Virtual Model Control of a Bipedal Walking Robot

Jerry Pratt
MIT Leg Laboratory, Cambridge, MA 02139
http://www.ai.mit.edu/projects/leglab/

Abstract

The transformation from high level task specification to low level motion control is a fundamental issue in sensorimotor control in animals and robots. This paper describes a control scheme called virtual model control which addresses this issue.

Virtual model control is a motion control language which uses simulations of imagined mechanical components to create forces, which are applied through real joint torques, thereby creating the illusion that the virtual components are connected to the robot. Due to the intuitive nature of this technique, designing a virtual model controller requires the same skills as designing the mechanism itself. A high level control system can be cascaded with the low level virtual model controller to modulate the parameters of the virtual mechanisms. Discrete commands from the high level controller would then result in fluid motion.

Virtual model control has been applied to a physical bipedal walking robot. A simple algorithm utilizing a simple set of virtual components has successfully compelled the robot to walk continuously over level terrain.

1 Introduction

Design and analysis of control systems for nonlinear systems such as robots seems to be much more difficult than for linear systems. A particular class of robots which suffers from lack of powerful techniques for its control is dynamic legged locomoting robots. These robots are extremely difficult to control since they are nonlinear and operate throughout the range of their state space; act in a gravity field; interact with a semi-structured, complex environment; are nominally unstable; are Multi Input, Multi Output (MIMO); exhibit time variant and intermittent dynamics; and require both continuous control and discrete control (for step-to-step transitions).

In addition, the performance measures of such robots are much different than typical notions of performance such as command following and disturbance rejection. Performance for these robots is usually defined in terms of biological similarity, efficiency, locomotion smoothness, top speed, and robustness to rough terrain.

Because of these difficulties, the only acceptable tools for analyzing such systems are often simulation or experimentation and the only good design tools are often physical intuition, parameter iteration, and "hand tweaking".

Anyone who wishes to expand the toolbox of analysis and design techniques for such a class of robots will typically either make a large advancement in control system design and analysis mathematics or exploit physical intuition. In this paper we present a control technique, dubbed virtual model control, which attempts the latter.

1.1 Virtual Model Controllers

Virtual model control is a language for describing interactive force behaviors. This control technique uses simulations of virtual mechanical components to generate real actuator torques (or forces), thereby creating the illusion that the simulated components are connected to the real robot. Because any imaginable component can be used, virtual model control requires no loss of generality, and traditional control techniques can be utilized in the same context. For example, a proportional-derivative (PD) servo at a joint can be implemented as a virtual torsional spring-damper with spring constant equal to servo proportional gain, damping constant equal to derivative gain, and set point equal to desired joint angle. Virtual components can even contain adaptive and learning elements [9].

One of the reasons virtual model control is being developed is that motion control is generally difficult to describe. For example, how would one describe what he or she is doing when performing deep knee bends, and how would one control a robot to perform this task? A quick and easy method would be to attach a virtual spring between the robot's feet and its waist and change the rest-length in a sinusoidal fashion.

Another complex task which is difficult to describe using traditional control methodologies is the throwing of a punch by a robotic boxer. A simple virtual model controller might consist of a virtual force source connected between the robot's chest (to throw a jab) and the end of its hand, producing a virtual force in the direction of the opponent. Virtual components could be connected between the feet and the body to help maintain balance (see Figure 1). If the robot wanted to throw an upper-cut using everything it has, virtual actuators could be attached to its hand from its feet, hip, and chest. Whereas the jab controller would only cause chest and arm torques to be exerted, the upper-cut controller would use every available joint torque in the robot's body.

Some benefits of virtual model control are that it is compact, requires relatively small amounts of computation, and can be implemented in a distributed manner (see [10, 8] for information on how to imple-

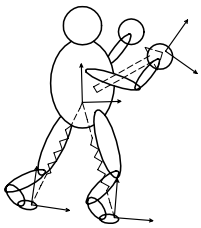

Figure 1: Robotic Boxer with virtual components attached from the feet to the body to maintain balance and from the body to the hand to throw a jab.

ment virtual components). These benefits are due to the fact that no matrix or function inverses need be computed for serial links; all equations can be precomputed in closed form and optimized; and multiple virtual components can be independently computed and superposed since their outputs, joint torques, are linearly additive.

Furthermore, a high level controller could be implemented as a state machine which simply changes virtual component connections or parameters at the state transitions. Even though a discrete high level controller would be used, the overall motion would be fluid since the virtual components are continuous.

Note that virtual model control does not use dynamic inversion in an attempt to alter the behavior of the robot. We like to call dynamic inversion the "virtual robot" approach. We believe that the virtual robot approach should only be used when high performance requirements or other extreme situations dictate. This is because plant inversion adds computational complexity; fighting the natural dynamics of the robot can be inefficient; undesirable natural dynamics is an indication that the real robot was designed improperly; and overcompensation can lead to instability.

Also note that with virtual model control, we usually talk in terms of spring set points, for example, and not *commanded* positions. Except for actuator nonidealities, we can perfectly implement virtual components whereas very few control algorithms can perfectly track a commanded trajectory. In this light we in the Leg Laboratory believe that robots cannot be *commanded* to perform a task; they can only be given *hints* and *suggestions*.

Virtual model control has been used to control a dynamic walking bipedal robot (described below) and an agile 3D hexapod in simulation [13].

2 Bipedal Walking Robot

Figure 2 is a photograph of Spring Turkey, our bipedal walking robot. Spring Turkey was designed and built by Peter Dilworth and Jerry Pratt in 1994. It has an actuated hip and knee on each leg. An unactuated boom constrains Spring Turkey's roll, yaw, and

Figure 2: Spring Turkey, our bipedal walking robot. There are four actuators attached to the body. Power is transmitted to the hips and knees via cables. The unactuated feet consist of a strip of rubber. A boom is used to prevent motion in the lateral, roll, and yaw directions. Note the spring packs which are used to implement series elastic actuation.

lateral motion thereby reducing it to a planar robot. All of Spring Turkey's motors are located in its upper body, with power being transmitted to the joints via cable drives. Series Elastic Actuation [7] is employed at each degree of freedom, allowing for accurate application of torques and a high degree of shock tolerance. The maximum torque that can be applied to the hips is approximately 12 Nm while approximately 18 Nm can be applied to the knees. The force control bandwidth we achieve is approximately 20 Hz. Spring Turkey weighs in at approximately 22 lbs (10 kg) and stands 2 ft (60 cm) tall.

Potentiometers at the hips, knees, and boom measure joint angles and body pitch. Extension springs are used at the hips and knees to implement Series Elastic Actuation [7]. Linear potentiometers measure the stretch in the springs.

2.1 Turkey Walking

Virtual model control was applied to Spring Turkey to allow it to perform simple walking. The algorithm is as follows:

- Maintain a constant height and pitch.
- Transition from double support to single support if the body's x position becomes close to the point where a foot contacts the ground.
- Transition from single support to double support if the body's x position becomes far away from the support foot's ground contact point.
- Swing the non-stance leg so that the foot is placed a nominal stride length away from the support foot when transitioning to double support.
- During double support correct for velocity disturbances.

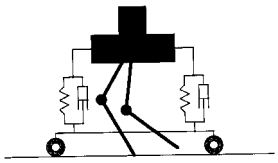

Figure 3: Spring Turkey with virtual granny walker mechanism.

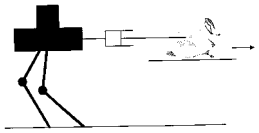

Figure 4: Spring Turkey with virtual dog track bunny mechanism.

To implement turkey walking, we use a simple set of virtual components and a state machine. Figure 5 shows the state machine and Table 1 lists the trigger and branch events and the virtual components which are utilized in each state. During both double support and single support, a virtual granny walker with spring-damper mechanisms (Figure 3) maintains a constant height and regulates the pitch angle to zero. During double support, a virtual dogtrack bunny with damper mechanism (Figure 4) applies a virtual force in the forward horizontal (x) direction to help maintain a desired velocity. The swing leg is controlled via a virtual linkage with springs and dampers which compel the swing leg to mirror the stance leg while clearing the ground and to set down at the nominal stride length before transitioning back to double support.

States Left Support 2 and Right Support 2 are used as buffer states between single and double support. Because Spring Turkey has no foot switches to detect ground contact, in these states the swing leg is simply made limp (zero torque applied to the joints) for a set delay time, allowing for the swing leg to fall to the ground before the large forces, which double support require, are applied.

The various virtual spring, damper, and force variables and walking parameters were chosen using physical insight and a manual search. The virtual granny walker spring-damper constants were experimentally varied while physically examining their effects (resistance to being pushed on, decay rate, etc.) until the desired effects were achieved; the walking parameters and virtual dogtrack bunny damper were changed through trial and error until the robot successfully walked. These walking parameters consisted of nominal stride length and percent of stride length spent in single support.

Walking was initiated in the single support phase. A slight push was applied to the robot to propel it forward. After the push, no external intervention was required.

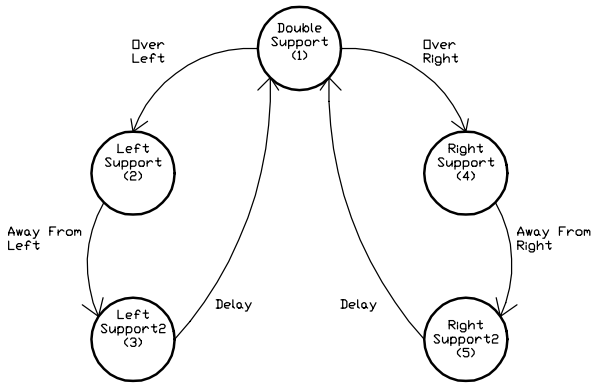

Figure 5: State machine used in the turkey walking algorithm.

Table 1: Details of turkey walking state machine.

State	Trigger Event	Virtual Components
1 Double Support	Delay after left or right support2	Granny Walker Dogtrack Bunnie
2 Left Support	Body nearly over left foot.	Granny Walker Swing Leg Linkage
3 Left Support2	Body away from left foot.	Granny Walker
4 Right Support	Body nearly over right foot.	Granny Walker Swing Leg Linkage
5 Right Support2	Body away from right foot.	Granny Walker

Figure 6 shows experimental data from Spring Turkey while performing turkey walking. The upper left graphs show the body's horizontal position (x), vertical position (z), and pitch (theta), and the corresponding spring set points (dotted). The upper right graphs show the virtual forces applied to the body due to the virtual components. The horizontal velocity, along with the virtual dogtrack bunny velocity (dotted) is plotted in the lower left graph. The state of the state machine is plotted in the lower right graph.

The data in Figure 6 is plotted in graphical form in Figure 7. The snapshots in Figure 7 are approximately 0.5 seconds apart. Lines are drawn to show the path of the tips of the feet and the center of the body.

Spring Turkey walked continuously at approximately 0.5 m/s (1.125 mph). The data shows approximately 6 steps (left to right or right to left support transitions) in 4 sec, giving a step time of 0.5 seconds. It deviated a maximum of 3 cm from the nominal height of 54 cm and pitch was confined to ± 0.10 radians (± 5.2 deg).

2.2 Stupid Walking

Another algorithm which we tried, called stupid walking, was identical to turkey walking except there

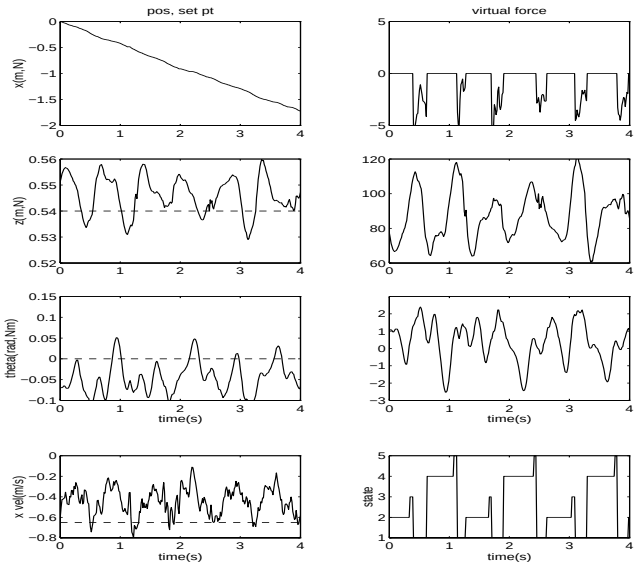

Figure 6: Turkey walking data. Upper left graphs display the x, z, and θ positions and virtual spring set points (dashed). Upper right graphs display the resultant forces applied to the body due to the virtual components. The lower left graph shows the body velocity and the dogtrack bunny velocity (dotted). The lower right graph shows the state machine transitions.

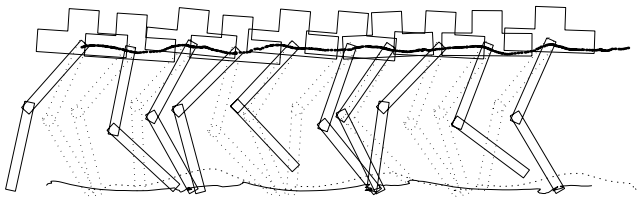

Figure 7: Elapsed time snapshot of the bipedal walking data in Figure 6. The drawings of the robot are spaced approximately 0.5 seconds apart. The left leg is dotted while the right leg is solid. Lines show the path of the tips of the foot and the center of the body.

was no speed control mechanism (no virtual dogtrack bunny with damper). With this algorithm, the state trajectory converged to a stable limit cycle for an appropriate choice of initial conditions and compelled the robot to walk as many as 14 steps. However, it was extremely dependent on initial conditions, ground conditions, virtual component parameters, etc. We have not attempted to perform an analysis on why the stupid walking algorithm converges to a limit cycle for the appropriate initial conditions. We only speculate that mechanisms similar to those present in McGeer's passive dynamic walker [3] and Mochon and McMahon's balastic walking model [6] are in force.

3 Conclusions

Spring Turkey walked continuously through the use of virtual model control techniques. We stress here that we *augmented* the natural dynamics of the robot with simple virtual components, rather than attempted to *cancel* the natural dynamics. In no case did we assume linear dynamics.

The ease of implementing virtual model controllers is promising. One of the major incentives of developing virtual model controllers is to make designing robot control algorithms easier and more intuitive. The algorithm designer is given additional incentive to use virtual model controllers since they require minimal computational resources and are straightforward to derive. One of our goals is to automate this process.

We are hopeful that virtual model control will be useful in producing more robust walking. Future work will focus on developing such algorithms. We are currently designing Spring Flamingo, a bipedal robot similar to Spring Turkey with the addition of feet and actuated ankles. Preliminary simulations using virtual model control have shown that ankles and feet may allow for improved speed control, higher efficiency, and smoother walking.

Acknowledgements

This work was supported in part by the Office of Naval Research, Grant #N00014-93-1-0333.

References

[1] E. Dunn and R. Howe. Towards smooth bipedal walking. *IEEE Conference on Robotics and Automation*, 1996.

[2] S. Kajita and K.Tani. Experimental study of biped dynamic walking in the linear inverted pendulum mode. *IEEE Conference on Robotics and Automation*, 1995.

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

[4] W. T. Miller. Real time neural network control of a biped walking robot. *IEEE Control Systems Magazine*, pages Feb:41–48, 1994.

[5] H. Miura and I. Shimoyama. Dynamic walk of a biped. *International Journal of Robotics Research*, 3(2):60–74, 1984.

[6] Simon Mochon and Thomas A. McMahon. Ballistic walking: An improved model. *Mathematical Biosciences*, 52:241–260, 1979.

[7] Gill A. Pratt and Matthew M. Williamson. Series elastic actuators. *IEEE International Conference on Intelligent Robots and Systems*, 1:399–406, 1995.

[8] J. Pratt, A. Torres, P. Dilworth, and G. Pratt. Virtual actuator control. *IEEE International Conference on Intelligent Robots and Systems*, 1996.

[9] Jerry E. Pratt. Learning virtual model control of a biped walking robot. Unpublished Project Report for MIT's Machine Learning Class (6.858), 1994.

[10] Jerry E. Pratt. Virtual model control of a biped walking robot. Master's thesis, Massachusetts Institute of Technology, August 1995.

[11] Marc H. Raibert. *Legged Robots That Balance*. MIT Press, Cambridge, MA, 1986.

[12] Shin-Min Song and Kenneth J. Waldron. *Machines That Walk*. MIT Press, Cambridge, MA., 1989.

[13] Ann L. Torres. Implementation of virtual model control on a walking hexapod. May 1996. Undergraduate Thesis, Massachusetts Institute of Technology.