The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Compliant Quadruped Locomotion Over Rough Terrain

Jonas Buchli, Mrinal Kalakrishnan, Michael Mistry, Peter Pastor, and Stefan Schaal

Computational Learning and Motor Control Lab, University of Southern California
Los Angeles, CA 90089
Email: {buchli,kalakris,mmistry,pastorsa,sschaal}@usc.edu

***Abstract*— Many critical elements for statically stable walking for legged robots have been known for a long time, including stability criteria based on support polygons, good foothold selection, recovery strategies to name a few. All these criteria have to be accounted for in the planning as well as the control phase. Most legged robots usually employ high gain position control, which means that it is crucially important that the planned reference trajectories are a good match for the actual terrain, and that tracking is accurate. Such an approach leads to conservative controllers, i.e. relatively low speed, ground speed matching, etc. Not surprisingly such controllers are not very robust – they are not suited for the real world use outside of the laboratory where the knowledge of the world is limited and error prone. Thus, to achieve *robust* robotic locomotion in the archetypical domain of legged systems, namely complex rough terrain, where the size of the obstacles are in the order of leg length, additional elements are required. A possible solution to improve the robustness of legged locomotion is to maximize the compliance of the controller. While compliance is trivially achieved by reduced feedback gains, for terrain requiring precise foot placement (e.g. climbing rocks, walking over pegs or cracks) compliance cannot be introduced at the cost of inferior tracking. Thus, model-based control and – in contrast to passive dynamic walkers – active balance control is required. To achieve these objectives, in this paper we add two crucial elements to legged locomotion, i.e., floating-base inverse dynamics control and predictive force control, and we show that these elements increase robustness in face of unknown and unanticipated perturbations (e.g. obstacles). Furthermore, we introduce a novel line-based COG trajectory planner, which yields a simpler algorithm than traditional polygon based methods and creates the appropriate input to our control system. We show results from both simulation and real world of a robotic dog walking over non-perceived obstacles and rocky terrain. The results prove the effectivity of the inverse dynamics/force controller. The presented results show that we have all elements needed for robust all-terrain locomotion, which should also generalize to other legged systems, e.g., humanoid robots.**

I. Introduction

For fast and precise movement in *known* environments, high gain position control of robots can achieve accurate results, and control mechanisms can remain simple. However, when the task involves contact with the environment, and especially if these contacts have a stochastic element, the advantage of high gain position control can turn into a severe disadvantage: the controller will try to satisfy the position goal at all cost and with all available force, at the danger of unbalancing the robot. A further drawback of high gain position control is the dependence on a "perfect" plan: a perfect kinematic model of the robot and the environment and velocity matching when making ground contact. The latter is important, since if contact with the environment occurs at moments when the end-effector has non-zero velocity, a perturbation is created at the end-effector and due to the high gain position control in each joint, this perturbations gets transmitted into every element of the kinematic chain.

These issues are particularly problematic in legged locomotion, where we have no fixed contact with the ground. An inadvertent impact of the swing leg with an obstacle or the terrain will create a perturbation to the body as well as to all other feet, which can jerk the body around, and in the case of difficult terrain with small and non-ideal footholds, induce critical slipping and/or tumbling of the robot. Ideally one wants to decouple the center of gravity (COG) from the end-effector of the (swing) legs. This requires low gain control. Low gain control without sacrificing precision in the executed movements requires a feedforward element based on an inverse dynamic model. This controller is able to predict the required torque for accurate control in a *feedforward* fashion.

In legged locomotion over rough terrain, compliant control is very desirable [1] as it is unavoidable that the planned trajectories will be executed with errors, thus leading to unplanned contacts due to slip, modeling errors, and imprecise knowledge of the terrain. In essence we need to be able to combine compliant walking with active balance control, which is in contrast to passive [2] or almost-passive dynamic walkers [3] which have robustness margins that are too small, and are not versatile enough to be of use in complex terrain.

While theoretically inverse dynamics control addresses issues of accurate control and balance control, its practical application for autonomous robots such as legged robots has been hindered by several issues. In the next section we will discuss these issues and show how they can be addressed by a novel, analytically correct, floating-base inverse dynamics law. Inverse dynamics requires desired joint trajectories, which are twice differentiable. In order to convert the task space of locomotion, i.e., the space of foot movement, into usable joint space trajectories, the end-effector trajectories need to be completed with a COG trajectory. In this paper we will thus focus on two aspects: First, we will review the novel Inverse Dynamics control law for floating-base systems as first introduced in [4], then we will explain our approach for COG path planning. We will show that stability triangles are not explictly needed for quasi steady state locomotion, but the criterion reduces to lines formed by diagonal pairs of footholds.

Results from simulation and on the real robot will be presented showing the advantage of the low gain inverse dynamics force control in presence of not or wrongly perceived obstacles and environments.

II. Inverse Dynamics Control

While inverse dynamics control for a fixed-base robot, such as an industrial manipulator, is textbook knowledge [5], it is not so easy to develop a general inverse dynamics control law for a floating-base system such as a legged robot. Practical application of floating-base inverse dynamics has been hindered by the following issues: (a) Dependence on precise dynamics models, amplified by numerical problems that can arise due to matrix inversions, in particular the inversion of the rigid body dynamics (RBD) inertia matrix [6], (b) external forces need to be measured, and (c) no general analytically correct framework for floating-base systems for arbitrary constraints from the environment and closed loop kinematic chains.

In a recent development [4], we demonstrated how we can address (b),(c). As it turns out the solution, while still dependent on a model, is less susceptible to modeling errors as it avoids inversion of the RBD inertia matrix, such that it also alleviates issue (a). (c) is also partially taken care of by planning constraint satisfying reference trajectories.

Our novel way of computing inverse dynamics control for floating-base systems can be applied to arbitrary robots with multiple and dynamically changing constraints. Because of the problems that are caused by working with measured contact forces directly, we have developed an approach that avoids knowledge of the contact forces. This solution is accomplished by computing the analytically correct inverse dynamics torques in the reduced dimensional Null-space of the constraints, as realized by an orthogonal decomposition of the constraint Jacobian.

In the following we will shortly review the requirements and the central results of our floating-base inverse dynamics framework. For further details please refer to [4].

A. Floating-Base Dynamics

The configuration of the full floating-base system is described as

$$\mathbf{q} = \left[\begin{array}{cc} \mathbf{q}_r^T & \mathbf{x}_b^T \end{array} \right]^T \tag{1}$$

where $\mathbf{q}_r \in \mathbb{R}^n$ is the joint configuration of the rigid body robot with n joints and $\mathbf{x}_b \in \mathbb{R}^6$ is the position and orientation of the coordinate system attached to the robot base, and measured with respect to an inertial frame (cf. [4] for further details). When the robot is in contact with the environment, the equations of motion with respect to an inertial frame are given by

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{S}^T\tau + \mathbf{J}_C^T(\mathbf{q})\lambda \tag{2}$$

with variables defined as follows:

- $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n+6\times n+6}$: the floating base inertia matrix
- $\mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n+6}$: the floating-base centripetal, Coriolis, and gravity forces.
- $\mathbf{S} = \left[\begin{array}{cc} \mathbf{I}_{n\times n} & \mathbf{0}_{n\times 6} \end{array} \right]$: the actuated joint selection matrix
- $\tau \in \mathbb{R}^n$: the vector of actuated joint torques
- $\mathbf{J}_C \in \mathbb{R}^{k\times n+6}$: the Jacobian of k constraints
- $\lambda \in \mathbb{R}^k$: the vector of contact forces

In order to be able to derive an inverse dynamics control law that is not depending on measured forces we make the following assumptions

1) $\dot{\mathbf{x}}_C = \ddot{\mathbf{x}}_C = \mathbf{0}$ is maintained, i.e., there is sufficient friction and stiffness at the constraint locations to prevent motion.
2) The system is not over-constrained: The matrix $\mathbf{J}_C$ should remain full row rank, i.e., every constraint can be simultaneously satisfied.
3) The system is sufficiently constrained to eliminate under-actuation: if we divide the constraint Jacobian into its parts ($\mathbf{J}_C = \left[\begin{array}{cc} \partial\mathbf{x}_C/\partial\mathbf{q}_r & \partial\mathbf{x}_C/\partial\mathbf{x}_b \end{array} \right]$), the constraint Jacobian related to base motion ($\partial\mathbf{x}_C/\partial\mathbf{x}_b$), must have a rank equal to 6.

It is important to note that while these assumptions are needed to ensure that the calculated torques are *analytically* correct they are not numerically critical. This means even when some of the assumptions are violated (which is likely to happen on a real system), the algorithms are still numerically stable. This means while the torques will not be 100% accurate they can not grow to infinity. They remain meaningful and useful for practical purposes. Refer to [7] for further discussion of these assumptions. Such graceful degradation in respect to the violation of the assumptions together with avoiding the inversion of inertia matrices makes our method applicable to *real* systems.

B. QR decomposition of the constraint Jacobian

Condition 2) and $Rank(\mathbf{J}_C) = k$ allows us to compute the $\mathbf{QR}$ decomposition of $\mathbf{J}_C^T$

$$\mathbf{J}_C^T = \mathbf{Q}\left[\begin{array}{c} \mathbf{R} \\ \mathbf{0} \end{array} \right] \tag{3}$$

where $\mathbf{Q}$ is orthogonal ($\mathbf{Q}\mathbf{Q}^T = \mathbf{Q}^T\mathbf{Q} = \mathbf{I}$), and $\mathbf{R}$ is an upper triangle matrix of rank k. Additionally, if $\mathbf{R}$ is constrained to have all positive diagonal elements, $\mathbf{Q}$ and $\mathbf{R}$ are unique.

It can then be shown [4] that the analytically correct inverse dynamics can be written as

$$\tau = \left(\mathbf{S}_u\mathbf{Q}^T\mathbf{S}^T\right)^+ \mathbf{S}_u\mathbf{Q}^T\left[\mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{h}\right] \tag{4}$$

where

$$\mathbf{S}_u = \left[\begin{array}{cc} \mathbf{0}_{(n+6-k)\times k} & \mathbf{I}_{(n+6-k)\times(n+6-k)} \end{array} \right] \tag{5}$$

and $(.)^+$ is the right pseudo-inverse ($A^+ = A^T\left(AA^T\right)^{-1}$).

Eq. (4) does not depend on the contact forces and produces the analytically correct inverse dynamics torques that will realize the desired joint accelerations $\ddot{\mathbf{q}}_d$. This control law is just a projection of the floating-base inverse dynamics (the term in square brackets), which can be computed with efficient algorithms that scale linearly with the number of the degrees-of-freedom of the robot [8].

III. Compliant low gain locomotion control for a quadruped robot

A. Platform

While the recent decades have seen a lot of research on walking robots and gait generation, only a few projects focus on the archetypical domain of legged locomotion, i.e. maneuvering in truly complex terrain. This fact is a bit surprising since it is only in complex terrain where legged locomotion gains advantages over wheeled locomotion. Thus, while a lot of work commonly cites these facts in the research motivation, little research is done towards robust solutions for complex terrain walking. Our Learning Locomotion project differs in that walking over rough terrain is its primary focus.

Our experimental setup consists of the LittleDog quadruped robot (Fig. 1) manufactured by Boston Dynamics. It is about 0.3m long, 0.18m wide, and 0.26m tall, weighs approximately 2.5kg, and has 3 degrees of freedom per leg. Joint angle references, force controller gains and feedforward commands are sent to the robot from a Linux host computer over a wireless connection at 100Hz, and the on-board low level controller servos each actuator using PD, feedforward and force control at 400Hz.

LittleDog has a 3-axis force sensor on each foot, position sensors to measure joint angles, and an on-board inertial measurement unit (IMU). An external motion capture system (VICON) provides information about the absolute world position and orientation of the robot.

The robot has to overcome steps of more than 10cm height and gaps of more than 15cm width (i.e. in the order of magnitude of the leg length of the robot). The goal of this project is to cross a variety of terrain, e.g. slopes, jersey barriers, rocks with cracks, round rocks, and logs, while reaching a goal state as fast as possible.

B. Control law & Dynamics model

To realize the inverse dynamics controller we use a standard control law that combines the feedforward torque computed based on the desired acceleration $\ddot{\mathbf{q}}_d$ and the current state of the robot $\mathbf{q}, \dot{\mathbf{q}}$ with a negative feedback correction computed via a standard PD controller:

$$\tau = \tau_{\text{ID}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_d) - \mathbf{K}_P\left(\mathbf{q} - \mathbf{q}_d\right) - \mathbf{K}_D\left(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d\right) \quad (6)$$

where τ_{ID} is inverse dynamics feedforward term computed using Eq. 4. $\mathbf{K}_P > 0$ and $\mathbf{K}_D > 0$ are the proportional and differential gain respectively. The dynamics model uses a mix of CAD based data and estimated parameters.

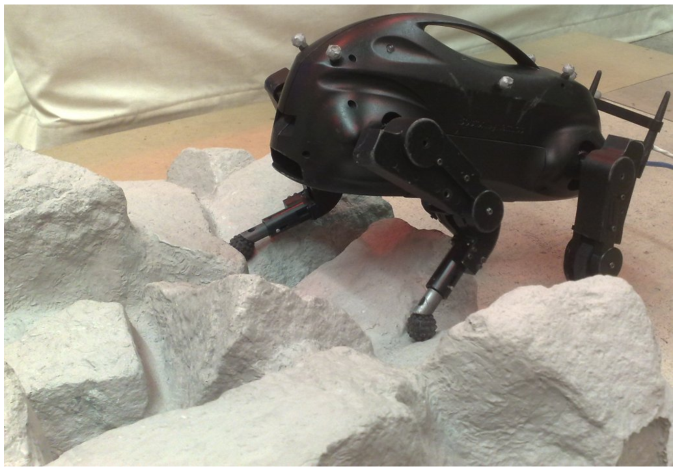


Fig. 1. The quadruped robot LittleDog and a sample terrain to be crossed.

Force control: We can use the inverse dynamics to predict the contact forces at the stance feet (cf. [4]):

$$\lambda = \mathbf{R}^{-1}\mathbf{S}_c\mathbf{Q}^T\left[\mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{h} - \mathbf{S}^T\tau\right] \quad (7)$$

where

$$\mathbf{S}_c = \left[\begin{array}{cc} \mathbf{I}_{k\times k} & \mathbf{0}_{k\times(n+6-k)} \end{array}\right] \quad (8)$$

Since we do not rely on measured contact forces to compute the inverse dynamics, we can instead use the predicted contact forces (Eq. 7) to add a force control term to our controller:

$$\tau_F = \mathbf{J}_F^T\left(\mathbf{F}_m - \lambda\right) \quad (9)$$

where $\mathbf{F}_m$ are the measured foot forces and $\mathbf{J}_F^T$ is the Jacobian transpose of the foot forward kinematics.

This term will in case of unexpected terrain contact counteract the PD controller, thus having the robot not continuing to push into the terrain with all force. This active compliance makes the behavior more robust as will be shown in the results.

IV. Trajectory generation

Inverse dynamics requires a desired joint trajectory $\mathbf{q_d}, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$. In order to get satisfactory performance the trajectory $\mathbf{q}$ should be twice differentiable, i.e. should have a continuous acceleration profile. Furthermore, the joint trajectory should be compatible with the constraints from the environment, or, more technically, should be in the null-space of the constraints. These requirements arise naturally in our case by having three of four stance legs in the different phases of walking, which do not move in task space, i.e., relative to the ground. However to get a completely defined joint-space trajectory we need to complete the leg motion with a COG motion.

A. Path planning, Foothold search

For terrain of medium to high difficulty (comparable with the difficulty of humans scrambling over large boulders or rock climbing) locomotion becomes foothold oriented. In other words, it is crucially important that the robot is able to choose and reach good footholds as otherwise it might slip, fall into cracks, or roll over etc. The COG is a secondary planning objective, which, given the footholds, has to satisfy the stability (force, moment distribution) requirements. We assume that a COG path can be planned given the step-sequence, i.e. it is possible to find a path that is kinematically feasible.

In our approach, a series of steps is planned first based on the estimated quality of footholds and kinematic constraints, and subsequently, a COG plan is realized which is in line with the constraints imposed by the foothold sequence.

In this paper, we do not focus on the problem of finding the footholds. Foothold selection is explained in more detail in another contribution [9]. We assume that we are already given a feasible step sequence.

B. *COG trajectory generation*

In the following we explain how the COG path is planned for a given step sequence. This path has to satisfy the condition of differentiability (as above) and kinematic feasibility.

While kinematic feasibility is naturally satisfied with joint-space oriented trajectory generation methods (e.g. CPG/oscillator based), it is hard for this methods to satisfy the task space goals (footholds) and constraints (i.e. a trajectory that is in the null space of the constraints). Constraint incompatible trajectories often introduce discontinuities when the trajectory gets projected into the null-space of the constraints, and discontinuities cause slipping and falling.

Therefore, for our task and platform we opt for a task space oriented finite state machine approach.

For statically stable walk the usual way to address stability is to examine the polygons formed by the stance legs [10]. As long as the center of pressure (which in the quasi static regime coincides with the projection of the COG) lies in the support polygon the robot is statically stable. In static walk, at any moment of time, either 3 or 4 feet are in ground-contact and act as stance legs. This means in case a leg is swinging, the support polygon is a triangle, usually called the support triangle. COG planning for a statically stable walk therefore involves finding a path for the COG that traverses the sequence of support triangles in an efficient and perturbation-robust manner.

There are a total of six swing possible leg sequences. However, only one sequence satisfies the requirement that it is possible to move forward (and not backward) with the COG at *all* times (given a minimal step length). This sequence is [LH,LF,RH,RF], i.e. on each side of the robot, first the hind leg swings, then the front leg, then the other side's hind leg, and finally the other side's front leg. Therefore, we chose this pattern for steady state operation, i.e. as long as no larger event requires a complete switch of movement strategy.

In order to get a *robust* plan, a safety margin is applied to the support triangles (see [11] for discussion of different margins), which in essence reduces the usable area of the support triangle. Reducing the area of the support triangles also leads to the fact that support triangles might become disjoint, requiring a four leg support phase in which we have a support polygon spanned by the four feet. The margins can also account for the fact that the COG is only an approximation of the COP, to avoid a full ZMP or a even more involved dynamic model based planning scheme [11].

A line based COG path trajectory generator: To plan a fast, efficient path where the COG never has to move back in such a sequence we can make use of the intersection of two subsequent support triangles of hind and front swing. The intersection of this triangle is called a "double support triangle" (DST) [10], [12] and is often used in COG path planning for static walk. To make use of the DST we need to look ahead four steps (to get two subsequent DST).

While indeed the support polygon determines the stability of a walking gait, as we will show, we do not have to consider the complete polygons to plan a fast COG trajectory for complex terrain locomotion. This insight reduces the complexity of the algorithms tremendously, in particular, there is no numerical optimization needed as in, for instance, most ZMP approaches. The method we will present also extends and generalizes our earlier work on a simple COG trajectory planner for rough terrain [13].

In the following we will show how to develop the line-based COG trajectory generator, starting from the traditional support triangles and polygons. We will show that as long as there are no large perturbation requiring special recovery behaviors, the only relevant elements to plan a COG trajectory (that supports fast movements) are lines defined by diagonal pairs of legs. This reduces the complexity of the COG planning algorithms and especially does not require memorization a history of past footholds.

First of all let us state important design aspects for the COG trajectory:

- Avoid unnecessary accelerations and jerks: The COG should move at near constant velocity for fast and smooth locomotion with minimal slipping. This requires proper velocity boundary conditions.
- Avoid unnecessary movements (such as moving back as in figure-8 patterns) – This requirement is achieved by using the equivalence of a DST. However, as we will show, we will not have to compute the double support triangles explicitly.
- Optimal sway (velocity dependent) – The faster the COG is moving the more the path is straightened out (see calculation of v_y below).
- Appropriate safety margins (velocity/acceleration dependent) – The margins facing towards the next and previous stability triangle are reduced proportionally to the velocity, to account for the fact that the momentum of the body is pushing the COG into the next support triangle. The sideways margins remain constant (and are normally not directly relevant in a steady gait as shown below).

In the following discussion we assume a coordinate system that is fixed on the robot, the x-axis points in the forward direction and the y-axis to the left. Furthermore to label feet in relation to the current swing leg we designate the side on which the current swing leg is with I (ipsilateral) and the opposite as C (contralateral) concatenated with the standard

F, H for front leg/hind leg respectively. The “diagonal cross over point” (DCO) is used to label the point where the COG would cross the diagonal in case of zero stability margin. SI and SO label the points in which the COG enters the (velocity dependent) stability region or leaves it respectively.

Figure 2, develops the idea of the line based stability criterion. Please refer to the figure caption for an explanation.

Finally, we present an algorithm that uses the ideas introduced in Fig. 2 to generate a COG path. Parameters of the algorithm are: $\overline{v}_x$: average desired forward velocity, m_s: “static” stability margin, g: gain to modify the stability margin dependent on velocity – g also influences the velocity boundary conditions. Given the step sequence and the parameters, the algorithm proceeds as follows:

1) Calculate next cross over point – The mid point on the IH-CF between the intersection of this line with the next IF-CH and previous IF-CH yields the next DCO (Fig. 2d, star).
2) Calculate time needed for hind and front swing based on the distance of the last and next DCO. $t_s = \frac{||(DCO(n),DCO(n+1))||}{\overline{v}_x}$
3) Apply velocity dependent stability margins by moving the IF-CH and IH-CF lines by $m = m_s - g|v_x|$, if $m < 0 \rightarrow m = 0$ (cf. Fig. 3).
4) Apply velocity dependent y-velocity boundary conditions $|v_y| = |v_x|(1 - g|v_x|/m_s)$,if $|v_y| < 0 \rightarrow |v_y| = 0$.
5) Calculate SI and SO – To calculate these points a line with the direction (v_x, v_y) through DCO is intersected with the relevant line (SI: IH-CF or SO: IF-CH), modified by the stability margins (from Step 3). These points are taken as SI and SO.
6) Spline the movement from the SI to SO. (For four leg support a spline segment from SO to SI is used).
7) Find the mid point of the spline at $\frac{t_s}{2}$ to separate between hind swing and front swing.

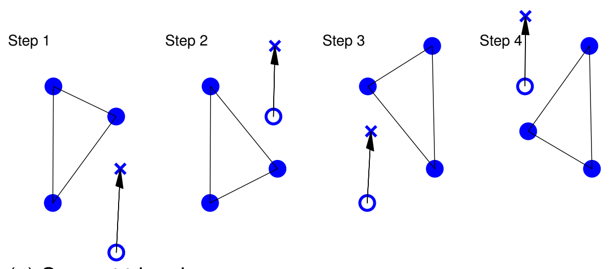
We use splines to connect the critical points (SI,SO) and generate the final path for each control cycle of the robot. Fifth order splines are used to ensure continuous acceleration profiles required by the inverse dynamics controller, although other methods could be used.

Swing leg trajectory: While the COG path together with the stance legs positions (from the step sequence) define the trajectories for the DOFs of stance legs (three DOFs per leg for our robot), the swing leg has to be specified separately. There are several constraints that have to be fulfilled, including to avoid impact with the terrain, appropriate timing, and velocity constraints. We use a simple convex-hull computed over the obstacles of the terrain and a spline based swing leg trajectory.

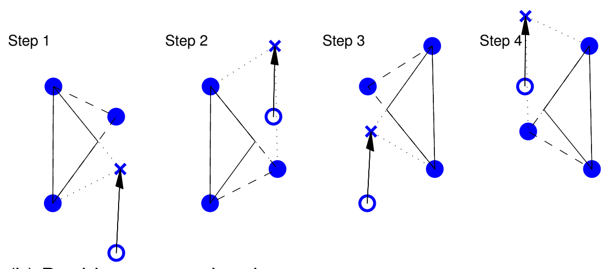
V. Experimental Results

A. Improvement of robustness over non-perceived obstacle

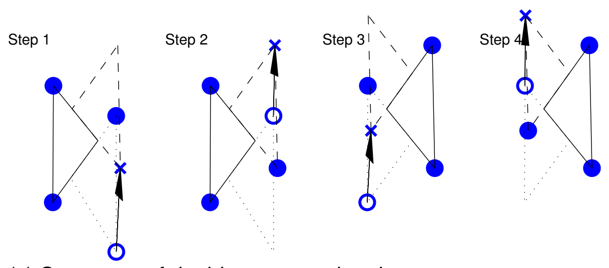
A total of 12 different controllers were tested in a physics based simulator [14]. Three different gain settings (HG) high gains, (MG) medium gains (half of the HG settings) and (LG) low gains at 1/6 of the HG settings. All of these three gain



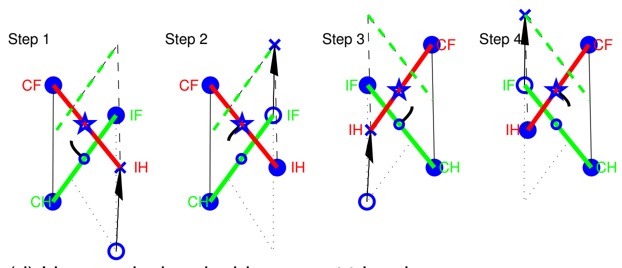
(a) Support triangles



(b) Double support triangles



(c) Sequence of double support triangles



(d) Lines replacing double support triangles

Fig. 2. This figure develops step-by-step the idea of using the diagonal feet pairs to replace the double support triangles in calculation of the COG path. All rows show the same four footsteps of a typical step sequence [RH,RF,LH,LF], see text for definition of labels – (a) The hollow circle depicts the origin and the cross the goal of the swing leg. The solid circles depict the stance legs and they form the support triangle, illustrated with solid lines. (b) The intersection of two subsequent unilateral (hind then front) swing leg movements yields the double support triangle (solid line, hind leg support dashed, front leg support dotted). (c) The double support triangles are drawn (solid) with the previous (dotted) and following (dashed) double support triangle. A path through these triangles will allow the COG to always have a forward movement. (d) The diagonal feet pairs yield the critical lines, which hold the same information as the full double support triangles. IH-CF yields the forward limit (red line), IF-CH yields the rearward limit (green line), i.e. the COG must be ahead of the green line and not cross the red line. The next IF-CH diagonal (green dashed) determines the sideways margins – this line is where the *same* foot pair will be two steps later, i.e. the red line needs to be crossed behind the green dashed lines, the middle point of this segment yields the largest achievable stability margin (stars) and is taken as the DCO. The critical diagonals with an example COG path are drawn in Fig. 3 a)

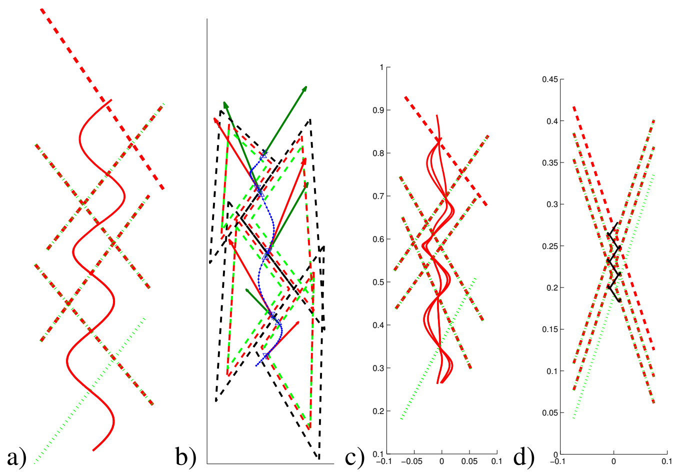


Fig. 3. (a) A typical path corresponding to the footsteps in Fig. 2. (b) The sequence shown with the velocity dependent margins applied. The arrow indicates the velocity used for the spline boundary conditions. (c) Paths found by holding all parameters constant but but changing the desired velocity. The faster the COG travels the straighter the path gets. (d) For short steps the algorithm finds naturally to the figure eight pattern [13] since satisfying the stability constraints in combination with short footsteps requires the COG to either move backward or come to halt (note the different scale).

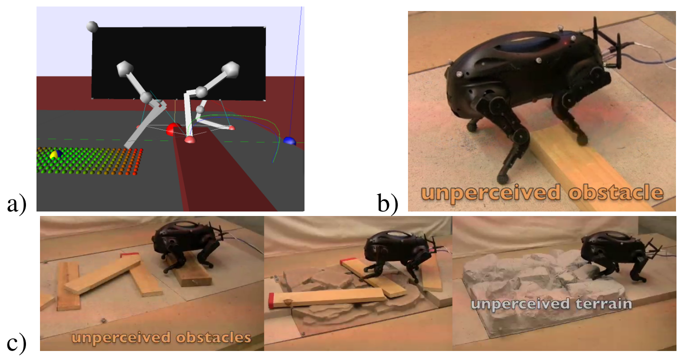
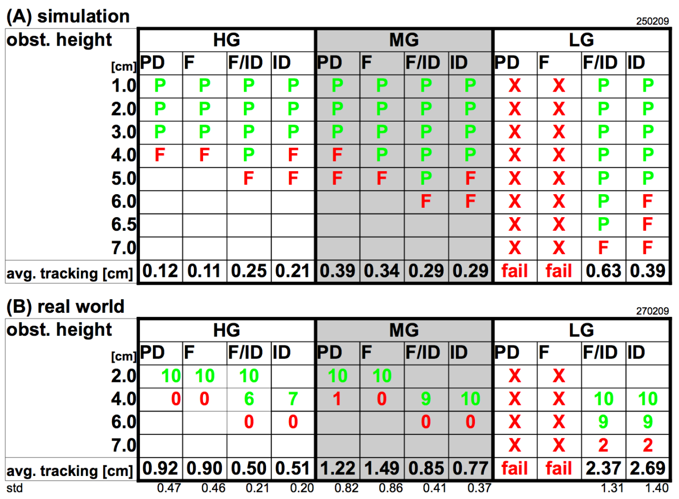


Fig. 4. Setups to test the robustness of the controllers towards non-perceived obstacles (a) simulation (b) real world. The brown respectively wooden obstacle are not perceived by the robot. (c) Terrains to assess the performance of the controller on more realistic setups, see video supplement.



(A) simulation 250209

obst. height [cm]	HG PD	HG F	HG F/ID	HG ID	MG PD	MG F	MG F/ID	MG ID	LG PD	LG F	LG F/ID	LG ID
1.0	P	P	P	P	P	P	P	P	X	X	P	P
2.0	P	P	P	P	P	P	P	P	X	X	P	P
3.0	P	P	P	P	P	P	P	P	X	X	P	P
4.0	F	F	P	F	F	P	P	P	X	X	P	P
5.0			F	F	F	F	P	F	X	X	P	P
6.0							F	F	X	X	P	F
6.5									X	X	P	F
7.0									X	X	F	F
avg. tracking [cm]	0.12	0.11	0.25	0.21	0.39	0.34	0.29	0.29	fail	fail	0.63	0.39

(B) real world 270209

obst. height [cm]	HG PD	HG F	HG F/ID	HG ID	MG PD	MG F	MG F/ID	MG ID	LG PD	LG F	LG F/ID	LG ID
2.0	10	10	10		10	10			X	X		
4.0	0	0	6	7	1	0	9	10	X	X	10	10
6.0			0	0			0	0	X	X	9	9
7.0									X	X	2	2
avg. tracking [cm]	0.92	0.90	0.50	0.51	1.22	1.49	0.85	0.77	fail	fail	2.37	2.69
std	0.47	0.46	0.21	0.20	0.82	0.86	0.41	0.37			1.31	1.40

TABLE I

Tracking results and results of robustness step on non-perceived obstacles for different controller types. (HG) high gains (MG) medium (half) gains (LG) low (1/6) gains. Control laws: (PD) PD position control only (F) PD + force control (F/ID) PD + force and inverse dynamics (ID) PD + inverse dynamics. Bottom row, avg. foothold tracking. Top panel: Simulation results (Pass/Fail) - Bottom panel: real robot, number of passed trials out of 10.

settings were run with four different control laws: (PD) PD position control only, (F) PD + force control, (F/ID) PD + force and inverse dynamics, and (ID) PD + inverse dynamics.

All controllers were subjected to an unperceived obstacle of variable height (cf. Fig. 4) from 1cm to 7cm (corresponding to up ∼50% of the leg length). Table I lists the results. A P denotes passing the obstacles without falling and F denotes failure and falling over. The experiments were repeated 10 times and despite the simulator having stochastic elements due to the "penalty method" employed to model ground contacts, the results were consistently either pass or fail for a given setup and obstacle height.

While lowering the gains (MG) already gives higher robustness, it is not strictly mandatory to use inverse dynamics with these gain settings. The robot fulfills the task pretty well with PD control only. However, in the case of LG, the PD only (and PD/F) control is not able to track well enough that the robot is even able to take a single step (cf. tracking results). To lower the gains that much inverse dynamics is the crucial element to achieve even basic task-fulfillment. From the last two columns in the table we see that the lower gains achieve the highest tolerance for non-perceived obstacles, with the combination of ID/F control giving the best performance.

As outlined above in our case reaching the footholds is of utmost importance. Hence, the tracking quality was assessed by evaluating the average distance of the swing foot from the desired foothold at touch-down and the results are listed in the last row in Table I.

Looking at the tracking results it is clear that with reducing the gains, as expected, we reduce the accuracy but at the same time gain a large amount of robustness in presence of not or wrongly perceived obstacles. The reduction of foothold tracking is still acceptable. The benefits in robustness thus justify the use of such a low gain inverse dynamics and force controller.

B. Real world tests

The same tests were repeated on the real robot. Table Ib) shows the results. The results match the predictions of the simulation very well. Again, the LG-F/ID controller is the most robust one. However, in contrast to the simulation the LG-ID controller seems to perform equally well as the combined LG-F/ID controller. The tracking performance degrades more significantly in the low gain condition, although increased robustness is still observed in the same way as in the simulation tests. It should be noted that our inverse dynamics model was not of high quality – it was partially derived from CAD data, and partially from parameter estimation techniques, and we noticed various deficiencies in the quality of the model. Future work will address how to improve this issue, and tracking performance in low gain control will be improved.

In order to demonstrate the usefulness of the LG-F/ID controller in more realistic scenarios we ran the real robot

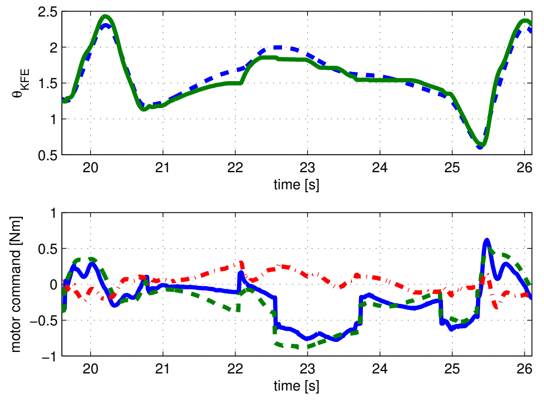


Fig. 5. Top: Tracking of the Right hind knee joint in a steady walk; Bottom: The total motor torques τ (blue line) comprised of the feedforward τ_{ID} (green dashed) and the feedback command $\tau_F B$ (red dash-dotted). The feedforward is the major contributor to the motor command most of the time and the feedback torque is comparatively very low. The effects of the constraint switches are clearly visible as they create discontinuities in τ_{ID}.

over non-perceived scattered obstacles of 2-4cm height and a non-perceived rock board (Fig. 4). In such terrain the HG-PD controller fails while the LG-F/ID controller create enough robustness for successful completion of the runs (see video supplement or [15]). These results show that the efficiency of ID/LG also holds on the real robot despite an imperfect model.

C. Joint space tracking results

In Figure 5, we show a representative example of the tracking of the knee joint along with the total motor command trade-off in terms of feedback command vs. feedforward command. Typically, it can be seen that the total motor command is mostly accounted for by the feedforward contribution. However, while in theory the feedback command should be almost negligible, in the real robot we have a continued clear contribution of the feedback controller. This is due to the imperfect dynamics model of the robot, and, therefore, the feedforward command is not able to track the desired trajectory perfectly. It is worth noting that the floating-base inverse dynamics controller has a graceful degradation despite a rather imprecise model of the inverse dynamics parameters. Other approaches, which require inversion of the rigid body dynamics inertia matrix [6] can create significant problems in face of modeling errors.

VI. Conclusion and Discussion

We have shown how to achieve floating-base inverse dynamics control on arbitrary robots with multiple and dynamically changing constraints. The controller has been applied to a quadruped robot to implement an inverse dynamics and force based controller to achieve robust and compliant locomotion over rough, unperceived and moving terrain. We have introduced a novel, simple and efficient way of computing a COG path given a foothold sequence for a statically stable walk over uneven terrain (i.e. varying step length and relative foothold position).

We have shown that the combination of inverse dynamics and force control greatly enhances the robustness against non-perceived obstacles. Results in simulation and on a real quadruped robot show the feasibility and effectivity of the controllers.

Future work: Future work will show the application of these control principles to other robots such as humanoid robots in complex terrains and a combination with other tasks such as manipulation, carrying payloads, etc. An additional advantage of low gain control is increased safety – a crucial design aspect for strong robots to operate among humans. Our approach can help implementing controllers needed to improve safety considerations in such scenarios.

Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, the DARPA program on Learning Locomotion, NASA grant AC98-516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Computational Neuroscience Laboratories. J.B. was supported by a prospective researcher fellowship from the Swiss National Science Foundation.

References

[1] U. Saranli, M. Buehler, and D. Koditschek, "RHex – a simple and highly mobile hexapod robot," *International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[2] T. McGeer, "Passive dynamic walking," *International Journal of Robotics Research*, vol. 9, pp. 62–82, 1990.

[3] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient Bipedal Robots Based on Passive-Dynamic Walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[4] M. Mistry, "The representation, learning, and control of dexterous motor skills in humans and humanoid robots," Ph.D. dissertation, University of Southern California, 2009.

[5] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. Springer, 2000.

[6] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational space control: A theoretical and emprical comparison," *International Journal of Robotics Research*, no. 6, pp. 737–757, 2008. [Online]. Available: http://www-clmc.usc.edu/publications/N/nakanishi-IJRR2008.pdf

[7] M. Mistry, J. Nakanishi, G. Cheng, and S. Schaal, "Inverse kinematics with floating base and constraints for full body humanoid robot control," in *IEEE International Conference on Humanoid Robotics*, 2008.

[8] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, New York, 2007.

[9] M. Kalakrishnan, J. Buchli, and S. Schaal, "Learning locomotion on rough terrain using terrain templates," in *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2009, in print.

[10] R. McGhee and A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, no. 3–4, pp. 331–351, 1968.

[11] P. Gonzales de Santos and E. J. Garcia, E., *Quadrupedal Locomotion*. Springer London, 2006.

[12] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "A control architecture for quadruped locomotion over rough terrain," in *Proceedings IEEE Int. Conference on Robotics and Automation*, 2008, pp. 811–818.

[13] D. Pongas, M. Mistry, and S. Schaal, "A robust quadruped walking gait for traversing rough terrain," in *International Conference on Robotics and Automation (ICRA 2007)*, 2007, pp. 1474–1479.

[14] S. Schaal, "The sl simulation and real-time control software package," Tech. Rep., 2009. [Online]. Available: http://www-clmc.usc.edu/publications/S/schaal-TRSL.pdf

[15] Videos available at http://www-clmc.usc.edu.