Robot Limbo: Optimized Planning and Control for Dynamically Stable Robots Under Vertical Obstacles

Kasemsit Teeyapan, Jiuguang Wang, Tobias Kunz, and Mike Stilman

***Abstract*— We present successful control strategies for dynamically stable robots that avoid low ceilings and other vertical obstacles in a manner similar to limbo dances. Given the parameters of the mission, including the goal and obstacle dimensions, our method uses a sequential composition of IO-linearized controllers and applies stochastic optimization to automatically compute the best controller gains and references, as well as the times for switching between the different controllers. We demonstrate this system through numerical simulations, validation in a physics-based simulation environment, as well as on a novel two-wheeled platform. The results show that the generated control strategies are successful in mission planning for this challenging problem domain and offer significant advantages over hand-tuned alternatives.**

***Index Terms*— dynamic limbo, sequential controller composition, stochastic optimization, two-wheeled balancing robot**

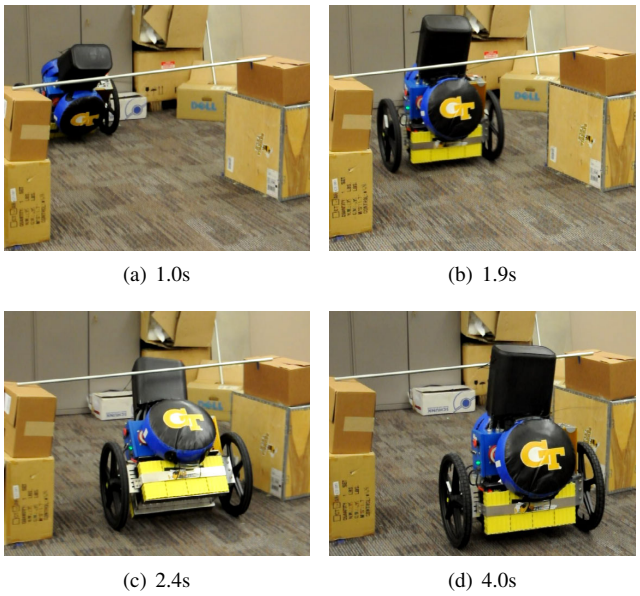
I. Introduction

Search and rescue robots that enter disaster areas will need to go around fallen debris, go over rubble and go under partially collapsed supports and hanging wires. The former two types of navigation can be solved by existing algorithms in motion planning [1–3] with stable and adaptive control [4, 5]. However, *passing under obstacles* remains a challenging open problem. We present multiple solutions for dynamically stable robots that navigate underneath obstacles. Furthermore, we use stochastic optimization to choose parameters for a sequence of controllers and times to switch between the individual controllers.

Mobile manipulators and other tall robots with multiple wheels such as Pearl [6], Xavier [7], and Minerva [8] remain balanced due to their statically stable support structures. However, they cannot naturally duck under obstacles since they easily become dynamically unstable when the workspace is steep, or when the robots make abrupt changes to their velocity. Furthermore, if the workspace is limited, their navigation ability may be restricted due to a larger base of support or a greater turn radius.

In contrast, robots like the Segway RMP [3], JOE [9], uBot [10], Robonaut [11], and Ballbot [12] are dynamically stable. Their method of stabilization is similar to that of humans, allowing greater flexibility in control. When such systems are affected by external disturbances, they dampen the oscillations and gradually return to an equilibrium state. These robots are consequently more robust to external forces as well as rapid acceleration and deceleration. With only one or two wheels, they possess a near-zero turn radius for moving in a limited space. We show that dynamically stable robots can also dynamically pass under obstacles.

The authors are with the Center for Robotics and Intelligent Machines (RIM) at the Georgia Institute of Technology, Atlanta, Georgia, 30332, USA. Email: {kasemsit, j.w, tobias, golem}@gatech.edu



(a) 1.0s (b) 1.9s (c) 2.4s (d) 4.0s

Fig. 1. A two-wheeled robot executing the hand-tuned hybrid limbo motion.

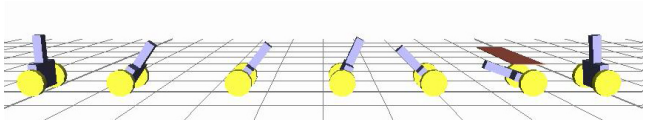


Fig. 2. Time-lapsed simulation of the optimized hybrid limbo motion.

This paper explores a novel autonomous planning algorithm that allows a two-wheeled robot to generate a series of motions to move or duck under an obstacle. The robot is shown in Fig. 1. When the obstacle is not movable, a statically stable robot might not accomplish this particular task. With the dynamically stable capability, the robot's choice is either to lean forward or backward while it is beneath the obstacle. In the latter case, the action is similar to limbo, a West Indian dance in which a dancer leans backward to go under a fixed-height stick.

To avoid the vertical obstacle, the robot motion is divided into two stages and controlled by two separate controllers. We present a hybrid controller as a sequential composition [13] of two controllers that both use input-output (IO) feedback linearization [14] with different gains and references. Consequently, this is a planning task where stochastic optimization is used to decide the optimal switching boundary and the parameters of each controller.

The rest of the paper is organized as follows: Section III provides the state-representation for the system dynamics. Section IV describes the controller design based on feedback linearization. Section V gives an overview of the planner formulated using stochastic optimization. The simulation and experimental results are presented in Section VI while the preliminary test done on the robot platform is described in Section VII. Section VIII gives the concluding remarks and directions for future work.

II. Related Work

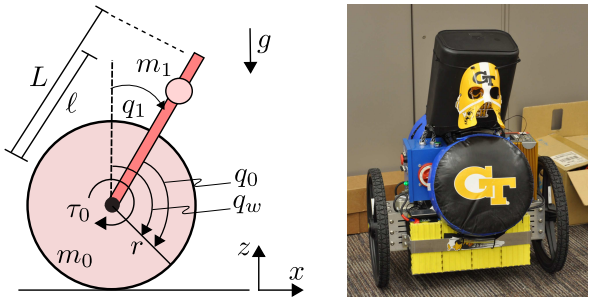
There exist numerous research studies on two-wheeled manipulators. However, they primarily focus on dynamics and balance [9, 10, 15]. Since two-wheeled platforms are underactuated they do not lend themselves to efficient control methods such as *feedback linearization*. [16] showed that linearization can be done around the active or passive joints, referred to as collocated and non-collocated partial feedback linearization. Yet, the general control problem remains an active topic of investigation. Recent work describes position and velocity control [17], pose control [18], as well as adaptive approaches to motion control [19]. However, it does not deals with motion strategies for robots that move among obstacles. We are particularly interested in cases where the obstacles are in the vertical position, hanging over the workspace.

Recent work has demonstrated limbo with a humanoid robot [20]. By applying genetic algorithms, this approach succeeded in designing joint motions without significant study of robot dynamics. Due to the complexity of the platform, the generated motion was relatively slow and potentially inefficient. Performance can be improved with two-wheeled robots because of their increased agility. Robots with wheels can achieve faster movements that can be combined to complete maneuvers.

Although two-wheeled balancing platforms are well-known to be nonlinear, a linear-quadratic (LQ) regulator is widely used to solve the balancing problem as presented in [15]. Since the controller is based on the linearized system dynamics, the performance is reliable only around the equilibrium. Partial feedback linearization is an alternative control scheme that was explored in [17] and [18]. This approach is more applicable to nonlinear systems, particularly when trying to control the tilt angle of the robot. We apply the less complex input-output feedback linearization (IO-linearization) which is adequate to control the tilt angle of the robot as well as its position. We introduce a sequential composition technique that merges a series of IO-linearization controllers. Our method lowers computational cost while providing better accuracy than other controllers based on linear approximations of the dynamic system.

III. Dynamic Model

In this section, the dynamics of the robot are derived using Lagrangian mechanics [21]. To simplify the problem, we use the robot schematic shown in Fig. 3 corresponding to the parameters in Table I. The subscripts 0 and 1 refer to the robot's wheels and body, respectively. The wheels have



(a) Schematic (b) Golem 1

Fig. 3. The schematic (a) of our two-wheeled robot (b).

TABLE I
List of symbols

m_0	Mass of the wheels
m_1	Mass of the body
I_0	Inertia of the wheels around the center of mass
I_1	Inertia of the body around the axle of the wheels
r	Wheel radius
ℓ	Distance from the wheel axis to the body's center of mass
L	Distance from the wheel axis to the top of the body
g	Gravity
q_w	Rotation angle of the wheels w.r.t the world frame
q_0	Rotation angle of the wheels w.r.t the robot body
q_1	Tilt angle of the robot body
τ_0	Motor torque

radius r and are driven by DC servo motors with encoders. The posture of the complete system is described by the angular position of the wheels q_w and the inclination angle of the robot's body q_1. However, for our system, the position is measured by encoders on the motors. What we obtain is therefore the angular position of the wheels relative to the body, which we call q_0. The relation between q_w and q_0 is simply $q_w = q_0 + q_1$.

In our problem formulation, we choose the generalized coordinates of the system as $\mathbf{q} = [q_0, q_1]$ where $q_0 \in \Re$ and $q_1 \in (-\frac{\pi}{2}, \frac{\pi}{2})$. We also define the generalized forces $\tau = [\tau_0, 0]$ such that $\tau_0 \in \Re$ is the torque applied to the wheels. The equations of motion are obtained from Lagrange's equation,

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathscr{L}}{\partial q_i} = \tau_i, \qquad i = 0, 1 \tag{1}$$

where $\mathscr{L}$ is the Lagrangian function and is computed from the difference between the kinetic energy and the potential energy of the system as $\mathscr{L} = T - U$. The resulting dynamic equations for our system are as follows:

$$\tau_0 = (\ddot{q}_0 + \ddot{q}_1)I + m_1 r\ell\ddot{q}_1 \cos q_1 - m_1 r\ell\dot{q}_1^2 \sin q_1 \tag{2}$$

$$0 = (I + I_1 + 2m_1 r\ell\cos q_1)\ddot{q}_1 - g m_1 \ell \sin q_1 + (I + m_1 r\ell\cos q_1)\ddot{q}_0 - m_1 r\ell\dot{q}_1^2 \sin q_1 \tag{3}$$

where $I = I_0 + (m_0 + m_1)r^2$. These equations are second-order nonlinear differential equations of the form

$$\mathbf{M}(\mathbf{q}) + \mathbf{V}(\mathbf{q}, \ddot{\mathbf{q}}) = \tau, \tag{4}$$

where $\mathbf{M}(\mathbf{q})$ is the inertia matrix. $\mathbf{V}(\mathbf{q}, \ddot{\mathbf{q}})$ represents coriolis/centrifugal terms and gravity forces.

Rewriting the dynamics equations in the state-space representation yields Eq. 5,

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u) = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u, \tag{5}$$

where $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T$ is the state vector and u is the scalar input. By choosing $x_1 = q_0$, $x_2 = \dot{q}_0$, $x_3 = q_1$, $x_4 = \dot{q}_1$, and $u = \tau_0$, we obtain

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_2 \\ (f[2] - f[4])/\Delta \\ x_4 \\ f[4]/\Delta \end{bmatrix}, \quad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} 0 \\ (g[2] - g[4])/\Delta \\ 0 \\ g[4]/\Delta \end{bmatrix} \tag{6}$$

The values of $f[2]$, $f[4]$, $g[2]$, $g[4]$, and Δ are shown in the appendix. We found that the equilibrium state of the system is a set $\bar{\mathbf{x}} = \begin{bmatrix} x_1 & 0 & 0 & 0 \end{bmatrix}^T$ which means that the robot is standing upright at any horizontal position.

IV. Control

Input-output feedback linearization [14] is a common approach to the control of nonlinear systems. In this section, we present a generic controller enabling the robot to move/duck under the obstacle or balance at the desired position.

Consider the dynamics equations in (2) and (3). We can solve (3) for $\ddot{q}_0$. By substituting it into (2), q_0 and its derivatives in (2) can be completely eliminated. As a result, τ_0 can be rewritten as a function of only q_1 and its derivatives, yielding the following forms of (2):

$$\tau_0 = f_1(q_1)\ddot{q}_1 + f_2(q_1, \dot{q}_1) \tag{7}$$

or

$$u = f_1(x_3)\ddot{y} + f_2(x_3, x_4). \tag{8}$$

f_1 is a function of x_3, and f_2 is a function of x_3 and x_4.

Equation (8) represents an explicit relationship between the state variables and the input torque. We design a controller that can either achieve a target position or a target tilt angle of the robot.

Let the desired goals of the controller be the position $\delta_1 \in \Re$ and the tilt angle $\delta_3 \in (-\frac{\pi}{2}, \frac{\pi}{2})$. The double integrator term $\ddot{y}$ can be linearized by state errors,

$$\ddot{y} = -k_1(x_1 - \delta_1) - k_2 x_2 - k_3(x_3 - \delta_3) - k_4 x_4. \tag{9}$$

As a result, the combination of (8) and (9) provides us with a general controller for the horizontal position and the tilt angle of the robot. However, we cannot keep a desired position and a tilt angle different from zero at the same time. Instead we choose to control one of them. To control only the tilt angle, we set k_1 and k_2 to zero and the robot is expected to keep accelerating in order to maintain a specific tilt angle δ_3. On the other hand, to control the position, the desired tilt angle δ_3 is chosen to be zero to allow the robot to balance at the horizontal position δ_1.

In this paper, the hybrid controller, is a sequential composition of general controllers with distinct parameterizations. Each controller performs a different function and no single controller can easily satisfy the entire task. It is more effective for the robot to switch between control strategies depending on whether it needs to control the horizontal position or the tilt angle.

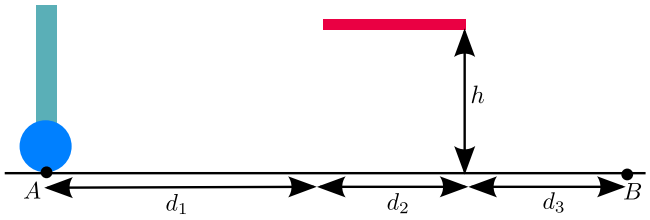


Fig. 4. Starting point, goal, and obstacle

V. Performance Optimization

A. Overview

Using the proposed feedback linearization controller from the previous section, we can achieve any desired tilt angle in order to duck under obstacles. However, there are many possible actions the robot might select in order to complete a particular ducking/limbo maneuver. Therefore, it is unclear exactly which set of reference angles and control parameters are optimal for a given situation. Using performance optimization methods, we seek to compute a set of control actions that minimize a predefined cost over a broad space of gains and parameters for the controllers. In this work, we focus on framing the optimization problem on top of the existing controller, keeping the same structural design but varying the controller parameters through stochastic optimization for performance improvements.

We selected a particular method called Particle Swarm Optimization (PSO) [22, 23], a stochastic, population-based evolutionary computing technique inspired by social interaction. PSO designs trajectories for a group of potential solutions called "particles", which traverse the solution space simultaneously and search for extrema points. Unlike traditional optimization algorithms that rely on gradient information, PSO does not explicitly compute the gradient but rather estimates the search direction through interactions with neighboring particles. At each time instance, a fitness function evaluates the quality of the solution obtained by each particle and shares the value with neighboring particles. Each particles is attracted to its own best solution as well as the group's best solution such that over time, the group as a whole is drawn stochastically towards the global optimum.

Previously, we have successfully applied PSO in both linear and nonlinear control designs. [24, 25] While the dynamics of the system in this implementation are not as challenging as those of our previous systems, the combined parameter space for the controllers is much larger due to the sequential composition of multiple controller instances. The composition results in a difficult, large-scale optimization problem. Since the merits of PSO lie in its ability to quickly converge to globally optimal solutions, even in large and non-convex solution spaces, ducking/limbo under obstacles is a particularly suitable application. The lack of dependence on gradients bypasses a computationally expensive process and the use of multiple particles ensures that the algorithm is not easily trapped in local minima. Our approach can also be parallelized in future applications that require greater speed.

TABLE II

SIMULATION PARAMETERS

(a) Robot parameters

m_0	3.139	kg	r	0.23	m
m_1	67.8	kg	ℓ	0.0762	m
I_0	0.1661	kg m^2	L	0.7834	m
I_1	1.21	kg m^2	g	9.81	m/s^2

(b) Obstacle parameters

Scenerio I			Scenerio II		
d_1	2.0	m	d_1	8.0	m
d_2	1.0	m	d_2	0.5	m
d_3	7.0	m	d_3	1.5	m
h	0.75	m	h	0.65	m

TABLE III

RESULTS

(a) Controller parameters: Hand-tuned VS. PSO

Parameters	Scenario I		Scenario II	
	Hand-tunded	PSO	Hand-tuned	PSO
k_{13}	40	176.11	40	188.93
k_{14}	20	143.71	20	200.00
k_{21}	4	5.8398	3	3.6623
k_{22}	6	12.685	6	5.5288
k_{23}	40	100.00	43	35.626
k_{24}	16	41.689	20	15.884
x_{3d} [rad]	1.0996	1.1924	0.6109	0.7136
d_{12} [m]	3	2.9289	5	4.5383

(b) Cost: Hand-tunded VS. PSO

Parameters	Hand-tuned			PSO		
	J	J_1	J_2	J	J_1	J_2
Scenerio I	8036	6032	10.0	7656	5826	9.1
Scenerio II	6963	4425	12.7	5837	4267	7.8

B. Formulation

Consider the scenario in Fig. 4 where the robot is initially balancing at point A and its mission is to reach point B. The path is partially blocked by an obstacle of length d_2 at a height h above the ground. Assume that the robot is taller than the height of the obstacle and there is no better way to reach the destination than passing under it.

In this scenario, we split the motion of the robot into two stages controlled by two separate controllers. Let k_{ij} represent the gain k_j in stage i, and similarly, δ_{ij} is the reference δ_j in stage i. During the first stage, feedback linearization with tilt control is in action. The controller gains involved in this stage are therefore k_{13} and k_{14}. The corresponding reference angle is δ_{13}. In the last stage, IO-linearization with position control is applied to allow the robot to stop at the goal. The controller parameters involved in this stage are k_{21}, k_{22}, k_{23}, k_{24}, and δ_{21}. The reference δ_{21} is the desired stopping position equal to $d_1+d_2+d_3$. Our planner designs a combination of these two stages resulting in a sequential composition of the two controllers. Since deciding when to switch the controllers is also important to help the robot reach the goal, we denote d_{12} as the horizontal distance from the starting point to make the transition from stage one to stage two.

There is an continuous space of possibilities for the values of all controller parameters. However, some of them are invalid since they make the robot collide with the obstacle or the ground. In addition, we are looking for parameters that minimize a pre-defined cost. Therefore, at this point, PSO plays a significant role in searching for the best set of parameters. We define a combined cost function consisting of two parts: the total squared control effort J_1, and the total time to complete the maneuver J_2 as shown in (10).

$$J = J_1 + \beta J_2 \tag{10}$$

β is a weight scalar. When T denotes the total time, we have

$$J_1 = \int_0^T \tau_0^2(t)\,dt, \qquad J_2 = T. \tag{11}$$

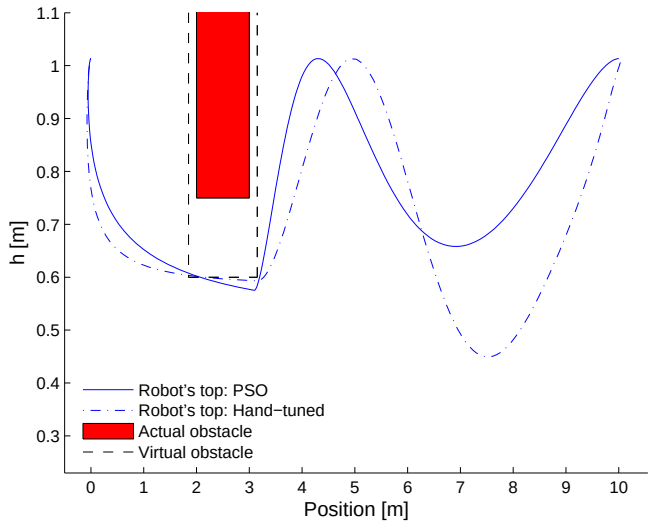
Control parameters that lead to a collision have infinite cost. The set of optimal parameters we seek are thus k_{ij} for all i and j, δ_{13}, and d_{12}. Using PSO, the parameters under consideration are encoded within particles, with appropriate restrictions placed as boundaries for the search space. Twenty particles are used in the search, initialized randomly within the search space. The algorithm iterates for a fixed number of iterations and returns the best solution found.

VI. SIMULATION AND RESULTS

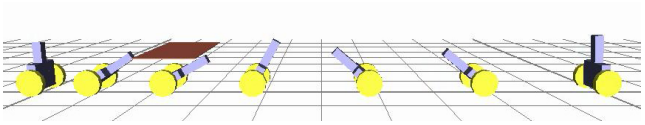
The performance of the optimized controllers was evaluated relative to a set of hand-tuned parameters. Parameter selection was first done by manual tuning followed by PSO. We conducted experiments for two different cases, Scenario I and Scenario II, using MATLAB and SNU Robotics Library (srLib) [26], a physics-based simulation library. In Scenario I, the obstacle position is close to the stating point. In Scenario II it is close to the goal. The simulation parameters are shown in Table II. Table III presents the set of parameters returned from PSO and the other set determined by trial and error. The overall costs are also compared in Table III(b). Since the real robot differs from our original skeleton model in thickness of the body, additional margins were added to the size of the obstacle to compensate for such errors as indicated by the virtual obstacle.

In Scenario I, an obstacle with 1.0 meter length was placed close to the starting position. The robot autonomously decided to perform a ducking action. This is reasonable since the obstacle is so close that the robot just leaned forward and passed under it as illustrated from the optimized result in Fig. 5(b). It is hard to achieve a low cost by trial and error since all six gains and references need to be selected in a way that provides a smooth, stable, and collision-free trajectory for all the stages. From the result, PSO provides superior performance to hand tuning in terms of the overall cost, or even the total squared control effort and the total time to reach the goal. Although PSO took around 15 minutes to return the optimal solution, the trial-and-error approach can take far longer to obtain a good result since it relies heavily on the experience of the operator.

In Scenario II, the position of the 0.5m-long obstacle was further away from the robot and close to the goal. One choice of strategies for the robot could again be ducking. However, this is not the best choice since the overall torque is critical in our consideration. Leaning forward with IO-linearization

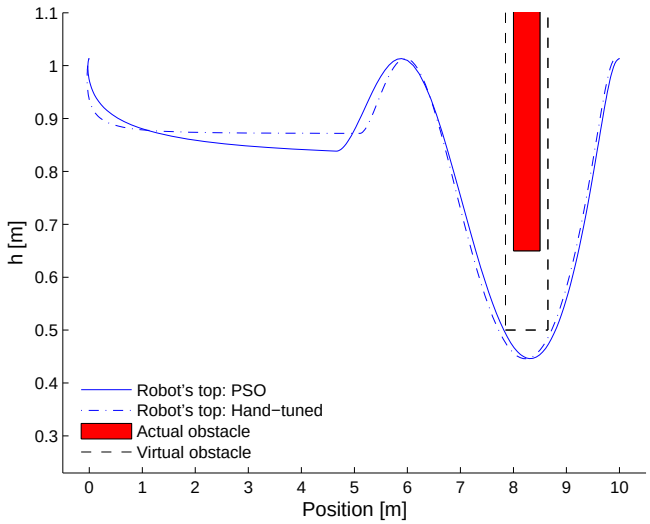


(a) Trajectories of the robot's top: Hand-tuned VS. PSO

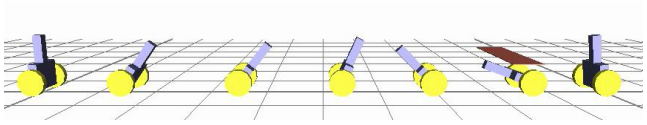


(b) srLib simulation with optimized parameters

Fig. 5. Simulation in MATLAB and srLib for Scenario I (Ducking). An obstacle of length 1m was placed close to the starting point and the resulting motion plan choose to lean forward (duck) to pass under the obstacle.



(a) Trajectories of the robot's top: Hand-tuned VS. PSO



(b) srLib simulation with optimized parameter

Fig. 6. Simulation in MATLAB and srLib for Scenario II (Limbo). An obstacle of length 0.5m was placed close to the goal and the resulting motion plan choose a limbo maneuver to pass under the obstacle.

accelerates the robot. In this case, the velocity would be very high and would require a lot of torque to stop after it has passed under the obstacle. PSO returned a set of parameters that results in the robot performing a limbo action as we expected. The successful result is shown in Fig. 6(b) and the overall costs in Table III(b) support the effectiveness of PSO, similar to the results from Scenario I. According to Fig. 6(a), the manual-tuned trajectory is nearly identical to the result by PSO as the trajectory is very close to the obstacle edges. Yet, Table III(b) shows that the hand-tuned controller performs significantly worse than autonomous optimization, especially in terms of time. These results show that PSO yields superior performance to hand-tuning since it provides better parameters without the need for trial and error.

Observe that *the sequential controller specifications used in both cases are identical.* PSO develops two entirely different strategies from this simple basis. In Scenario II the controller in the last stage is not just a stopping controller. The robot uses the same controller to travel under the obstacle. This is unlike the ducking in Scenario I. We classify the robot action into three parts. The first part is the first stage where the robot moves forward. The second and the third parts compose the second stage of limbo where the robot leaned backward to exploit its own dynamics to pass the obstacle and stop in an equilibrium state at the goal. This implies that the limbo strategy provided a better maneuver in terms of the total cost to accomplish the mission.

VII. Experimental Platform

Preliminary tests were conducted to study the validity of the dynamics and controllers. The experimental platform was a two-wheeled mobile robot, shown in Fig. 3. The robot is part of a larger wheeled humanoid developed by the Humanoid Robotics Lab at Georgia Tech [27]. It was equipped with an inertia measurement unit (IMU) and DC motors with encoders. Sensor data were Kalman filtered to estimate positions and velocities for wheels and robot tilt.

Experiments were performed without external sensors in order to verify that the distinct sequences of controllers could be applied in a physical system. Confirmation was completed in both individual controller tests as well as the sequential compositions for both ducking and limbo missions. The sample video sequences for limbo are presented in Fig. 1. Fig. 7 shows the tilt angle and velocity as well as the torque applied by the motors.

VIII. Conclusion

In this paper, we presented a sequential composition control design using a combination of two identical IO-linearization controllers that achieved entirely different motions for a two-wheeled robot when avoiding vertical obstacles. Stochastic optimization was applied in order to select the controller parameters and the appropriate time to switch the controllers. This yielded improved performance over manually-tuned parameters. With the combination of these simple components, the robot autonomously generated ducking and limbo

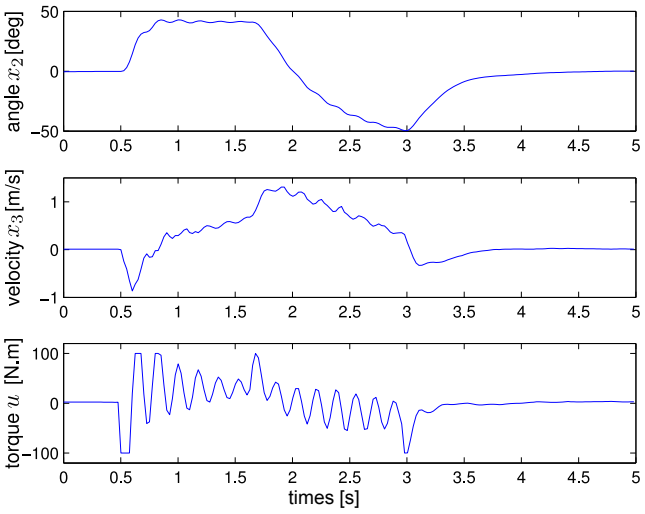


Fig. 7. Data collected from the actual robot performing limbo in Fig. 1

behaviors depending on whether the obstacle was close to the starting point or the goal, respectively.

Future work on this topic will incorporate sensing through vision and a laser scanner. We will also extend the method to handle multiple obstacles. In order to improve the performance of the controller, adaptive control approaches will be applied to make the robot more robust to uncertainty.

APPENDIX

Detailed expressions for (6) in Section III:

$$f[2] = m_1 r\ell I_1 x_4^2 \sin(x_3) - m_1^2 r\ell^2 g \sin(x_3)\cos(x_3) \quad (12)$$

$$g[2] = I_1 + m_1 r\ell \cos(x_3) \quad (13)$$

$$f[4] = m_1 g\ell I \sin(x_3) - m_1^2 r^2 \ell^2 x_4^2 \sin(x_3)\cos(x_3) \quad (14)$$

$$g[4] = -I - m_1 r\ell \cos(x_3) \quad (15)$$

$$\Delta = I_1 I - m_1^2 r^2 \ell^2 \cos^2(x_3) \quad (16)$$

$$I = I_0 + (m_0 + m_1) r^2 \quad (17)$$

ACKNOWLEDGMENTS

The authors are grateful to Ray Marceau for his contributions to the development of the robot and the experiments. We also thank the two anonymous reviewers for their comments and suggestions on an earlier version of this manuscript.

REFERENCES

[1] M. Montemerlo and S. Thrun, "A multi-resolution pyramid for outdoor robot terrain perception," in *National Conference on Artificial Intelligence*. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004, pp. 464–469.

[2] M. Kobilarov and G. Sukhatme, "Time optimal path planning on outdoor terrain for mobile robots under dynamic constraints," *University of Southern California Technical Report CRES-04-009*, 2004.

[3] C. Ye and J. Borenstein, "Obstacle avoidance for the segway robotic mobility platform," in *ANS 10th Int. Conf. on Robotics and Remote Systems for Hazardous Environments*, 2004, pp. 107–114.

[4] J. Kim and J. Oh, "Realization of dynamic walking for the humanoid robot platform KHR-1," *Advanced Robotics*, vol. 18, no. 7, pp. 749–768, 2004.

[5] H. Hirukawa, S. Hattori, S. Kajita, K. Harada, K. Kaneko, F. Kanehiro, M. Morisawa, and S. Nakaoka, "A pattern generator of humanoid robots walking on a rough terrain," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2181–2187.

[6] M. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, S. Engberg, J. Matthews, J. Dunbar-Jacob, C. McCarthy *et al.*, "Pearl: A mobile robotic assistant for the elderly," in *AAAI workshop on automation as Eldercare*, vol. 2002, 2002.

[7] R. Simmons, R. Goodwin, S. Koenig, J. O'Sullivan, and G. Armstrong, "Xavier: An Autonomous Mobile Robot on the Web," *Beyond Webcams: an introduction to online robots*, p. 81, 2001.

[8] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte *et al.*, "MINERVA: A second generation mobile tour-guide robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[9] F. Grasser, A. D' Arrigo, S. Colombi, and A. Rufer, "JOE: a mobile, inverted pendulum," *IEEE Transactions on industrial electronics*, vol. 49, no. 1, pp. 107–114, 2002.

[10] P. Deegan, B. Thibodeau, and R. Grupen, "Designing a self-stabilizing robot for dynamic mobile manipulation," in *Robotics: Science and Systems-Workshop on Manipulation for Human Environments*, 2006.

[11] R. Ambrose, R. Savely, S. Goza, P. Strawser, M. Diftler, I. Spain, and N. Radford, "Mobile manipulation using NASA's robonaut," in *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04*, vol. 2, 2004.

[12] 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*, 2006, pp. 2884–2889.

[13] R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, p. 534, 1999.

[14] H. Khalil, *Nonlinear systems*. Prentice Hall, 2002.

[15] Y. Kim, S. Kim, and Y. Kwak, "Dynamic analysis of a nonholonomic two-wheeled inverted pendulum robot," *Journal of Intelligent and Robotic Systems*, vol. 44, no. 1, pp. 25–46, 2005.

[16] M. Spong, "Partial feedback linearization of underactuated mechanical systems," in *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, vol. 1, 1994.

[17] K. Pathak, J. Franch, and S. K. Agrawal, "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 505–513, June 2005.

[18] D. S. Nasrallah, H. Michalska, and J. Angeles, "Controllability and posture control of a wheeled pendulum moving on an inclined plane," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 564–577, 2007.

[19] Z. Li and J. Luo, "Adaptive Robust Dynamic Balance and Motion Controls of Mobile Wheeled Inverted Pendulums," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 1, pp. 233–241, 2009.

[20] D. Aydemir and H. Iba, "Evolutionary Behavior Acquisition for Humanoid Robots," *Lecture Notes in Computer Science*, vol. 4193, p. 651, 2006.

[21] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New Jersey, 2006.

[22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, 1995.

[23] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[24] J. Wang, B. T. Brackett, and R. G. Harley, "Particle Swarm-Assisted State Feedback Control: From Pole Selection to State Estimation," in *2009 American Control Conference*, June 2009.

[25] M. Stilman, J. Wang, K. Teeyapan, and R. Marceau, "Optimized Control Strategies for Wheeled Humanoids and Mobile Manipulators," in *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, December 2009.

[26] J. Haan, B. Kim, and J. Lee, "srLib - SNU Robotics Library," June 2009. [Online]. Available: http://r-station.co.kr/srlib/index.html

[27] M. Stilman, J. Olson, and W. Gloss., "Golem Krang: Dynamically Stable Humanoid Robot for Mobile Manipulation," in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, Alaska, USA, May 2010.