In Proceedings of the 1996 International Conference on Intelligent Robots and Systems (IROS '96)

Virtual Actuator Control

Jerry Pratt, Ann Torres, Peter Dilworth, Gill Pratt
MIT Leg Laboratory, Cambridge, MA 02139
http://www.ai.mit.edu/projects/leglab/

Abstract

Robots typically have an individual actuator at each joint which can result in a non-intuitive and difficult control problem. In this paper we present a control method in which the real joint actuators are used to mimic virtual actuators which can be more intuitive and hence make the control problem more straightforward.

Our virtual actuator control method requires a solution to the force distribution problem when applied to parallel mechanisms. An extension of Gardner's Partitioned Actuator Set Control method is presented. This extended method allows for dealing with constrained degrees of freedom in which the torque cannot be specified but can be measured.

A simulated hexapod robot was developed to test the proposed control method. The virtual actuators allowed textbook control solutions to be used in controlling this highly non-linear, parallel mechanism. Using a simple linear control law, the robot walked while simultaneously balancing a pendulum and tracking an object.

1 Introduction

Due to design constraints, robots typically have actuators at the joint level and hence the actuation space is non-intuitive. For example, it is often not clear how to specify torques at the ankle, knee, and hip joints of a legged robot in order to produce smooth motion of its torso. Control of robots may be made easier if they are equipped with a more intuitive set of actuators. In this paper we develop a technique for mimicking "virtual actuators" on a robot by applying joint torques in such a way that the result on the robot is identical to the result the virtual actuators would induce. For example, we could implement virtual actuators on a walking robot to produce forces directly between each foot and the center of mass. The virtual actuators may produce a more intuitive control space, which can result in a more straightforward control problem.

We describe a method for creating a single virtual actuator out of several serial chains acting in parallel. This method is useful for controlling multi-legged creatures or multi-manipulator work cells. Such systems require a solution to the force distribution problem. We present a solution which is an extension to Gardner's Partitioned Actuator Set Control Technique (PASC) [2, 3]. Our solution allows for the incorporation of actuator constraints such as unactuated degrees of freedom or degrees of freedom in which the torque cannot be commanded but can still be measured or inferred.

We have applied the virtual actuator technique to a simulated hexapod robot. The virtual actuators allowed us to ignore the non-linearities due to the legs and instead treat the hexapod as if it were a space ship with virtual thrusters and reaction wheels. We could then apply simple linear control laws with the virtual actuator generalized forces as the input to the plant. Using this technique, the hexapod walked with a tripod gait while simultaneously balancing a pendulum and tracking an object.

2 Virtual Actuator Implementation

The implementation of virtual actuators is straightforward. There are four major steps: definition of the virtual actuator reference frames, computation of the forward kinematics, calculation of a Jacobian matrix, and computation of the joint torques.

For parallel virtual actuators, one must divide the generalized force among the individual serial paths. This requires solving a system of equations (i.e. inverting a matrix). We describe an extension of Gardner's Partitioned Actuator Set Control Technique method which minimizes the necessary computational requirements. All equation derivation can be performed offline.

2.1 Defining the Virtual Actuator Frames

Each virtual actuator requires three coordinate frames. These are the action frame {B}, the reaction frame {A}, and the reference frame {O} (Figure 1). The action frame defines the virtual actuator connection upon which the generalized forces act. The reaction frame defines the second attachment point of the virtual actuator. The reference frame is the coordinate system in which all displacements, forces, etc are expressed.

In most treatments of similar control methods, the reaction frame is assumed to be fixed to ground and usually is not even mentioned. However, one must remember Newton's Third Law. One cannot simply describe a force *acting at a point*. One must describe forces acting *between two points*. Similarly, in the virtual actuator context, one cannot simply define a generalized force acting on a frame but must define a generalized force acting between two frames.

The action, reaction, and reference frames do not need to be inertial, nor cartesian, nor be directly attached to parts of the physical robot. All three frames

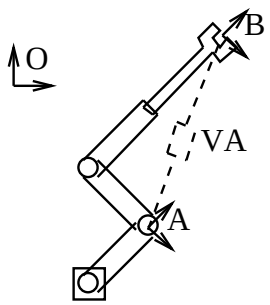

Figure 1: Virtual Actuator Reference Frames. {B} is the action frame. {A} is the reaction frame. {O} is the reference frame. VA represents the virtual actuator.

simply need to be well defined. In most cases, however, all three frames will be cartesian; they will be connected to logical points on the robot such as joints, links, or end effectors; and the reference frame, {O}, will either be inertial or coincide with {A} or {B}.

In our use of the reference frame, {O}, we are not concerned about its location, and in fact it need not be specified. All that is needed is the relative rotation between the reference frame and the action and reaction frames.

Choice of the virtual actuator frames is a major part of defining a virtual actuator and must be done carefully if the desired results are to be obtained. For example, both the action and reaction frames might be defined so that there cannot be a relative rotation between them in any orientation of the robot. This is perfectly valid but it then makes it impossible to produce a relative torque between the two. The best tools for determining how to attach the virtual actuator's frames is physical intuition, insight, and experience.

2.2 Deriving the Forward Kinematics

Computing the forward kinematic map, ${}^A_B\vec{X}$, is well documented [1]. For any serial manipulator with revolute and prismatic joints, ${}^A_B\vec{X}$ will be a closed form function of the joint angles and prismatic displacements, $\vec{\Theta}$, lying between frames {A} and {B}.

To express the kinematics between frames {A} and {B} with reference to frame {O}, we use the rotation matrix between {O} and {A}

$$ {}^O({}^A_B\vec{X}) = {}^O_A R \; {}^A_B\vec{X} \tag{1} $$

Note that we ignore the displacement between {O} and {A}. This is because we are concerned with the relative kinematics between the reaction and action frames and not the absolute location of any of the frames, with reference to {O} or World Coordinates.

All virtual actuators do not necessarily need to provide actuation. They can be used with the forward kinematics alone and act simply as virtual sensors.

2.3 Deriving the Jacobian Matrix

Let the Jacobian Matrix for a virtual actuator be defined as:

$$ {}^A_B J = \frac{\partial}{\partial \vec{\Theta}} \, {}^A_B\vec{X} \tag{2} $$

The Jacobian can be used for several mappings. It can be used to calculate generalized velocities by

$$ {}^A_B\dot{\vec{X}} = {}^A_B J \; \dot{\vec{\Theta}} \tag{3} $$

which holds by definition. To express velocities with respect to the reference frame, we again use the rotation matrix,

$$ {}^O({}^A_B\dot{\vec{X}}) = {}^O_A R \; {}^A_B\dot{\vec{X}} \tag{4} $$

The Jacobian is also used to transform generalized forces to joint torques as discussed next.

There are several techniques to compute the Jacobian for the 3D case [6, 1]. One method described in [1] is to recursively compute the joint to cartesian velocity relationship (Equation 3) and extract the Jacobian Matrix.

2.4 Computing the Joint Torques

To compute the joint torques which will successfully emulate the virtual actuator, we use the following equation

$$ \vec{\tau} = {}^A_B J^T \; {}^A_B\vec{F} \tag{5} $$

where $\vec{\tau}$ is the vector of joint torques (or forces for prismatic joints) and ${}^A_B\vec{F}$ is the generalized force vector acting on action frame {B} from reaction frame {A} defined with respect to frame {A}. The generalized force vector will typically consist of a force and moment vector.

Equation 5 can be derived starting from the energy balance $\vec{\tau}^T \delta\vec{\Theta} = \vec{F}^T \delta\vec{X}$ [1]. It requires that the generalized forces which act on action frame {B} be specified in terms of reaction frame {A}. If the forces are expressed in reference frame {O}, we must use the rotation matrix from {A} to {O} to express them in {A},

$$ {}^A_B\vec{F} = {}^A_O R \; {}^O({}^A_B\vec{F}) \tag{6} $$

We can combine Equations 5 and 6 to get

$$ \vec{\tau} = {}^A_B J^T \; {}^A_O R \; {}^O({}^A_B\vec{F}) = {}^O({}^A_B J)^T \; {}^O({}^A_B\vec{F}) \tag{7} $$

where ${}^O({}^A_B J)^T = {}^A_B J^T \; {}^A_O R$.

One point of note is that "admissible" generalized forces lie in the row space of ${}^O({}^A_B J)^T$. By admissible, we mean forces which can be realized using the available joint actuators. For example, if no relative motion can be realized along a certain direction, then no matter how large a force is produced along that direction by the virtual actuators, no effect will result on the robot (assuming rigid links). Similarly, any generalized forces which lie in the null space of ${}^O({}^A_B J)^T$ will produce no torque at the joints and hence have no effect on the robot. Whether or not such an inadmissible generalized force should be allowed probably depends on the implementation. In any case, an inadmissible force should be detectable. An easy test is to

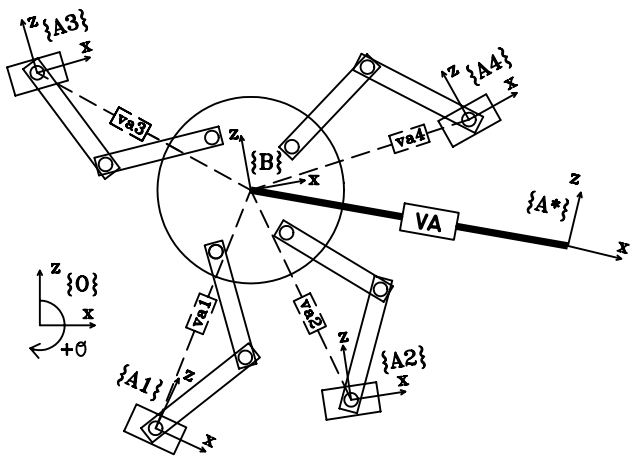

Figure 2: Parallel Virtual Actuator Reference Frames. Frame {B} is the action frame. Frames $\{A_i\}$ are the reaction frames. Frame {O} is the reference frame. VA represents the conglomerate virtual actuator which is comprised of the individual virtual actuators VA_i. Frame $\{A^*\}$ is an imaginary construct which represents the reaction frame of the conglomerate virtual actuator.

see if it lies in the row space of the Jacobian transpose. When implementing parallel virtual actuators, as described below, it is important that the inadmissible force constraints be known for each of the serial paths of the virtual actuator so that the desired generalized force can be accurately divided among the individual serial paths.

2.5 Parallel Mechanisms and Multi-Frame Virtual Actuators

With a serial link structure all the necessary equations are computationally inexpensive. With a parallel structure a matrix inversion is necessary in solving the force distribution problem.

We start by defining one action frame {B}, one reference frame {O} and multiple reaction frames $\{A_i\}$ as in Figure 2. Frame $\{A^*\}$ is a construct used to represent all of the frames $\{A_i\}$ and can be viewed as the conglomerate reaction frame. The parallel virtual actuator can be considered as multiple serial sub-actuator acting on the same action frame. Each serial sub-actuator has a corresponding Jacobian which is calculated as in Section 2.3. We combine these to get

$$\begin{bmatrix} \vec{\tau_1} \\ \vec{\tau_2} \\ \vdots \\ \vec{\tau_p} \end{bmatrix} = \begin{bmatrix} {}^O({}^{A_1}_{B}J)^T & 0 & \cdots & 0 \\ 0 & {}^O({}^{A_2}_{B}J)^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & {}^O({}^{A_p}_{B}J)^T \end{bmatrix} \begin{bmatrix} {}^O({}^{A_1}_{B}\vec{F}) \\ {}^O({}^{A_2}_{B}\vec{F}) \\ \vdots \\ {}^O({}^{A_p}_{B}\vec{F}) \end{bmatrix} \tag{8}$$

We now have a mapping from sub-actuator generalized forces to joint torques. However, we wish to specify a single generalized force to act on the action frame. Since the action frame and reference frame are coincidental for all the sub-actuators, the vector sum of the individual generalized forces must equal the desired generalized force,

$$\sum_{i=1}^{p} {}^O({}^{A_i}_{B}\vec{F}) = {}^O({}^{A^*}_{B}\vec{F}) \tag{9}$$

We need to solve for ${}^O({}^{A_i}_{B}\vec{F})$ in terms of ${}^O({}^{A^*}_{B}\vec{F})$. To do this we must add a number of constraints. Some of these constraints will arise due to the inadmissibility of certain individual force directions. These constraints can be determined by examining the row space of the individual ${}^O({}^{A_i}_{B}J)^T$. Others will arise from constraints on the robot such as unactuated joints. The rest of the constraints can be used as design degrees of freedom.

Once enough constraints have been determined, we will have a square invertible constraint matrix, K, so that the constraints can be written in the form

$$\begin{bmatrix} {}^O({}^{A^*}_{B}\vec{F}) \\ \vec{c} \end{bmatrix} = K \begin{bmatrix} {}^O({}^{\vec{A_1}}_{B}\vec{F}) \\ {}^O({}^{\vec{A_2}}_{B}\vec{F}) \\ \vdots \\ {}^O({}^{\vec{A_p}}_{B}\vec{F}) \end{bmatrix} \tag{10}$$

and the individual sub-force vectors solved for by inverting the constraint matrix, K.

The elements of $\vec{c}$ can be extra control variables, such as individual interaction forces, or 0 for constraints which are solely a function of the forces ${}^O({}^{A_i}_{B}\vec{F})$. Of course, we need not retain the columns of K^{-1} which are multiplied by 0. We can now substitute back into 8 to get the final force to torque relationship.

2.5.1 Inverting the Constraint Matrix

Solving Equation 10 requires inverting a potentially large sparse matrix. We show here a method for taking advantage of the structure of the constraint matrix, K, in order to reduce computational requirements. The method is an extension of Gardners's Partitioned Actuator Set Control Technique [3, 2].

Gardner partitions the actuators into a Minimum Actuator Set (MAS) and a Redundant Actuator Set (RAS). We stress here that we are not dealing with actuators at this level but rather virtual force distribution. Therefore we specify the Minimum Force Set (MFS) and the Redundant Force Set (RFS). We extend Gardner's method by adding the Constrained Force Set (CFS) for dealing with natural constraints, such as underactuated legs, point feet, and limp joints.

We assume that all the serial paths of the parallel virtual actuators are of the same structure, with the

same number of constraints. We do this only to simplify the following explanation. The mathematics is easily extended to the general case.

We define the following constants,

n	dimension of the generalized force to be applied
p	number of serial paths of the parallel virtual actuator
l	number of constraints in each serial path
d = n-l	number of non-constrained degrees of freedom per serial path
r = pd - n	number of redundant serial path virtual force actuators

The number of force elements in the Minimum Force Set will be n; in the Constrained Actuator Set l per serial path (pl total); in the Redundant Force Set r. How the forces are partitioned into these sets depends on the design constraints one wishes to implement and the limitations placed on these constraints by the extended partitioned force set method.

As an example, suppose we have a 100 leg millipede with 2 joints per leg and point feet. We would have $n = 6$ for the 3 elements of the force vector and 3 elements of the moment vector which we wish to exert; $p = 100$ for the 100 legs; $l = 4$ for the 3 constraints of no torques at the feet and the 1 constraint provided by the underactuation of having only 2 joints per leg; $d = 2$ meaning each leg can provide a 2 dimensional force vector from the 6 dimensional space; and $r = 194$ meaning we have 194 redundancies and therefore can specify up to 194 design constraints. The Minimum Force Set would contain 6 elements; the Constrained Force Set 400; and the Redundant Force Set 194.

We first partition the virtual forces into the Constrained Force Set (CFS), f^{ib}, and those not in the CFS, f^{ia}, and rearrange Equation 10 into the following form,

$$\begin{bmatrix} f_d^a \\ f_l^b \\ t_l^{1c} \\ t_l^{2c} \\ \vdots \\ t_l^{pc} \\ c_r \end{bmatrix} = \begin{bmatrix} I_{d\times d} & 0_{d\times l} & I_{d\times d} & 0_{d\times l} & \cdots & I_{d\times d} & 0_{d\times l} \\ 0_{l\times d} & I_{l\times l} & 0_{l\times d} & I_{l\times l} & \cdots & 0_{l\times d} & I_{l\times l} \\ J_{l\times d}^{1a} & J_{l\times l}^{1b} & 0_{l\times d} & 0_{l\times l} & \cdots & 0_{l\times d} & 0_{l\times l} \\ 0_{l\times d} & 0_{l\times l} & J_{l\times d}^{2a} & J_{l\times l}^{2b} & \cdots & 0_{l\times d} & 0_{l\times l} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{l\times d} & 0_{l\times l} & 0_{l\times d} & 0_{l\times l} & \cdots & J_{l\times d}^{pa} & J_{l\times l}^{pb} \\ D_{r\times d}^{1a} & D_{r\times l}^{1b} & D_{r\times d}^{2a} & D_{r\times l}^{2b} & \cdots & D_{r\times d}^{pa} & D_{r\times l}^{pb} \end{bmatrix} \begin{bmatrix} f_d^{1a} \\ f_l^{1b} \\ f_d^{2a} \\ f_l^{2b} \\ \vdots \\ f_d^{pa} \\ f_l^{pb} \end{bmatrix} \tag{11}$$

where I is the identity matrix, 0 is the zero matrix, J^{ia} , J^{ib} are natural constraint matrices, and D^{ia}, D^{ib} are design constraint matrices. The Constrained Force Set consists of the forces f^{ib} while the forces f^{ia} belong in the Minimum and Redundant Force Sets. The subscript on each matrix block show the size of the block.

The constrained torques, t^{ic} are those which can be measured or inferred but not controlled. For example, with an unactuated limp joint, $t^c = 0$ while for a joint containing a passive linear spring, $t^c = k\theta$.

We can reduce the size of the matrix in equation 11 by taking advantage of the sparseness of the natural constraint blocks. Each natural constraint row can be written as

$$t^{ic} = J^{ia} f^{ia} + J^{ib} f^{ib}$$

Solving for f^{ib} we have,

$$f^{ib} = Q^i f^{ia} + (J^{ib})^{-1} t^{ic} \tag{12}$$

where,

$$Q^i = [-(J^{ib})^{-1} J^{ia}]_{l\times d} \tag{13}$$

We can substitute this back into Equation 11 to get a reduced set of equations,

$$\begin{bmatrix} f_d^a \\ f_l^b - u_l \\ c_r - v_r \end{bmatrix} = \begin{bmatrix} I_{d\times d} & I_{d\times d} & \cdots & I_{d\times d} \\ Q^1 & Q^2 & \cdots & Q^p \\ S^1 & S^2 & \cdots & S^p \end{bmatrix} \begin{bmatrix} f_d^{1a} \\ f_d^{2a} \\ \vdots \\ f_d^{pa} \end{bmatrix} \tag{14}$$

where,

$$\begin{aligned} u_l &= \sum_i (J^{ib})^{-1} t^{ic} \\ v_r &= \sum_i D^{ib} (J^{ib})^{-1} t^{ic} \\ S^i &= [D^{ib} Q^i + D^{ia}]_{r\times d} \end{aligned} \tag{15}$$

We have now reduced the matrix size from $np \times np$ to $dp \times dp$ by eliminating the Constrained Force Set (CFS). Note that Equation 14 is still in the general form, i.e. we have put no restrictions on the design constraint equations. We could stop here and invert the new $dp \times dp$ matrix, if it were computationally feasible. In order to further reduce the size of the matrix, we can eliminate the Redundant Force Set (RFS) by specifying our design constraints in a proper manner.

Similar to [3], we specify the Redundant Force Set, f^r, in terms of the Minimum Force Set, f^m.

$$f_r^r = c_r - B_{r\times n} f_n^m \tag{16}$$

where f^r is the Redundant Force Set, f^m is the Minimum Force Set, c is a vector of variables or constants, and B is the design constraint matrix.

Writing the design constraints in terms of Equation 16 requires that

$$D^{ib} = 0 \; \forall i$$

and D^{ia} are restricted so that we can rearrange Equation 14 into the following form,

$$\begin{bmatrix} f_n \\ c_r \end{bmatrix} = \begin{bmatrix} A_{n\times n}^m & A_{n\times r}^r \\ B_{r\times n} & I_{r\times r} \end{bmatrix} \begin{bmatrix} f_n^m \\ f_r^r \end{bmatrix} \tag{17}$$

We can now eliminate the RFS, f^r, from Equation 17 to by substituting Equation 16 into 17 to get

$$[f - A^r c]_n = [A^m - A^r B]_{n \times n} [f_n^m] \tag{18}$$

To solve for the MFS, f^m, we must invert the $n \times n$ matrix in Equation 18. The Redundant Force Set is then computed by plugging back into equation 16. Finally, we can rearrange the Minimum and Redundant Force Sets to get f^{ia} and substitute back into Equation 12 to solve for the Constrained Force Set.

In order to use the above method, one must write all the design constraints in the form of Equation 16. In writing these constraints, one must specify the Redundant Force Set and the Minimum Force Set. The remaining forces are the Constrained Force Set. One must make sure that the choice of design constraints allows for a solution of Equation 18 to exist.

Examining Equations 14 to 18 we see that we take p $[l \times l]$, and 1 $[n \times n]$ matrix inversions in solving for the virtual forces applied to each serial path. These inversions will take significantly less computational resources to perform than the original $np \times np$ inversion. Since the computational complexity of matrix inversion scales with the cube of the matrix size, the above method is $O(pl^3 + n^3)$ whereas inverting the original matrix is $O(p^3 n^3)$.

2.5.2 Pseudo-Inversion of the Constraint Matrix

The above discussion assumed that we specified r design constraints. In some cases, we may not wish to specify as many design constraints, and we may not wish to be limited to specifying all the constraints in terms of Equation 17. Suppose we wish to specify only s design constraints, where $s < r$. If we specify the design constraints in the form,

$$f_s^{r2} = c - B^1 f^m - B^2 f^{r1} \tag{19}$$

then instead of Equation 17, we will have,

$$\begin{bmatrix} [f - A^{r2} c]_n \\ c_s \end{bmatrix} = \begin{bmatrix} A^m_{n \times n} & A^{r1}_{n \times (r-s)} & A^{r2}_{n \times s} \\ B^1_{s \times n} & B^2_{s \times (r-s)} & I_{s \times s} \end{bmatrix} \begin{bmatrix} f_n^m \\ f_{r-s}^{r1} \\ f_s^{r2} \end{bmatrix} \tag{20}$$

Utilizing the structure of this matrix, we get the following set of Equations,

$$\left[f_n \right] = \left[[A^m - A^{r2} B^1]_{n \times n} \; [A^{r1} - A^{r2} B^2]_{n \times r-s} \right] \begin{bmatrix} f_n^m \\ f_{r-s}^{r1} \end{bmatrix} \tag{21}$$

The above matrix is non-square. To solve the above equation for f^m and f^{r1}, we can use the Moore-Penrose pseudo-inverse,

$$A^+ = A^T (A A^T)^{-1} \tag{22}$$

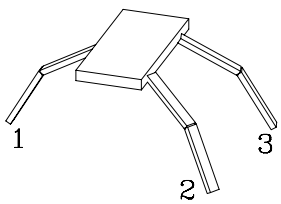

Figure 3: Drawing of hexapod, used in the 3D example, standing on three of its six legs. The other three legs are not shown for clarity.

and then solve for f^{r2} by substituting back into Equation 19.

The pseudo-inverse still requires inverting an $n \times n$ matrix so our computational requirements are about the same as the previous method.

3 Implementation on a Hexapod

We present a three dimensional example of a hexapod robot alternately being supported by tripods composed of three of its six legs. Using three legs for support allows the hexapod to be statically stable and is a typical snapshot of the tripod walking gait.

Figure 3 shows the hexapod model standing on three legs. Figure 4 shows a close up view of one of the legs. There is one actuated degree of freedom at the knee and two at the hip. The knee angle is θ_{k_i} and the hip angles are θ_{h1_i} and θ_{h2_i}. If the joint angles are all zero, the leg will point straight down from the body. Each leg is comprised of an upper and lower link, both of length L. The location of the leg with respect to the action frame {B} is $(P_x, P_y, 0)$. We assume that we can measure only the angles of the three joints of each of the three legs.

We wish to specify a generalized force, ${}^B({}^{A^*}_{B} F)$ which acts on the body from the three legs. We must determine the mapping from the desired generalized force to the virtual forces applied by each leg and finally to the joint torques.

The Jacobian from the reaction frame to the action frame for a given leg, ${}^B({}^{A_i}_{B} J)$ is determined via an iterative computation of the joint velocity to cartesian velocity mapping, as described in [1]. We can extract the elements of the Jacobian transpose which map the virtual model generalized forces to joint torques for the ith leg,

$$\begin{bmatrix} \tau_{k_i} \\ \tau_{h1_i} \\ \tau_{h2_i} \end{bmatrix} = J^T \begin{bmatrix} f_{x_i} \\ f_{y_i} \\ f_{z_i} \\ n_{x_i} \\ n_{y_i} \\ n_{z_i} \end{bmatrix} \tag{23}$$

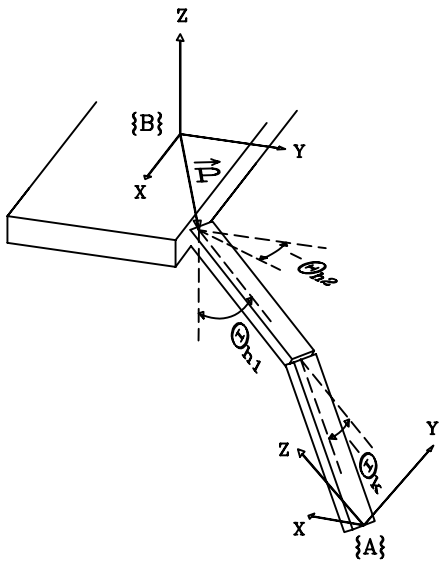

Figure 4: Diagram of a single leg of the hexapod example. There is one actuated degree of freedom at the knee and two at the hip. The knee angle is θ_k and the hip angles are θ_{h1} and θ_{h2}. The leg links are both of length L. The location of the leg with respect to frame {B} is (P_x, P_y, 0).

$$J = \begin{bmatrix} -L\, s_{h2_i} c_{h1_i} & 0 & -P_{y_i} \\ L\, c_{h2_i} c_{h1_i} & 0 & +P_{x_i} \\ L\, s_{h1_i} - P_{x_i}\, s_{h2_i} + P_{y_i}\, c_{h2_i} & -P_{x_i}\, s_{h2_i} + P_{y_i}\, c_{h2_i} & 0 \\ -c_{h2_i} & -c_{h2_i} & 0 \\ -s_{h2_i} & -s_{h2_i} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{24}$$

where $c_x = \cos(\theta_x)$ and $s_x = \sin(\theta_x)$.

Since no moments can be applied at the point feet each leg has three natural constraints. Following the method in Section 2.5.1 we have

$$n = 6,\ p = 3,\ l = 3,\ d = 3,\ r = 3 \tag{25}$$

To reduce the size of the matrix to be inverted, we must partition the individual components of the virtual forces into the Minimum Force Set (6 elements), Redundant Force Set (3 elements), and Constrained Force Set (9 elements). For our 3 design constraints, we decide to match the horizontal forces of legs 2 and 3 and match the lateral force of leg 1 with the sum of the lateral forces of legs 2 and 3,

$$f_{x3} = f_{x2},\ \ f_{y3} = f_{y2},\ \ f_{y1} = 2f_{y2} \tag{26}$$

These design constraints are written in the terms of Equation 16 if we partition the virtual forces as follows,

$$\begin{aligned} MFS &= \{f_{x1}, f_{z1}, f_{x2}, f_{y2}, f_{z2}, f_{z3}\} \\ RFS &= \{f_{x3}, f_{y3}, f_{y1}\} \\ CFS &= \{n_{x1}, n_{y1}, n_{z1}, n_{x2}, n_{y2}, n_{z2}, n_{x3}, n_{y3}, n_{z3}\} \end{aligned}$$

The terms of the constraint equation (11) are now,

$$f^a = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix},\ \ f^b = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix},\ \ c = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$D^{1a} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},\ \ D^{2a} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -2 & 0 \end{bmatrix},\ \ D^{3a} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$D^{1b} = D^{2b} = D^{3b} = 0 \tag{27}$$

The elements of the natural constraints, J^{ia} and J^{ib} are extracted from the Jacobian Transpose. The Minimum and Redundant Force Sets are acted on by J^{ia} where

$$\begin{aligned} [J^{ia}]_{1,1} &= -Ls_{h2_i}(c_{k_i+h1_i} + c_{h1_i}) \\ [J^{ia}]_{1,2} &= Lc_{h2_i}(c_{k_i+h1_i} + c_{h1_i}) \\ [J^{ia}]_{1,3} &= Ls_{k_i+h1_i} + Ls_{h1_i} + P_{y_i}\, c_{h2_i} - P_{x_i}\, s_{h2_i} \\ [J^{ia}]_{2,1} &= -Lc_{h2_i}(1 + c_{k_i}) - P_{y_i}\, s_{k_i+h1_i} \\ [J^{ia}]_{2,2} &= -Ls_{h2_i}(1 + c_{k_i}) + P_{x_i}\, s_{k_i+h1_i} \\ [J^{ia}]_{2,3} &= -P_{y_i}\, s_{h2_i} c_{k_i+h1_i} - P_{x_i}\, c_{h2_i} c_{k_i+h1_i} \\ [J^{ia}]_{3,1} &= Lc_{h2_i} s_{k_i} - P_{y_i}\, c_{k_i+h1_i} \\ [J^{ia}]_{3,2} &= Ls_{h2_i} s_{k_i} + P_{x_i}\, c_{k_i+h1_i} \\ [J^{ia}]_{3,3} &= P_{y_i}\, s_{h2_i} s_{k_i+h1_i} + P_{x_i}\, c_{h2_i} s_{k_i+h1_i} \end{aligned}$$

while the Constrained Force Set is acted on by J^{ib} where

$$(J^{ib}) = \begin{bmatrix} -c_{h2_i} & -s_{h2_i} & 0 \\ s_{h2_i} c_{k_i+h1_i} & -c_{h2_i} c_{k_i+h1_i} & -s_{k_i+h1_i} \\ -s_{h2_i} s_{k_i+h1_i} & c_{h2_i} s_{k_i+h1_i} & -c_{k_i+h1_i} \end{bmatrix} \tag{28}$$

The first six rows of the constraint matrix state that the sum of the virtual force and moment vectors for each of the individual legs equals the total virtual force and moment vector acting on the body at frame {B}. The next three sets of three rows define the natural constraints of zero torque at each foot. The last three rows are the design constraints chosen in Equation 26.

Note that the angles the foot makes with frame $\{A_i\}$ do not appear in any of the previous equations. In order to derive these equations, X-Y-Z Euler angles were used at the feet. The foot angles did not appear in the torque-force relationship (Equation 23). However, they did appear in the natural constraints, J^{ia} and J^{ib}. The natural constraint equations define

a 3 dimensional subspace of the 6 dimensional virtual force space. This subspace is the space in which "admissible" virtual forces can be applied. We verified that this subspace is the same for any foot angles. Therefore we were able to arbitrarily set the foot angles to zero.

An intuitive explanation for why the foot angles do not matter is that virtual forces are being applied from frame $\{A_i\}$ to frame {B} *with respect to frame* $\{B\}$. How we define frame $\{A_i\}$ therefore doesn't matter, as long as we can specify the foot angles in some way. Therefore, it is arbitrary what the foot angles are and we can set them to zero in order to eliminate them from our equations.

The submatrix, J^{ib}, happens to be orthonormal so that its inverse is simply its transpose,

$$(J^{ib})^{-1} = (J^{ib})^T \tag{29}$$

We eliminate the Constrained Force Set to get the following elements in Equation 14,

$$S^i = D^{ia}, \quad u_l = 0, \quad v_r = 0 \tag{30}$$

$$\begin{aligned}
[Q^i]_{1,1} &= 0 \\
[Q^i]_{1,2} &= Lc_{k_i+h1_i} + Lc_{h1_i} \\
[Q^i]_{1,3} &= P_{y_i} + Lc_{h2_i}(s_{k_i+h1_i} + s_{h1_i}) \\
[Q^i]_{2,1} &= -L(c_{k_i+h1_i} + c_{h1_i}) \\
[Q^i]_{2,2} &= 0 \\
[Q^i]_{2,3} &= Ls_{h2_i}(s_{k_i+h1_i} + s_{h1_i}) - P_{x_i} \\
[Q^i]_{3,1} &= -Lc_{h2_i}(s_{k_i+h1_i} + s_{h1_i}) - P_{y_i} \\
[Q^i]_{3,2} &= -Ls_{h2_i}(s_{k_i+h1_i} + s_{h1_i}) + P_{x_i} \\
[Q^i]_{3,3} &= 0
\end{aligned}$$

We can eliminate the Redundant Force Set as in Equations 17 and 18, to get

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & s_{4,2} & 0 & s_{4,4} & s_{4,5} & s_{4,6} \\ s_{5,1} & s_{5,2} & s_{5,3} & 0 & s_{5,5} & s_{5,6} \\ s_{6,1} & 0 & s_{6,3} & s_{6,4} & 0 & 0 \end{bmatrix} \begin{bmatrix} f_{x1} \\ f_{z1} \\ f_{x2} \\ f_{y2} \\ f_{z2} \\ f_{z3} \end{bmatrix} \tag{31}$$

where,

$$\begin{aligned}
s_{4,2} &= P_{y_1} + Lc_{h2_1}(s_{h1_1+k_1} + s_{h1_1}) \\
s_{4,4} &= -2L\, s_{h1_1} s_{k_1} - Ls_{h1_2} s_{k_2} - Ls_{h1_3} s_{k_3} \\
&\quad + Lc_{h1_2}(c_{k_2} + 1) + Lc_{h1_3}(c_{k_3} + 1) + 2L\, c_{h1_1}(c_{k_1} + 1) \\
s_{4,5} &= P_{y_2} + Lc_{h2_2}(s_{h1_2+k_2} + s_{h1_2}) \\
s_{4,6} &= P_{y_3} + Lc_{h2_3}(s_{h1_3+k_3} + s_{h1_3}) \\
s_{5,1} &= -Lc_{h1_1+k_1} - Lc_{h1_1} \\
s_{5,2} &= -P_{x_1} + Ls_{h2_1}(s_{h1_1+k_1} + s_{h1_1}) \\
s_{5,3} &= Lc_{h1_2}(-c_{k_2} - 1) + Lc_{h1_3}(-c_{k_3} - 1) \\
&\quad + Ls_{h1_2} s_{k_2} + Ls_{h1_3} s_{k_3} \\
s_{5,5} &= -P_{x_2} + Ls_{h2_2}(s_{h1_2}(c_{k_2} + 1) + c_{h1_2} s_{k_2}) \\
s_{5,6} &= -P_{x_3} + Ls_{h2_3}(s_{h1_3+k_3} + s_{h1_3}) \\
s_{6,1} &= -P_{y_1} - Lc_{h2_1}(s_{h1_1+k_1} + s_{h1_1}) \\
s_{6,3} &= -P_{y_2} - P_{y_3} - Lc_{h2_2}(s_{h1_2+k_2} + s_{h1_2}) \\
&\quad - Lc_{h2_3}(s_{h1_3+k_3} + s_{h1_3}) \\
s_{6,4} &= 2\,P_{x_1} + P_{x_2} + P_{x_3} - 2\,Ls_{h2_1}(s_{h1_1+k_1} + s_{h1_1}) \\
&\quad - Ls_{h2_2}(s_{h1_2+k_2} + s_{h1_2}) - Ls_{h2_3}(s_{h1_3+k_3} + s_{h1_3})
\end{aligned}$$

To solve for the Minimum Force Set, we need to invert the 6×6 matrix in Equation 31. The design constraints (Equation 26) are then used to solve for the Redundant Force Set in terms of the Minimum Force Set. Equation 12 is used to solve for the Constrained Force Set in terms of the Minimum and Redundant Force Sets and finally Equation 23 is used to compute the joint torques.

We now have a relatively small set of equations for coordinating three legs of a hexapod robot.

4 Hexapod Simulation

A hexapod robot was created using simulation software which emulates physically based dynamics [9]. The simulated hexapod has a mass of approximately 3.8kg, is 45cm in length, and stands 15cm tall. Virtual Actuator Control was applied to the hexapod in order to make it perform a simple walking algorithm as detailed by Torres [14].

Virtual actuators were used to emulate simple linear spring and damper virtual components (i.e. cartesian PD controllers) [8]. Movements in the z, roll, and pitch directions were position controlled to a constant desired value using a virtual spring-damper system. A virtual damper was used in the x, y, and yaw directions in order to velocity control the movements in these directions. A different set of virtual spring-damper mechanisms were used to position control the legs of the swing tripod. The swing and support tripods were cycled by a state machine which changed states based on the distance between the body and the tripods.

The simplicity of the virtual components facilitated the use of a minimal set of high level user inputs to trace out various walking paths. By specifying the magnitude and angle of the body velocity and the magnitude of the yaw velocity, the robot walked paths ranging from a straight line, to a circle, to a sine wave while facing a direction independent of its travel direction. The hexapod was able to walk up to 0.8m/s.

Virtual Actuator Control was also used to balance a pendulum on the back of the hexapod while walking. A pendulum of length 30cm and mass 0.18kg was attached via a pin joint in the x direction as seen in Figure 5. A simple LQR controller was used to regulate the movement in the x direction in order to stablize the pendulum. The pendulum and bug exhibited the typical response of a non-minimum phase system as shown in Figure 6. The robot begins by traveling in

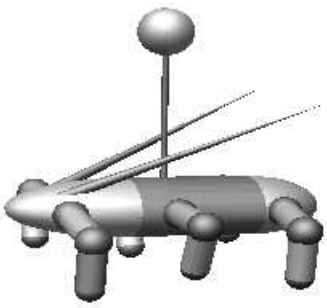

Figure 5: Picture of the simulated hexapod with a pendulum attached to its back.

a direction opposite to the desired velocity in order to have the pendulum fall in the direction of the desired velocity. The robot then changes its velocity direction and slowly ramps to the desired value of 0.2 m/s. After an initial peak in pendulum angle at 0.055 radians, the angle reduces to approximately 0.015 radians. The final plot of the walking state shows that the bug is able to balance the pendulum without disrupting its walking cycle.

5 Summary and Conclusions

We have introduced a control method in which joint actuators are used to mimic virtual actuators. Virtual Actuator Control provides a more intuitive control space and hence, simpler control laws.

The implementation of virtual actuators is relatively straightforward. Equations have to be derived only once for a given mechanism. For a serial mechanism, computational requirements are low since no functional or matrix inversions are required. For a parallel mechanism, a matrix inversion is necessary. However, by taking advantage of the form of the matrix, the size of the inverse can be minimized.

Virtual Actuator Control, in conjunction with a simple linear controller, was applied to a simulated hexapod. The hexapod successfully walked while balancing a pendulum and tracking an object.

Acknowledgements

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

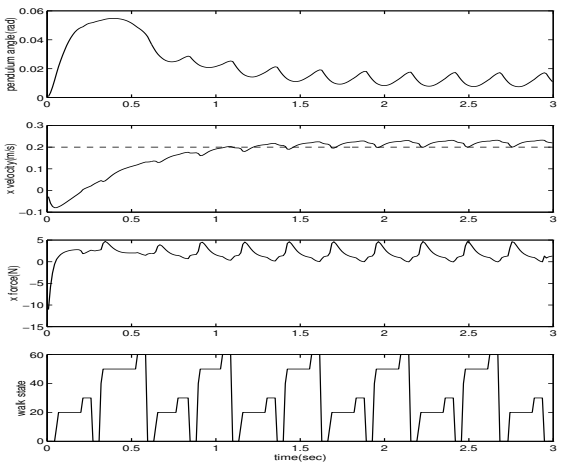

Figure 6: Plots of data generated while the hexapod walked and balanced a pendulum. The first displays the angle of the pendulum. The second displays the actual velocity of the hexapod in the x direction with the desired velocity indicated by the dashed line. The third graph shows the resultant virtual x force due to the virtual components. The final graph shows the steady, periodic nature of the walking cycle.

References

[1] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.

[2] J. F. Gardner. Force distribution in walking machines over rough terrain. *J. of Dynamic Systems, Measurement and Control*, 113:754–758, 1991.

[3] J. F. Gardner, K. Srinivasan, and K. J. Waldron. A solution for the force distribution problem in redundantly actuated closed kinematic chains. *J. of Dynamic Systems, Measurement and Control*, 112:523–526, 1990.

[4] N. Hogan. Impeadance control: An approach to manipulation: Part i - theory, part ii - implementation, part iii - applications. *J. of Dynamic Systems, Measurement and Control*, 107:1–24, 1985.

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

[6] David E. Orin and W.W. Schrader. Efficient jacobian determination for robot manipulators. *Robotics Research: The First International Symposium*, 1984.

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

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

[9] Robert Ringrose. The creature library. Unpublished reference guide to a C library used to create physically realistic simulations, 1992.

[10] K. Salisbury. Active stiffness control of a manipulator in cartesian coordinates. *19th IEEE Conference on Decision and Control*, pages 83–88, Dec. 1980.

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

[12] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, Wellesley, MA., 1993.

[13] C. Sunada, D. Argaez, S. Dubowsky, and C. Mavroidis. A coordinated jacobian transpose control for mobile multi-limbed robotic systems. *IEEE Journal of Robotics and Automation*, pages 1910–1915, 1994.

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