

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

Kasemsit Teeyapan, Jiuguang Wang, and Mike Stilman

**Abstract**—We present three 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 such as the dimensions of the obstacle, our method utilizes a LQ/IO-linearization hybrid controller in a sequential composition and uses stochastic optimization to automatically compute an optimized selection of controller gains as well as switching time for the overall torque. 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 a stochastic optimization to choose among sequences of controllers and automatically computing their parameters.

Mobile manipulators 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 where workspace is steep, or when the robots make an abrupt change in velocity. In addition, if the workspace is limited, their navigation ability may be restricted since they typically require a greater turn radius.

In contrast, dynamically stable robots like JOE [9], uBot [10], Robonaut [11], Segway RMP [3], and Ballbot [12] possess a near-zero turn radius for moving in a limited space. Furthermore, their method of stabilization is similar to that of humans, allowing greater flexibility in control. Such robots are more robust to applied external forces as well as rapid acceleration and deceleration. Moreover, due to the fact that they require only two actuators to achieve a maneuver, they simplify planning and control for complex tasks such as search and rescue.

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

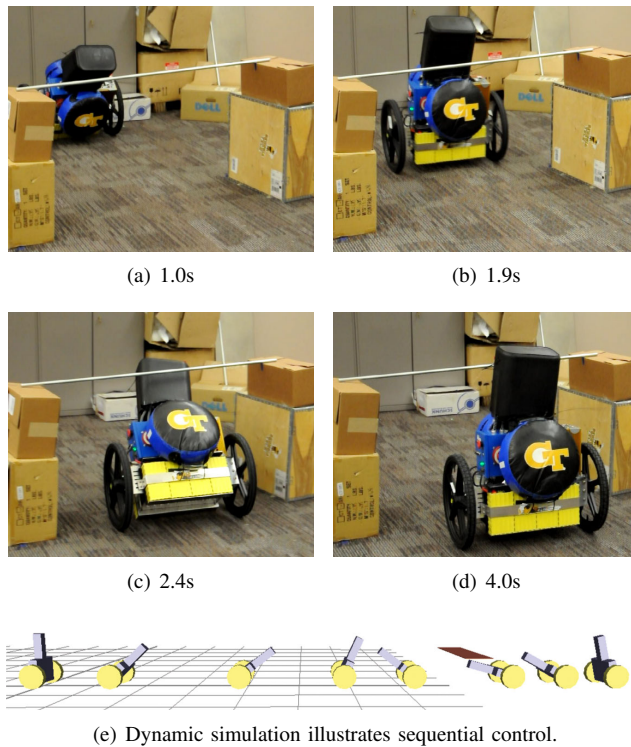


Fig. 1. Robot execution of the hybrid limbo motion.

This paper explores the ability of a two-wheeled robot (Fig. 1) in generating motions to move or duck under obstacles. Generally, obstacles are immovable so statically stable robots might not easily 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 later case, the action is similar to the so-called limbo, a West Indian dance in which the dancer leans backward to go under the fixed-height stick.

To generate a robot’s motion, the trajectory is divided into three stages and is controlled by three separate controllers. We present a hybrid controller consisted of linear quadratic controller (LQ) [13] and input-output feedback linearization [14]. Since each controller is assigned to different function, this kind of strategy is called “sequential composition technique” [15]. Consequently, this is a planning task where a stochastic optimization decides the actual switching boundaries and the parameters of the controller.

The rest of the paper is organized as follows: Section III provides the state-representation for the dynamics of the system. Section IV describes the design of the controllers, LQ controller and feedback linearization controller. In Section V, an overview of an optimization algorithm named “particle

swarm optimization” is given followed by the formulation to our problem. The simulation and experimental results are presented in Section VI while the preliminary test done on the robot platform is described in Section VII.

## II. RELATED WORK

Many research studies on two-wheeled manipulators have been conducted, but their main focuses have exclusively been on dynamics and balance [9] [10] [16]. Being underactuated, two-wheeled platforms also experience difficulty in achieving the efficient control approach since such the systems are not *feedback linearizable*. [17] showed that linearization can be done around the active or passive joints, referred to as collocated and non-collocated partial feedback linearization. Consequently, the control problem remains a current research issue. There are some recent work focusing on the position and velocity control [18], pose control [19], and also some adaptive approach to motion control [20]. However, none of them so far has regard to motion strategies when the robots operate among the obstacle, especially when the obstacles are in the vertical position.

In humanoid robot communities, there are recently some experiments regarding the limbo with a humanoid robot [21]. By applying genetic algorithms, the work succeeded without the needs of understanding the detailed dynamics of the robot. Unfortunately, due to the complexity of the platform, the generated motion was too sluggish and cumbersome. The performance can be different with two-wheeled robots because of their increased agility. With wheels, the robots are allowed faster movements to complete maneuvers.

Although two-wheeled balancing platform is well-known to be nonlinear, LQ regulator is widely used to solve balancing problem, for example in [16]. Since the controller is based on linearized version of the system, the performance is reliable only around the equilibriums. This is typically sufficient and requires only small computational costs. Partial feedback linearization, another control scheme, was also explored in [18] and [19]. This approach can better deal with the nonlinearity of the system but needs more computation, particularly when trying to simultaneously control the position and the tilt angle of the robot. However, there exists a less complex version called input-output feedback linearization (IO-linearization), capable of just the inclination control which is adequate to vertically control the robot. As a result, a sequential composition technique composed of LQ and IO-linearization is introduced to control to keep lowering the computational complexity.

## III. DYNAMIC MODEL

In this section, the dynamics of the robot is derived with Lagrangian mechanics. To simplify the problem, we use the robot schematic as shown in Fig. 2 corresponded to the parameters in Table I. The robot’s wheels are driven by DC servo motor with encoders. The body is connected to the wheels through the passive joints. The posture of this whole system can be fully described by the angular position  $q_w$  and the inclination angle  $q_1$  of the robot. However, for our system, the position is measured by the encoders on the

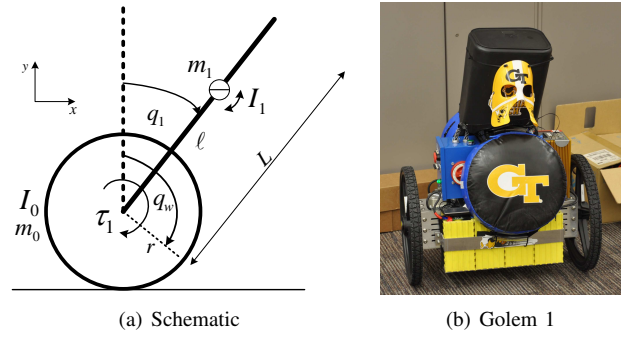


Fig. 2. The schematic (a) of our two-wheeled robot (b).

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 center of mass
$r$	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
$\tau_1$	Motor torque

motors. What we obtain is therefore the position of the robot relative to the body in which we call  $q_0$ . The relation between  $q_w$  and  $q_0$  is simply  $q_w = q_0 + q_1$ .

As a result, we choose the generalized coordinate of the system as  $\mathbf{q} = [q_0, q_1]$  where  $q_0 \in \mathfrak{R}$  and  $q_1 \in (-\frac{\pi}{2}, \frac{\pi}{2})$ . We also define the generalized force  $\boldsymbol{\tau} = [\tau_1, 0]$  in which  $\tau_1 \in \mathfrak{R}$  is the applied torque to the wheels and due to the passivity of the joint, the second element of  $\boldsymbol{\tau}$  is zero. The equations of motion are obtained from Lagrange’s equation,

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

where  $L$  is Lagrangian function and is computed from the difference between the kinetic energy and the potential energy of the system as  $L = T - U$ . The dynamics of the system we obtain is

$$\tau_1 = (\ddot{q}_0 + \ddot{q}_1)I + m_1 r l \ddot{q}_1 \cos q_1 - m_1 r l \dot{q}_1^2 \sin q_1 \quad (2)$$

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

where  $I = I_0 + (m_0 + m_1)r^2$ .

These equations can be considered as the second-ordered nonlinear differential equations of the form

$$\mathbf{M}(\mathbf{q}) + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} \quad (4)$$

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

In order to use the dynamics to design the controllers in the next section, the dynamic equation is then reformulated into the state-space representation,

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

where  $\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4]^T$  is the state vector and  $u$  is the scalar input. By choosing  $x_1 = q_0$ ,  $x_2 = q_1$ ,  $x_3 = \dot{q}_0$ ,  $x_4 = \dot{q}_1$ , and  $u = \tau_1$ , we obtain

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} x_3 \\ x_4 \\ f[3]/\Delta \\ f[4]/\Delta \end{bmatrix}, \quad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} 0 \\ 0 \\ g[3]/\Delta \\ g[4]/\Delta \end{bmatrix}. \quad (6)$$

The values of  $f[3]$ ,  $f[4]$ ,  $g[3]$ , and  $g[4]$  are shown in the appendix. We found that the equilibrium state of the system is a set  $\bar{\mathbf{x}} = [x_1 \ 0 \ 0 \ 0]^T$  which means that the robot is standing upright at any distance.

#### IV. CONTROL

##### A. Linear Quadratic Controller

Linear quadratic control [13] is one of the most fundamental control strategies in optimal control theory. In this paper, we consider its extension to tracking problems. First, by the linearization of the system equation (5) around the equilibriums, we obtain the system of the form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \quad (7)$$

where  $\mathbf{A} \in \mathfrak{R}^{4 \times 4}$  and  $\mathbf{B} \in \mathfrak{R}^4$ .

The output equation of the system is defined directly in terms of the states of the wheels i.e.,  $\mathbf{y} = \mathbf{x}$ . Our goal is to design a LQ controller that allows the output to track a constant reference value  $\mu$ .

Before designing the feedback controller, the rank test can verify that the system (7) is *controllable* and *observable*.

To begin with the controller design, we assume that the problem is infinite-horizon. For the linearized system (7), the cost functional for LQ problem is defined as

$$J = \int_0^{\infty} ((\mathbf{y} - \mu)^T \mathbf{Q}(\mathbf{y} - \mu) + Ru^2) dt \quad (8)$$

where  $\mathbf{Q}$  is the weight matrix which is restricted to be positive semi-definite and  $R$  is the positive-definite weight of the input.

As derived in [13], the optimal value  $u$  that minimizes the defined cost  $J$  is given by

$$u = -R^{-1}\mathbf{B}^T(\mathbf{P}\mathbf{x} + \mathbf{w}) \quad (9)$$

where the  $\mathbf{P}$  is the solution to the algebraic Riccati equation.

$$0 = -\mathbf{A}^T\mathbf{P} - \mathbf{P}\mathbf{A} - \mathbf{Q} + \mathbf{P}\mathbf{B}R^{-1}\mathbf{B}^T\mathbf{P} \quad (10)$$

and

$$\mathbf{w} = (\mathbf{A} - \mathbf{B}R^{-1}\mathbf{B}^T\mathbf{P})^{-T}\mathbf{Q}\mu \quad (11)$$

However, due to the infinite-horizon assumption we made, the reference  $\mu$  is valid only if  $\mu \in \bar{\mathbf{x}}$ . In other words, the proposed control law can only drive the state  $\mathbf{x}$  to any values which is in  $\bar{\mathbf{x}}$ . This is useful since we can apply the controller to control the position of the robot.

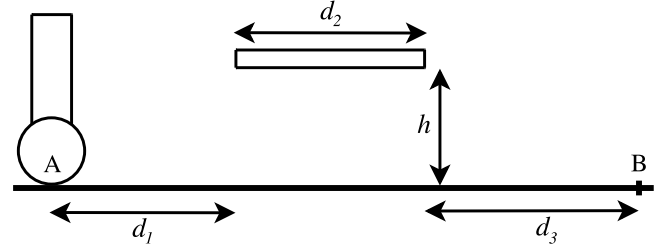


Fig. 3. Stating point, goal, and obstacle

##### B. Input-Output Feedback Linearization

In the previous part, we obtain the LQ controller to do the balance and position control of the robot. In this section, we introduce the input-output feedback linearization [14], a common approach to control nonlinear systems, to control the tilt angle. This can allow the robot to move/duck under the obstacle. Therefore, the output equation we are interested here is  $y = x_2 = q_1$ .

Consider the original dynamics equations, (2) and (3). From (3), we can obviously solve for  $\ddot{q}_0$ . By substituting it into (2), the terms directly related to  $q_0$  in (2) can be completely eliminated. As a result,  $\tau_1$  can be rewritten as a function of only  $q_1$  and its derivatives and (2) becomes the equation of the form

$$\tau_1 = f_1(q_1)\ddot{q}_1 + f_2(q_1, \dot{q}_1) \quad (12)$$

or

$$u = f_1(x_2)\ddot{y} + f_2(x_2, \dot{x}_4) \quad (13)$$

where  $f_1$  is a function of  $x_2$ , and  $f_2$  is a function of  $x_2$  and  $x_4$ .

Equation (13) represents an explicit relation between the state variables and the input torque. Let the goal of the controller control the tilt angle to a specific reference angle  $\delta \in (-\frac{\pi}{2}, \frac{\pi}{2})$ . The second-ordered output terms can be linearized by a state feedback controller,

$$\ddot{y} = -k_1(x_2 - \delta) - k_2x_4 \quad (14)$$

where  $s^2 + k_2s + k_1$  needs to be Hurwitz for stabilization. As a result, by combining (13) and (14) together, we finally obtain the controller to control the tilt angle of the robot where it deals only with the angle but the not the position. Hence, we expect the robot to keep accelerate while maintain its inclination at a constant angle.

In this paper, the hybrid controller which is behind the robot motion is therefore the combination of the two types of controllers we described. No single controller can easily achieve our problem scenerio. LQ controller can perform balancing and positioning. Although it also can be designed to control the tilt angle, the performance may not be good enough since it is based on the linearized dynamics. In addition, that controller will have to solve the differential Riccati equation which increases the computational complexity. Therefore, it is more efficient for the robot to switch between the two controllers whenever it needs to control the position and balance or the inclined angle.

## V. PERFORMANCE OPTIMIZATION

### A. Overview

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

We selected a particular optimization tool called Particle Swarm Optimization (PSO) [22], a stochastic, population-based evolutionary computing technique inspired by the paradigm of social interaction. PSO relies on the trajectories of a group of potential solutions called “particles”, which traverses the solution space simultaneously to search for extrema points. Unlike traditional optimization algorithms that rely on gradient information, PSO do not explicitly compute the gradient but rather estimate the search direction through interactions with neighboring particles. At each time instance, a fitness function evaluates the quality of the solutions obtained by each particle and the value is shared across neighboring particles. The particles are attracted to its own best solution as well as the group’s best solution, and 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 [23] [24]. While the dynamics of the system in this implementation is not as challenging as our previous systems, the combined parameters space for the control and obstacle parameters is much larger, resulting in a difficult, large-scale optimization problem. As 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, which can also be easily parallelized in future applications that require greater speed.

### B. Formulation

Consider the scenario in Fig. 3 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 the height  $h$  above the ground. Supposedly, the robot is taller than the position of the obstacle and there is no better way to reach the destination than passing under it.

With this scenario, we plan the motion of the robot into three sequences controlled by three separate controllers. In the first two stages, feedback linearization is in action. The

TABLE II  
SIMULATION PARAMETERS

(a) Robot parameters					
$m_w$	3.139	kg	$r$	0.23	m
$m_p$	67.8	kg	$\ell$	0.0762	m
$I_w$	0.1661	kg m <sup>2</sup>	$L$	0.7834	m
$I_p$	1.21	kg m <sup>2</sup>	$g$	9.81	m/s <sup>2</sup>

(b) Obstacle parameters					
Scenario I			Scenario II		
$d_1$	2.0	m	$d_1$	8.0	m
$d_2$	1.5	m	$d_2$	0.5	m
$d_3$	6.5	m	$d_3$	1.5	m
$h$	0.75	m	$h$	0.65	m

TABLE III  
OPTIMIZED PARAMETERS

(a) Scenario I: Ducking					
Parameters	Scenario I		Scenario II		
	Hand-tuned	PSO	Hand-tuned	PSO	
$k_{11}$	20	9.3133	40	11.8003	
$k_{21}$	20	8.2533	10	24.9041	
$k_{12}$	5	5.0000	40	5.3196	
$k_{22}$	10	5.0000	10	29.3483	
$\delta_1$ [rad]	1.2566	1.1864	0.7679	1.3142	
$\delta_2$ [rad]	0.5236	1.2033	1.0123	1.2256	
$d_{12}$ [m]	3.2200	1.6149	2.3000	2.0590	
$d_{23}$ [m]	3.3350	3.3776	4.6000	5.0000	

(b) Cost: Hand-tuned VS. PSO

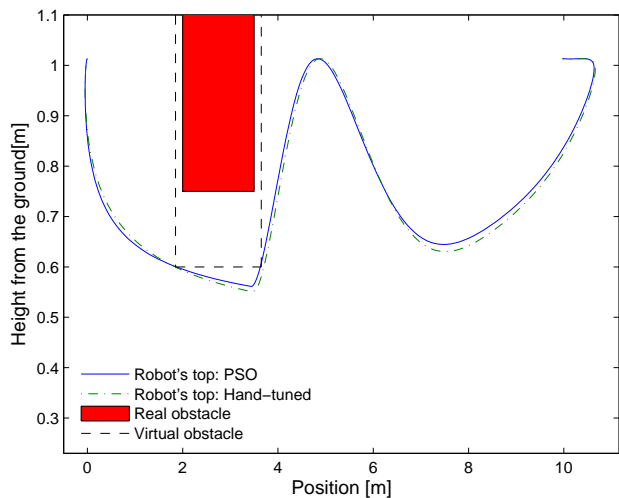
	Cost	Hand-tuned	PSO
Scenario I		6024.0	5920.2
Scenario II		6784.9	6295.6

controller gain  $k_i$  for the stage  $j$  where  $i, j \in 1, 2$  is simply called  $k_{ij}$ . The reference angle corresponding to each stages is also denoted as  $\delta_j$ . In the last stage, LQ controller is applied to allow the robot to stop at the goal. All three stages of planning result in a sequential composition. The positions to switch the controllers are consequently important parameters. We also denote  $d_{j(j+1)}$  as the position from the starting point to pass from the stage  $j$  to  $j+1$ . In our case, we have  $d_{12}$  and  $d_{23}$ .

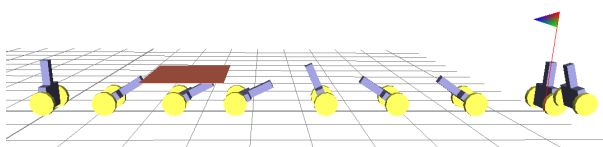
There are infinite possibilities to assign values to all controller parameters. However, some of them are invalid since they lead the robot trajectory either to the obstacle or to the ground. In addition, we are more concerned about the parameters that minimize some pre-defined costs. Therefore, at this point, PSO plays a significant role in searching the best set of parameters. We define the cost function to be the total squared control effort to reach the goal:

$$J = \int_0^T \tau_1^2(t) dt \quad (15)$$

where  $T$  is the total time to complete the maneuver. For simplicity, the LQ gain for the final stage is fixed. A set of optimal parameters we seek are thus  $k_{ij}$  and  $d_{j(j+1)}$ . Using PSO, the controller parameters under consideration are encoded within particles, with appropriate restrictions placed as boundaries for the search space. Five 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.

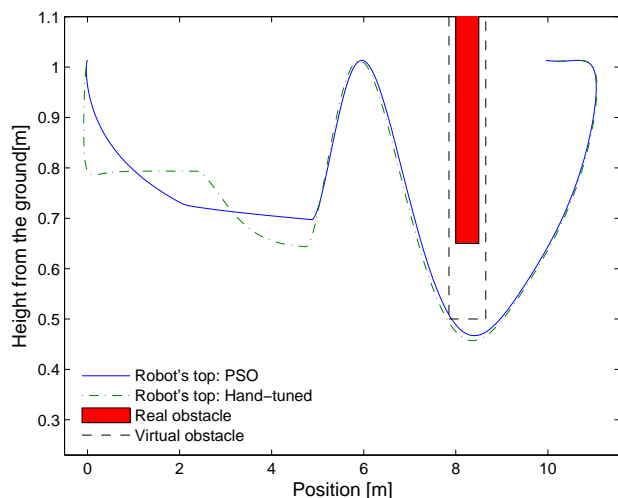


(a) Trajectories of the robot's top: Hand-tuned VS. PSO

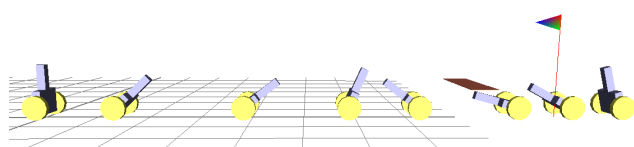


(b) srLib with optimized parameters

Fig. 4. Simulation in MATLAB and srLib for Scenario I



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



(b) srLib with optimized parameter

Fig. 5. Simulation in MATLAB and srLib for Scenario II

## VI. SIMULATION AND RESULTS

The performance of the optimized controllers was evaluated relative to a set of hand-tuned parameters. Experiments were conducted in two different cases: Scenario I and Scenario II on both a planar model in MATLAB and a 3D model using SNU Robotics Library (srLib). The simulation parameters are shown in Table II. Table III presents the set of parameters returned from PSO and the other set formed 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 such errors as indicated as the virtual obstacle.

### A. Scenario I: Ducking

In Scenario I, the obstacle with 1.5 meters long was placed close to the starting position of the robot. As a result, the robot in this case performed ducking action. This is reasonable since the obstacle is so close that the robot just leaned forward by IO-linearization controller and passed under it as illustrated from the optimized result in Fig. 4(b). Tuning parameters by trial and error in this case was not too difficult and provided the total cost quite close to a set of parameters generated by PSO. This point is also noticeable in the MATLAB plot since the trajectories of the robot's top are nearly overlapped though the parameter values are almost different in Table III(a). In overall, however, PSO is still successful since it can provide better parameters without the need of trial and error.

### B. Scenario II: Limbo

In Scenario II, the position of the 0.5m-long obstacle was further away and close to the goal. Apparently, one choice of the robot again could be ducking. This, however, is not the best choice since the overall torque is critical in our consideration. Leaning forward by IO-linearization typically accelerates the robot so in this case its velocity will be very high and requires much torque to stop after it passes under the obstacle. By PSO, the optimal set of parameters was returned to create a limbo action as we expected. The successful result is shown in Fig. 5(b). One interesting thing to note is that the controller used to control the robot to pass the obstacle was LQ controller in the final stage. This is unlike ducking scenario where IO-linearization was used. The result implies that limbo provided a better maneuver in terms of torque when the obstacle was close to goal. Manual tuning could also produce a set of parameters. However, as can be seen from Fig. 5(a) and Table III(b), the obtained performance is disappointing. By PSO, the trajectory is perfectly touched the virtual obstacle edges. This, thus, evidences that PSO provides superior performance to manually-tuned method.

## 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. 2, developed by the Humanoid Robotics Lab at Georgia Tech. The robot 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.

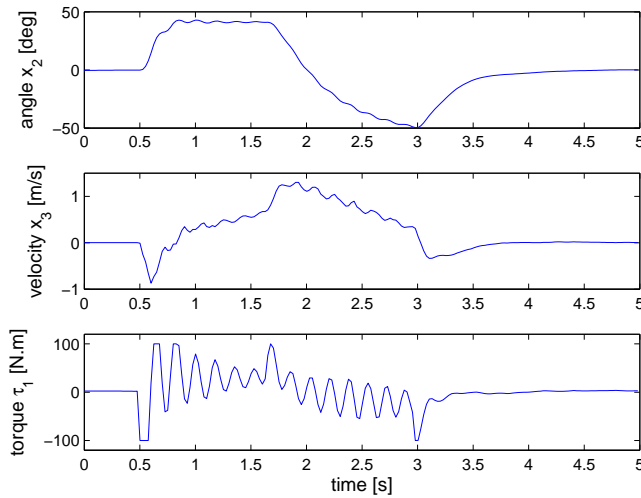


Fig. 6. Data collected from the actual robot performing limbo in Fig. 1

Experiments were performed without external sensors in order to verify that sequences of controllers actually worked in the physical system. The 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. 6 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 as a combination of LQ controller and IO-linearization for two-wheeled manipulators to generate a motion to avoid a vertical obstacle. A stochastic optimization algorithm was also applied in order to help select the optimal control parameters and the appropriate time to switch the controllers. The most interesting result was that these parameters were automatically generated by PSO and provided an optimal result over the manually-tuned parameters. With the combination of the simple components, the robot is allowed to accomplish ducking and limbo task if the obstacle is close to the starting point and to the goal respectively.

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

### APPENDIX

Detailed expressions for (6) in Section III:

$$f[3] = m_1 l \sin(x_2) [(I_1 + m_1 l^2) r x_4^2 - gI - m_1 r l (g - r x_4^2) \cos(x_2)] \quad (16)$$

$$g[3] = I + I_1 + m_1 l^2 + 2m_1 r l \cos(x_2) \quad (17)$$

$$f[4] = [m_1 g l I - (m_1 r l x_4)^2 \cos(x_2)] \sin(x_2) \quad (18)$$

$$g[4] = -(I + m_1 r l \cos(x_2)) \quad (19)$$

$$\Delta = (I_1 + m_1 l^2) I - (m_1 r l \cos(x_2))^2 \quad (20)$$

$$I = I_0 + (m_0 + m_1) r^2 \quad (21)$$

### 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] A. Locatelli, *Optimal Control: An Introduction*. Birkhauser, 2001.
- [14] H. Khalil, *Nonlinear systems*. Prentice Hall, 2002.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] K. Pathak, J. Franch, and S. Agrawal, "Velocity and position control of a wheeled inverted pendulum by partial feedback linearization," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 505–513, 2005.
- [19] D. 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.
- [20] 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.
- [21] D. Aydemir and H. Iba, "Evolutionary Behavior Acquisition for Humanoid Robots," *Lecture Notes in Computer Science*, vol. 4193, p. 651, 2006.
- [22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, 1995.
- [23] 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.
- [24] 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.