Intelligent Control of a Pole Balancing Robot

Meng Joo Er*, Nikos Mastorakis#, Bak Heng Kee* and Chee Chong Tan*

*School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, Nanyang Avenue, Singapore 639798.
E-mail : emjer@ntu.edu.sg, 7573734C@ntu.edu.sg and smarty@singnet.com.sg

#Military Institutions of University Education
Hellenic Naval Academy
Terma Hatzikyriakou, 18539, Piraeus, GREECE.
E-mail : mastor@ieee.org

ABSTRACT

In this paper, an intelligent controller capable of static balancing as well as dynamic balancing of a pole mounted on a motorized robot is designed and developed. The brain of the intelligent controller lies in the Fuzzy Inference System (FIS), which receives as its inputs displacement, velocity and acceleration information. An embedded instrumentation system onboard the robot measures the displacement of the robot and the angle of inclination of the pole from the vertical position. For static balancing, the controller needs to maintain the pole in an upright position while the robot is free to move on a flat surface. For dynamic balancing, the robot needs to balance the pole while performing transitions up and down a ramp. Furthermore, the robot needs to steer itself back to the center to prevent it from falling off the ramp.

**Keywords:** Intelligent Control; Static Balancing; Dynamic Balancing; Pole-Balancing Robot.

1. **INTRODUCTION**

The inverted pendulum model has proven to be an important precursor to more complex work on active balancing, particularly in the field of legged machines and locomotion studies for robotics. The first recorded attempt for this fundamental problem was probably by [1]. He used the parts from an erector set to build a machine that balanced an inverted pendulum on top of a small powered truck. The truck drove back
and forth in response to tipping movements of the pendulum as sensed by a pair of switches at its base. In order to move from one place to another, the truck first had to drive it away from the goal to unbalance the pendulum toward the goal. In order to balance again at the destination, the truck moved past the destination until the pendulum was again upright with no forward velocity. It then moved back to the goal. The inverted pendulum model would become the primary tool for studying balancing in legged systems and the importance of active balancing in legged locomotion has been widely recognized since 1938, see [2]-12. However, progress in building physical legged systems that employ such principles was slowed down by perceived difficulty of the task. It was not until the late 1970s that experimental work on balancing for legged systems gained momentum [13].

The major drawback of the previous work is that the algorithms are very complicated. In this paper, a new algorithm for balancing the pole is presented. We show that by using a three-dimensional fuzzy logic controller, the controller is capable of performing dynamic and static balancing with high consistency and repeatability.

2. STATIC BALANCING

2.1 Design and Development of Robot Hardware

2.1.1 Mechanical Design

The robot was designed to support an inverted pendulum that is free to swing about a horizontal axis with one degree of freedom. The inverted pendulum was secured on a pole-support mechanism and balanced by moving the pivoted support parallel to the plane of swing. Other than the robot’s wheels, no parts of the body touch the balancing platform.

On the pole-support mechanism, ball bearings were used in the axle of rotation supporting the pole. A potentiometer was used to feedback the rotating pole angle. An optical encoder was mounted on one side of the front wheel instead of the back driving wheels in order to reduce skidding error. The robot is driven by a DC motor which receives commands from the FIS.
The hardware of the robot consists of the following main modules:
1. DC Motor
2. Motor and wheel holders
3. Robot base plate
4. Pole pivot bracket
5. Potentiometer to feedback pole angle
6. Optical encoder to feedback robot position
7. Transceiver board with Intel 8751 micro-controller

The intelligence of the robot lies in the computer, which communicates with the Intel 8751 micro-controller reading sensory feedback signals and driving the motor. The algorithm on the computer is designed to control the robot throughout the duration of its operation without human intervention.

2.1.2 PCB Design of Embedded 8751 System
The PCB was designed to provide an interface for capturing and processing two essential data and controlling the driving motor. The data are sent via the serial
A communication link to the computer where they are processed to determine the required direction and magnitude of the force to drive the motor.

The two data that are captured by the embedded system are the optical encoder (16 bits) and potentiometer (12 bits). One output data is generated for the DC Motor PWM (pulse-width modulation).

The tasks performed by the embedded system are as follows:
1. Serial port communication
2. Optical encoder feedback circuitry
3. Potentiometer feedback circuitry
4. Motor driver
5. Interrupt generator
6. Micro-controller to handle individual sub-modules

![Diagram of the embedded 87C51 system](image-url)

*Fig. 4 Block diagram of the embedded 87C51/2 system*
2.2 Design and Development of Robot Software

2.2.1 Proposed Control Strategy

The control system for the pole-balancing robot is separated into two main parts, namely the $\dot{\theta}/d\dot{\theta}/d\ddot{\theta}$ system and the $x/dx/ddx$ system. The basic block diagram of the proposed control strategy is shown below.

The three key parameters are the displacement, velocity and acceleration of the pole angle and the robot position. They are explained as follows:

![Proposed control strategy diagram](image1)

**Fig. 5 Proposed control strategy**

![Flowchart of the 87C51/2 microcontroller routine](image2)

**Fig. 6 Flowchart of the 87C51/2 microcontroller routine**
2.2.2 $\dot{\theta}/d\dot{\theta}$ System
This system is the primary control system to keep the pole upright at all times. The inputs are $\theta$, the pole angle with respect to the vertical position, $d\theta$ is the velocity of pole movement and $d^2\theta$ is the acceleration of pole movement. Taking the sampling time into consideration, we can incorporate $d\theta$ from $\theta$ by using the following relationship between distance, speed and time. This additional information enables the system to predict whether the pole swing is slowing down or vice versa. This is similar to adding a derivative component in the classical PID controller. This additional input to the fuzzy system makes the system more intelligent. The same reasoning applies to $d^2\theta$.

2.2.3 $x/dx/ddx$ System
Besides trying to balance the pole, we want it to maintain that action at a particular position. This is done by the $x/dx/ddx$ system. The inputs are $x$, robot displacement with respect to a predefined position, $dx$, velocity of the robot and $ddx$, acceleration of the robot. The output of the $x/dx/ddx$ system is the required offset angle to compensate for the system so that the motor will return to the desired balancing point on the platform. Similarly, the additional parameters $dx$ and $ddx$ will provide anticipatory actions like what the $\dot{\theta}/d\dot{\theta}$ system has achieved with $d\theta$ and $d^2\theta$.

2.2.4 Incorporation of Fuzzy Logic
Implementation of the $\dot{\theta}/d\dot{\theta}/d^2\theta$ system will be similar to that of the $x/dx/ddx$ system except that the variables are different. The potentiometer and encoder readings are fuzzified before being read by the FIS, which is implemented using C programming language. A total of 5 membership functions for each variable are created and this results in 125 rules in either the $\dot{\theta}/d\dot{\theta}/d^2\theta$ system or the $x/dx/ddx$ system.
As illustrated above, the inverted pendulum consists of an arm attached to the motor at a particular point. The arm rotates in the plane OXY, the motor moves along the X-axis forward and backward under the influence of a force. The input parameters are the angle $\theta$ and the velocity $d\theta$. The output parameter is the force. The control objective is for any pair $(\theta, d\theta)$, which specifies the position and velocity of the system to find and apply a corresponding force to the robot such that the pole is balanced, i.e. does not fall over the motor.

2.2.5 Evaluation of System Performance

Real-time data taken during a successful run have been recorded and plotted using Matlab. Each variable can be analyzed individually and trends of instability in the system can be identified.
Figure 8 shows the variation of theta, i.e., the pole angle from the vertical position. The working range of theta in the $\vartheta/d\vartheta/dd\vartheta$ Fuzzy Inference Engine (FIE) is $\pm5^\circ$. From the graph, the pole angle never exceeds $\pm1^\circ$. It is evident that the 3D rule base has improved the system reliability and accuracy significantly. Also, Graph 1b shows the deviation of the robot position from the desired spatial point on the platform. Here, the working range of $x$ in the $x/dx/ddx$ FIE is $\pm5$ cm. The graph shows that the robot displacement fluctuates between +3 cm to −7 cm, with a mean position of −2 cm. This is due to the un-leveled ground that the robot is balancing on. Unless the controller compensates for this uncertainty, the robot will always tend to bias to one side of the balancing point. This is the main idea of balancing the robot on a slope. The variable used in the C language code is “t_slope”. This variable is added to each defuzzification membership function of both the $\vartheta/d\vartheta/dd\vartheta$ and the $x/dx/ddx$ FIEs. Note that there are other possible causes to this observation. These include the uneven distribution of weights on the robot itself and the driving wheels positioned at the rear of the robot causing an uneven push-pull effect. Despite the above uncertainties and the fact that the robot dynamics is changing with time, the FIE is able to control the effects of these unknowns using simple tuning parameters: “$K_T$”, “$K_DT$”, “$K_X$”, “$K_DX$” and other parameters that decide the range of physical inputs to be fuzzified and manipulated by the controller.
The terms “K_PWM” and “K_THETAOFFSET” will decide the range of defuzzified output to be delivered to the next stage of the control system. For example, “K_PWM” will decide the range of effective PWM signal computed by the FIE. If it is set too high, the PWM signal computed will always be saturated at its peak. On the other hand, if it is set too low, the robot may not move at all despite the fact that there are some small variations in the PWM signal computed. The aforementioned observations are evident in the following plots of theta_offset and PWM. The working range of theta_offset is varying within ±0.4° because “K_THETAOFFSET” is set to 0.4. This argument only holds for the x/dx/ddx system because its output is added directly to the fuzzified theta input. As such, the output of the θ/dθ/ddθ system is the scaled PWM signal which is not recorded. The plot in Graph 1d shows the actual count for the PWM duty cycle. This is sent to the 8751 microcontroller to produce the corresponding PWM signal. It has been observed that the fast varying PWM signal is due to high sensitivity of the θ/dθ/ddθ system. Undoubtedly, this causes the output of the x/dx/ddx system to be masked by the higher prioritized θ/dθ/ddθ system. In fact, this is an inherent problem within our chosen control strategy. However, there are many reasons to stick to the current strategy. This will be explained further in the section for dynamic balancing.

In addition to the two main inputs and two main outputs of the system, i.e. theta, x_pos, theta_offset and pwm, the following plots show how the inputs relate to their derivatives. The values of all the derivatives are relatively consistent about their means, which are near zero. This further proves the robustness of both the θ/dθ/ddθ system and the x/dx/ddx system. One important observation here is to relate the distribution of the variables with their corresponding membership functions. This will allow accurate analysis of the effectiveness of the chosen tuning parameters. To illustrate this idea, the data of theta and dtheta can be analyzed as shown below.
3. DYNAMIC BALANCING

3.1 Design and Development of Robot Hardware

Dynamic balancing of the pole will take place on the top of the balancing platform depicted in Figure 13. The platform has a slight slope at the beginning (Region A) and the end (Region C). The surface of the platform is covered by a layer of rubber in order to increase the friction on the robot’s wheels. A reflective tape is placed along the centre of the platform for the robot to sense its alignment using infrared detectors.

To begin with, the robot will be placed within Region A, the pole will be held in an upright position and released after the system is turned on. The robot will try to balance the pole for a minimum duration of 20 seconds without the pivot point of the pole crossing the line x-x'. After this initial balancing act, the robot will move across the line...
x-x', through Region B, until the pivot point of the pole crosses the line y-y'. After crossing the line y-y', the robot will retrace its initial original path, through Region B until the pivot point of the pole crosses the line x-x' again. When the robot returns to its original position, it will be counted as one lap. During this dynamic transition across Region B, the robot need not stay for any length of time in any regions. The robot will continue to repeat these cycles within 5 minutes. However, for all these laps to be counted as successful cycles, the robot must perform at least 20 seconds of static balancing in Region A before the end of 5 minutes.

3.1.1 Mechanical Design
Based on requirements for dynamic balancing plus the experience gained from static balancing, the idea of including steering in the new design was mooted. This calls for an innovative way to drive the robot such that it can steer itself in all directions. As such, two motors are used to drive the left and right wheels. Using feedback signals from the infrared sensors, the robot will be able to orientate itself back to the center to avoid falling off the platform after numerous transitions. An initial design using CAD software is shown below. Further details on the necessary changes following this initial idea will be described in the sequel.

Some photographs of first prototype capable of dynamic balancing are shown below. In particular, note the changes made to drive the DC motor.
The prototype above uses direct drive for both DC motors, but it turns out that there is not enough starting torque to move the robot. As a result, the structure was modified to include gear trains that will produce a much larger torque.

3.2 Design and Development of Robot Software
3.2.1 Proposed Control Strategy

For dynamic balancing, a different X system is adopted. However, the THETA system is left unchanged to simplify system tuning. By keeping the THETA system intact, we need only to tune the X system to achieve dynamic balancing. On the contrary, if both systems are being tuned simultaneously for dynamic balancing, it will be very difficult to learn the characteristics of the robot during dynamic balancing. This idea is in fact similar to that used in static balancing, except that the velocity, rather than the position, of the robot is now being controlled. As such, it will be more difficult to set correct tuning parameters for the control system.

In order to complete numerous laps, the control system must be reliable and robust. This calls for a velocity profile that is smooth. The basic idea is to accelerate and decelerate gradually. The momentum of the moving robot will also be taken into account to compensate for its inertia. This is especially important when the mechanical structure of the robot becomes heavier in the second prototype. After weeks of
experimentation with the robot on the platform, the following velocity profile was adopted.

Fig 18  Velocity profile of the Dynamic system
The profiles of each different region are classified as follows:

1. Region A First Lap
2. Region B Forward
3. Region C Braking
4. Region B Backwards
5. Region A Braking
6. Final Region A Last Lap

Note that section 2,3,4,5 in the proposed velocity profile will repeat itself during subsequent laps that follow within the 5 minutes duration. The robot will initially take off in Section 1, enters dynamic transitions and finally, performs static balancing in Section 6 (Region A).

In this work, the fuzzy algorithm was implemented by using the windows programming tool – Microsoft Visual C++ version 6.0. In addition, a graphical user interface was created to facilitate trouble-shooting, tuning and monitoring of the robot performance.
The program was developed according to the following sequence:

```
Increment Timer

timer>= static_time ?

Ready to do dynamic balancing ?

x_pos<120 and LAP=0 ?

status= REG_A_FIRSTLAP
direction= FORWARD

x_pos<=80 and 0<LAP<LASTLAP-2 ?

status= REG_A_BRAKE
direction= FORWARD

80<x_pos<220 and direction=FORWARD ?

status= REG_B_FORWARD
direction= BACKWARD

80<x_pos<220 and direction=BACKWARD ?

status= REG_B_BACKWARD
direction= BACKWARD

x_pos>=220 ?

status= REG_C_BRAKE
direction= BACKWARD

X_POS<120 and LAP=LASTLAP-1 ?

status= REG_A_LASTLAP
direction= BACKWARD

X_POS<81 and LAP=LASTLAP ?

status= DO_STATIC
direction= BACKWARD

reg_a_firstlap();
reg_a_brake();
reg_b_forward();
reg_b_brake();
reg_b_backward();
reg_c_brake();
do_static();

dyn_sample= NDYN ?

dynamic x system
```

Fig 20 Flowchart showing switching of the main program between different regions

(1) RS232 Serial Communication Link
The only link between fuzzy algorithm and the micro-controller on the robot. In static mode, it helps to convey 4 bytes of feedback information and one byte of PWM signal. However, in dynamic mode, 2 more bytes of information are required. The first extra byte is the sensor feedback and another byte is for the second PWM signal.

(2) Graphical User Interface (GUI)
The GUI reflects the whole system status and provides an interface between users and the fuzzy algorithm. It displays some of the useful information as follows:
3.2.2 $\dot{\theta}/\ddot{\theta}/\dddot{\theta}$ System

The entire $\dot{\theta}/\ddot{\theta}/\dddot{\theta}$ FIE is common with the one used for static balancing. Only the $x/dx$ FIE is modified to cater to the velocity profile in the dynamic mode.

3.2.3 $x/dx/ddx$ System

The output of the dynamic $x/dx$ system, THETAOFFSET, is used to control the velocity of the robot during transitions on the balancing platform. The fuzzy rule base of the dynamic $x/dx$ system is shown below:

<table>
<thead>
<tr>
<th>x / dx</th>
<th>0.588</th>
<th>0.513</th>
<th>0.438</th>
<th>0.363</th>
<th>0.288</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.0</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
</tr>
<tr>
<td>60</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
</tr>
<tr>
<td>80</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>MP</td>
<td>MP</td>
</tr>
<tr>
<td>100</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>MP</td>
<td>LP</td>
</tr>
<tr>
<td>120</td>
<td>ZE</td>
<td>SP</td>
<td>MP</td>
<td>LP</td>
<td>LP</td>
</tr>
</tbody>
</table>

Table 1  Fuzzy rule base for region A first lap
<table>
<thead>
<tr>
<th>x / dx</th>
<th>0.663</th>
<th>0.588</th>
<th>0.513</th>
<th>0.438</th>
<th>0.363</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.0</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
</tr>
<tr>
<td>60</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
</tr>
<tr>
<td>80</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>MP</td>
<td>MP</td>
</tr>
<tr>
<td>100</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>MP</td>
<td>SP</td>
</tr>
<tr>
<td>120</td>
<td>ZE</td>
<td>SP</td>
<td>MP</td>
<td>SP</td>
<td>SP</td>
</tr>
</tbody>
</table>

Table 2 Fuzzy rule base for region A braking and last lap

<table>
<thead>
<tr>
<th>X / dx</th>
<th>0.34</th>
<th>0.40</th>
<th>0.45</th>
<th>0.49</th>
<th>0.53</th>
<th>0.55</th>
<th>0.57</th>
<th>0.58</th>
<th>0.58</th>
<th>0.59</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>LN</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
</tr>
<tr>
<td>110</td>
<td>LN</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
</tr>
<tr>
<td>120</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
</tr>
<tr>
<td>130</td>
<td>LN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
</tr>
<tr>
<td>140</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
</tr>
<tr>
<td>150</td>
<td>MN</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
</tr>
<tr>
<td>160</td>
<td>MN</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td>MP</td>
</tr>
<tr>
<td>170</td>
<td>SN</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td>MP</td>
<td>LP</td>
</tr>
<tr>
<td>180</td>
<td>SN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td>MP</td>
<td>LP</td>
<td>LP</td>
</tr>
<tr>
<td>190</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td>MP</td>
<td>LP</td>
<td>LP</td>
<td>LP</td>
<td>LP</td>
</tr>
<tr>
<td>200</td>
<td>ZE</td>
<td>SP</td>
<td>SP</td>
<td>MP</td>
<td>MP</td>
<td>MP</td>
<td>LP</td>
<td>LP</td>
<td>LP</td>
<td>LP</td>
</tr>
</tbody>
</table>

Table 3 Fuzzy rule base for Region B forward motion
Table 4  Fuzzy rule base for Region B backwards motion

<table>
<thead>
<tr>
<th>x / dx</th>
<th>-0.288</th>
<th>-0.363</th>
<th>-0.438</th>
<th>-0.513</th>
<th>-0.588</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
</tr>
<tr>
<td>200</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
</tr>
<tr>
<td>220</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>MN</td>
<td>MP</td>
</tr>
<tr>
<td>240</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
<td>MN</td>
<td>LP</td>
</tr>
<tr>
<td>260</td>
<td>ZE</td>
<td>SP</td>
<td>MN</td>
<td>LP</td>
<td>LP</td>
</tr>
</tbody>
</table>

Table 5 Fuzzy Rule for Region C braking

<table>
<thead>
<tr>
<th>x / dx</th>
<th>-0.288</th>
<th>-0.363</th>
<th>-0.438</th>
<th>-0.513</th>
<th>-0.588</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>LN</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
</tr>
<tr>
<td>200</td>
<td>LN</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>SP</td>
</tr>
<tr>
<td>220</td>
<td>MN</td>
<td>SN</td>
<td>ZE</td>
<td>MN</td>
<td>LP</td>
</tr>
</tbody>
</table>

The FIE for the dynamic mode follows the basis of its static counterpart, except for the change in the rule base of the x/dx system. The flowchart of the dynamic FIE is shown in the next section.
It can be seen clearly that all the different region profiles must be tuned separately. The switching between subsequent profiles must be smooth to avoid sudden change in the system response that may make it unstable. Tuning the robot for dynamic balancing is much more complicated than tuning it for static balancing. As a result, a substantial amount of time must be invested to tune the dynamic mode of the system to perform dynamic balancing satisfactorily. The major problems faced are slow acceleration, deceleration and braking of the physical system. This is likely the cause for not being able to follow the proposed velocity profile set in the x/dx rule base.

### 3.2.4 Evaluation of System Performance

The following real-time plots of the robot velocity clearly show that the system is working very well. The first plot shows the superimposition of velocity profiles of all 5 laps on the theoretical plot. To study the performance of the dynamic system, the velocity profile is compared with real-time plots of each lap individually.
The performance of the robot in subsequent laps i.e. laps 2,3 and 4 are relatively similar, thus only the intermediate lap, lap 3, is explained here. The only difference from the first lap is the calling of “Region A Braking” profile instead of “Region A First Lap”. Thus, there is a difference in the values of theta_offset. In fact, the braking in Region A is smoother in response as depicted in the plot. However, the physical performance
does not differ much from Region C braking. This is partly due to the fact that the final response of the robot is influenced by the $\frac{d^3\theta}{dt^2}$ system to a large extent. The theta_offset from the x/dx system has very slight effect on the system response when the main priority is to balance the pole. If the pole is relatively upright at all times, the theta_offset will create a more noticeable effect on the system response.

Finally, the last lap prior to static balancing will again call for the “Region A Last Lap” profile. This will begin at x_pos=120cm when the robot moves backwards in Region B. The profile will last till x_pos=80cm. The reason why the theta_offset value is relatively small in this region is due to the profile designed. The profile for Region B backwards already slows down the speed of the robot when it reaches x_pos=100cm. As such, the “Region A Last Lap” profile will not cause much of a braking effect but to ensure a
smooth transition back to static balancing. If braking is used, it will cause the robot to move back into Region B again.

4. Conclusions

In this paper, an intelligent controller capable of performing static balancing and dynamic balancing of an inverted pendulum mounted on a motorised vehicle is presented. Experimental studies show that the robot is capable of performing dynamic balancing as well as static balancing with great consistency and repeatability. Currently, other advanced control algorithms are being developed and will be implemented in the near future.

References


