2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Differentially Flat Trajectory Generation for a Dynamically Stable Mobile Robot

Michael Shomin[1] and Ralph Hollis[2]

Abstract— **This work presents a method to find analytic and feasible trajectories for underactuated, balancing robots. The target application is fast computation of achievable trajectories for the ballbot, a dynamically stable mobile robot which balances on a single spherical wheel. To this end, assumptions are made to form the system as differentially flat, and a method of deriving feasible trajectories is shown. These trajectories are tracked on the real robot, demonstrating both point to point motions and recovery from nonzero initial conditions.**

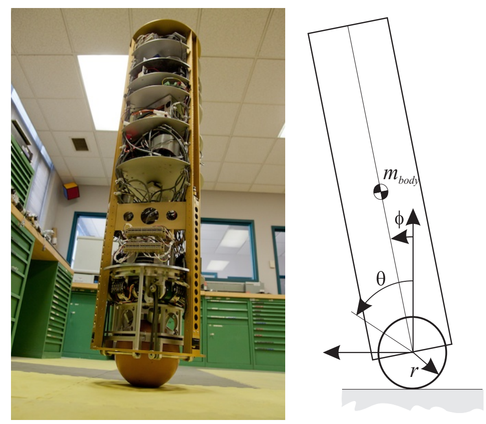
I. INTRODUCTION

The ballbot [1], [2] is an omnidirectional, dynamically stable mobile robot. It is a human-sized robot that balances on a single spherical wheel. As an underactuated, shape-accelerated system, it moves by leaning and cannot directly control its position. Similar to the classical controls problem of a cart-pole inverted pendulum, but with a few key differences discussed later, the ballbot presents many interesting dynamics and controls problems. Ballbot provides many advantages over a statically-stable robot [2] such as inherent compliance and small footprint. Being tall and thin allows easier navigation amongst cluttered human environments.

Navigation is a unique challenge with such a robot, as the planning and control are tightly coupled. Previous work has shown that lean-forward, lean-back motion can create straight line motions with segments of constant velocity [3]. More complicated motions can be realized through *a priori* nonlinear least-squares minimization to find feasible motions [4], [5] which can then be sequentially composed for planning and navigation [6]. Navigation can then be achieved through graph search across the space of the motion primitives. Although this method has proven effective, it is restricted to the set of motions computed offline. Therefore, movements to arbitrary locations or recovering from poor initial conditions cannot be handled. This motivates the need for a method of trajectory generation that can quickly create feasible trajectories to and from arbitrary conditions. This paper shows that this can be achieved through a relaxation of the dynamic equations and a transformation into a lower dimensional flat output space. The equations of motion of the ballbot are introduced first, then the system is linearized and flat outputs are found. Trajectory generation via the flat outputs will then be explored, followed by control of the system, experimental results, and conclusions.

*This work was supported by NSF Grant IIS-11165334.

[1]M. Shomin is a PhD Candidate at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 15213, USA mshomin at cmu.edu
[2]R. Hollis is a Research Professor, also at the Robotics Institute rhollis at cs.cmu.edu



(a) (b)

Fig. 1. Ballbot: (a) balancing on its single spherical wheel, (b) planar ballbot model notation diagram.

II. LINEARIZATION AND FLAT OUTPUTS

A. Equations of Motion

Before deriving the flat outputs for the ballbot, the equations of motion must be examined. Henceforth, for the purpose of derivation and simplification, the 2-D ballbot model will be used, shown in Fig. 1(b). The model has two degrees of freedom: lean angle and ball angle with one control input, the torque between ball and body.

As computed in [3], the planar model equations of motion are as follows, with lean angle ϕ, ball angle θ, height of the center of mass (from the center of the ball) l, radius of the ball r, mass of the sphere m_s, moment of inertia of the sphere I_s, mass of the body m_b, moment of inertia of the body I_b, and gravitational constant g:

$$M\ddot{q} + C\dot{q} + D + G = U, \quad (1)$$

$$q = \begin{bmatrix} \theta \\ \phi \end{bmatrix}, M = \begin{bmatrix} I_s + m_b r^2 + m_s r^2 & -m_b l r \cos\phi \\ -m_b l r \cos\phi & m_b l^2 + I_b \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & m_b \dot{\phi} l r \sin\phi \\ 0 & 0 \end{bmatrix}, G = \begin{bmatrix} 0 \\ -m_b g l \sin\phi \end{bmatrix}, U = \begin{bmatrix} \tau \\ -\tau \end{bmatrix}.$$

The contact point of the ball with the floor is simply $x = r\theta$, so the ball angle should be considered as the position in space of the robot, which the system essentially has direct control over, while the lean angle is unactuated. D is the vector of frictional terms and is omitted for the purpose of flat output derivation. U is the vector of control inputs. In this system the only control input is torque between body and ball. Eq. 1 yields two equations. Adding these two equations and replacing the second equation with the result yields:

$$M'(q)\ddot{q} + C'(q,\dot{q}) + G'(q) = \begin{bmatrix}\tau \\ 0\end{bmatrix}. \tag{2}$$

This formulation is useful as the control input (torque) does not appear in the second equation. The second equation from Eq. 2 is referred to as the internal system dynamics and is equal to zero.

B. Differential Flatness

Differential Flatness [7] is a property of some systems that can reduce the complexity of trajectory generation [8]. In defining a flat system, the goal is to find "flat output" variables, which are defined as functions of the state variables and their derivatives. There are as many flat outputs as control inputs to the system. For a flat output to be feasible, the state variables and control inputs must have a functional mapping from the flat outputs S and their derivatives:

$$S = Y(x, \dot{x}, \ddot{x}, ..., x^{(r)}) \tag{3}$$
$$x = A(S, \dot{S}, \ddot{S}, ..., S^{(\beta)}) \tag{4}$$
$$u = B(S, \dot{S}, \ddot{S}, ..., S^{(\beta+1)}). \tag{5}$$

In other words, if x is vector of state variables, the functions Y, A, and B must exist with finite r and β. If flat outputs exist for a given system, trajectory generation may become simpler, as trajectories of the flat output variables yield trajectories for the state variables and the control inputs. The flat output trajectory must simply have enough derivatives to satisfy the mapping back to the state variables and control inputs. In the case of a system like the planar ballbot, there will be one flat output variable corresponding to the one control input. A trajectory in this variable will map to trajectories in lean angle and ball position. Therefore, if a flat output can be found, the task of generating a feasible trajectory will be reduced to simply finding a trajectory in the flat output with sufficient derivatives. In the general case, the flat output trajectories can have any basis. For the purpose of easily satisfying boundary conditions and having finite smooth derivatives, polynomials are used for this work.

C. Flat Outputs

Given the ballbot equations of motion in Eq. 2, some assumptions must be made to modify the equations such that flat outputs can be defined. Previous work has been done in finding flat outputs for a cart-pole [9] with similar assumptions. The major difference in these systems is that the ballbot has extra terms due to the control input being torque between body and ball, and not just on the ball. The force on the cart in the inverted pendulum does no virtual work on the pole, simplifying the equations slightly, whereas the generalized force in the planar ballbot does virtual work on both body and ball.

To find flat outputs, the system must be linearized. Because the robot's lean angle is generally bounded $-5^\circ < \phi < 5^\circ$ in normal operation, it is reasonable to use the small angle approximation about equilibrium, *i.e.*, $\sin\phi \approx \phi$ and $\cos\phi \approx 1$. This makes the Eq. 2 become

$$\tau = \alpha\ddot{\theta} + (\alpha+\beta)\ddot{\phi} - \beta\phi\dot{\phi}^2 \tag{6}$$

and

$$0 = (\alpha+\beta)\ddot{\theta} + (\alpha+\gamma+2\beta)\ddot{\phi} - \beta\phi\dot{\phi}^2 + \frac{\beta g \phi}{r}, \tag{7}$$

with $\alpha = I_{\text{ball}} + (m_{\text{ball}} + m_{body})r^2$, $\beta = m_{\text{body}} r l$, $\gamma = I_{\text{body}} + m_{\text{body}} l^2$, each constants.

Next, to further linearize the system, the term $\beta\phi\dot{\phi}^2$ must be assumed negligible. Intuition dictates that since ϕ should be nominally zero and $\dot{\phi}$ should be small, this term should be very small, as it is the product of three small quantities. With this assumption, Eq. 6 becomes:

$$\tau = \alpha\ddot{\theta} + (\alpha+\beta)\ddot{\phi} \tag{8}$$

and Eq. 7, the internal dynamics, becomes:

$$\underbrace{r(\frac{\alpha}{\beta}+1)}_{\lambda_1}\ddot{\theta} + \underbrace{\frac{r}{\beta}(\alpha+\gamma+2\beta)}_{\lambda_2}\ddot{\phi} = g\phi, \tag{9}$$
$$\lambda_1\ddot{\theta} + \lambda_2\ddot{\phi} = g\phi. \tag{10}$$

λ_1 and λ_2 are constants. With this equation, the flat output S is defined as:

$$S = \lambda_1\theta + \lambda_2\phi, \tag{11}$$

with derivatives:

$$\dot{S} = \lambda_1\dot{\theta} + \lambda_2\dot{\phi}, \tag{12}$$
$$\ddot{S} = \lambda_1\ddot{\theta} + \lambda_2\ddot{\phi} = g\phi \tag{13}$$

from Eq. 10, and

$$\dddot{S} = g\dot{\phi} \tag{14}$$
$$S^{(4)} = g\ddot{\phi}. \tag{15}$$

This mapping is chosen after recognizing the relationship between ϕ, $\ddot{\theta}$, and $\ddot{\phi}$ in Eq. 10. This gives the mapping of the two state variables θ and ϕ to the flat output variable S in Eq. 11. Four derivatives of S are shown because these are required to get the inverse mapping to the state variables and their derivatives present in Eqs. 6 and 7. This is also necessary because for the flat output to be valid, there must be a mapping to the control input τ. Because $\ddot{\phi}$ appears in the equation for τ, a mapping to $\ddot{\phi}$ is required. Leaving g on the right hand side, outside of the λ constants emphasizes that the system is shape accelerated, meaning it's acceleration comes from nonzero lean angle. It should be noted that with these four derivatives of S, there is a mapping from both the state variables and a finite number of derivatives to S and a mapping from S and a finite number of derivatives to the state variables and the control input. Therefore, S is a flat output of the linearized system in Eqs. 8 and 9. Using this relation, trajectory generation can now be simplified.

III. TRAJECTORY GENERATION

The ballbot is accelerated by its lean angle ϕ, but the more important state variable in high level planning is the ball angle θ, as it can be thought of as the global position of the robot. From the previous section, it has been shown that from a flat output trajectory, the feasible θ trajectory can be found for the linearized system in Eqs. 8 and 9. In the control law derived in the next section, it will be shown that τ depends on the fourth derivative of S; as such S needs to be at least C^4 for a point-to-point motion, with desired initial and final configurations, making ten conditions. With these constraints and the choice of a polynomial basis, a nonic (ninth order) polynomial is needed. After constructing this polynomial S_d from the constraints, the state variable trajectories are simply found by solving Eqs. 11 and 13 for θ and ϕ:

$$\theta^* = \frac{1}{\lambda_1}\left(S_d - \frac{\lambda_2}{g}\ddot{S}_d\right) \quad (16)$$

$$\phi^* = \frac{\ddot{S}_d}{g}. \quad (17)$$

This means that given a smooth function in S (in this case a nonic polynomial), a feasible trajectory in θ^* and ϕ^* can be easily calculated which satisfies the internal system dynamics. Because there is a mapping in the opposite direction (Eq. 11), the S trajectory can be constructed with initial and final conditions given by state variables. Fig. 2 shows an example of a rest-to-rest motion and feasible θ trajectory that comes out. S has been rescaled by λ_1 for the purpose of comparison. This is a rather agressive trajectory, moving the robot two meters in three seconds. As such, a very large lean angle is required. As can be seen from the θ output trajectory in Fig. 2, the ball must first roll back before moving forward to achieve that lean angle. It then overshoots and comes back to achieve the lean back required for deceleration. The fact that this behavior comes out naturally from the flat output trajectory is a significant advantage of this method. The polynomial in S was constructed from initial conditions of $\theta_0 = 0, \dot{\theta}_0 = 0, \phi_0 = 0, \dot{\phi}_0 = 0, \ddot{\phi}_0 = 0$ and final conditions $\theta_f = rx_f = 2r \approx 24\,\text{rad}, \dot{\theta}_f = 0, \phi_f = 0, \dot{\phi}_f = 0, \ddot{\phi}_f = 0$.

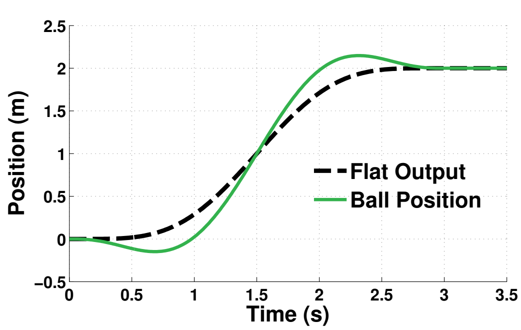


Fig. 2. Desired flat output trajectory and corresponding feasible θ trajectory.

As stated previously, the flat output trajectory is a nonic polynomial, and in the previous example shown in Fig. 2, ten constraints were used to generate it. However, for many cases, it may be beneficial to leave some of those parameters free. In a quick stopping trajectory, for example, the final position may not be important, but coming to zero velocity quickly is important. As seen in quadrotor trajectory generation [10], smooth, well-formed, (in the sense of the control input) trajectories can be found by minimizing the snap (fourth derivative of position) over the trajectory. The intuition for this cost function comes from Flash and Hogan [11] who found that human arm trajectories minimize jerk (third derivative of position). The quadrotor model control laws contained one higher derivative than the human arm model, so instead of jerk, snap was minimized. The planar ballbot model contains one higher derivative still: $S^{(4)}$, the snap of the flat output, which appears in the control law. As such, the cost function used in this work finds feasible trajectories with free parameters by minimizing $S^{(5)}$, the crackle of the flat output. Note also that $S^{(5)}$ is proportional to $\dddot{\phi}$, so this is equivalent to minimizing the jerk of the lean angle, which has been found empirically to produce good trajectories in previous work [3].

Minimizing $S^{(5)}$ over the whole trajectory can be formulated as a quadratic program (QP), since S is simply a ninth order polynomial:

$$\begin{aligned} \min \quad & c^T H c \\ \text{s.t.} \quad & Ac \leq b. \end{aligned} \quad (18)$$

where c is a vector of the polynomial coefficients and non-free constraints become equality contraints $Ac = b$ in the QP. H is a positive definite matrix and as such, the QP can be solved in polynomial time. This allows for real time, optimal trajectory generation online. The ballbot updates its controller at 500 Hz, and using this formulation, the QP can be solved in less than 1 ms, allowing for trajectory generation in two orthoganol directions in one controller update step.

One huge advantage of the flat output trajectories is that they can be scaled spatially and temporally since they are simply polynomials, compared with numerical trajectories which have no such property. This means that any safety requirements such as a maximum lean angle can be satisfied by lengthening the time of a trajectory, since all the derivatives of S go to zero as t goes to infinity. When temporally rescaled, derivatives of S simply have a scale factor of $(t_{old}/t_{new})^k$ where t_{new} is new time length of the trajectory, t_{old} is the original, and k is the derivative to be scaled. Note that this scales the initial and final conditions (other than the zero derivative) if they are nonzero. This means that the time length of the trajectory must be known *a priori* if the initial or final derivatives are nonzero. Scaling time is also useful for numerical stability in solving the QP. Since S is a nonic polynomial in t, t^9 appears in the minimization. As such, using unit time of $0 < t < 1$ keeps the numerical solution from going unstable.

A. *Rest-to-Rest Motions*

Possibly the most useful for normal navigation, rest-to-rest motions are trajectories which begin and end with all flat output derivatives equal to zero. An example motion can

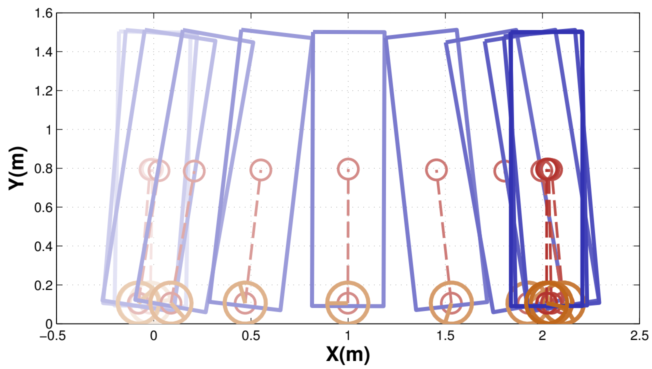


Fig. 3. MATLAB simulation of an aggressive rest to rest motion.

be seen in the 2-D simulation, shown in Fig. 3. Notice that to achieve forward motion, the ball first must roll backwards to achieve a positive lean angle. This simulated motion is the same ball trajectory shown in Fig. 2. Rest-to-rest motions were previously explored in [3] and [6]. A comparison of the differentially flat approach to the approach in [6] can be seen in Fig. 4. The previous approach utilized minimization of the deviation from the nonlinear internal system dynamics. Ball position and velocity are within 1% for the whole trajectory while the lean angle trajectory differs by a maximum of 12%. This difference could be attributed to the differences between the linearized and nonlinear system. As will be shown in Section V, this difference in negligible in practice. The key difference between these methods is that the differentially flat trajectory can be computed in less that 1 ms, allowing for online generation, unlike nonlinear minimization which must be done offline.

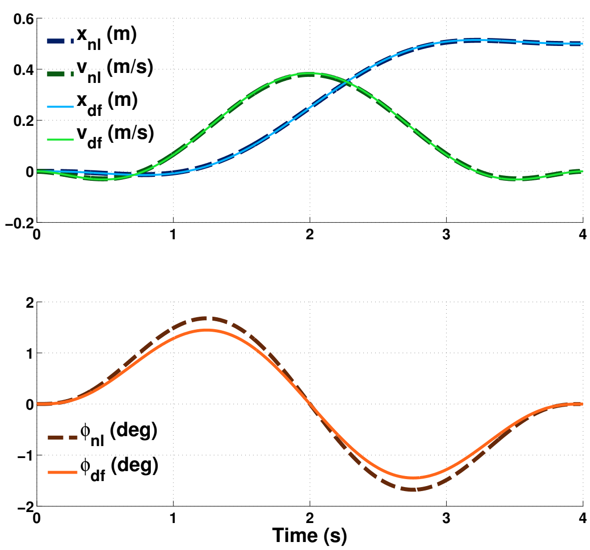


Fig. 4. Comparison of Differentially Flat (df) Trajectory with corresponding trajectory found via offline nonlinear (nl) minimization

B. Arbitrary Motion and Free Constraints

Differentially flat trajectory generation scales to more complex situations than rest-to-rest motions. Examples include recovering from an outer loop controller going unstable or significant disturbances. Generating a feasible trajectory in a single controller time step before applying torque to recover allows for feasible feedforward terms. This means less vulnerability to higher order modes becoming excited. One current drawback of the method is that the length of time the trajectory will take must be known a prior to satisfy the initial conditions. In practice, it has been found that trajectories of 4 s or less can recover from any reasonable condition without exiting the safe domain of the balancer. Possible better solutions will be discussed in Section VI. In situations when recovery is important but the final position is not, the constraints on the final ball angle are left free. This is a significant advantage of the QP formulation, as it can still construct the polynomial within one time step, but also find the best smooth path that minimizes lean angle jerk without regard to final location.

IV. CONTROL

A. Inverse Dynamics Control

Using the flat output formulation, it is reasonably straightforward to derive an inverse dynamics control law. The equation for feedforward torque was shown in Section II to show differential flatness. From the external system dynamics (Eq. 6), the control input for a given desired state is

$$\tau = \alpha\ddot{\theta} + (\alpha + \beta\cos\phi)\ddot{\phi} - \beta\dot{\phi}^2\sin\phi. \tag{19}$$

Although the linearized version of the internal dynamics was used to derive the flat outputs, the nonlinear version is used here because as much information as is known about the system should be used to get the control input. From [9], the feedback terms of the control law can be wrapped into $\ddot{\phi}$, replaced with $\ddot{\phi} = \frac{\nu}{g}$ where $\nu = gS^{(4)} + K_e$. $gS^{(4)}$ is the original feedforward term and K_e incorporates P feedback. The vector of errors in the flat outputs is e, and K is vector of gains. This becomes an inverse dynamic controller with feedforward terms on $\ddot{\theta}$ and ϕ, $\dot{\phi}$,$\ddot{\phi}$, and feedback on θ, $\dot{\theta}$,ϕ, and $\dot{\phi}$. Although this controller showed good performance in MATLAB simulation, it was not tested on the robot, as will be discussed in Section VII.

B. Cascaded PID

Currently, the ballbot uses an inner and outer loop control scheme as discussed extensively in [6]. A PID controller is run on the desired lean angle, with an outer loop PD controller for trajectory following. This method has the advantage of placing limits on lean angle and ensuring that as long as the balancing controller is stable within the lean angle limits, the robot will not fall over. Although this may not yield the same performance as an inverse dynamics or LQR controller, having an inner loop balancer is a more robust approach as substantiated by experiments in [12]. As such, this method of control is utilized to test the feasibility of the generated trajectories. An inner loop balancer has the notion built in of staying upright as the most mission-critical task.

V. RESULTS

A. Experimental Setup

Testing of the trajectories generated via the methods presented was done on the ballbot. The system has a low-level computer running the QNX real-time operating system controlling the inner-loop balancer, driving the motors and reading data from the IMU. The four motors drive the ball via an inverse mouse-ball drive [1].

Lean angles are commanded via socket communication from another onboard computer, running Ubuntu Linux. This higher level computer handles all tasks that are not real-time critical. For this work, it was used for generating the trajectories and communicating with a laptop to start and stop trajectories. This machine runs ROS [13] for communication between processes; ease of visualization; and in other applications, mapping and localization.

For the purpose of visualizing the trajectories, a strip of LEDs was attached vertically to the side of the robot. Each of the 45 exposed RGB LEDS can be commanded to a 7-bit value via an I^2C bus. A ROS node sends message via serial to an Arduino microcontroller which then commands the LED chain. By taking a long-exposure photograph, a trail of the robot can be seen throughout the trajectory. This technique has been used for tracing robot paths before [14], [15] and has proven effective as a tool to visualize a dynamic trajectory.

B. Rest-to-Rest Motion

Rest-ro-rest motions for distances of 0.5 m to 2.0 m were run with the robot. Time was varied to affect the maximum lean angle achieved in the trajectory. Maximum angles from $1.2°$ to $5°$ were achieved. Examples of this motion can be seen in the attached video. A characteristic trajectory can be seen in Fig. 5 with the corresponding tracking data in Fig. 6(a). In Fig. 5, the LEDs on the robot produce light trails and pulse brightly at 3 Hz. The pulses transition from red at the beginning of the trajectory to green at the end. Of note is the significant lean forward and lean back behavior at the start and end. The necessity to roll the ball backward to lean forward and vice versa can also be seen from this image. The maximum angle achieved in this trajectory is over $5°$, which corresponds to an instantaneous acceleration of over 2 m/s^2. This is a significant lean angle for the ballbot.

All of the trajectories attempted were run open loop, using only the feedforward lean angle for the balancer. This was done simply to show this method is a feasible means to create ballbot trajectories. This also allowed a look at how good feedforward-only performance can be. Feedback on the lean angle can also produce discontinuities, as a P term on position will give a discontinuity at the beginning of a trajectory if there is any deviation from the starting condition. Despite the lack of feedback, the tracking is very acceptable, which shows that the trajectories are certainly feasible. For the trajectory shown in Fig. 6(a), the maximum velocity tracking error is 15% with the trajectory slightly undershooting, but ending 10% short of the goal. This error

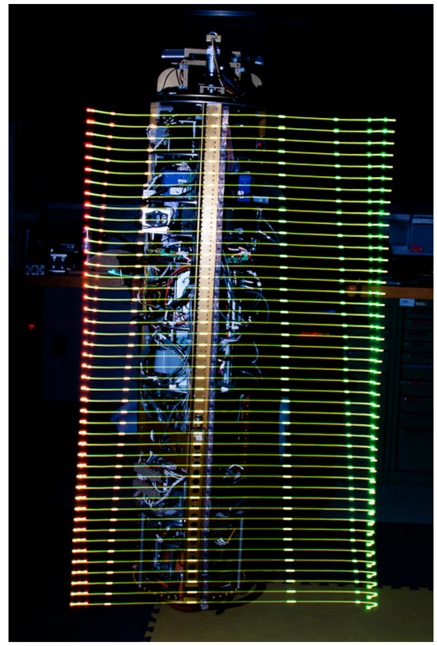


Fig. 5. Long exposure photograph of rest-to-rest motion with a $5°$ lean. LED pulses are at 3 Hz with continuous low level orange light. A flash was used at approximately 1.5 s to illuminate the robot.

could be significantly reduced with feedback in the future, possibly with a variant of the controller demonstrated in the past [3].

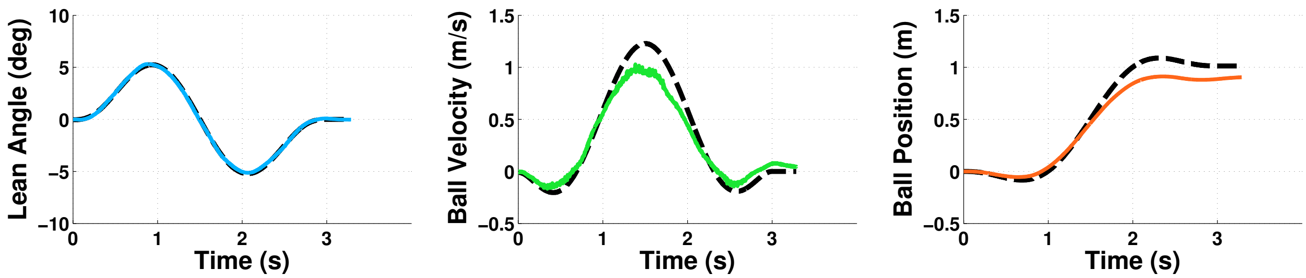
C. Recovery Trajectories

Feasibility of recovery trajectories was accessed by testing recovery from both an initial lean angle and an initial velocity. A successful recovery from a $2°$ lean angle can be seen in Fig 6(b). The lean angle was slowly ramped up to $2°$ while the robot was held in place. The robot was let go, and it began to accelerate. After being let go, the trajectory planner generated a stopping trajectory and executed it. This has been attempted 6 times, all successful recoveries. This motion can be seen in the attached video. Recovery from initial velocity was also tested by pushing the robot and generating a stopping trajectory. This was also successful in bringing the robot safely to a stop from initial velocities up to 1 m/s.

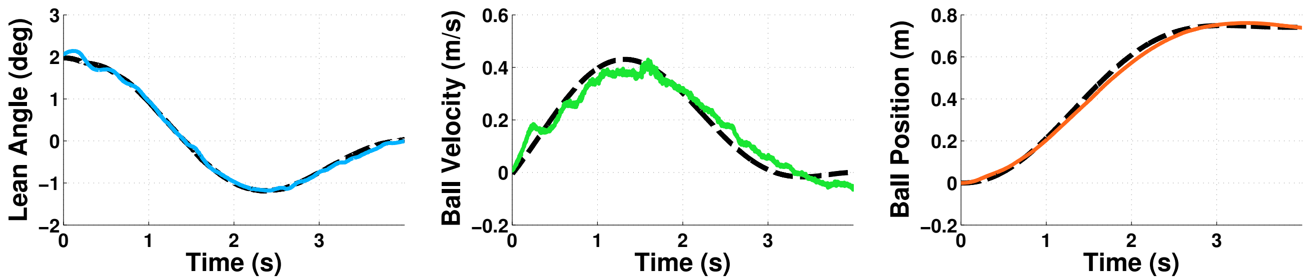
This recovery motion can be compared to an outer-loop station keeping controller or simply the balancing controller trying to achieve a $0°$ lean angle from the initial offset. Unfortunately, even with soft gains, the experimental system has not been able to recover stably by simply using the balancer. As such, this trajectory-based method of recovery is not only feasible, but also very useful, as there isn't another good option for these scenarios.

VI. CONCLUSIONS

A method of fast, analytic trajectory generation for dynamically stable balancing robots has been presented. Relaxing the equations of motion to form a differentially flat system allows for analytic generation of feasible trajectories. Furthermore, the method has been extended to use free constraints



(a) Rest-to-rest motion from $x = 0$ to $x = 1m$ in $4s$. Max lean angle is approximately 5 degrees.



(b) Recover from an initial lean angle of 2 degrees. Recovery takes 4 seconds and

Fig. 6. Lean angle, velocity, and position data from a rest-to-rest motion and a recover motion. Desired trajectories are shown in dashed black with the colored lines showing actual data. Both experiments were done open loop with only feedforward lean angles given to the balancing controller.

for recovering from poor initial conditions. This method was experimentally verified with the ballbot, a single-wheeled balancing system. The derivation of this technique also confirmed some empirically-found notions from previous work [3]. These include using a nonic polynomial for trajectories and minimizing the jerk of lean angle.

VII. FUTURE WORK

The method can be extended with closed loop position feedback. The possibility of solving the QP iteratively can also be explored in an attempt to avoid specifying a time length before generating trajectories. This work could also be extended to piecing trajectories together to form more complicated paths. Planning could be done between keyframes on the path with lean angle jerk minimized over the total path. Finally, the trajectories presented could be used as motion primitives. Since they are so quick to compute, the set of motion primitives could be updated dynamically to adjust to the needs of the system.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant IIS-11165334. The authors thank Umashankar Nagarajan and Paul Vernaza for their assistance in running experiments.

REFERENCES

[1] T. Lauwers, G. Kantor, and R. Hollis. A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2884 –2889, may 2006.

[2] R. Hollis. Ballbots. *Scientific American*, 18:72–78, October 2008.

[3] U. Nagarajan, G. Kantor, and R. Hollis. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3743 –3748, may 2009.

[4] U. Nagarajan. Dynamic constraint-based optimal shape trajectory planner for shape-accelerated underactuated balancing systems. In *Proceedings of 2010 Robotics: Science and Systems*, June 2010.

[5] U. Nagarajan, G. Kantor, and R. Hollis. Hybrid control for navigation of shape-accelerated underactuated balancing systems. In *CDC'10*, pages 3566–3571, 2010.

[6] U. Nagarajan, G. Kantor, and R. Hollis. Integrated planning and control for graceful navigation of integrated planning and control for graceful navigation of shape-accelerated underactuated balancing mobile robots. In *Robotics and Automation, 2012. ICRA '12. IEEE International Conference on*, page to appear, may 2012.

[7] M. Fliess, J. Lévine, and P. Rouchon. Flatness and defect of nonlinear systems: Introductory theory and examples. *International Journal of Control*, 61:1327–1361, 1995.

[8] H. Sira-Ramírez and S. Agrawal. *Differentially Flat Systems*. Control Engineering. Taylor & Francis, 2004.

[9] A. Mansour and H. Jerbi. An exact differential flatness control for a non minimum phase model of an inverted pendulum. In *Proceedings of the 2009 Third Asia International Conference on Modelling & Simulation*, AMS '09, pages 465–469, Washington, DC, USA, 2009. IEEE Computer Society.

[10] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[11] T. Flash and N. Hogans. The coordination of arm movements: An experimentally confirmed mathematical model. *JOURNAL OF NEUROSCIENCE*, 5:1688–1703, 1985.

[12] U. Nagarajan, B. Kim, and R. Hollis. Planning in high-dimensional shape space for a single-wheeled balancing mobile robot with arms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 130–135, 2012.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[14] S. Kuznetsov, E. Paulos, and M. Gross. Wallbots: interactive wall-crawling robots in the hands of public artists and political activists. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, DIS '10, pages 208–217, New York, NY, USA, 2010. ACM.

[15] M. Shomin and J. Fiene. Teaching manipulator kinematics by painting with light. *ASME Conference Proceedings*, 2011(54839):677–683, 2011.