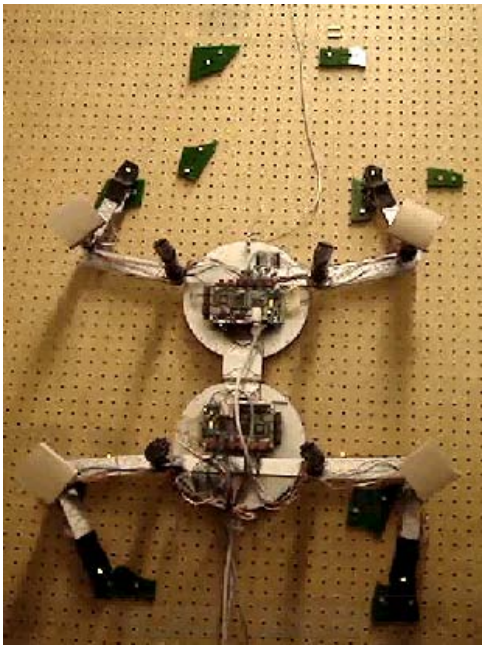


Figure 5.2: Capuchin at its initial configuration on the terrain of Experiment I.

any time. Finger positioning errors during climbing are negligible, so vision-based control was not activated.

Figures 5.3 and 5.4 show snapshots of Capuchin successfully climbing up this terrain with open-loop position control and lazy force control, respectively. There is almost no visible difference between the two sequences, although as we will see that lazy force control made several force adjustments. It is quite remarkable that Capuchin could successfully execute the six-step (12 moves) climbing motion with only open-loop position control. In fact, we initially did several such open-loop climbing experiments on simple terrains and we were surprised that Capuchin could reach its goal in a significant number of them. This observation gave us the initial inspiration to develop our lazy force control approach, which is essentially an open-loop position control performing force adjustments only when they seem needed.

The plots in Figures 5.5 and 5.6 represent the variations over time (horizontal axis) of the measured vertical component F_{1y} of the reaction force on finger 1 (vertical axis) for open-loop position control and lazy force control, respectively. In Figure 5.6, the intervals marked in red along the time axis correspond to the control cycles

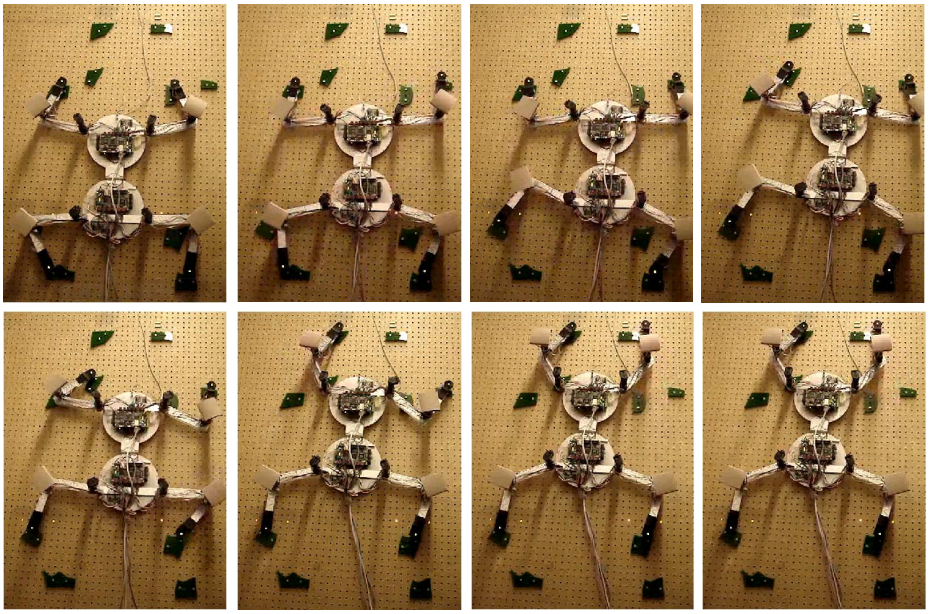


Figure 5.3: Climbing motion for Experiment I with open-loop position control.

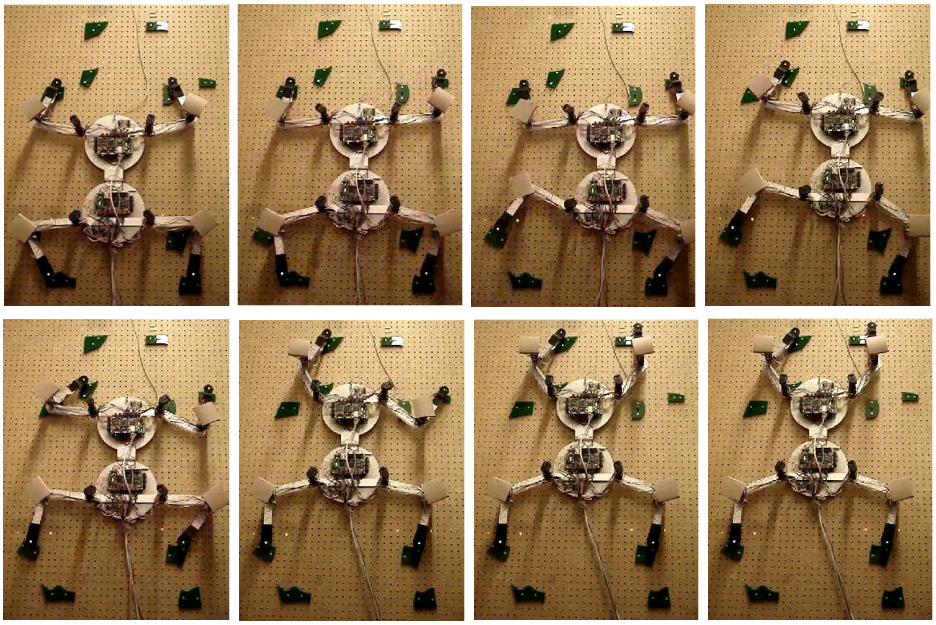


Figure 5.4: Climbing motion for experiment I with lazy force control.

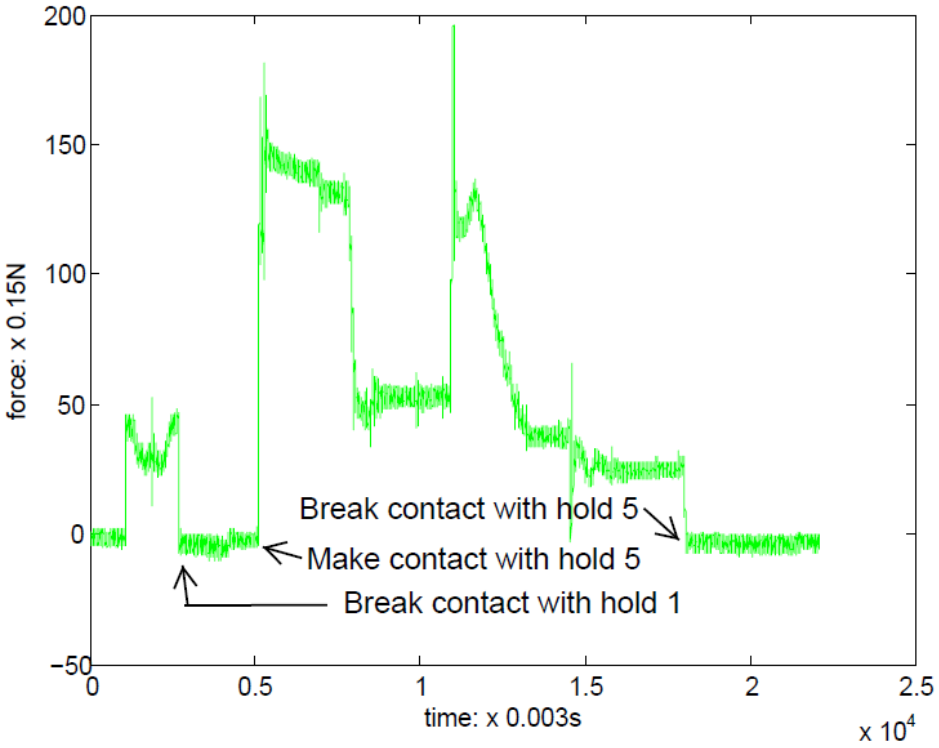


Figure 5.5: Vertical component F_{1y} of the reaction force on finger 1 with open-loop position control.

during which the controller performs force adjustments at the finger. There were nine adjustments accounting for 5.6% of the total execution time. For easier comparison, Figure 5.7 plots the vertical components of all the contact forces for both open-loop and lazy control and all four fingers. Note that the vertical axes in the four plots have different scales. Figure 5.7 shows that, overall, the reaction forces with lazy force control are better distributed over the four fingers than with open-loop control. Specifically, with open-loop control, reaction forces on fingers 3 and 4 are much larger than on fingers 1 and 2 most of the time.

Figure 5.8 provides additional insight into the two control processes. It plots the reaction forces measured on the four fingers at all control cycles between the instant when finger 4 breaks contact at hold 4 and the instant when finger 2 makes contact at hold 10. This interval of time lies between 1.2 and 1.9 on the time axis in Figure 5.7.

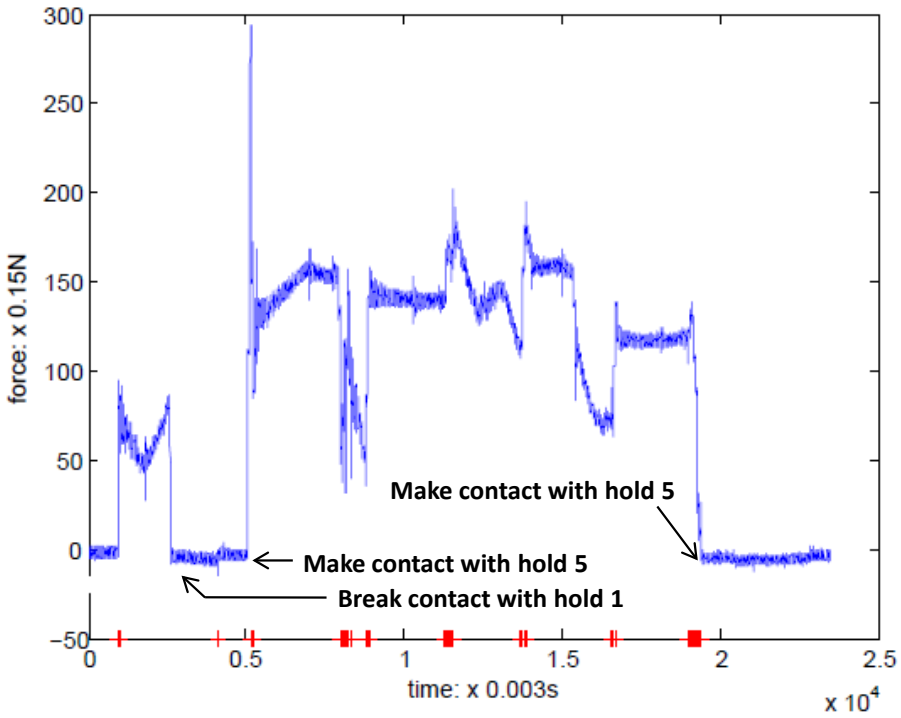
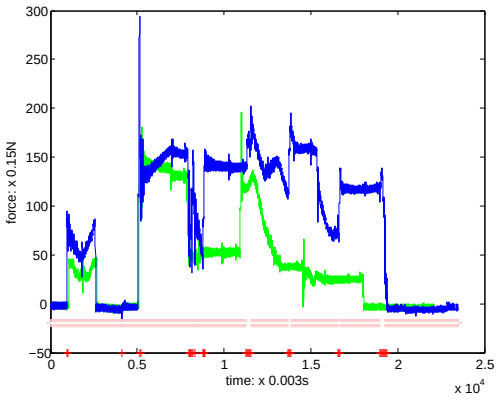
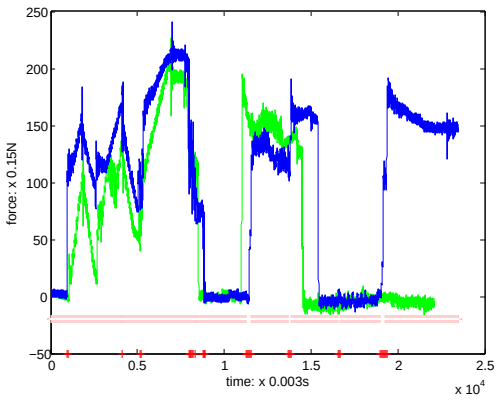


Figure 5.6: Vertical component F_{1y} of the measured reaction force on finger 1 with lazy force control. The short intervals marked in red along the time axis correspond to the control cycles during which the controller performs force adjustments. They represent 5.6% of the total execution time.

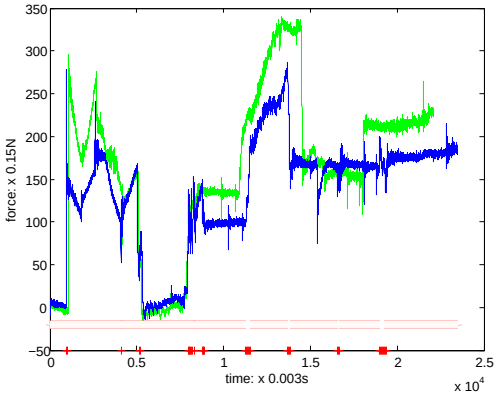
During this interval, each finger makes contact at a single hold. The plots in green (left column) correspond to open loop control and the plots in blue (right column) to lazy force control. Each plot shows all the measurements relative to the safe regions (red contour) of the friction cones (blue contour) at the corresponding hold. Each safe region has an angular range of 74 degrees and an upper limit of 45N (300 $\times$ 0.15N) on the force magnitude. We observe that with open-loop control a significant portion of the contact forces lie outside their safe regions: orientation violations for the force on finger 2 (Figure 5.8(c)) and magnitude violations for the forces on fingers 3 and 4 (Figures 5.8(e) and 5.8(g)). With lazy force control, four force adjustments were performed, two due to orientation violations (Figures 5.8(b) and 5.8(d)) and two due to magnitude violations (Figures 5.8(f) and 5.8(h)). As a result, almost all forces lie



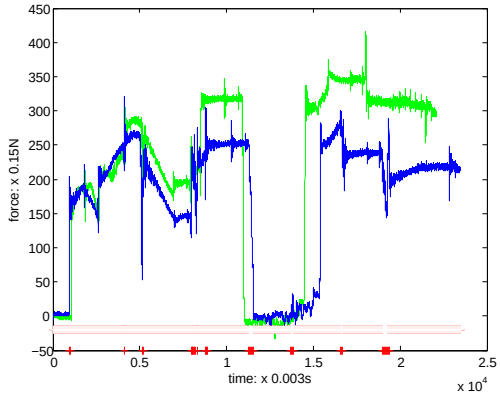
(a) Vertical component F_{1y} of the reaction force on finger 1



(b) Vertical component F_{2y} of the reaction force on finger 2



(c) Vertical component F_{3y} of the reaction force on finger 3



(d) Vertical component F_{4y} of the reaction force on finger 4

Figure 5.7: Vertical components of the measured reaction forces for Experiment I at all four fingers. In green: open-loop control. In blue: lazy force control. In red: intervals during which lazy force control performs force adjustments (5.6% of the total time).

in the safe regions. Forces smaller in magnitude than 2N are considered to be too noisy to be reliable and do not lead to any adjustments.

Figure 5.9 shows the same set of measured forces as Figure 5.8(f) (for finger 3 on hold 7), but here the forces measured during adjustment are plotted in red. Only one adjustment is due to a (magnitude) violation at this finger. Figure 5.10 provides a magnified view of the correction after that violation. Adjustment starts

at point S when the contact force reaches 45N and proceeds toward the selected force in the safe region. The convergence of the adjustment toward the selected force is globally satisfactory, even though it is not perfectly smooth locally. Other adjustments yielding more red points are caused by force violations at other contacts.

Figures 5.11, 5.12, and 5.13, show similar force plots for finger 4. This finger is first in contact at hold 4. It later moves to make contact at hold 8. When it is not moving it plays a major role in supporting a large fraction of the robot's weight. Figure 5.11 plots the contact forces when it is in contact at hold 4. The force measurements during force adjustments are shown in red and the other measurements in blue. Four adjustments were performed due to force violations at finger 4, one for angular violation and three for magnitude violations. Other adjustments yielding more points in red were performed due to violations at other fingers. Figure 5.12 magnifies the area of the adjustment for angular violation at finger 4. Here, one may notice that adjustment did not start exactly as soon as the violation occurred. When a force is between 2 and 20N in magnitude, as is the case here, its orientation is too noisy for a single measurement to be trusted. In that case, our controller waits until it has detected 10 successive angular violations to start an adjustment. Figure 5.13 plots the reaction forces on finger 4 at hold 8. Only one force adjustment was performed due to magnitude violation at this contact.

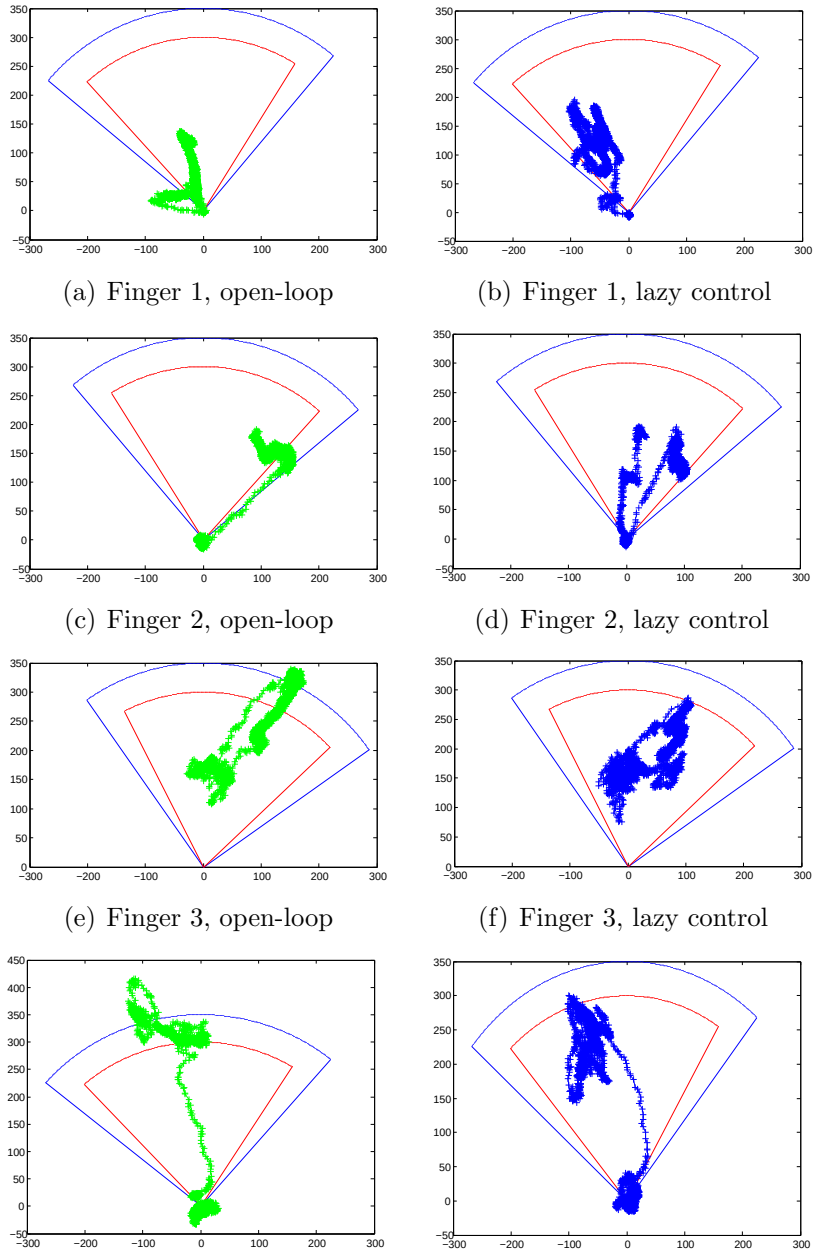
Figure 5.14 plots the position of the robot center during Experiment I. The trajectory of the center computed by the planner is shown in green and the trajectory generated with lazy force control in blue. The maximum error is less than 5mm.

If quasi-static motion is assumed, the joint torques can be estimated by summing the forces and torques on each joint of the robot and setting the system velocities and accelerations to zero. These forces include the contact forces on the finger and the forces of gravity acting at the CM of each link (Figure 5.15). Equation 5.1 and 5.2 give the torques on shoulder joints and elbow joints, respectively.

$$\tau_{is} = f_i x \times l_{xs} + f_i y \times l_{ys} + m_1 g \times l_{1s} + m_2 g \times l_{2s} \tag{5.1}$$

$$\tau_{ie} = f_i x \times l_{xe} + f_i y \times l_{ye} + m_2 g \times l_{2e} \tag{5.2}$$

Figure 5.16 plots the torques on the four shoulder joints estimated using Equation 5.1, for both open loop and lazy force control. We observe that no torque exceeds the torque limit of the actuators, 7Nm.



(a) Finger 1, open-loop (b) Finger 1, lazy control

(c) Finger 2, open-loop (d) Finger 2, lazy control

(e) Finger 3, open-loop (f) Finger 3, lazy control

(g) Finger 4, open-loop (h) Finger 4, lazy control

Figure 5.8: Plots of the reaction forces measured on all fingers during Experiment I, between the instant when finger 4 breaks contact at hold 4 and the instant when finger 2 makes contact at hold 10. With open-loop control, the forces often lie outside their safe regions, sometimes by large amounts. In contrast, with lazy force control, they rarely escape from the safe regions and, when they do, it is by small amounts during short durations.

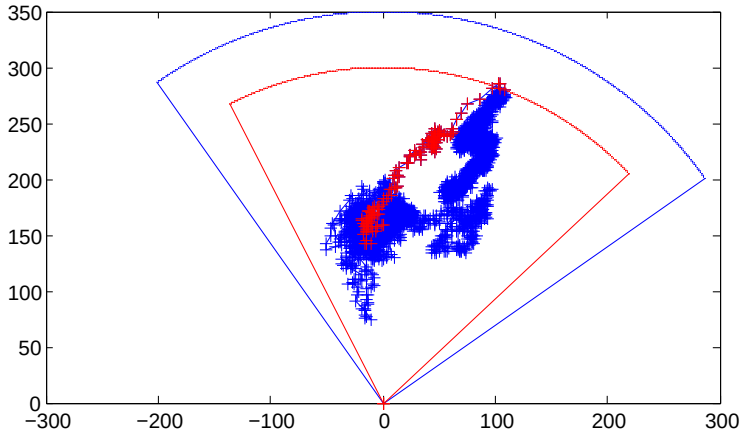


Figure 5.9: Force adjustment for finger 3 at hold 7. This plot is the same as the one in Figure 5.8(f), except that the forces measured during adjustments are now plotted in red.

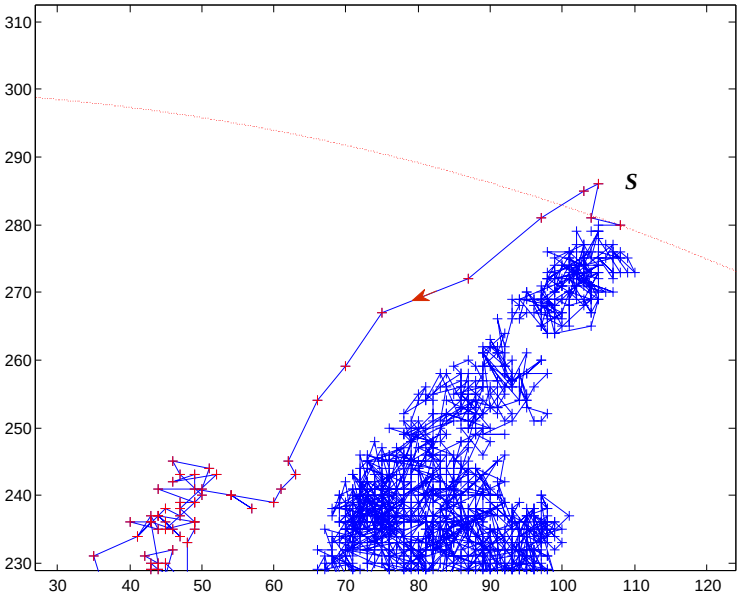


Figure 5.10: Magnified view of the force adjustment process when the magnitude of the reaction force on finger 3 grows too large.

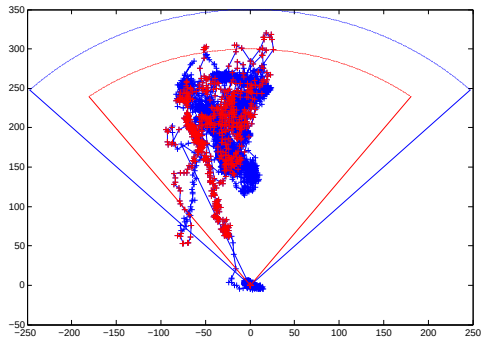


Figure 5.11: Force measurements for finger 4 at hold 4.

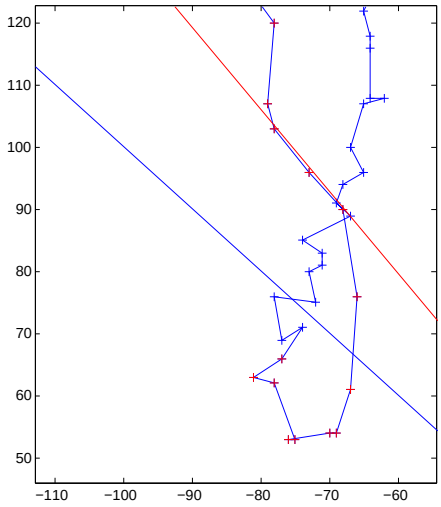


Figure 5.12: Magnified view of the force adjustment process caused by angular violation in Figure 5.11 (see text).

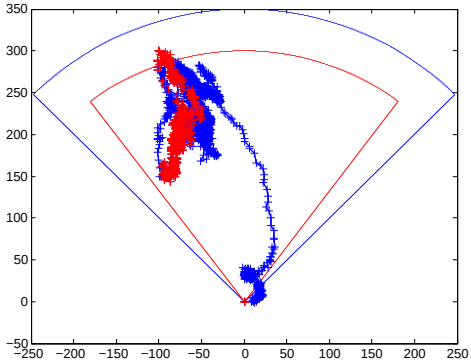


Figure 5.13: Force measurements for finger 4 at hold 8.

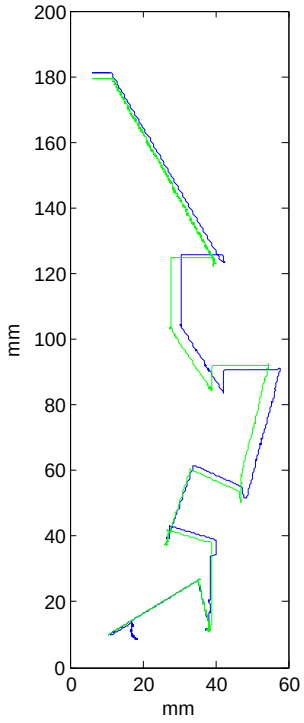


Figure 5.14: Trajectories of Capuchin's center during Experiment I. In green: planned trajectory. In blue: trajectory generated with lazy force control.

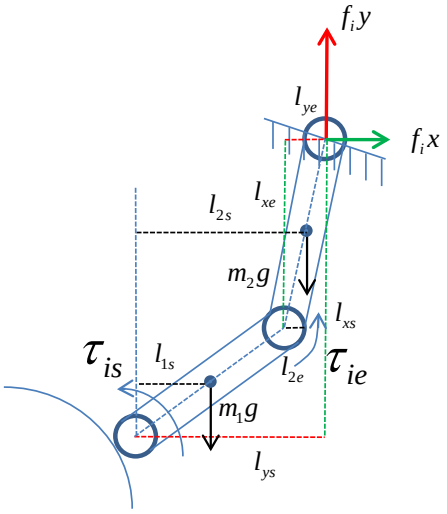
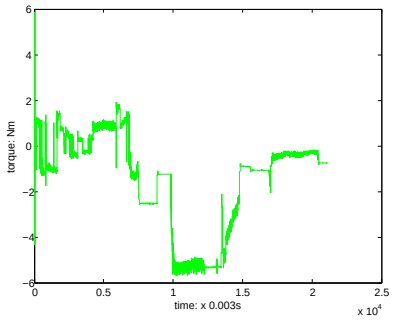
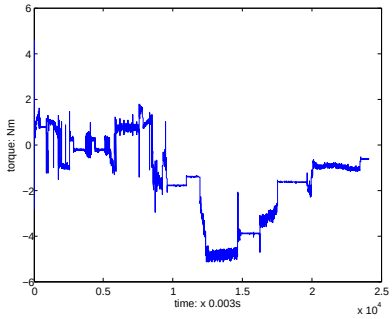


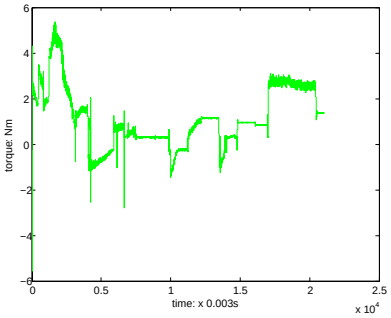
Figure 5.15: Shoulder and elbow joint torque estimation based on the measured finger contact force.



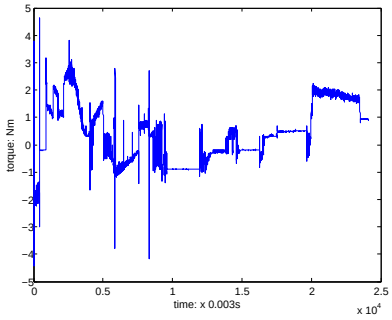
(a) Shoulder 1, open-loop



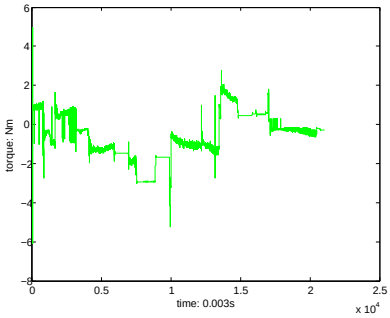
(b) Shoulder 1, lazy control



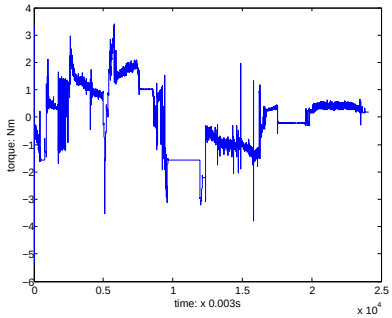
(c) Shoulder 2, open-loop



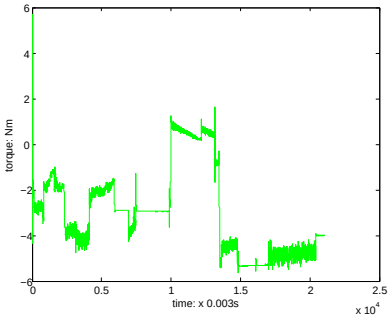
(d) Shoulder 2, lazy control



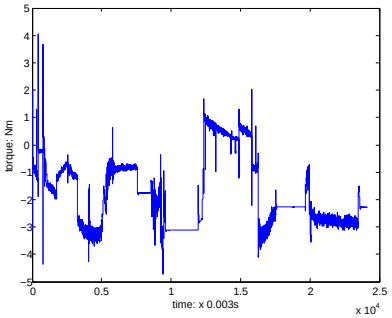
(e) Shoulder 3, open-loop



(f) Shoulder 3, lazy control



(g) Shoulder 4, open-loop



(h) Shoulder 4, lazy control

Figure 5.16: Plots of the estimated shoulder joints torques during Experiment I

5.2 Experiment II: Prevention of slipping

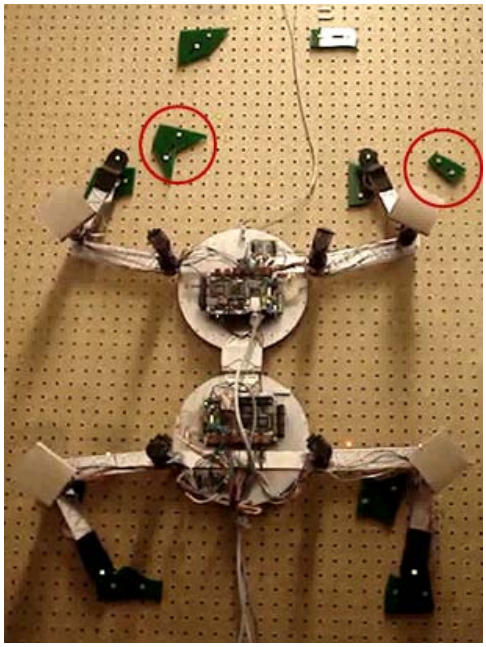
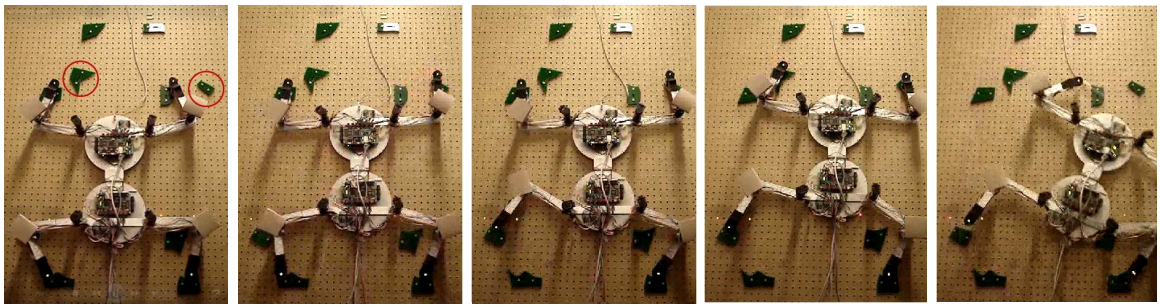


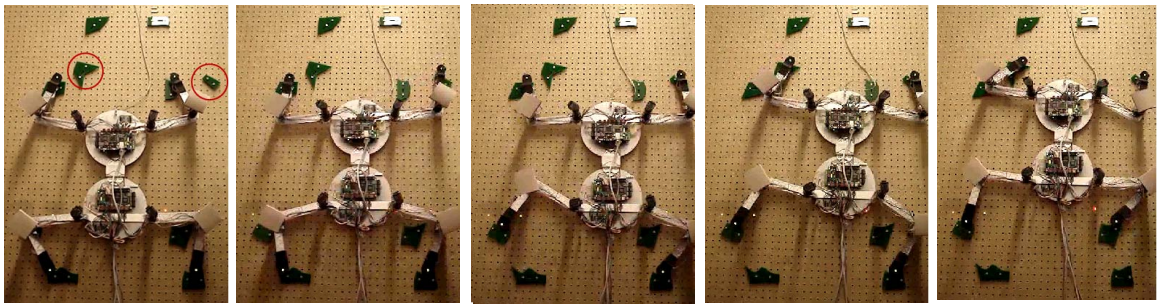
Figure 5.17: Capuchin at its initial configuration on the terrain of Experiment II.

In this experiment, like in Experiment I, the robot system was given an accurate geometric model of the terrain and a precise description of the robot's initial configuration. Vision-based control was not activated. Although the terrain looks similar, it is actually more challenging than in Experiment I. The reason is that holds 5 and 6 (circled in red in Figure 5.17 and the first snapshots of Figure 5.18) have normals heading in roughly the same non-vertical direction (due to a different orientation of the supporting features). During a portion of the climb fingers 1 and 2 must make contact at these two holds, respectively, while moving finger 4 from hold 4 to hold 8. Then it is particularly difficult to distribute the forces exerted by fingers 1, 2, and 3 so that the robot remains in equilibrium.

Figure 5.18 shows two series of snapshots of Capuchin climbing on this new terrain with open-loop and lazy force control, respectively. In the fourth snapshot of each series, fingers 1 and 2 make contact with holds 5 and 6, respectively, as instructed by the planner. But in the fifth snapshot of the first series (open-loop control),



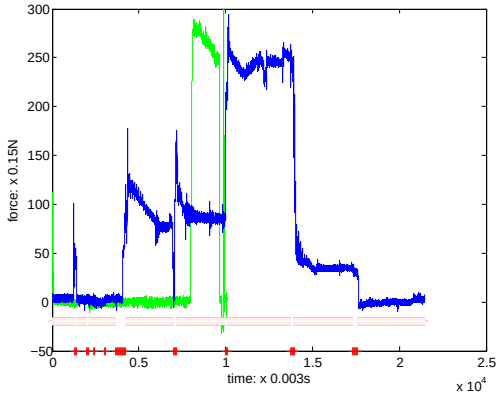
Open-loop position control



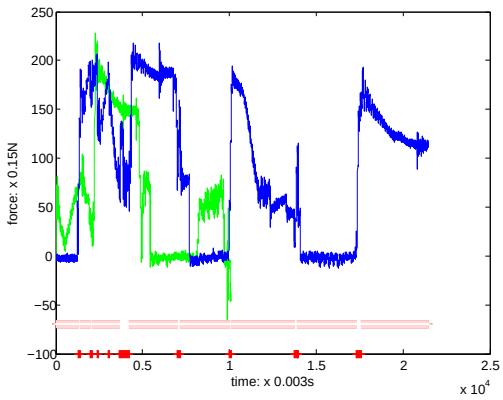
Lazy force control

Figure 5.18: Climbing motions for Experiment II with open-loop control and lazy force control.

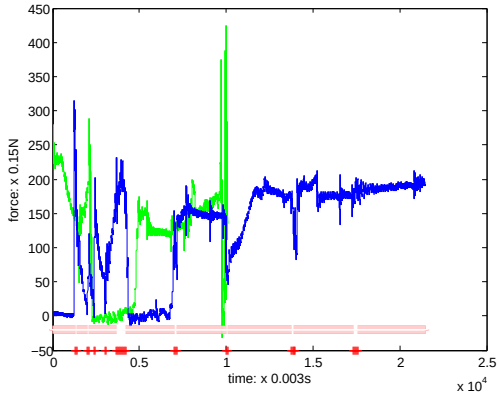
the fingers slip off their holds and the robot falls. A detailed analysis of the forces (Figures 5.19 and 5.20) reveals that at some point the reaction force on finger 4 becomes too large and, as a result, the force on finger 1 gets too small, which causes the forces on fingers 2 and 3 to exit their friction cones. In contrast the lazy force control algorithm performs a force adjustment when the force on finger 4 exceeds 45N. Soon after, it detects that the force at finger 1 is out of its safe region and performs another adjustment. Overall, lazy force control adjusts the forces properly and timely to achieve proper forces on all fingers. This allows the robot to safely lift finger 4 from hold 4 to hold 8 (fifth snapshot in the second series) and eventually succeed in climbing the entire terrain. In total there were 9 force adjustments accounting for 7.8% of the total execution time.



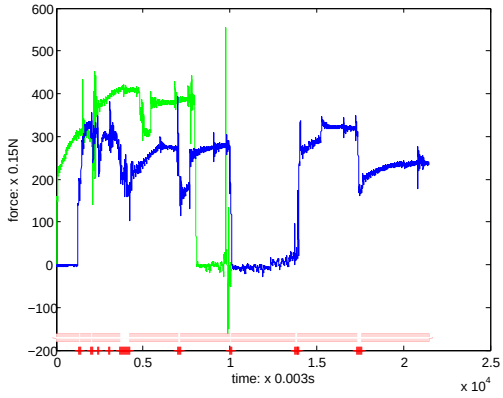
(a) Vertical component F_{1y} of the reaction force on finger 1



(b) Vertical component F_{2y} of the reaction force on finger 2



(c) Vertical component F_{3y} of the reaction force on finger 3

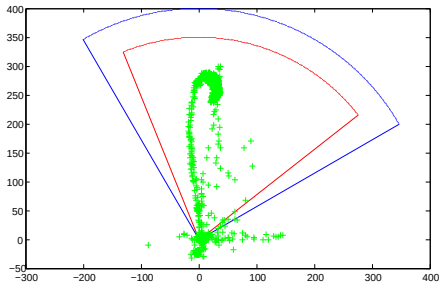


(d) Vertical component F_{4y} of the reaction force on finger 4

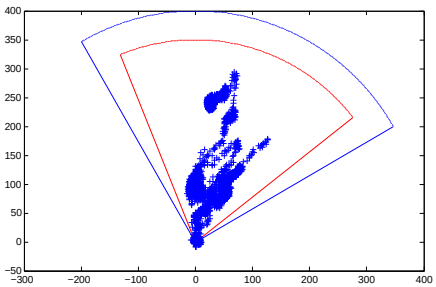
Figure 5.19: Variations of the vertical components of the measured reaction forces at all four fingers for experiment II. In green: open-loop control. In blue: lazy force control. In red: intervals during which lazy force control performs force adjustments (7.8% of the total time).

Like Figure 5.8 for Experiment I, Figure 5.20 now plots the reaction forces measured on all fingers during Experiment II, between the instant when finger 4 breaks contact at hold 4 and the instant when finger 2 makes contact at hold 10. This portion of the motion lies between 1.0 and 1.7 on the time axis of Figure 5.19. Each finger makes contact with a single hold. As mentioned above, the plots for open-loop control (in green) show that the magnitude of the reaction force at finger 4 exceeds

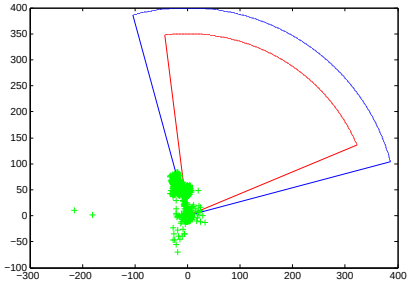
its limit by a large amount, the force at hold 1 is quasi-null, and the forces at holds 2 and 3 exit their friction cones. Figure 5.20(e) contains several measurements of the reaction force on finger 3 that are significantly outside the friction cone. These are measurements taken while the robot had started slipping. In contrast, the plots (in blue) on the right-hand side of the figure show that lazy force control succeeds in keeping reaction forces within their safe regions, except for a few measurements of small amplitude for which the orientations can be quite noisy.



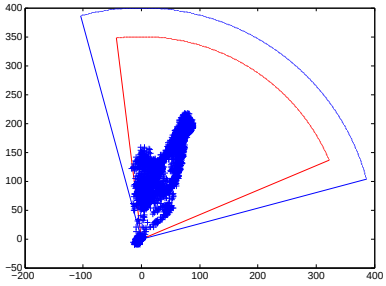
(a) Finger 1, open-loop



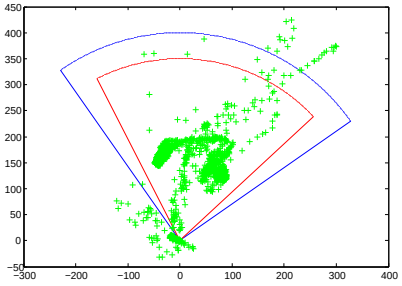
(b) Finger 1, lazy control



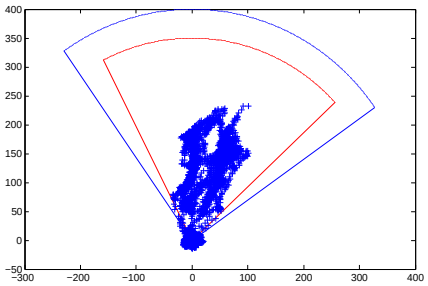
(c) Finger 2, open-loop



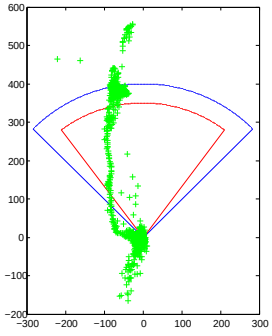
(d) Finger 2, lazy control



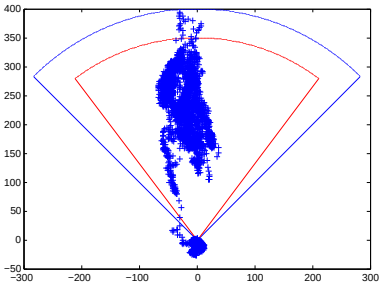
(e) Finger 3, open-loop



(f) Finger 3, lazy control



(g) Finger 4, open-loop



(h) Finger 4, lazy control

Figure 5.20: Plots of the reaction forces measured on all fingers during Experiment II. Open-loop: 0.6 to 1.0 on time axis; Lazy control: 0.3 to 1.2 on time axis.

5.3 Experiment III: Prevention of force saturation

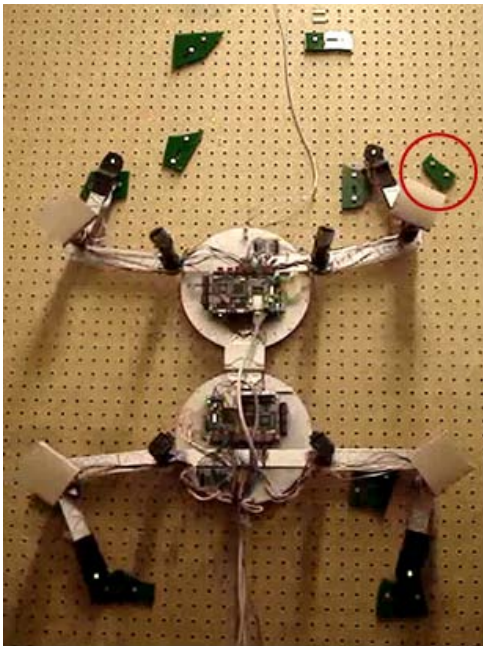


Figure 5.21: Capuchin at its initial configuration on the terrain of Experiment III.

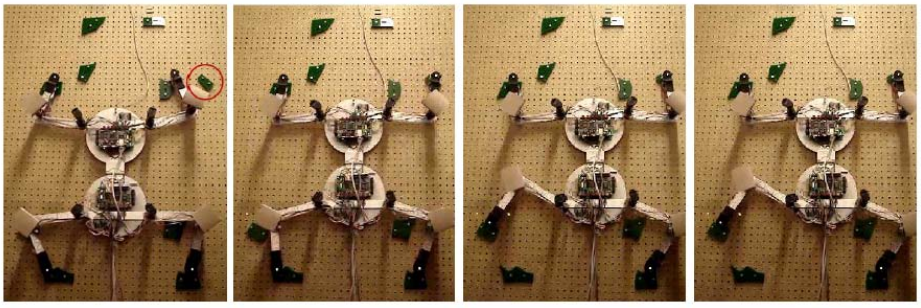


Figure 5.22: Climbing motion for Experiment III with open-loop position control.

Again, in this experiment the robot system was given an accurate geometric model of the terrain and a precise description of the robot's initial configuration. Vision-based control was not activated. Here the normal of hold 5 (circled in red in Figure 5.21) is more inclined than in Experiment II and only makes a 45-degree angle

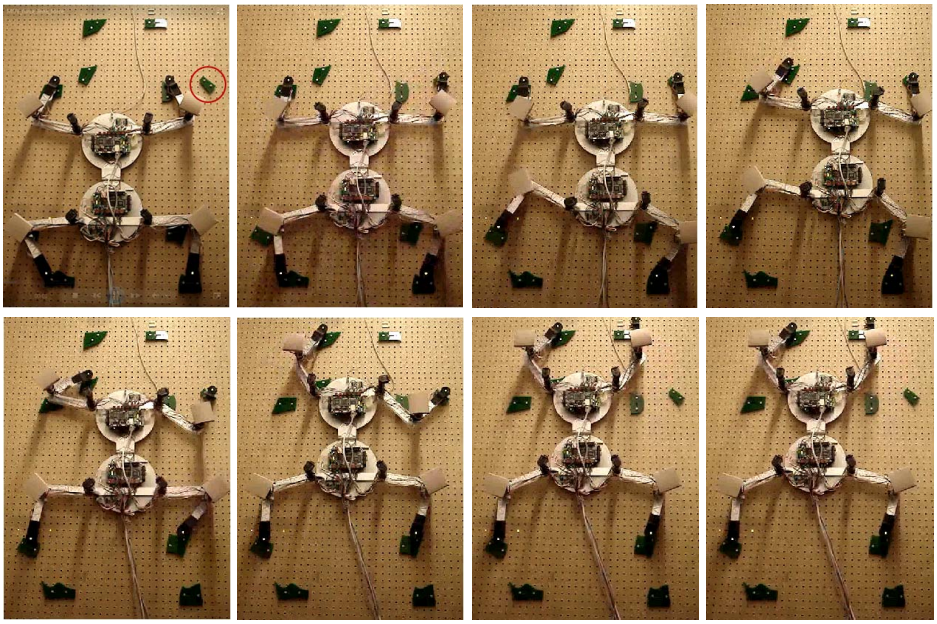
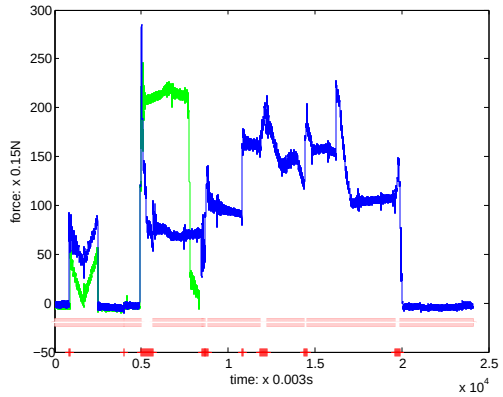


Figure 5.23: Climbing motion for Experiment III with lazy force control.

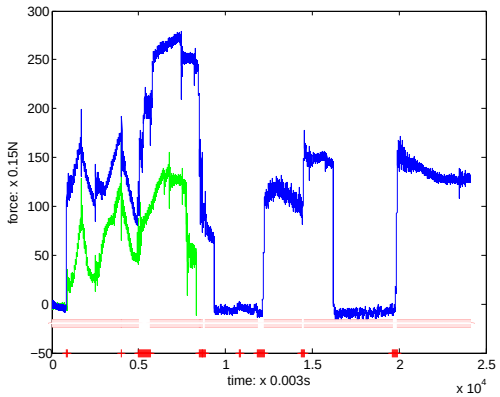
with the horizontal line, making it more difficult for Capuchin not to slip on this hold. However, hold 6 has the same more favorable orientation as in Experiment I. Figures 5.22 and 5.23 show series of snapshots of the climbing motions with open-loop and lazy force control, respectively. Figure 5.24 plots the variations over time of the vertical components of the measured reaction forces at all four fingers.

Open-loop control eventually fails to make Capuchin climb this terrain. Due to the more favorable orientation of hold 6, Capuchin climbs further than in Experiment II, but then gets stuck at a configuration where fingers 1, 2, 3, and 4 are at holds 5, 6, 7, and 8, respectively (last snapshot in Figure 5.22). The magnitude of the contact force on finger 4 grows too large and the robot cannot produce enough torques in its joints to proceed further. Instead, lazy force control adjusts forces when the force on finger 4 reaches 45N. Simultaneously, it is able to keep the force at hold 5 in its safe region to avoid slipping off the highly sloped hold. There were 8 adjustments accounting for 8.3% of the total execution time.

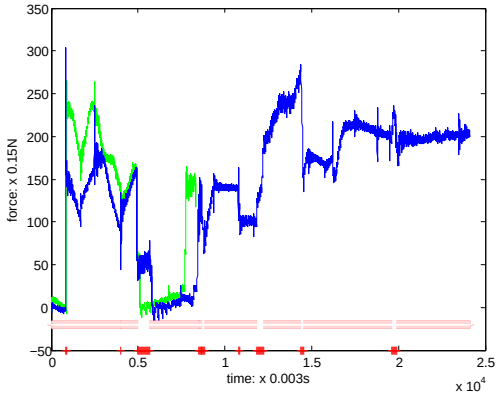
Figure 5.25 plots the reaction forces measured on all fingers during Experiment III, between the instant when finger 3 breaks contact at hold 3 and the instant when



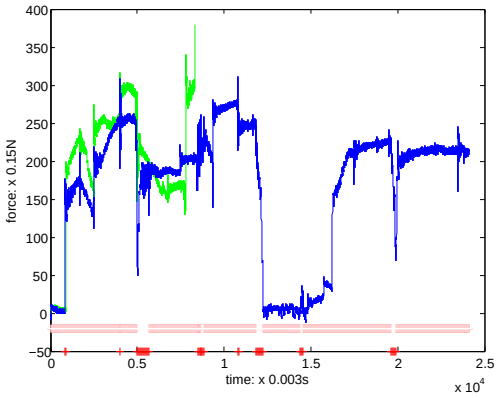
(a) Vertical component F_{1y} of the reaction force on finger 1



(b) Vertical component F_{2y} of the reaction force on finger 2



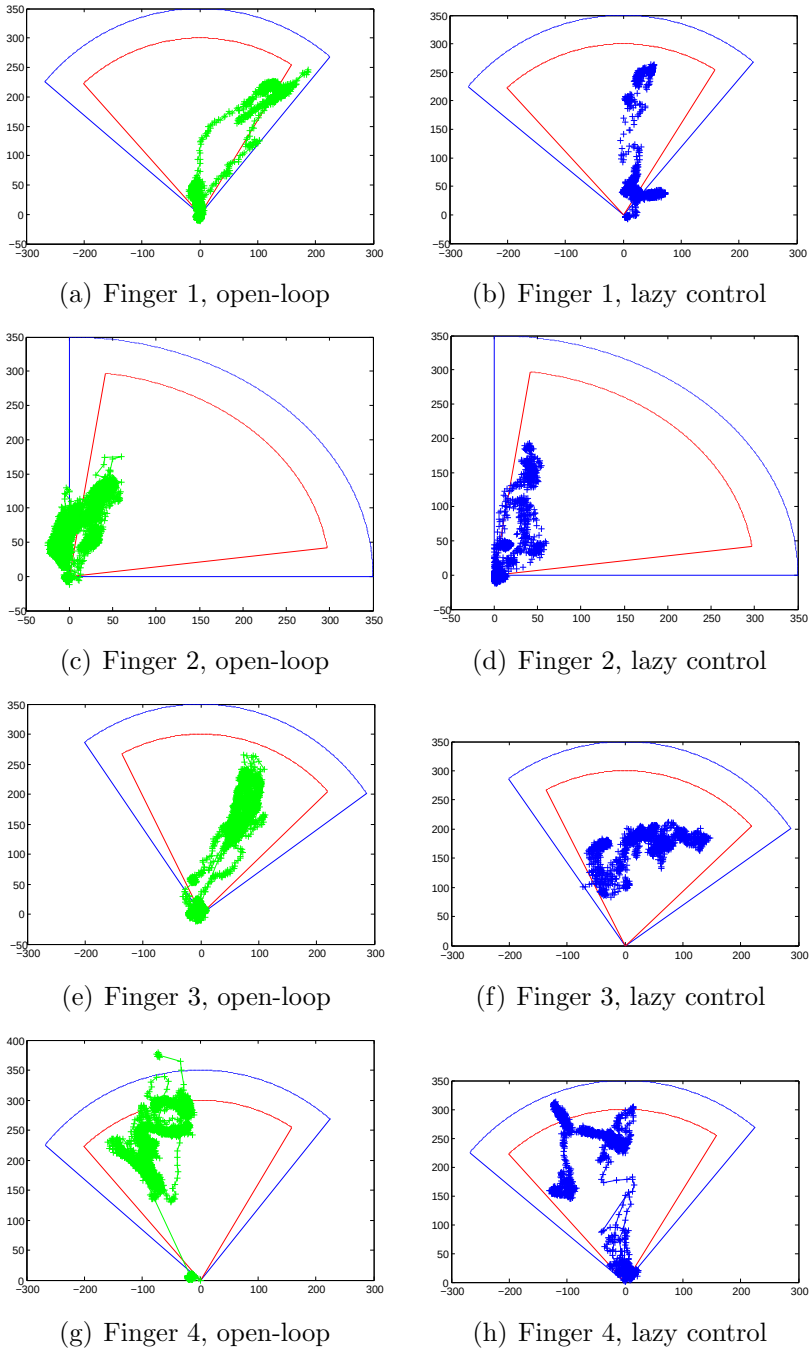
(c) Vertical component F_{3y} of the reaction force on finger 3



(d) Vertical component F_{4y} of the reaction force on finger 4

Figure 5.24: Variations of the vertical components of the measured reaction forces at all four fingers for experiment III. In green: open-loop control. In blue: lazy force control. In red: intervals during which lazy force control performs force adjustments (8.3% of the total time).

finger 2 breaks contact at hold 2. This portion of the motion lies between 0.5 and 0.8 on the time axis in Figure 5.24(a). Each finger has contact with a single hold. The friction cones are shown in Figure 5.25. The plots for open-loop control (in green) show that the contact force on finger 4 eventually grows too large. With lazy force control, the majority of the force measurements remain inside the safe region, except for forces with small magnitudes.



(a) Finger 1, open-loop

(b) Finger 1, lazy control

(c) Finger 2, open-loop

(d) Finger 2, lazy control

(e) Finger 3, open-loop

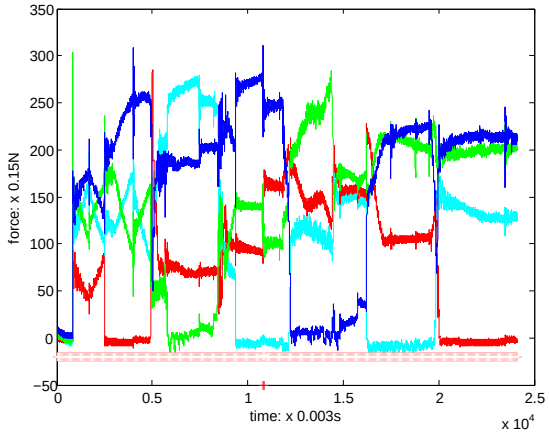
(f) Finger 3, lazy control

(g) Finger 4, open-loop

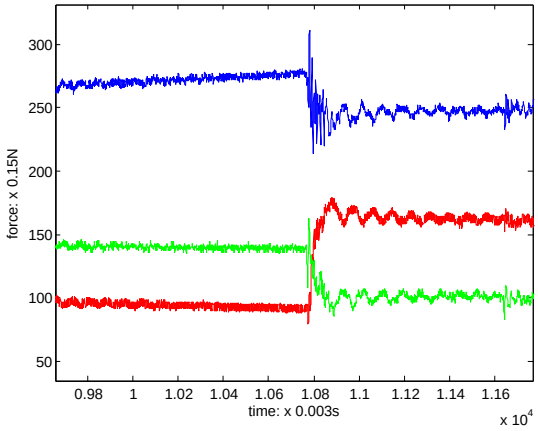
(h) Finger 4, lazy control

Figure 5.25: Plots of the reaction forces measured on all fingers during Experiment III, between the instant when finger 3 breaks contact at hold 3 and the instant when finger 2 breaks contact at hold 2.

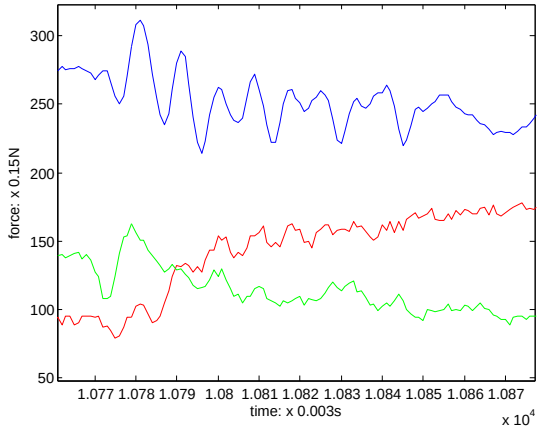
As discussed in Section 4.2.4, each force adjustment performed by the lazy force control algorithm is achieved by redistributing simultaneously all the forces exerted by the fingers in contact with the terrain, in order to achieve a target set of safe reaction forces. Figure 5.26 visualizes the concurrent evolution of several reaction forces during an adjustment. Figure 5.26(a) shows in a single diagram the variations of the vertical components of the reaction forces on all four fingers during the climbing motion with lazy force control. (They are the same plots as in Figure 5.24.) One force adjustment is highlighted in red on the horizontal axis. During this adjustment finger 2 is moving and is therefore not in contact with the terrain. Figures 5.26(b) and 5.26(c) magnify the evolution of the three contact forces during the adjustment. They show that the contact forces converge quickly (in less than 0.2 seconds) to their target values.



(a) Vertical components of the contact forces on fingers 1 (cyan), 2 (red), 3 (green), 4 (blue), during Experiment III with lazy force control



(b) Zoom-in on the force adjustment marked in red on the time axis of Figure 5.26(a)



(c) Larger zoom-in on the same force transition

Figure 5.26: Analysis of a force adjustment during experiment III. The interval of time during which this adjustment occurs is marked in red on the time axis of Figure 5.26(a).

5.4 Experiment IV: Dealing with a map error

In this experiment (Figure 5.27) the robot system is given a terrain model in which the position of feature 5 is off its actual position by 3cm (horizontally). Consequently the position of hold 5, which is defined relative to this feature, is also incorrect. Lazy force control without vision feedback to correctly dock fingers fails to make the robot climb this terrain. Finger 1 reaches the position of hold 5 assumed by the planner (Figure 5.28). Lazy force feedback then tries to achieve a proper reaction force by adjusting the finger position (as described in Section 4.2). In this case the finger actually moves, since it is not in contact with a terrain. So, lazy force control fails to achieve a proper force and the controller ends up stopping the motion with failure (although the robot did not fall). With vision feedback, the controller activates vision-based docking when hold 5 appears into the field of view of the camera located above finger 1 (Figure 5.29(a)). The controller detects that finger 1 is incorrectly positioned relatively to this hold (see Section 4.1.2) and switches to an updated trajectory to reach hold 5. In this experiment, the combination of lazy force control and vision-based docking succeeds to make the robot climb the terrain (Figure 5.30). Figure 5.29 shows camera views when fingers 1, 3, 2, and 4 reach holds 5, 7, 6, and 8, respectively.

This experiment shows that for imperfectly modeled terrains, vision-based docking is necessary for successful climbing. But large positioning errors could still be difficult to handle, e.g., if the targeted hold did not enter the field of view of the moving camera. One possibility would then be to perform exploratory moves aimed at detecting the hold in a larger region. But such moves might be limited by the equilibrium constraints. Calling back the planner may then be necessary.

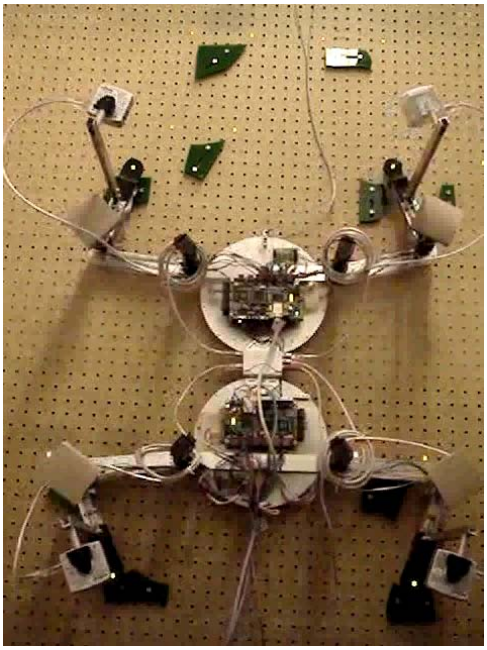


Figure 5.27: Capuchin at its initial configuration on the terrain of Experiment IV.

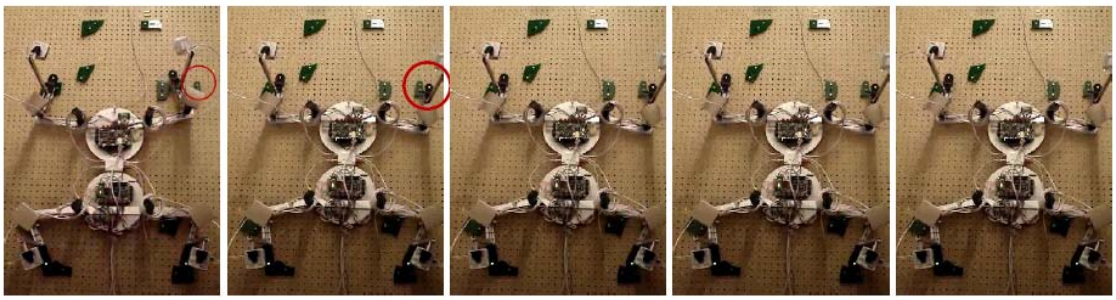
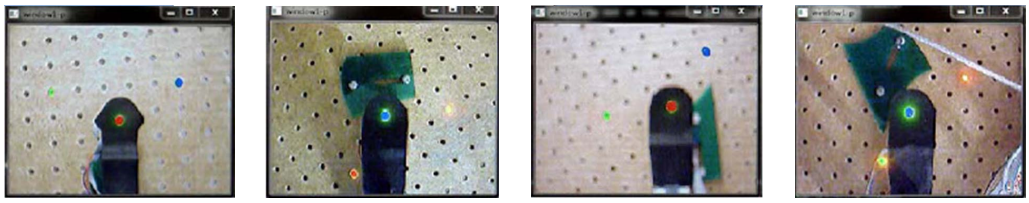


Figure 5.28: Lazy force control without vision feedback control.



(a) Finger 1 docking on hold 5 (b) Finger 3 docking on hold 7 (c) Finger 2 docking on hold 6 (d) Finger 4 docking on hold 8

Figure 5.29: Four camera views showing fingers 1, 3, 2, and 4 docking at holds 5, 7, 6, and 8.

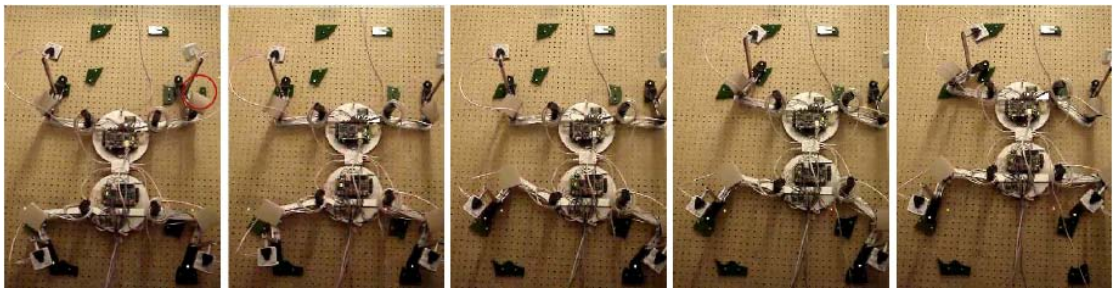


Figure 5.30: Lazy force control with vision feedback control.

Chapter 6

Conclusion and future research

In this project, we designed and built a four-limbed free-climbing robot, Capuchin, capable of climbing autonomously planar vertical terrains equipped with arbitrarily shaped and distributed features. While climbing, Capuchin uses no special tools and only relies on frictional contacts between its fingers and terrain features to maintain (static) equilibrium. An off-line motion planner developed by Bretl and Hauser automatically computes the motion trajectories from a given map of the terrain. We equipped each finger of Capuchin with a force sensor and a camera. A vision-based motion controller allows Capuchin to accurately position its fingers at contact holds selected by the planner, even when the terrain map is slightly imperfect. A force-based controller implements a novel lazy force control approach that adjusts the forces exerted by the robot at the contact holds only when some reaction forces are close to violate the equilibrium constraints. The integrated system was implemented and tested on several terrains. The tests demonstrate that Capuchin can reliably and smoothly free-climb vertical artificial climbing walls. Detailed analysis of the contact forces measured while climbing also validates our lazy force control approach. Perhaps the most interesting result is that forces only need to be monitored continuously; adjustments are needed only when some forces are close to violate the equilibrium constraints. In almost all our experiments, such adjustments took less than 10% of the total execution time of a climbing motion plan. We believe that our lazy force control approach could be applied as well in other applications requiring multiple frictional

contacts between a robot and its environment, e.g., dexterous manipulation.

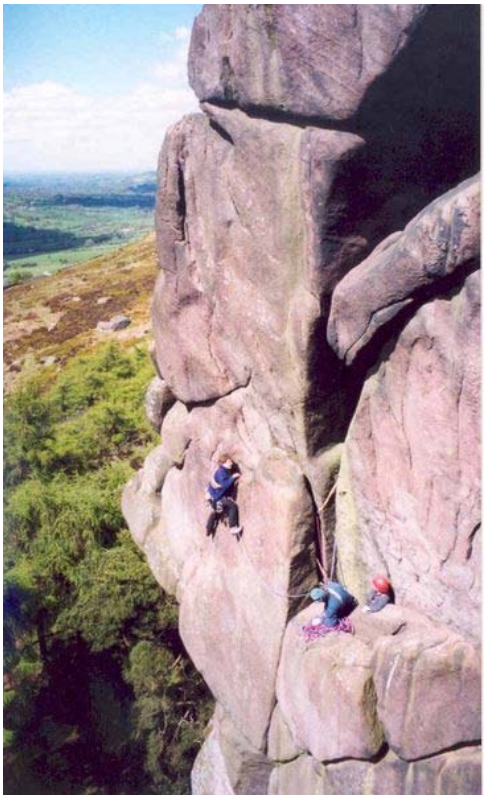


Figure 6.1: There is still a long way before one can build a robot capable of free-climbing natural crags like the one shown in this photo.

However, there is still a long way before one can build a robot capable of free-climbing natural crags like the one shown in Figure 6.1. To achieve such a goal, a number of research directions must be explored further, in particular:

Extension to 3-D robot. We originally designed Capuchin as a 3-D robot (see Section 2.1.1) capable of climbing non-planar terrain. Lack of time and design complexity led us to scale down our initial design to a 2-D version of Capuchin. An exciting and challenging work would now be to extend Capuchin to a 3-D robot. Based on our initial 3-D design, at least one more degree of freedom would have to be added at each shoulder/hip joint. Another joint would have be inserted between the upper and lower body of the robot so that it can twist its body like human climbers

often do. It is not clear yet if this joint should be actuated, or just passive (and spring-loaded). The camera on each robot finger should be replaced by a pair of stereo cameras or another light 3-D sensor to localize holds on 3-D features.

Incremental sensing and planning. A global terrain model, even a low-resolution one, is often not available in advance. In order to climb on a large terrain, the robot should be able to sense the terrain incrementally and call the planner periodically to plan additional moves or to modify some previously planned moves. This would require improving the planner to make it faster. The new planner should also be able to plan "exploratory" motions aimed at detecting new interesting features and holds.

Holds detection and characterization. Some properties of the contact surface are very critical to climbing, such as normal direction, friction, and surface roughness. 3-D vision sensors could be used to detect, localize, and characterize potential holds. But tactile sensing would also be helpful for model detailed properties of the terrain like roughness. Further research on free-climbing robots should take advantage of recent progress in developing robust extensible skin-like tactile surfaces [43].

Internal forces Free climbing can also be considered as a grasping problem if we regard the climbing wall as an "object". Capuchin has 8 actuators and the "object" has only 3 degrees of freedom. The redundant actuators are able to provide independent internal forces and moments. As described in Section 4.2.4, in our implementation the contact forces are generated by a random sampling approach. Although the generated contact forces satisfy the static equilibrium constraints and the resulted joint torques are kept within the safe range of the actuators, Capuchin's contact forces could still be optimized by the control of the internal forces, using approaches like linkage modeling. This will be especially helpful for challenging moves like chimney climbing.

Dynamic moves In our work we forced Capuchin to move slowly enough in order to neglect dynamic effects, like momentum. Although human climbers also climb

relatively slowly and do not often use dynamics, dynamic moves can still be helpful to reach distant holds, or to save time and energy. Human climbers carefully decouple dynamic moves from their hands and feet contacting the terrain to avoid moving them, as even small disturbances could lead to slipping. Allowing a free-climbing robot to perform dynamic moves is particularly challenging. It would definitively require a good dynamic model of the robot, more sensors, and a more precise model of the contacts.

Bibliography

[1] M. Abderrahim, C. Balaguer, A. Gimenez, J.M. Pastor, and V.M. Padron. ROMA: A climbing robot for inspection operations. *IEEE Int. Conf. Robotics and Automation*, Detroit, MI, 1999.

[2] M. Almonacid, R. Saltarn, R. Aracil, and O. Reinoso. Motion planning of a climbing parallel robot. *IEEE Trans. Robot. Automat.*, Vol. 19, No.3, pp. 485-489, 2003.

[3] Abd Alsalam Sh. I. Alsalameh, Shamsudin H.M. Amin, and Rosbi Mamat. Mechanical design of a quadruped robot for horizontal ground to vertical wall movement. *TENCON*, volume 1, pages 213-217, 2000.

[4] H. Amano. A vertically moving robot able to grip handrails for fire-fighting. *Advanced Robotics*, 16(6):557-560, 2002.

[5] A.T. Asbeck, S. Kim, and M.R. Cutkosky. Scaling hard vertical surfaces with compliant microspine arrays. *The Int. J. of Robotics Research*, 25: 11651179, 2006.

[6] K. Autumn, M. Sitti, Y.A. Liang, A.M. Peattie, W.R. Hansen, S. Sponberg, T.W. Kenny, R. Fearing, J.N. Israelachvili, and R.J. Full. Evidence for van der waals adhesion in gecko setae. *PNAS*, 99(19):11252-12256, Jan 1999.

[7] F.-W. Bach, H. Haferkamp, J. Lindemaier, and M. Rachkov. Underwater climbing robot for contact arc metal drilling and cutting. *IEEE Int. Conf. Industrial Electronics, Control, and Instrumentation*, pages 1560-1565, Taipei, Taiwan, 1996.

[8] C. Balaguer, A. Gimenez, J.M. Pastor, V.M. Padron, and M. Abderrahim. A climbing autonomous robot for inspection applications in 3D complex environments. *Robotica*, Vol. 18, pp. 287-297, 2000.

[9] D. Bevly, S. Dubowsky, and C. Mavroidis. A simplified cartesian computed torque controller for highly geared systems and its application to an experimental climbing robot. *ASME J. of Dynamic Systems, Measurement, and Control*, 122(1):27-32, 2000.

[10] D. Bevly, S. Farritor, and S. Dubowsky. Action module planning and its application to an experimental climbing robot. *IEEE Int. Conf. Robotics and Automation*, volume 4, pages 4009-4014, 2000.

[11] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. *IEEE Int. Conf. Robotics and Automation*, 2000.

[12] T. Bretl, T. Miller, S.M. Rock, and J.C. Latombe. Climbing robots in natural terrain. *7th Int. Symp. on Artificial Intelligence, Robotics and Automation in Space*, Nara, Japan, May 2003.

[13] T. Bretl, S.M. Rock, and J.C. Latombe. Motion planning for a three-limbed climbing robot in vertical natural terrain. *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, 2003.

[14] T. Bretl, S. Rock, J.C. Latombe, B. Kennedy, and H. Aghazarian. Free-climbing with a multi-use robot. *Int. Symp. Exp. Rob.*, Singapore, 2004.

[15] T. Bretl. *Multi-step motion planning: application to free-climbing robots.* Ph.D Dissertation, Dept. of Aero & Astronautics, Stanford University, Stanford, CA, June 2005.

[16] T. Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: the free-climbing robot problem. *Int. J. of Robotics Research*, Vol. 25, No. 4, pp. 317-342, 2006.

[17] L. Briones, P. Bustamante, and M. A. Serna. Wall-climbing robot for inspection in nuclear power plants. *IEEE Int. Conf. Robotics and Automation*, pages 1409-1414, San Diego, CA, 1994.

[18] I.M. Chen and S.H. Yeo. Locomotion of a two-dimensional walking climbing robot using a closed-loop mechanism: From gait generation to navigation. *Int. J. Rob. Res.*, 22(1):21-40, 2003.

[19] Y. Chen, I. Walker, and J. Cheatham. A new approach to force distribution and planning for multifingered grasps of solid objects. *IEEE Conference on Robotics and Automation*, pages 890-896, Sacramento, CA. April 1991.

[20] H. Dulimarta and R. L. Tummala. Design and control of miniature climbing robots with nonholonomic constraints. *WCICA*, Shanghai, P.R.China, 2002.

[21] Y. Fujimoto and A. Kawamura. Simulation of an autonomous biped walking robot including environmental force interaction. *IEEE Robotics and Automation Magazine*, pages 33-42, June 1998.

[22] S. Galt, B.L. Luk, and A.A. Collie. A tele-operated semi-intelligent climbing robot for nuclear applications. *Mechatronics and Machine Vision in Practice*, pages 118-123, Toowoomba, Australia, 1997.

[23] J. C. Grieco, M. Prieto, M. Armada, and P. G. de Santos. A six-legged climbing robot for high payloads. *IEEE Int. Conf. Cont. App.*, Trieste, Italy, 1998.

[24] G. Guo and W.A. Gruver. Fingertip force planning for multifingered robot hands. *IEEE Conference on Robotics and Automation*, pages 665-672, Sacramento, CA. April 1991.

[25] K. Hashimoto. A review on vision-based control of robot manipulators. *Advanced Robotics* (AR), 17(10):969-991, 2003.

[26] K. Hauser. *Motion planning for legged and humanoid robots.* Ph.D. Thesis, Dept. Computer Science, Stanford University, Stanford, CA, September 2008.

[27] K. Hauser, T. Bretl, J.C. Latombe, K. Harada, and B. Wilcox, Motion planning for legged robots on varied terrain. *Int. J. of Robotics Research*, 27(11-12):1325-1349, 2008.

[28] K. Hauser and J.C. Latombe. Multi-modal motion planning in non-expansive spaces. *Int. J. of Robotics Research*, 29(7):897-915, 2010.

[29] S. Hirose, A. Nagabuko, and R. Toyama. Machine that can walk and climb on floors, walls, and ceilings. *IEEE Int. Conf. Robotics and Automation*, Pisa, Italy, pp. 753-758, 1991.

[30] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. *IEEE Int. Conf. Robotics and Automation*, Leuven, Belgium, 1998.

[31] N. Hogan. Impedance control: an approach to manipulation. Part I: Theory. Part II: Implementation. Part III: Applications. *Trans ASME J. Dyn. Syst. Meas. Control*, 107, 1-24, 1985.

[32] D. Hsu. *Randomized single-query motion planning in expansive spaces*. Ph.D. Dissertation, Dept. of Computer Science, Stanford University, Stanford, CA, 2000.

[33] K. Iagnemma, A. Rzepniewski, S. Dubowsky, P. Pirjanian, T. Huntsberger, and P. Schenker. Mobile robot kinematic reconfigurability for rough-terrain. *Sensor Fusion and Decentralized Control in Robotic Systems III*, 2000.

[34] J. Joh and H. Lipkin. Lagrangian wrench distribution for cooperating robotic mechanisms. *IEEE Conference on Robotics and Automation*, pages 224-229, Sacramento, CA. April 1991.

[35] T. Kang, H. Kim, T. Son, and H. Choi. Design of quadruped walking and climbing robot. *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, Las Vegas, NV, 2003.

[36] L.E. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. and Automat.*, Vol. 12. No. 4, pp. 566-580, 1996.

[37] S. Kim, M. Spenko, S. Trujilo, B. Yeyneman, V. Mattoli, and M. Cutkosky. Whole body adhesion: hierarchical, directional and distributed control of adhesive forces for a climbing robot. *IEEE Intl. Conf. on Robotics and Automation*, pages 1268-1273, April 2007.

[38] J. Kuffner, K., S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. *Int. Symp. Robotics Research*, Siena, Italy, 2003.

[39] V. Kumar and K.J. Waldron. Force distribution in closed kinematic chains. *IEEE Conference on Robotics and Automation*, pages 114-119, Philadelphia, PA. April 1988.

[40] H. Lee, Y. Shen, C. Yu, G. Singh, and A. Y. Ng. Quadruped robot obstacle negotiation via reinforcement learning. *IEEE Int. Conf. Robotics and Automation*, 2006.

[41] A. Leeper, K. Hsiao, E. Chu, and K. Salisbury. Using near-field stereo vision for robotic grasping in cluttered environments. *Proc. Int. Symp. on Experimental Robotics*, New Delhi, India. Dec, 2010.

[42] S.P. Linder, E. Wei, A. Clay. Robotic rock climbing using computer vision and force feedback. *Proc. IEEE Int. Conf. Robotics and Automation*, pages 4685-4690, 2005.

[43] D. J. Lipomi, M. Vosgueritchian, B. C-K. Tee, S. L. Hellstrom, J. A. Lee, C. H. Fox, and Z. Bao. Skin-like pressure and strain sensors based on transparent elastic films of carbon nanotubes. *Nature Nanotechnology*, 2011.

[44] A. Madhani and S. Dubowsky. Motion planning of mobile multi-limb robotic systems subject to force and friction constraints. *IEEE Int. Conf. Robotics and Automation*, 1992.

[45] M. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-11,418-432, 1981.

[46] R.B. McGhee and D.E. Orin. A mathematical programming approach to control of joint positions and torques in legged locomotion systems. *Proc. of ROMANSY-76 Symposium*, 1976.

[47] T. G. Miller, T. Bretl, and S. Rock. Control of a climbing robot using real-time convex optimization. *Proc. of the IFAC Symposium on Mechatronic Systems*, Heidelberg, Germany, September 2006.

[48] T. G. Miller. *Control of a climbing robot using real-time convex optimization.* Ph.D Dissertation, Dept. of Aero & Astronautics, Stanford University, Stanford, CA, December 2007.

[49] A. Nagakubo and S. Hirose. Walking and running of the quadruped wall-climbing robot. *IEEE Int. Conf. Robotics and Automation*, pp. 1005-1012, 1994.

[50] Y. Nakamura. Minimizing object strain energy for coordination of multiple robotic mechanisms. *American Control Conference*, pages 499-504, 1998.

[51] W. Neubauer. A spider-like robot that climbs vertically in ducts or pipes. *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, Munich, Germany, pp. 1178-1185, 1994.

[52] M. Nilsson. Snake robot - free climbing. *IEEE Control Systems Magazine*, Vol. 18, pp. 21-26, Feb 1998.

[53] J. Park and O. Khatib. Robot multiple contact control. *Robotica*, Vol. 26, No. 5, pp. 667-677, September 2008.

[54] S. P. Patarinski and R. G. Botev. Robot force control: a review. *Mechatronics*, Vol. 3, No. 4, pp. 377-398, 1993.

[55] P. Pirjanian, C. Leger, E. Mumm, B. Kennedy, M. Garrett, H. Aghazarian, S. Farritor, and P. Schenker. Distributed control for a modular, reconfigurable cliff robot. *IEEE Int. Conf. on Robotics and Automation*, 2002.

[56] M. Raibert and J. Craig. Hybrid position/force control of manipulators. *Trans. ASME J. Dyn. Syst. Meas. Control*, 102, 126-133, 1981.

[57] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter. BigDog, the rough-terrain quaduped robot. *Proc. of the 17th World Congress The Int. Federation of Automatic Control*, Seoul, Korea, July 6-11, 2008.

[58] Z.M. Ripin, B.S. Tan, A.B. Abdullah, and Z. Samad. Development of a low-cost modular pole climbing robot. *TENCON*, volume 1, pages 196-200, Kuala Lumpur, Malaysia, 2000.

[59] T. Robmann and F. Pfeiffer. Control of an eight legged pipe crawling robot. *Int. Symp. on Experimental Robotics*, pp. 353-346, 1997.

[60] S. W. Ryu, J. J. Park, S. M. Ryew, and H. R. Choi. Self-contained wall-climbing robot with closed link mechanism. *IEEE/RSJ Int. Conf. on Int. Rob. and Sys.*, Maui, HI, 2001.

[61] M. Saha. *Motion planning with probabilistic roadmaps.* Ph.D. Dissertation, Dept. of Mechanical Eng., Stanford University, Stanford, CA, 2006.

[62] J.K. Salisbury. Active stiffness control of a manipulator in Cartesian coordinates. *Proc. 19th 1EEE Conf. on Decision and Control*, 1980.

[63] G. Sanchezl. *Single-query Bi-directional probabilistic roadmap planner with lazy collision checking.* Ph.D. Dissertation, ITESM Campus Cuernavaca, Mexico, 2002.

[64] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic grasping of novel objects using vision. *Int. J. of Robotics Research*, (IJRR), 2008.

[65] T. Schlegl, M. Buss, T. Omata, and G. Schmidt. Fast dextrous regrasping with optimal contact forces and contact sensor-based impedance control. *IEEE Int. Conf. on Robotics and Automation*, pages 103-108, 2001.

[66] L. Sentis and O. Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *Int. J. of Humanoid Robotics*, 2(4):505-518, December 2005.

[67] L. Sentis, J. Park, and O. Khatib. Modeling and control of multi-contact centers of pressure and internal forces in humanoid robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 453-460, St. Louis, MO, USA, October 2009.

[68] P. Sinha and J. Abel. A contact stress model for multifingered grasps of rough objects. *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, pages 7-21, February 1992.

[69] W.T. Townsend and J.K. Salisbury. The efficiency limit of belt and cable drives. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 110, pp. 303-307, Sep 1988.

[70] D. Whitney. Force feedback control of manipulator fine motions. *Trans. ASME J. Dyn. Syst. Meas. Control*, 98, 91-97, 1977.

[71] B. H. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmed, A. Sirota, and B. Cooper. ATHLETE: a cargo handling and manipulation robot for the moon. *Journal of Field Robotics*, 24(5), DOI: 10.1002/rob.20193, 421-434, Apr 2007.

[72] D. Williams and O. Khatib. The Virtual Linkage: A model for internal forces in multi-grasp manipulation. *IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 1025-1030, Atlanta, GA, USA, May 1993.

[73] Y. Xu, H.B. Brown, M. Friendman, and T. Kanade. Control system of the self-mobile space manipulator. *IEEE Trans. Contr. Syst. Technol.*, 2(3):207-219, 1994.

[74] W. Yan, L. Shuliang, X. Dianguo, Z. Yanzheng, S. Hao, and G. Xuesban. Development and application of wall-climbing robots. *IEEE Int. Conf. Robotics and Automation*, Detroit, MI, 1999.

[75] M. Yim, S. Homans, and K. Roufas. Climbing with snake-robots. *IFAC Workshop on Mobile Robot Technology*, Jejudo, Korea, 2001.

[76] X. Yun. Coordination of two-arm pushing. *IEEE Conference on Robotics and Automation*, pages 182-187, Sacramento, CA. April 1991.

[77] A. Zagler and F. Pfeiffer. MORITZ a pipe crawler for tube junctions. *IEEE Int. Conf. Robotics and Automation*, pp. 2954-2959, Taipei, Taiwan, 2003.

[78] Y.F. Zheng and J.Y.S Luh. Joint torques for control of two coordinated moving robots. *IEEE International Conference on Robotics and Automation*, pages 1375-1380, San Francisco, CA. April 1986.

[79] Ruixiang Zhang, Prahlad Vadakkepat and Chee-Meng Chew. An evolutionary algorithm for trajectory based gait generation of biped robot. *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems* 2003

[80] Ruixiang Zhang and Prahlad Vadakkepat. Motion planning of biped robot climbing stairs. *FIRA Robot World Cup* Vienna, 2003.

[81] P Vadakkepat, NB Sin, D Goswami, RX Zhang, LY Tan. Soccer playing humanoid robots Processing architecture, gait generation and vision system. *Robotics and Autonomous Systems* 57 (8), 776-785

[82] R Zhang, P Vadakkepat, CM Chew, J Janardhanan. Mechanical design and control system configuration of a humanoid robot. *Proc. of 2nd Int. Conf. on Computational Intelligence, Robotics and Autonomous Systems* 2003

[83] Ruixiang Zhang. Design of a climbing robot: Capuchin. *Proc. 5th Intl. Conf. on Computational Intelligence, Robotics, and Autonomous Systems* 2008

[84] R Zhang, P Vadakkepat, CK Wai. Mechanical design and control system configuration of RoboSapien humanoid robot. *Proceedings of the International Conference on Computational Intelligence and Autonomous Systems* 2003

[85] Ruixiang Zhang. The development and walking control of biped robot. *Stanford University*

[86] http://www.cs.dartmouth.edu/s̃pl/Academic/Robots/Tenzing/

[87] http://www-robotics.jpl.nasa.gov/systems/system.cfm?System=5