Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Motion Generation for a Tumbling Robot Using a General Contact Model

J. V. Albro and J.E. Bobrow
Department of Mechanical and Aerospace Engineering
University of California, Irvine
Irvine, CA 92697
{jvargasc,jebobrow}@uci.edu

Abstract— **The goal of this research was to develop a general approach for the generation of motions for robots that make and break contact with the ground, such as hopping, walking, or tumbling robots. We first develop a general contact model and show how to smooth the forces arising from this contact to make them C^1 functions of the state. This is essential for our motion generation algorithm, which relies on the solution of an optimal control problem for different phases of the motion. We apply our approach to the generation of motion primitives to a simple experimental robot that is capable of producing hand-stands and tumbling.**

I. Introduction

Many interesting physical systems have changing dynamics because they make and break contact with their environment, *e.g.*, robots that are not fixed to the ground, legged robots, and legged organisms. The aim of this work is to model such systems in a general way. To do this in the context of our previous work, which relies on an optimal control approach to generate trajectories for robots, we had to develop a general model for the forces arising due to contact between physical bodies. For the optimal control problem to converge reliably, the forces in the contact model have to be C^1, so some functions are introduced to smooth the contact forces. Another requirement for the general contact model is the derivative describing how the distance between two bodies changes as a function of the robot joints' motion. The resulting general contact model is presented in this paper.

Optimal control has been used many times to generate trajectories for physical systems (including robots), ([17],[9],[8],[15]) but the complexity of the governing dynamics equations often renders the problem intractable. For this approach to succeed, it is important that analytic gradients of the equations of motion be used; approximate gradients are particularly problematic for systems that make and break contact with the ground, or have nonholonomic constraints. The implication for the system model is that the forces and accelerations it defines need to have continuous derivatives whose analytic form can be determined.

Park, Bobrow and Ploen [10] used the matrix exponential formulation of the dynamics and Lie group theory to express the equations of motion so that their analytic gradients come automatically from the formulation of the equations. This was implemented in the software program C-STORM [12], and used to solve hopping robot[1] and human walking [16] problems, but in both these cases, modeling the interaction with the environment was specific to the particular system being simulated. This formed the foundation for the current research – the general contact model would be implemented as an extension to C-STORM.

There are two general ways to deal with dynamic models that change due to making and breaking contact with the environment: (1) having different dynamic models for the different phases of motion and switching between them as needed; or (2) having one dynamics model throughout the motion, and modeling interactions with the environment as external forces acting on the system model [2], [4]. We have chosen to use the latter, as previously indicated. Yamane and Nakamura [18] give a good overview of how contact models are implemented. There are two categories of contact models, penalty-based methods and analytical methods. In the penalty-based methods, which we use, contact forces are modeled with virtual spring-damper systems at points of contact. Wang, Kumar, and Abel [14] looked more closely at the rigid body assumption in contact and how it can lead to violations of conservation of energy and other conservation laws. They simulate mechanical systems undergoing multiple frictional constraints with local, linearly elastic properties at points of contact. Our contact model has similarities to this approach, as well as that of [11], and is a modification of the approach described in Tenaglia et al [13].

II. Generating Motions

We use an optimal control approach to generate motions for our robots. The motions include the internal paths of the actuated robot joints, which we specify, along with path of the robot through space that result from these internal paths. Using optimal control to generate paths allows the paths to be *good* according to some

0-7803-8232-3/04/$17.00 ©2004 IEEE

objective measure, does not require a large amount of a priori information about the motion, and allows the motion to be specified in a more high-level manner. For example, rather than try to figure out what a fuel efficient path might look like, the objective measure (or cost function) being minimized in the optimal control problem can include the total fuel used throughout the motion. By using different objective measures, motions can be found with different characteristics, such as minimum effort used, minimum time to travel a given trajectory, or the maximum payload that can be moved from one position to another [15].

To get the trajectories of the actuated robot joints (the controls in the optimal control problem), we assume the trajectories minimize some cost function, subject to certain constraint equations. The cost function together with the constraint equations define the optimal control problem. Each unique cost function corresponds to a separate motion, so specifying the cost function equates to specifying the motion. The cost function is minimized subject to the equations of motion of the system, boundary conditions on joint positions and velocities, and joint limits on the robot model. The general form of cost function with its constraints is as follows:

$$\underset{q^a(t)}{\text{Min}} \; J(q^a) = \Psi[q^p(t_f), \dot{q}^p(t_f)] + \int_0^{t_f} L(q^p, \dot{q}^p, \tau^a) dt \quad (1)$$

Subject to:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau + \sum \Phi_q^T F \quad (2)$$

$$q(0) = q_0, \; \dot{q}(0) = \dot{q}_0 \quad (3)$$

$$\underline{q} < q < \overline{q} \quad (4)$$

$$\underline{\dot{q}} < \dot{q} < \overline{\dot{q}} \quad (5)$$

where J is the cost function, $q^a(t)$ is the position of the actuated joints of the robot, $q^p(t)$ is the position of the unactuated joints of the robot (some of which are used to keep track of the position of the robot in space), $\dot{q}^p$ is the velocity of the unactuated joints of the robot, $q = (q^a, q^p)$, the set of all robot joints, and τ^a is the motor torques on the active joints. The matrix Φ_q^T is the Jacobian of the contact point, and F is the external force applied at the contact point (the normal and/or friction forces, obtained from our contact model). The summation of $\Phi_q^T F$ represents the contribution of all the external forces applied to the system.

We solve the optimal control problem with a direct method: guess a solution $q^a(t)$, integrate the equations of motion with the guessed solution, evaluate J, and use gradient information to correct the solution so that the cost decreases until a minimum value for J is found. The guessed solutions to the paths of the active joints $q^a(t)$ are parameterized with quintic B-splines of the form

$$q^a(t) = \sum_{i=1}^{n} a_i B_{i,j} \quad (6)$$

where $B_{i,j}$ are the spline basis functions, and a_i are the scaling coefficients. This $q^a(t)$ also defines the active joint velocities and accelerations $\dot{q}^a(t), \ddot{q}^a(t)$. Moving the active joints of the robot also causes the passive (unactuated) joints q^p to move along some trajectory which can be solved for by integrating the equations of motion for the system. Because all the terms in the cost function are then determined, the problem is transformed into one of static parameter optimization. The spline parameters defining $q^a(t)$ are the parameters that are varied during the optimization.

In our C-STORM software, we have a recursive hybrid dynamics algorithm [12] which solves the dynamics equations for the active joint torques τ^a and the passive joint accelerations $\ddot{q}^p$. This algorithm takes as input the current state $(q, \dot{q})$ along with the active joint accelerations $\ddot{q}^a$ and passive joint torques τ^p. Let the recursive algorithm be referred to as g with:

$$(\tau^a, \ddot{q}^p) = g(q, \dot{q}, \ddot{q}^a, \tau^p). \quad (7)$$

Given an input $\ddot{q}^a(t)$, $\tau^p(t)$, and initial conditions $q(0), \dot{q}(0)$, we use (7) to solve the differential equation

$$\frac{d}{dt}\begin{Bmatrix} q^p \\ \dot{q}^p \\ \int_0^t L dt \end{Bmatrix} = \begin{Bmatrix} \dot{q}^p \\ \ddot{q}^p(q, \dot{q}, \ddot{q}^a, \tau^p) \\ L(q^p, \dot{q}^p, \tau^a) \end{Bmatrix} \quad (8)$$

for $q^p(t)$. The purpose of the third equation is to compute the second term in the cost function $\int_0^{t_f} L(q^p, \dot{q}^p, \tau^a) dt$. The exact form of L differs depending on the cost function being computed. Once the system has been integrated forward in time, using this algorithm, the cost function can be evaluated, and then the derivative of the hybrid algorithm is used to get the gradient of the cost function.

The derivative of the hybrid algorithm [12] is used to solve for the active joint torques, the passive joint accelerations, and the derivatives of both of those with respect to all of the current joint positions, velocities, and accelerations. This algorithm takes the same inputs as g, with the additional inputs of the derivatives of the torques applied on the passive joints τ^p with respect to all of the current joint positions, velocities, and accelerations. Let the derivative of the recursive algorithm be referred to as dg with:

$$\left(\tau^a, \ddot{q}^p, \frac{\partial(\ddot{q}^p, \tau^a)}{\partial q}, \frac{\partial(\ddot{q}^p, \tau^a)}{\partial \dot{q}}, \frac{\partial(\ddot{q}^p, \tau^a)}{\partial \ddot{q}}\right) = dg\left(q, \dot{q}, \ddot{q}^a, \tau^p, \frac{\partial \tau^p}{\partial q}, \frac{\partial \tau^p}{\partial \dot{q}}, \frac{\partial \tau^p}{\partial \ddot{q}}\right). \quad (9)$$

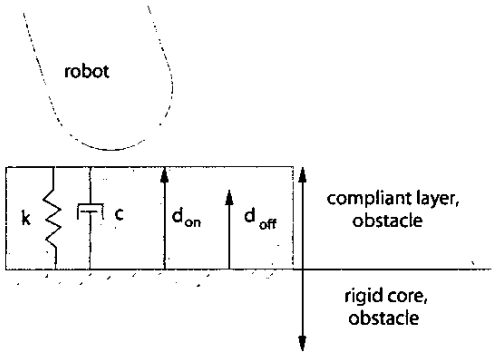

Fig. 1. Virtual Spring-Damper System Modeling Contact Forces.

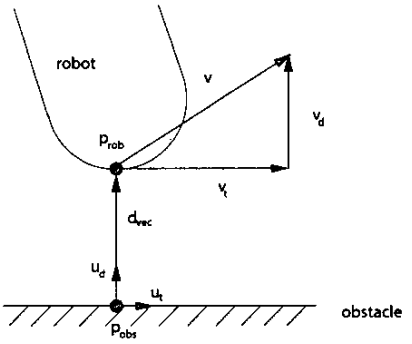

Fig. 2. Points, Vectors, and Unit Vectors used in General Collision Model.

Because the dg algorithm requires the inputs $\frac{\partial \tau^p}{\partial q}$, $\frac{\partial \tau^p}{\partial \dot{q}}$, and $\frac{\partial \tau^p}{\partial \ddot{q}}$, the applied forces to the robot from the environment must be continuous and continuously differentiable. The analytic gradient of the cost function can be calculated with the output from dg and passed to MATLAB's BFGS SQP function that performs parameter optimization. With the cost function and its gradient, the spline parameters can be varied so as to find the local minimum value of J, and the corresponding controls.

III. Contact model, Normal and Friction Forces

In our contact model, contact between two bodies results in two forces: a force in the direction normal to the shared contact surfaces, and a friction force in the plane tangential to the contact surfaces and in the opposite direction of the velocity of the contact point. As in the contact models of Tenaglia *et al.* [13], objects are modeled as though they have a rigid core inside a uniformly thick layer of compliant material. (This is like the "small region of interpenetration" in Reichler *et al.* [11].) The contact forces from the interpenetration of two objects in the compliant layer are modeled with virtual spring-damper systems at the point of contact. For simplicity, the robot is assumed to be rigid, the compliant layer is assumed to be on the obstacles it encounters (see Fig. 1), and the obstacles are assumed to remain stationary. Additionally, the forces are scaled by smoothing functions, to ensure that they are continuous and continuously differentiable, and that they begin at zero when contact is first made.

To know when the robot first makes contact with the compliant layer (which is when to start applying contact forces), and how far into the layer it has penetrated (which determines the deflection of the virtual spring), the distance between the two bodies must be determined. The relative velocity of the two bodies must also be known, to calculate the virtual damper term in the contact force model. For the contact model to be general, it must be formulated in terms of the local directions $\hat{u}_d$ and $\hat{u}_t$, where $\hat{u}_d$ is the direction of the unit vector between the near points and defines the normal direction, and $\hat{u}_t$ is a unit vector in the direction perpendicular to $\hat{u}_d$, in the plane tangential to the contact surfaces, pointing in the direction the robot is moving (see Fig. 2). The robot and obstacle may contact in more than one place, but for clarity the model will be discussed for just one pair of bodies that is contacting in one location.

We use a distance algorithm which finds the near points on each body in the pair – that is, the point on body A that is closest to body B, and vice versa. The point on the robot body closest to the obstacle is denoted p_{rob}, and the point on the obstacle body closest to the robot is denoted p_{obs} (see Figure 2). The length of the vector from p_{obs} to p_{rob}, denoted d_{vec}, gives the distance d between the two bodies, and is used to define the unit vector $\hat{u}_d$:

$$\hat{u}_d = \frac{d_{vec}}{||d_{vec}||} = \frac{d_{vec}}{d} \tag{10}$$

The other value needed for the contact model, the velocity of p_{rob}, can be obtained using the Jacobian $\Phi(q, p_{rob})$ and the vector of joint velocities $\dot{q}$ as follows:

$$\begin{bmatrix} \omega \\ v \end{bmatrix} = \Phi_p(q, p_{rob})\dot{q} \tag{11}$$

where v is the linear velocity of p_{rob}. The linear velocity v is then decomposed into components in the normal direction $\hat{u}_d$, denoted v_d, and the tangential direction $\hat{u}_t$, denoted v_t. The unit vector $\hat{u}_t$ is defined from the tangential velocity component as follows:

$$\hat{u}_t = \frac{v_t}{||v_t||}, \tag{12}$$

where $v_t = v - (v \cdot \hat{u}_d)\hat{u}_d$.

With these quantities defined, the normal contact force can be written as:

$$N = \beta(k(d_{on} - d) - c||v_d||)\hat{u}_d \tag{13}$$

where β is the smoothing function, k is the virtual spring constant, c is the virtual damper constant, and d_{on} is the

thickness of the compliant layer around the obstacle. The contact forces are only calculated when $d \leq d_{on}$, as that is when contact is defined as occurring.

The smoothing function $\beta(d)$ operates in a thin outer layer of the compliant region. $\beta(d)$ is defined such that it becomes active when the contact forces become nonzero, at d_{on}, but has no influence past a depth of d_{off}. It is defined as follows:

$$\beta(d) = \begin{cases} 0 & \text{if } d \geq d_{on} \\ f_\beta(d) & \text{if } d_{on} > d > d_{off} \\ 1 & \text{if } d_{off} \geq d \end{cases} \quad (14)$$

$$f_\beta(d) = a_0 + a_1 d + a_2 d^2 + a_3 d^3 \quad (15)$$

$$\beta(d_{on}) = 0,\ \beta'(d_{on}) = 0 \quad (16)$$

$$\beta(d_{off}) = 1,\ \beta'(d_{off}) = 0 \quad (17)$$

$\beta(d)$ is constructed so as to ensure that the analytic gradients of the equations of motion are continuous. The coefficients (a_0, a_1, a_2, a_3) are written in terms of (d_{on}, d_{off}), and the constants d_{on}, d_{off} are set by the user.

The friction force f_{fric} is defined using the normal force, and has the form:

$$f_{fric} = -\mu ||N|| \gamma \hat{u}_t \quad (18)$$

where μ is the Coulomb friction coefficient ($\mu = .3$ for our example), N is the normal force defined above and γ is the smoothing function for the friction force. γ is given by:

$$\gamma(||v_t||) = \begin{cases} 1 & \text{if } ||v_t|| > v_{max} \\ f_\gamma(||v_t||) & \text{otherwise} \end{cases} \quad (19)$$

$$f_\gamma(||v_t||) = 1.5(\frac{||v_t||}{v_{max}}) - 0.5(\frac{||v_t||}{v_{max}})^3 \quad (20)$$

$$\gamma(0) = 0,\ \gamma'(0) = 1 \quad (21)$$

$$\gamma(v_{max}) = 1,\ \gamma'(v_{max}) = 0 \quad (22)$$

where v_{max} is the value of $||v_t||$ at which γ goes to unity.

The normal and friction contact forces are found for each pair of bodies in contact with each other, and are included as F in the summation term in Eqn. (2) in the optimal control problem formulation. To use these forces with the algorithms g and dg (Eqns. (7) and (8)), the derivative of the contact forces with respect to the joint positions and velocities $(q, \dot{q})$ must be found. As the contact forces are functions of the distance between colliding bodies and the relative velocity of those bodies, their derivatives can be written with just the derivatives $\partial p_{rob}/\partial q, \partial p_{obs}/\partial q$, and by evaluating the force expressions directly for the derivatives with respect to $\dot{q}$. The next section discusses how the derivatives $\partial p_{rob}/\partial q, \partial p_{obs}/\partial q$ were evaluated.

IV. Derivative of Distance Functions

Within our optimal control approach, we need the derivatives of the contact forces N, f_{fric} with respect to the robot joint positions $q(t)$. N and f_{fric} are functions of the distance between the two bodies in contact, so the necessary derivatives are those that describe how the distance changes between the two bodies as the joints of the robot move. If $\partial p_{rob}/\partial q, \partial p_{obs}/\partial q$ are known, that is sufficient to define the derivatives $\partial N/\partial q, \partial f_{fric}/\partial q$; therefore, those are the derivatives that will be formulated here.

The change in the distance between bodies in contact is not just a function of the robot joints q, but is also affected by the shape of the surface of the robot and obstacle. This means that $\partial p_{rob}/\partial q$ is a function of both the robot shape and the obstacle shape, and the same holds true for $\partial p_{obs}/\partial q$. Therefore, the equations that describe the movement of p_{rob}, p_{obs} for each shape pair will be different (*i.e.*, the equations for a pair of spheres will be different from those for a sphere and a cylinder).

For obtaining the derivatives $(\partial p_{rob}/\partial q, \partial p_{obs}/\partial q)$, we require a set of equations that define p_{rob}, p_{obs} in terms of q and in the context of the shapes on which they exist. One such set of equations can be obtained by defining an optimization problem for which the solution is p_{rob}, p_{obs} and where the surface shapes of the robot and obstacle are expressed as constraints on the optimization [6], [7]. The general optimization problem is set up as follows:

$$\underset{p_{rob}, p_{obs}}{\text{Min}}\ f = \|p_{rob} - p_{obs}\|^2 \quad (23)$$

$$\text{subject to } h(p_{rob}, p_{obs}) = 0 \quad (24)$$

$$g(p_{rob}, p_{obs}) \leq 0 \quad (25)$$

where f is the objective function, $h(p_{rob}, p_{obs})$ are the equality constraints and $g(p_{rob}, p_{obs})$ are the inequality constraints. Not all of the inequality constraints apply to p_{rob}, p_{obs} at the same time, so the subset of active inequality constraints is denoted $g_i, i \in I$ where I is the set of indices of active inequality constraints. For compactness, the vector $x = [\ p_{rob}\ \ p_{obs}\ ]^T$ will be used in place of p_{rob}, p_{obs}.

The specific form of the constraints h, g depends on the particular shapes of the robot and obstacle. However, for all the shapes that can be modeled in this approach, the constraints are linear in x which allows them be factored as follows:

$$h(x) = Kx + b_1 \quad (26)$$

$$g_i(x) = Dx + b_2 \quad (27)$$

To obtain the solution $x = (p_{rob}, p_{obs})$ for the constrained optimization problem, the Kuhn-Tucker condi-

tions are solved:

$$\nabla f(x) + \lambda^T \nabla h(x) + \mu^T \nabla g_i(x) = 0 \quad (28)$$

$$h(x) = 0 \quad (29)$$

$$g_i(x) = 0, i \in I \quad (30)$$

Eqs. (28) – (30) can be written as

$$\begin{bmatrix} Q & K^T & D^T \\ K & 0 & 0 \\ D & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} = H \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ b_1 \\ b_2 \end{bmatrix} \quad (31)$$

where K, D, b_1, and b_2 are functions of the robot joint positions q, and Q is obtained from $\nabla f^T(p_{rob}, p_{obs}) = Q[\ p_{rob} \quad p_{obs}\ ]$.

To get $\partial p_{rob}/\partial q, \partial p_{obs}/\partial q$, we take the partial derivative of Eq. (31) with respect to q and rearrange it to obtain

$$\begin{bmatrix} \frac{\partial x}{\partial q} \\ \frac{\partial \lambda}{\partial q} \\ \frac{\partial \mu}{\partial q} \end{bmatrix} = H^{-1} \left\{ \begin{bmatrix} 0 \\ \frac{\partial b_1}{\partial q} \\ \frac{\partial b_2}{\partial q} \end{bmatrix} - \frac{\partial H}{\partial q} \begin{bmatrix} x \\ \lambda \\ \mu \end{bmatrix} \right\}. \quad (32)$$

We have expressions for H, b_1, b_2 as functions of q, so those partial derivatives are defined. The vector $[\ x \quad \lambda \quad \mu\ ]^T$ equals $H^{-1}[\ 0 \quad b_1 \quad b_2\ ]^T$, so the equation can be solved to obtain $\partial x/\partial q = [\ \partial p_{rob}/\partial q \quad \partial p_{obs}/\partial q\ ]^T$.

V. Simulation Results

Once our method was implemented in software, we chose a robot with which to test it. Initially, to check that the general contact model equations were correct, we duplicated some previous results with the hopping lamp simulation [1]. The general contact model gave the same final motion as the contact model specific to the hopping lamp, confirming that the equations were correct. This also showed that optimizations with the general contact model took the same amount of time, or very little more, to converge as optimizations with a contact model specific to the robot being simulated.

To see that the contact model was in fact general, we generated motions for the tumbler robot, a robot with three links actuated by two motors, as shown in Fig. 3. This is based on a real robot that has been built in our lab and is still undergoing design iterations. Its ongoing purpose is to serve as a testbed for the motions generated through this method. From the position shown in Fig. 3, each of the links with feet was allowed to rotate until they impinged on the middle link.

Three motions were generated for the tumbler with this method. These three motions, referred to as the handstand, the roll, and the downstand, can be concatenated in a particular order to form a longer sequence. Specifically, the handstand can be following by the downstand, or the handstand can be followed by the roll. The first of these patterns can be repeated in a loop, to form

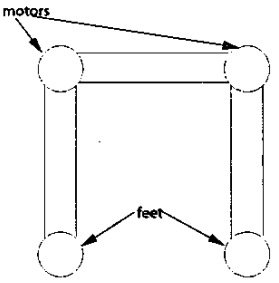

Fig. 3. The Tumbler Robot.

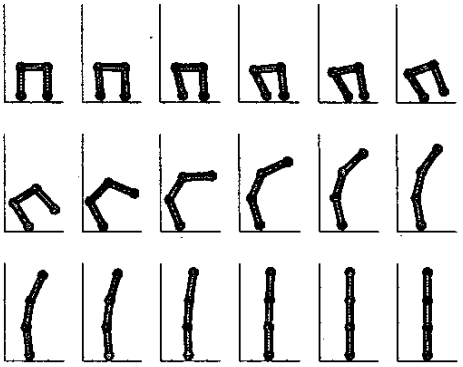

Fig. 4. The Tumbler Robot going up into a handstand.

a gait for the tumbler. The individual motions can be seen in Figs. 4 – 6. The optimizations took between six and thirty minutes to converge to a local minimum, depending on the motion. The simulations were run on a Pentium 700 MHz PC, integrating seven state equations and seventy-seven gradient equations for every iteration of the process.

The handstand motion was generated using the cost function

$$J = (q_3(t_f) - \frac{\pi}{2})^2 \quad (33)$$

where q_3 is the rotation of the middle link, and the position shown in Fig. 3 is where $q_3 = 0$. This cost function rewards motions that end with the middle link vertical, which resulted in the motion shown in Fig. 4. The initial and final positions for the motors, and thus the legs of the robot, are specified when the initial guess is made for the active joint paths $q^a(t)$, and is done so for all motions. The intermediate positions for the path $q^a(t)$ are defined by the a_i in (6) which are the parameters that are varied during the optimization. The roll motion was generated using the cost function

$$J = 50(q_1(t_f)) - q_3(t_f) \quad (34)$$

where q_1 measures translation in the horizontal direction and q_3 is the rotation of the middle link. The first term rewards motion that travels to the left, and the second

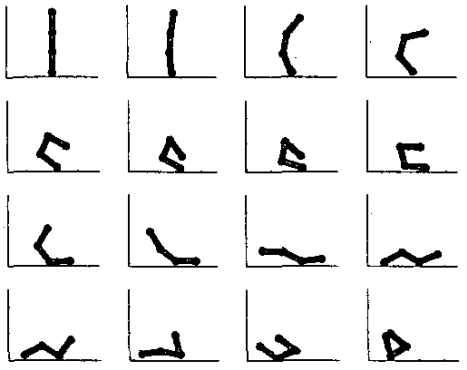

Fig. 5. The Tumbler Robot going from a handstand into a backwards roll.

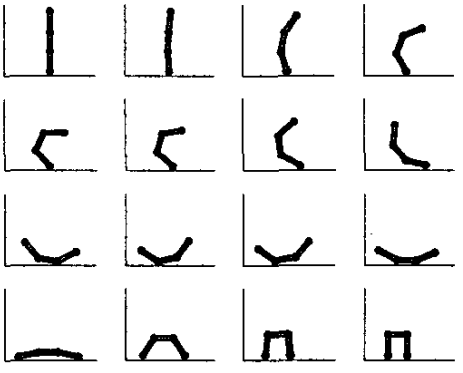

Fig. 6. The tumbler robot going from a handstand into a position with both feet on the ground.

term rewards rotation. The number of contact points with the ground change several times throughout the motion.

The downstand motion was generated using the cost function

$$J = 50(q_3(t_f) - \pi)^2 + \frac{1}{2}\int_0^{t_f} ||\tau^a||^2 dt \qquad (35)$$

where q_3 is the rotation of the middle link, which started this motion at a value of $\pi/2$, so this term encourages another rotation of $\pi/2$ from the initial position (or ending with the middle link horizontal again, but rotated π from the home position). The second term penalizes the amount of motor torque used to perform the motion, so it encourages a minimum amount of effort to be used in completing the motion.

VI. Conclusion

In this paper, we presented an optimal control approach to generating paths for robots, extended our contact model to apply generally rather than specifically, and discussed the derivatives that the general contact model in conjunction with the optimal control approach require. We simulated a tumbling robot to verify that this method is valid.

This research shows that it is possible to incorporate a general contact model into an optimal control approach to generating robot trajectories, and still have optimizations converge reliably to a local minimum with exact gradients of the cost function. Our contact model was able to resolve multiple contacts between a robot and its environment, and the use of smoothing functions on the contact forces did not distort the plausibility of the resulting motions. Despite adding some calculations to the computational burden of the optimization, the time taken to reach convergence was generally comparable to that of our previous work.

Our future research will involve using this approach to generate gaits for the tumbler robot mentioned here, and also to generate some motions for a hopping robot currently under construction in our lab. Another application for this method is simulation of walking robots, studying human locomotion, or finding trajectories for multiply-legged systems. An intelligent way to connect the pieces of motion generated with this approach is also an area of interest for our ongoing research.

VII. References

[1] J. V. Albro and J. E. Bobrow, "Optimal Motion Primitives for a 5 DOF Experimental Hopper", *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001.

[2] J. Betts, "Survey of Numerical Methods for Trajectory Optimization", *Journal of Guidance, Control and Dynamics*, vol. 21, 2, pps 193–207, 1997.

[3] A. Bryson and Y. Ho, *Applied Optimal Control: Optimization, Estimation and Control*, Hemisphere Publishing Corp., Bristol, PA., 1975.

[4] M. Buss, M. Glocker, M. Hardt, O. von Stryk, R. Bulirsch and G. Schmidt, "Nonlinear Hybrid Dynamical Systems: Modeling, Optimal Control, and Applications", *Modelling, Analysis and Design of Hybrid Systems*, vol. 279, pps 311–335, Springer-Verlag, 2002.

[5] Elmer G. Gilbert and Daniel W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles", *IEEE Journal of Robotics and Automation*, vol. RA-1, pps 21–30, 1984.

[6] E. G. Gilbert, D. W. Johnson and S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in a Three-Dimensional Space", *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pps 193–203, 1988.

[7] D. C. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Publishing Co., 1989.

[8] K. D. Mombaur, H. G. Bock, J. P. Schlöder and R. W. Longman, "Human-Like Actuated Walking that is Asymptotically Stable Without Feedback", *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pps. 4128–4133, 2001.

[9] J. P. Ostrowski, J. P. Desai, and V. Kumar, "Optimal gait selection for nonholonomic locomotion systems", *International Journal of Robotics Research*, vol. 19, no. 3, pps 225–237, 2000.

[10] F. C. Park, J. E. Bobrow and S. R. Ploen, "A Lie Group Formulation of Robot Dynamics", *The International Journal of Robotics Research*, vol. 14, no. 6, pps 609–618, December 1995.

[11] J. A. Reichler and F. Delcomyn, "Dynamics Simulation and Controller Interfacing for Legged Robots", *The International Journal of Robotics Research*, vol. 19, no. 1, pps 42–58, January 2000.

[12] G. A. Sohl, *Optimal motions for underactuated manipulators*, Ph.D thesis, University of California, Irvine, 2000.

[13] C. A. Tenaglia, D. E. Orin, R. A. LaFarge and C. Lewis, "Toward Development of a Generalized Contact Algorithm for Polyhedral Objects", *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Detroit, MI 1999.

[14] Y. Wang, V. Kumar and J. Abel, "Dynamics of Rigid Bodies Undergoing Multiple Frictional Contacts", *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pps 2764–2769, Nice, France, 1992.

[15] C-Y. E. Wang, W. K. Timoszyk and J. E. Bobrow, "Payload Maximization for Open Chained Manipulators: Finding Weightlifting Motions for a Puma 762 Robot," *IEEE Transactions on Robotics and Automation*, April 2001, Vol. 17, No. 2 pp:218-224.

[16] C-Y. E. Wang, J. E. Bobrow and D. J. Reinkensmeyer, "Dynamic Motion Planning for the Design of Robotic Gait Rehabilitation", to appear *ASME Journal of Biomechanical Engineering*.

[17] A. Witkin and M. Kass, "Spacetime constraints", *Computer Graphics (Proc. SIGGRAPH '88)*, Vol. 22, pps 159–168, 1988.

[18] K. Yamane and Y. Nakamura, "Dynamics Filter—Concept and Implementation of Online Motion Generator for Human Figures", *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pps 421–432, June 2003.