INTELLIGENT VELOCITY CONTROL OF A BOUNDING QUADRUPED ROBOT

MICHELE FARAGALLI

MASTER OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

MCGILL UNIVERSITY

MONTREAL, QUEBEC

MAY 2009

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Engineering

ACKNOWLEDGEMENTS

I would like to thank my supervisor Inna Sharf for her patience, guidance, and discipline throughout the course of my research, without her, none of this would have been possible. I'd like to thank Adam Harmat who was a great sounding board for ideas, both good and bad, and who was always there with me late at night debugging the robot. I'd also like to thank Chris Prahacs whose knowledge of electro-mechanical hardware was always a big help. To Philippe Giguère, who spent many hours helping me debug IMU issues, un gros merci! I must also thank Dr James Smith for keeping in touch and answering my questions. Of course, I have to thank all of those who have given me advice and company throughout the course of my masters': Nicolas Plamondon, David Cowan, Olivia Chiu, and Marco Masciola. Finally, I must thank my girlfriend Ashley: thanks for being there for me!

ABSTRACT

The Platform for Ambulating Wheels (PAW) is a hybrid quadruped wheeled-legged robot that can bound, gallop, roll and brake at high speeds, and perform inclined turning. In previous work, the PAW's controller used fixed touchdown and liftoff angles to achieve a stable bounding gait, and these angles were predetermined through an extensive trial and error process.

In this work, an intelligent velocity controller is developed to allow the robot to autonomously find the touchdown and liftoff angles to bound at a desired velocity. This enables the robot to track desired velocities between 0.9 and 1.3 m/s, as shown in a Matlab-Adams co-simulation model of bounding. The controller also demonstrates tracking capabilities in the presence of minor terrain changes.

To implement this controller on the physical platform, an Extended Kalman Filter (EKF) is developed to estimate the forward velocity of the robot required as a controller input. The EKF combines the data from an Inertial Measurement Unit and an estimate of forward velocity found kinematically using measurements from motor encoders and leg potentiometers. The accuracy of the EKF estimate of the forward velocity is validated in simulation and using high speed camera experiments.

Finally, the intelligent controller is implemented and tested on the physical platform demonstrating adequate velocity tracking for set points between 0.9 m/s and 1.3 m/s, as well as transitions between set points in this range.

ABRÉGÉ

Le « Platform for Ambulating Wheels » (PAW) est un robot quadrupède qui possède des roues au bout de ses quatre jambes. Sa combinaison de roues et jambes lui permet de rouler, d'effectuer des virages en inclinant son corps, de sauter, de bondir et de galloper. Dans les travaux précédents, le robot utilisait des angles fixes, trouvés par essais et erreurs, pour pouvoir bondir à une certaine vitesse. Un contrôleur intelligent capable de trouver les angles de façon autonome afin de suivre une vitesse prédéterminée est développé dans ce mémoire.

Premièrement, la performance du contrôleur est évaluée dans une simulation MSC Adams et MATLAB démontrant les capacités à suivre des vitesses entre 0.9 et 1.3 m/s. Le contrôleur démontre une capacité à suivre la vitesse désirée même en présence de changement de terrain mineur.

Ensuite, un filtre Kalman pour système non-linéaire est développé pour estimer la vitesse du robot, un paramètre nécessaire pour introduire le système de contrôle intelligent sur le robot. Les données d'une unité de mesure inertielle et une estimation de la vitesse par des équations cinématiques sont combinés dans le filtre pour estimer plus précisément la vitesse du robot. La précision du filtre est validée en comparant ses résultats contre ceux acquis en simulation et par une caméra à haute vitesse.

Finalement, le contrôleur intelligent est évalué sur le robot en utilisant la vitesse estimée par le filtre Kalman. Les résultats expérimentaux du contrôleur démontre qu'il est capable de bien suivre des vitesses entre 0.9 et 1.3 m/s.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Traversing rough terrain is one of the main reasons for developing dynamically stable running robots. While wheeled and tracked robots can overcome obstacles and soft soil, few offer the versatility that is seen in biological systems. Hence, more versatile autonomous legged robots are being developed to achieve the capabilities of animals or insects. New biologically inspired robots could prove useful for planetary surface exploration, military reconnaissance, and search and rescue missions. The Platform for Ambulating Wheels, or PAW, is a hybrid wheel-leg quadruped robot built to combine the advantages of both rolling and running robots. This thesis shall introduce an intelligent velocity controller that increases the locomotion capabilities of PAW and an extended Kalman filter (EKF) that allows for increased velocity sensing capabilities. Ultimately, the goal of this research is to enhance PAW's autonomy and adaptability to various terrains.

This chapter introduces dynamically stable running robots, provides a background of existing control methods for quadruped robots, describes the PAW platform, and explains why intelligent velocity control is necessary for increasing the capabilities of the robot. Chapter 2 describes the intelligent velocity controller for the bound gait and provides results from testing in a Matlab-Adams co-simulation environment, including an evaluation of its capabilities with changing terrain properties. Chapter 3 describes the sensing requirements and the extended Kalman filter used to estimate forward bounding velocity. Chapter 4 demonstrates the effectiveness of the intelligent controller on the PAW robot in an indoor test environment. Finally, Chapter 5 concludes this work by discussing the contributions of this thesis and suggesting avenues for future work.

1.1 DYNAMICALLY STABLE RUNNING ROBOTS

Legged robots can be classified as statically and dynamically stable robots. The former requires continuous ground support by the legs at all instants through the gait. The robot's center of mass (COM) projection onto the ground must be located within the support area provided by the legs at each instant to ensure the robot's static stability throughout the gait. They tend to be slow moving, with legs resembling traditional manipulator designs. Faster moving 'running' legged robots that do not require the COM to be located within the area of support provided by the legs are described as dynamically stable legged robots. Dynamically stable legged robots can have phases in their gait during which no legs are in contact with the ground, termed 'flight phases'. These running robots also take advantage of the legs' passive dynamics to simplify their controllers. Marc Raibert's monopod hopper was the first dynamically stable running robot [1] and consisted of a hip motor combined with a prismatic actuator and spring to mimic a leg. The robot was mounted on a pivoting boom and was capable of hopping at a speed of 1.2 m/s [1]. To produce stable hopping, the prismatic springs stored the potential energy in the first portion of the stride, and released it in the second portion to produce leg liftoff, similar to a pogo stick. The dynamics of the system was modeled as a spring-loaded inverted pendulum (SLIP). Figure 1.1 shows the 3D version of Raibert's monopod hopper in action, capable of hopping at 2.2 m/s [1], and Figure 1.2 provides a visual representation of the SLIP model.

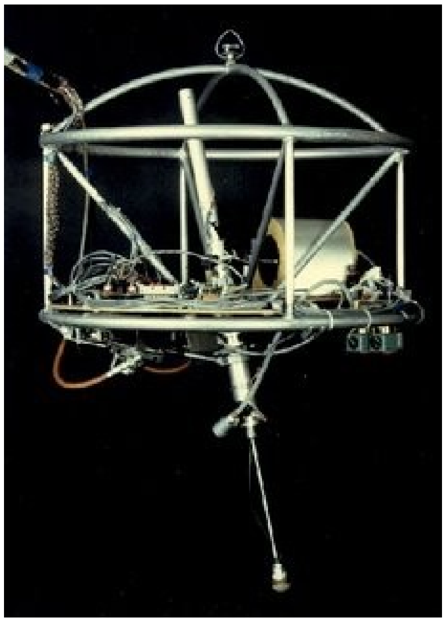


FIGURE 1.1 RAIBERT'S MONOPOD HOPPER [2]

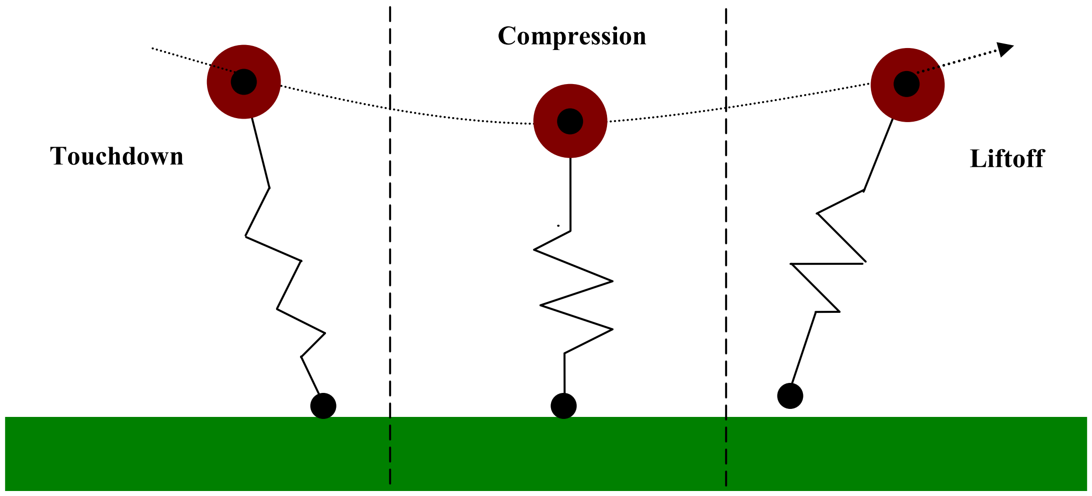
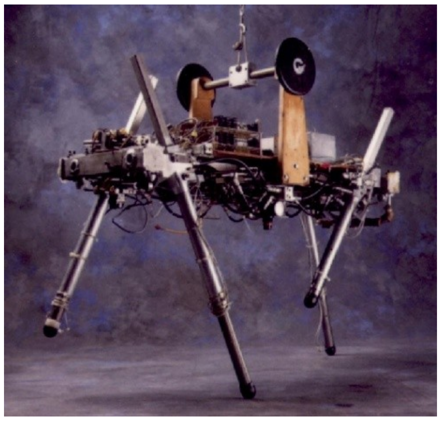
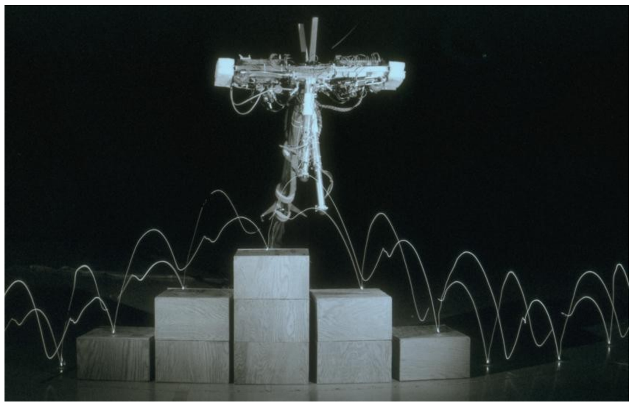


FIGURE 1.2 SPRING-LOADED INVERTED PENDULUM MODEL (SLIP)

This SLIP model was extended to achieve bipedal and quadrupedal locomotion as well. Figure 1.3 shows a two and a four legged running robot both developed by Raibert. Similar to the monopod, the bipedal and quadrupedal robots used a hip motor to control the leg position, and a prismatic actuator in the leg to control the energy injected in the system to maintain dynamic stability. His quadruped was capable of pacing, trotting, and bounding [3]; while the biped, mounted on a boom, was capable of step climbing [4] and running at up to 4.3 m/s [1]. Similar research on dynamically stable legged robots is being performed on the four legged KOLT robot, shown in Figure 1.4, which begun as a monopod called the OSU Dash-leg running on a treadmill [5]. The leg design differs from Raibert's as it uses pulleys to control hip and knee joints, offering one more degree of freedom per leg.



(A) RAIBERT'S QUADRUPED TROTTING [2]



(B) RAIBERT'S BIPED STEP CLIMBING [6]

FIGURE 1.3 RAIBERT'S ROBOT'S

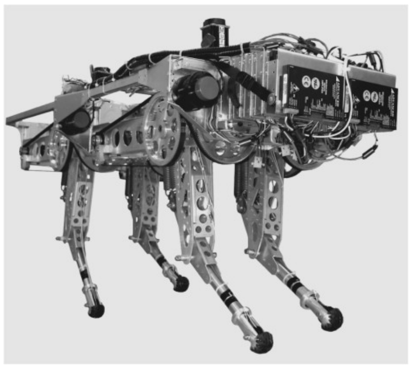


FIGURE 1.4 STANFORD'S KOLT ROBOT [7]

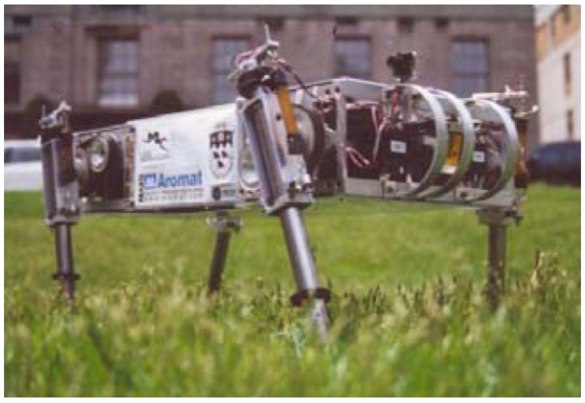
1.2 Control of Quadruped Robots

Several control strategies are available to quadruped robots. The traditional statically stable robots require precise foot placement to ensure stability at each moment in the robot's motion. These robots, considered *walking* robots, use a polygon support pattern to determine the feet positions. Conventional PID or PD laws are used at the motor position control level. Higher level controls optimize gait pattern generation and foot placement [8], and range from offline machine learning for parameter tuning [9] to online adaptive control methods [10]. In [11], the Sony AIBO robot's walking speed is increased by optimizing the gait's parameters offline using an evolutionary approach to a learning algorithm. Offline gait learning focuses on gait optimization for specific terrain types. Other researchers [12] [13] [14] focus on central pattern generators (CPG) for generating a locomotion gait in biologically inspired robots to adapt the gait according to the environment based on reward functions. The CPG is made up of coupled nonlinear oscillators to form a network capable of generating synchronization patterns used to coordinate actuation of the robot. Although approaches differ, the underlying objective is to optimize legged robots' performance in all terrain types.

While dynamically stable robots rely primarily on the natural dynamics of the system [15], control laws are still required to ensure stable locomotion. The robot's gait parameters are optimized based on the desired controller set point. These set points can include the desired attitude [16] [17], direction [18], velocity and hopping height [19], specific maneuvers [20], and minimizing energy consumption [21]. As running robots dynamics are complex and non-linear in nature, fuzzy control [22], evolutionary searches [23], and learning algorithms [24] are often used to determine the required controller outputs to achieve the desired set point. Raibert, on the other hand, simplified controllers by using virtual leg principles and relying on the dynamic stability of the system [1]. Ultimately, to make legged robots adaptable to their environment, they must be capable of sensing disturbances such as obstacles or terrain changes and decide which gait is best for completing the task. To do so, many robots often have state estimation or visual odometry feedback that alerts the controller of a change required in the outputs [25].

1.3 Platform for Ambulating Wheels (PAW)

At McGill University, two robots have been developed that use the principles of dynamically stable legged locomotion: Scout II [26] and PAW [27], shown in Figures 1.5(a) and (b) respectively. SCOUT II became the world's first physical robot to achieve stable galloping [26] and did not use prismatic actuators in the legs. The robot uses un-actuated compliant legs, controlled individually, that exploit the passive dynamics to store potential energy and release it to produce leg liftoff, leading to stable galloping at 1.3 m/s.


(a) SCOUT II [26]

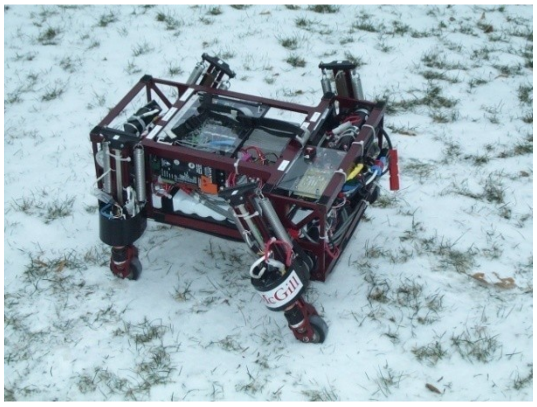
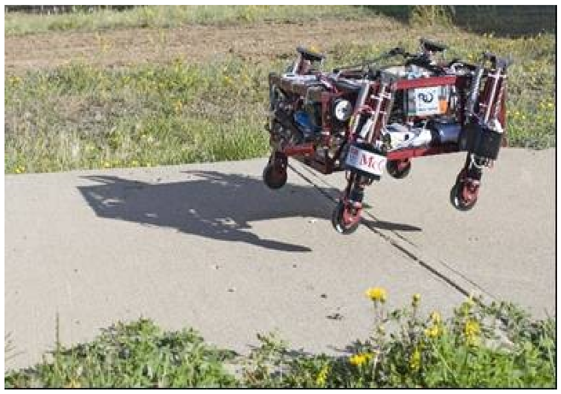

(b) PAW robot [27]

Figure 1.5 McGill Quadrupeds

Based on the success of SCOUT II, a smaller version, termed *hybrid* due to the addition of wheels at the end of each compliant leg, was developed. The Platform for Ambulating

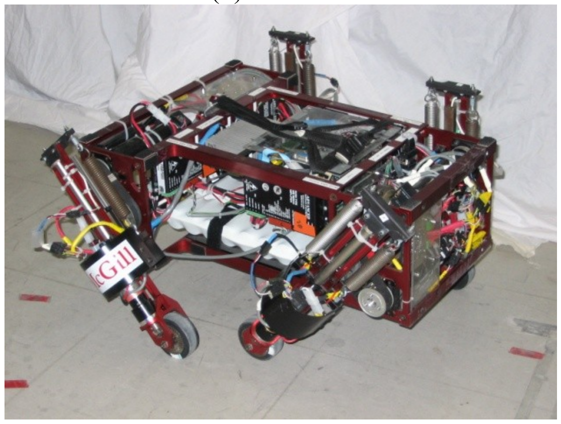
Wheels (PAW), an under-actuated robot with minimal sensing capabilities and simple controls, is able to achieve stable bounding and galloping at speeds over 1.0 m/s [28]. In addition, PAW can use the wheels to provide the forward motion while the actuated hips allow turning and braking at high speeds [27]. PAW's locomotion capabilities are illustrated in Figure 1.6.


(A) BOUNDING

(B) GALLOPING

(C) ROLLING

(D) SLOPE CLIMBING


(E) INCLINED TURNING

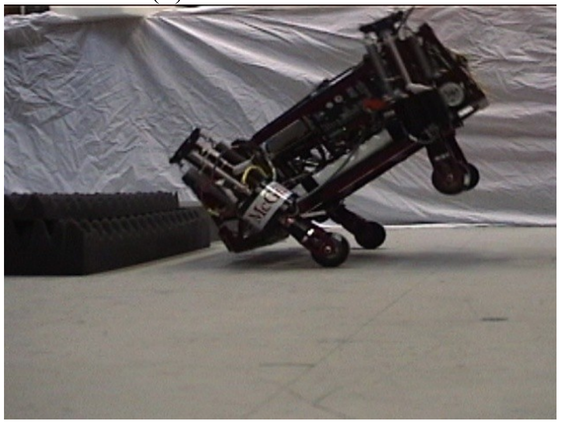

(F) BRAKING AT HIGH SPEEDS

FIGURE 1.6 PAW'S LOCOMOTION CAPABILITIES

PAW's motors are located in the hip for rotating the leg, and in the toes for powering the wheel, respectively. The galloping and bounding controllers require only linear potentiometers to measure spring compression in the legs, and encoders in the hip and wheel motors to measure angular position and velocity as inputs [29]. To achieve leg liftoff for the flight phase required for the gallop and bound gaits, potential energy must be stored in the leg springs during the stance phase. This limits the range of stability of the bound and gallop gaits as the robot relies on the un-actuated compliant nature of the legs to avoid toe dragging or stubbing, which can lead to a failure in the gait. An increase in maneuverability during running gaits will therefore require more intelligent control methods and sensing capabilities to take full advantage of the limited number of actuators [30]. A more detailed description of PAW's gaits, controls and important parameters is contained in Chapter 2.

1.4 Purpose of Intelligent Velocity Control on PAW

Intelligent control methods allow a system to track various set points and adapt to disturbances, thus increasing its autonomy. However, learning algorithms can significantly increase the complexity of the controllers, increase the computational time, and possibly require more information than is available on the robot's state. This section details the reasons for using intelligent control on PAW, the types of controls applicable to the platform, and why velocity control was implemented first.

1.4.1 Intelligent Controllers

PAW's existing controller uses specific hip angles combinations for the leg touchdown and liftoff instants to achieve stable running gaits. For bounding, the front legs and rear legs are actuated in pairs to fixed hip angles for touchdown and liftoff, whereas for galloping, there is a fixed angle phase difference between the pair's hip angles at touchdown and liftoff. The fixed values are predetermined through a trial and error process to achieve a stable gait. The control scheme for bounding is described in further detail in Chapter 2. Using an intelligent method for control allows the robot to modify these fixed values in order to adapt to the desired velocity, desired turning rates, and the terrain. As terrain properties change, the robot dynamics are affected. An intelligent controller can adapt to the change in robot states in order to maintain the desired set point. Increasing the robot's autonomy and versatility renders it applicable in more realistic situations where galloping and bounding would prove useful.

A plethora of intelligent control methods have been developed to increase the autonomy of robotic systems. Ideally, mimicking biological models would lead to the best solutions; however these would require neural network controllers [31]. Sets of artificial neural networks can provide pattern generation for various gaits and gait control. Neural networks, however, rely on extensive sensing capabilities and actuator coupling to create rhythmic patterns [31]. For PAW, the gait patterns are generated based on the passive dynamics of the system with each leg controller decoupled. In addition, our platform provides limited actuation and sensing capabilities that would render neural controllers ineffective. Fuzzy controllers provide a solution that uses some heuristic knowledge of the system to increase the robots adaptability. The fuzzy controller can be called at the apogee of the flight phase, or top-of-flight (TOF), to fuzzify the control parameters and find the optimal controller inputs based on a set of rules. Although Marhefka and Orin [22] demonstrate the capabilities of the fuzzy controller in a simulated galloping quadruped, the rule-base that specifies the membership function for each controller input requires additional parameter storage and computations per stride relative to the controller developed by Raibert [1]. In [24], the fuzzy controller is compared to a Levenberg-Marquardt (LM) learning algorithm that modifies the Raibert controller [1] in a simulated leg for a galloping robot. While achieving similar results as the fuzzy controller, the LM-Raibert controller would entail fewer modifications to our platform, while still improving PAW's tracking abilities.

1.4.2 Velocity Control

Dynamically stable legged robots have the advantage of not requiring continuous ground support throughout their running gait to ensure static stability. They are able to leap over obstacles, and traverse terrain that statically stable legged robots may not be able to. Although advantageous, these robots require precise foot positioning in "safe" locations in order to continue their stride. Thus, controlling foot placement is a key element in rough terrain traversal. Choosing the foot location can be based on visual odometry and other sensorial data. However, these foot locations greatly affect the stability, forward velocity, hopping height, and direction of travel of the system. Raibert's work in [4] addressed this issue by analyzing the parameters that define a running robot's step length. For steady state running, the step length is the sum of distance traveled during stance and flight. In his analysis, Raibert defines forward velocity as the key parameter in adjusting step length. Other researchers discuss possibilities of adjusting hopping height [19] useful for overcoming obstacles, or turning rates during running [7]; however, velocity control is an appropriate starting point for a dynamically stable robot with the dimensions of PAW. As will be discussed in Chapter 2, only minor modifications to the controller are needed to achieve bounding velocity control.

1.5 Thesis Organization

This thesis will focus on intelligent velocity control of the PAW robot during the bound gait. Chapter 2 will describe the existing control scheme on the robot, present the modifications necessary for PAW to achieve velocity tracking, and demonstrate the controller's performance in simulation. Chapter 3 will focus on the development and testing of an extended Kalman filter (EKF) used to estimate forward velocity during bounding on the robot, which is a necessary input to the intelligent controller. Chapter 4 will describe the controller's performance on the robot, as well as discuss some important parameters and limitations. Finally, Chapter 5 discusses the contributions made by this thesis and addresses areas for future work.

CHAPTER 2: INTELLIGENT VELOCITY CONTROLLER

This Chapter describes the existing bounding controller on PAW and outlines the modifications necessary to accomplish intelligent velocity control during bounding. To achieve intelligent velocity control, a Levenberg-Marquardt learning algorithm is called at the top-of-flight (TOF) instant of the bound stride to compute the necessary gains in a modified version of Raibert's controller [1], which in turn determines the hip angles required for achieving a particular forward bounding velocity. In order to verify the effectiveness of the controller, a model of the robot was created in the dynamics simulation software MSC Adams, which is used in conjunction with Matlab to control the robot in simulation. The results of this simulation include: a comparison with the existing controller, an evaluation of velocity tracking capabilities, an assessment of robustness to terrain changes, and the limitations of this method, which are presented in the final section of this chapter.

2.1 EXISTING BOUND CONTROL STRATEGY

The PAW robot's existing control strategy allows it to achieve two dynamically stable gaits: the bound and the gallop. Although this work focuses on the bound gait, an extension could easily be made to the gallop. To fully understand how the intelligent controller is implemented, it is important to first describe the bound gait and the existing control scheme.

2.1.1 Running Gaits

The footfall patterns for one stride of various quadruped running gaits are compared to the quadruped bound in Figure 2.1 below. The transverse and rotary gallop requiring the legs to touchdown individually, are considered as four beat gaits, whereas the bound and trot are two beat gaits as the legs touchdown in pairs. In biological systems, the bound gait is seen as a transition gait between trot and gallop; however some animals, like the squirrel, use the bound gait as a means of high speed locomotion [32].

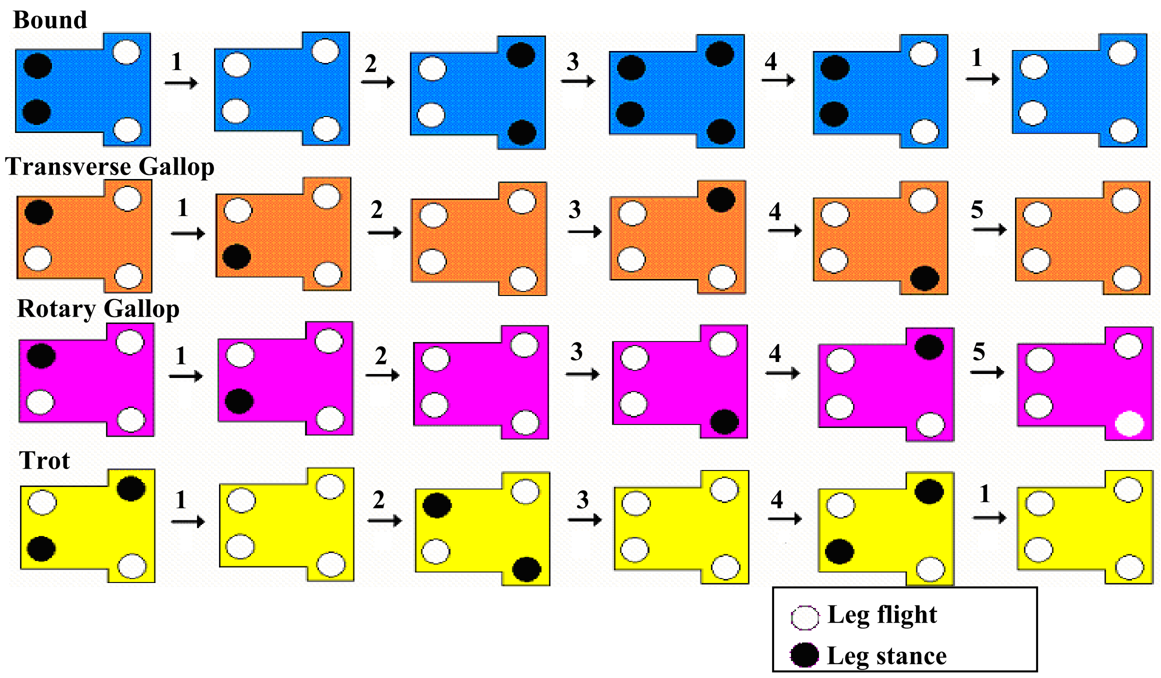


Figure 2.1 Footfall Patterns for Various Gaits

Although the gallop gait is traditionally more efficient with respect to energy consumption in running animals at high speeds [33], it has been demonstrated that in the case for PAW, bounding is a more efficient means of dynamically stable locomotion [29].

This is likely due to the dimensions of the robot, whereas a robot with longer legs, like Scout II, has a more efficient gallop gait at high speeds [29]. Thus, in this work, the intelligent controller is implemented solely for the bound gait. It is possible to pursue the intelligent controller strategy during galloping, as the gallop gait in PAW consists simply of adding a fixed angular phase difference in the hip angles for the front and rear leg pairs causing a change in the footfall pattern. However, this is out of the scope of this thesis.

2.1.2 Detecting State Changes on PAW

In order to achieve dynamically stable bounding, the individual leg controllers require detection of two states: stance and flight. A simplified visual representation of PAW bounding is seen in Figure 2.2.

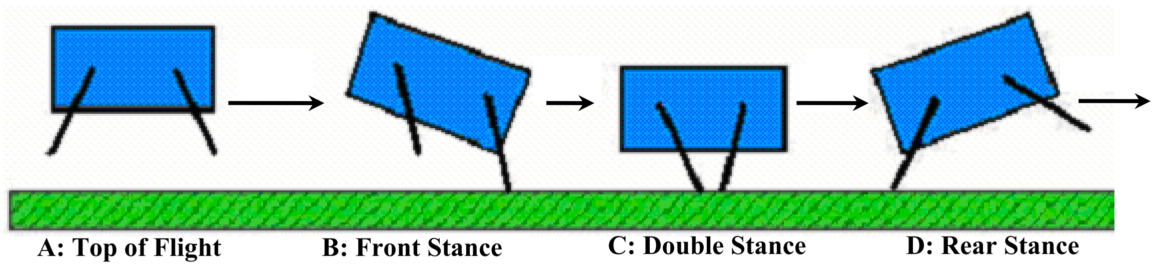


Figure 2.2 PAW's bound sequence

During bounding, the front legs are controlled symmetrically, as are the rear legs, creating virtual leg pairs at both the front and rear of the robot. The front and rear leg pairs are decoupled as they do not require knowledge of each other's states to provide appropriate leg actuation. Thus, once the state of a leg is detected, the controllers will servo the hip to the appropriate angle. During flight, the leg will want to achieve the correct touchdown angle for the stance phase while taking body pitch into account. During stance, the legs are actuated to the correct liftoff angle, at which point the energy stored in the springs is released, causing the legs to take flight.

Touchdown and liftoff detection is determined via potentiometers located on each robot leg. When the leg touches down, a predetermined compression threshold is achieved indicating to the robot controller that the leg has touched down. Similarly for takeoff, the leg extends beyond a predetermined threshold indicating that the leg is in flight. This method of determining leg states can be improved as vibration in the legs and inaccuracies in the potentiometer data may affect the performance of the state detection. As PAW has actuated wheels on the distal ends of the legs, it would be possible to determine the touchdown or liftoff instant using the wheel motor encoders. However, for the purpose of this work, the leg compression method for state determination is sufficient.

2.1.3 Motor Control

PAW's leg actuators, located at the hips, are controlled by traditional PD law of Equation (2.1) which is used during the gait cycle to servo the robot's legs to the desired angles:

$$\tau_{hip} = k_P(\gamma - \phi_D - \theta) + k_D\dot{\phi} \tag{2.1}$$

where: τ_{hip} is required motor torque,

γ is the actual hip angle with respect to the body,

ϕ_D is the desired leg angle with respect to the ground,

θ is the body pitch,

$\dot{\phi}$ is the actual leg angular rate, and

k_P and k_D are the controller gains.

The desired hip angular rate, $\dot{\phi}_D$, is zero in this case, allowing the derivative error of the traditional PD law of Equation (2.1) to be simplified to contain only the $\dot{\phi}$ term. Equation (2.1) is used during the leg flight phase, whereas during stance, the hip torque is commanded to the motor saturation limit. Once the hip's desired liftoff angle is reached in the stance phase, a stance brake state is detected and the control law in Equation (2.1) ensures that the hip is commanded to stay at the required angle, ϕ_D, until liftoff occurs.

The hybrid nature of PAW requires the wheel motors to be either actively controlled or mechanically blocked during dynamic legged locomotion. For active wheel control, the wheels are controlled to remain at rest via a PD position control law, reducing the wheel torque equation to:

$$\tau_{wheel} = k_P \theta_{wheel} + k_D \omega_{wheel} \tag{2.2}$$

where: τ_{wheel} is required wheel motor torque,

θ_{wheel} is the wheel angle,

ω_{wheel} is the wheel angular rate, and

k_P and k_D are the controller gains tuned via trial and error.

2.1.4 Touchdown and Liftoff Angles

From the previous subsections, it can be deduced that the critical parameters for PAW's bound gait are the touchdown and liftoff angles of the legs, which are controlled from the hip actuators. Due to the passive nature of the prismatic joint in PAW's legs, the hip angles will determine the amount of spring compression and extension to ensure liftoff and maintain dynamic stability. Specific combinations of these result in dynamically stable gaits at particular forward velocities and hopping heights. However, the manual tuning to find these values is time consuming; moreover, achieving alternate forward velocity set points requires re-tuning for both the leg touchdown and liftoff angles.

PAW's existing controller uses fixed leg touchdown and liftoff angles during the bounding gait developed by Smith in [29]. Table 2.1 demonstrates experimental results of combinations of hip angles resulting in various forward bounding velocities. The center of mass (COM) speed shown in Table 2.1 is calculated by dividing the distance traversed by the robot during the experiment, over the elapsed time.

Test	Front touchdown angle (deg)	Rear touchdown angle (deg)	Front liftoff angle (deg)	Rear liftoff angle (deg)	COM speed (m/s)
1	-20	-22	4	12	0.75
2	-20	-22	6	14	0.83
3	-20	-22	6	16	0.83
4	-20	-22	8	16	0.91
5	-20	-22	10	18	1.00

TABLE 2.1 PREDEFINED PAW BOUNDING TOUCHDOWN AND LIFTOFF ANGLES

2.2 INTELLIGENT VELOCITY CONTROLLER

As running robots have a flight phase in their gait, this allows them to overcome obstacles and difficult terrain. As discussed in section 1.4.2, foot placement plays a key role in the robot's locomotion. Moreover, in the case of a bounding robot that relies on the passive dynamics such as PAW, the desired foot placements must ensure that a sufficient amount of energy is injected into the system to maintain stable running. The foot location is determined from the leg kinematics relations between the hip angles and leg length as shown in Figure 2.3.

Raibert [4], as introduced in section 1.4.2, analyzed step length in steady state running. He defines step length as the sum of distance traveled during stance and flight between foot touchdown instants. This is shown in Equation (2.3).

$$L_{step} = \dot{x}_s T_s + \dot{x}_f T_f \tag{2.3}$$

In this equation: L_{step} is step length,

$\dot{x}_s$ and $\dot{x}_f$ is COM velocity during stance and flight phases respectively,

T_s and T_f is the time of stance and of flight respectively.

Although the duration of flight, the duration of stance, and forward velocity play a role in the step length, Raibert identifies forward velocity as the key parameter in adjusting step length [4]. From step length, it is possible to determine the foot touchdown locations, which are directly related to the hip angles.

2.2.1 Raibert's Velocity Controller

Raibert's controller, developed in [1] for velocity control of a monopod hopper and presented in Equation (2.4), finds the necessary foot placement at touchdown and liftoff instants for achieving a desired velocity set point. The desired velocity, in this case, is taken as the robot's center of mass (COM) forward velocity at the top-of-flight (TOF) during the hopping sequence. The first term in Equation (2.4) estimates the required foot placement for keeping constant running velocity. The second term corrects the velocity error by adding to the desired foot position.

$$x_{foot} = K_1 \frac{T_s}{2} v + K_2 (v - v_d) \tag{2.4}$$

In the above, T_s is the time of the previous stance period, v is the COM velocity at TOF, v_d is the desired TOF velocity, K_1 and K_2 are the gains.

This strategy can be translated to a quadruped robot as the legs are controlled independently during bound, as described in section 2.1.2. Thus, the controller is called at the TOF instant in the bound sequence to compute the desired foot placement at touchdown and liftoff for the front and rear leg pair, indicating that Equation (2.4) must be used four times, once for each desired foot placement, to track the desired velocity set point. Knowing the desired foot placement x_{foot}, the required hip angles are found kinematically using Equation (2.5) for the front and rear leg pairs. PAW's kinematic parameters are described in Figure 2.3. Subscripts f and r are used for front and rear leg pairs respectively.

$$\phi_{f,r} = \sin^{-1}\left[\frac{x_{foot_{f,r}}}{l_{f,r}}\right] \tag{2.5}$$

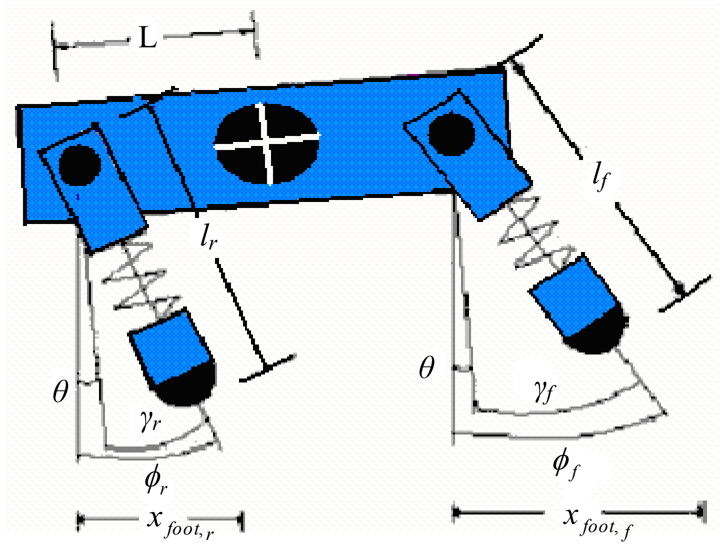


FIGURE 2.3 PAW PARAMETERS

Note that the body pitch is taken into account in the hip motor control of Equation (2.1) as foot position is found with respect to the local vertical axis.

2.2.2 MODIFICATIONS TO RAIBERT'S CONTROLLER

Raibert developed Equation (2.4) to control foot touchdown position for a monopod with a prismatic and a rotational actuator. In this case, the rotational actuator, i.e. the hip motor, servos the leg to the desired position, while the prismatic actuator ensures that enough energy is injected back into the system to avoid toe dragging or stubbing. Thus, this method relies on controlling the energy injected during the stance phase, whereas for PAW, this is determined by the passive dynamics of the system. To ensure that the foot placement equation takes the passive nature of PAW's legs into account, a third term, α_3, is added to Equation (2.4). Additionally, as the bound sequence in PAW is periodic in nature, we can assume constant stance time and simplify the first term of the equation, to give:

$$x_{foot} = \alpha_1 v + \alpha_2 (v - v_d) + \alpha_3 \tag{2.6}$$

For PAW's bound gait, Equation (2.6) is used to compute front and rear leg pair foot placement at touchdown and liftoff. This means the gains α_1, α_2, and offset α_3 must be tuned four times; for front and rear leg touchdown and liftoff combinations.

2.2.3 LEVENBERG-MARQUARDT LEARNING

To enable traversal of terrains with various physical properties, PAW must be adaptable to a range of velocity set points while maintaining dynamic stability. To avoid toe stubbing or dragging failures, the α gain tuning requires an extensive trial and error process for each desired forward velocity due to the compliant nature of PAW's prismatic legs. The approach developed in [24] and shown in Equation (2.7), which uses a Levenberg-Marquardt (LM) learning algorithm, is implemented on PAW allowing the gains in Equation (2.6) to be adjusted adaptively at each stride. This reduces the manual gain tuning process, requiring simply an initial value for each gain.

The LM intelligent algorithm solves the least squares problem to tune gains α_1 and α_2 of Equation (2.6) according to the following update law [24].

$$\alpha_{m_{j+1}} = \alpha_{m_j} + \frac{p_m}{p_m^2 + \lambda_m} e_j \qquad m = 1, 2 \tag{2.7}$$

where e_j is the system velocity error: $e_j = v_{d,j} - v_j$,

λ_m is the step size control variable,

p_1 is actual velocity v,

p_2 is the negative of the system velocity error, and

j is the stride index updated at each TOF instant.

At each TOF, Equation (2.7) is used to update the gains α_1 and α_2 for each leg pair for both touchdown and liftoff. According to [34], the α_3 term aids in compensating for the loss in energy that could lead to instability; this term remains constant for small velocity changes. The foot placement law of Equation (2.6) employs the newly tuned gains to compute the foot position, which is then employed in Equation (2.5) to find the appropriate hip angle as the hip motor controller input parameter.

2.3 SIMULATION RESULTS

To verify the validity of the proposed intelligent velocity controller, a simplified model of the PAW robot was created in the dynamics simulator MSC Adams. This program allows a Matlab interface where the robot controller is embedded. In the following subsections, we describe the simulation model, present a comparison between the intelligent velocity controller and the existing controller, demonstrate the ability of the intelligent controller to track varying set points, investigate its robustness to terrain modifications, and discuss some limitations.

2.3.1 MSC ADAMS MODEL

The dynamics model, described in [29] and developed in MSC ADAMS, is geometrically and inertially consistent with the actual robot; the main difference is that the foot wheels are modeled as toes without actuated wheels. This is an important discrepancy between the physical robot and dynamics model, as wheels cause some energy loss upon touchdown and liftoff. This will become apparent in Chapter 4 where we consider the physical bounding tests. Table 2.2 displays the robot's physical parameters and Figure 2.4 displays the MSC Adams PAW model [29].

Parameter	Value
Front body width	0.336m
Rear body width	0.240m
Body length	0.494m
Body height	0.170m
Body mass	15.7kg
Leg length	0.212m
Leg spring constant	3500N/m

TABLE 2.2 SIMULATION PARAMETERS

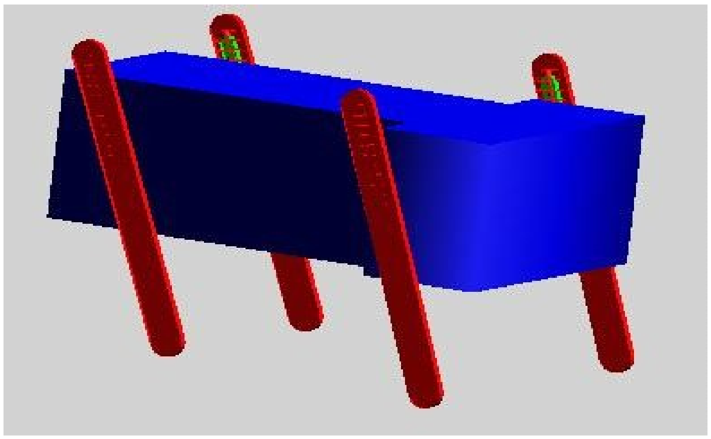


FIGURE 2.4 MSC ADAMS PAW MODEL

The robot's legs are controlled throughout the bound cycle, and the controller inputs for leg angles are found as discussed in Section 2.2, once per cycle at the top-of-flight. Figure 2.5 displays the block diagram of the controller developed in MATLAB for the MSC Adams dynamics model.

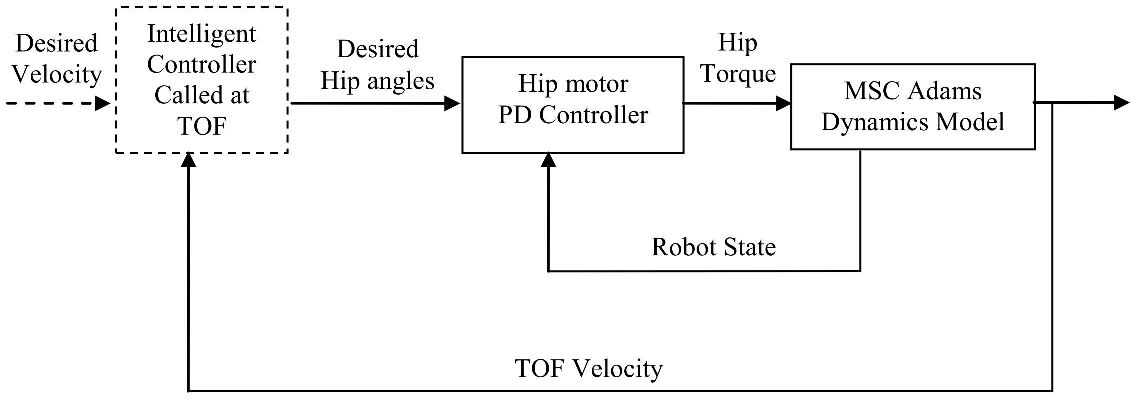


FIGURE 2.5 BLOCK DIAGRAM OF INTELLIGENT CONTROLLER IN SIMULATION

2.3.2 CONTROLLER COMPARISON

To demonstrate the intelligent controller's performance, we first compare it to the existing fixed-angle controller [29] for rate of convergence to a particular velocity set-point. The simulation is started by allowing the robot to fall from a fixed height above the ground. PAW is given an initial forward velocity of 0 m/s, with an initial forward pitch of 60º. The robot touches down and lifts off using known fixed hip angles to initiate stable bounding. The start up routine is used for both the intelligent and previously developed controller. For the intelligent controller, a fixed forward velocity of 1.3 m/s is commanded when the robot reaches its first top-of-flight after the start up sequence. The existing controller is commanded, throughout the bound cycle, to predefined forward and rear touchdown and liftoff angles that yield a forward velocity of 1.3m/s. These predetermined angles were found via trial and error in [29].

Figure 2.6 displays the COM forward velocity of the simulated bounding gait where PAW starts from rest and accelerates to the desired velocity of 1.3 m/s. The 'plateau' sections of the bound profile indicate the flight phase of the bound gait (phase A from Figure 2.2). The small spike and deceleration is explained by the front leg touchdown (phase B from Figure 2.2), followed by the double stance phase (phase C from Figure 2.2). The robot then accelerates as the rear legs are actuated prior to liftoff (phase D from Figure 2.2), after which the 'plateau' sections appear to indicate flight has occurred. The responses from the two controllers demonstrate the faster convergence of the intelligent controller. The phases for one bound stride are indicated in Figure 2.6.

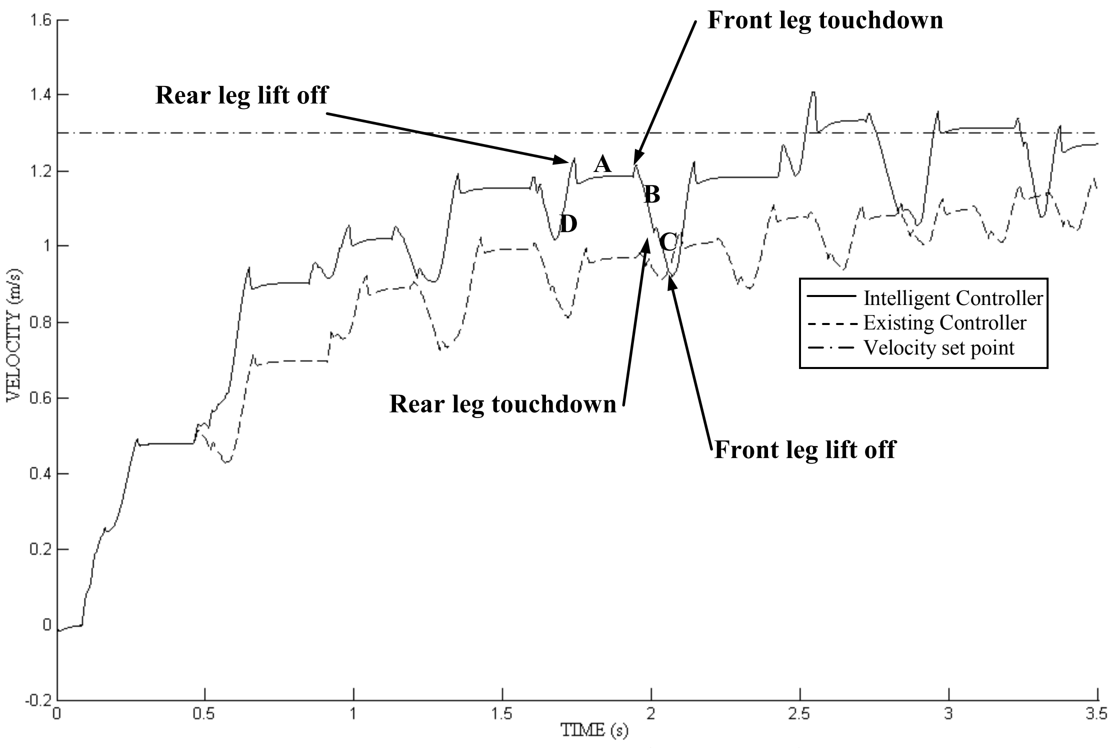


FIGURE 2.6 INTELLIGENT VS. EXISITING (FIXED-ANGLE) CONTROL

The intelligent controller allows the hip angles to adapt quickly to the desired velocity set point while the existing controller does not achieve the desired velocity after 3.5s (approximately 10 strides). The intelligent controller is able to adjust the hip angles to accelerate to the set point in 2.5s (approximately 6 strides).

2.3.3 VELOCITY VARIATION

The real benefits of the intelligent controller, however, can be gleaned from Figure 2.7 where adaptation to the variable velocity set-point is demonstrated. Unlike the existing bounding controller, the intelligent controller is able to track quite accurately the velocities between 0.9 m/s and 1.3 m/s and furthermore, to seamlessly transition between the different set-points. Near the end of the test, a desired velocity of 0 m/s was used to successfully stop the robot without losing stability.

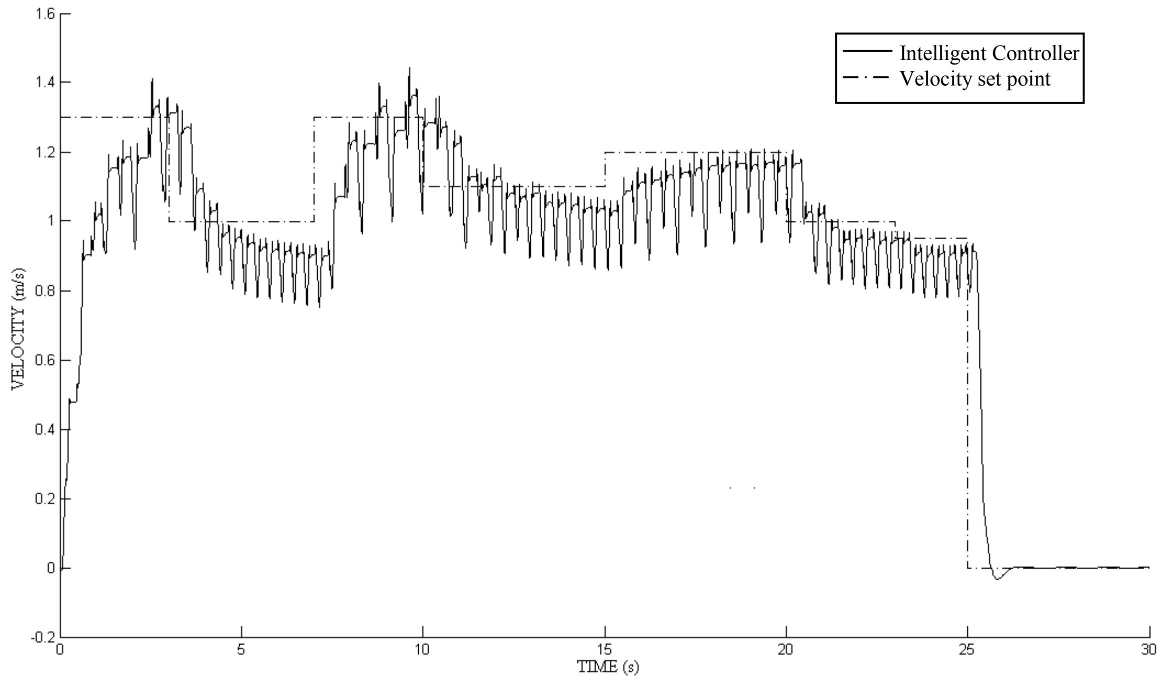


FIGURE 2.7 TRACKING VELOCITY VARIATIONS

In this simulation, the velocity set point was varied 8 times during a 30 second bound trial. The velocity intervals in this test varied between 3 and 5 seconds in length, yielding 7 to 16 bound strides per interval. Within five strides of each change in set point, the robot reached the new velocity set point. Once the convergence to the new set point occurred for each interval, the robot continued to maintain the bound velocity with errors less than 0.1 m/s.

The effects on the desired hip angles of the adaptation with the intelligent controller can be seen in Figures 2.8 and 2.9. In Figure 2.8, the robot is commanded to accelerate from 1.0m/s to 1.3m/s. The intelligent controller achieves these results by increasing the magnitudes of both the desired touchdown and liftoff angles. In these experiments, the negative hip angle indicates that the leg is forward from the hip joint thus preparing for touchdown; whereas a positive hip angle indicates that the leg is behind the hip position, preparing for liftoff. At higher speeds, the 'plateau' portion of the velocity plot is longer, indicating an increase in flight time.

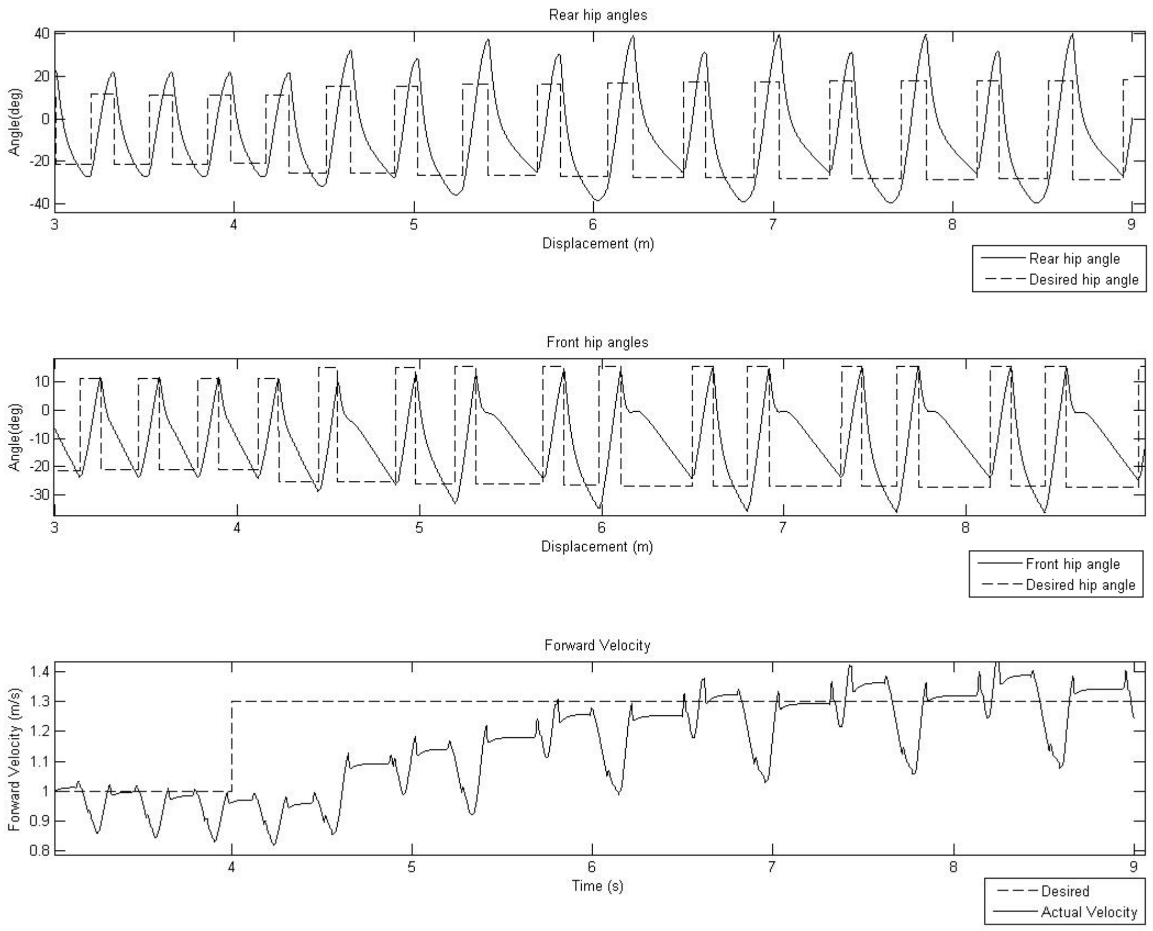


FIGURE 2.8 PAW ACCELERATING FROM 1.0 TO 1.3M/S

An example of the effects of deceleration using the intelligent controller is shown in Figure 2.9. Here, the robot's desired velocity is varied from 1.3m/s to 1.0m/s, causing the hip angle fluctuations to shrink and the flight time to decrease.

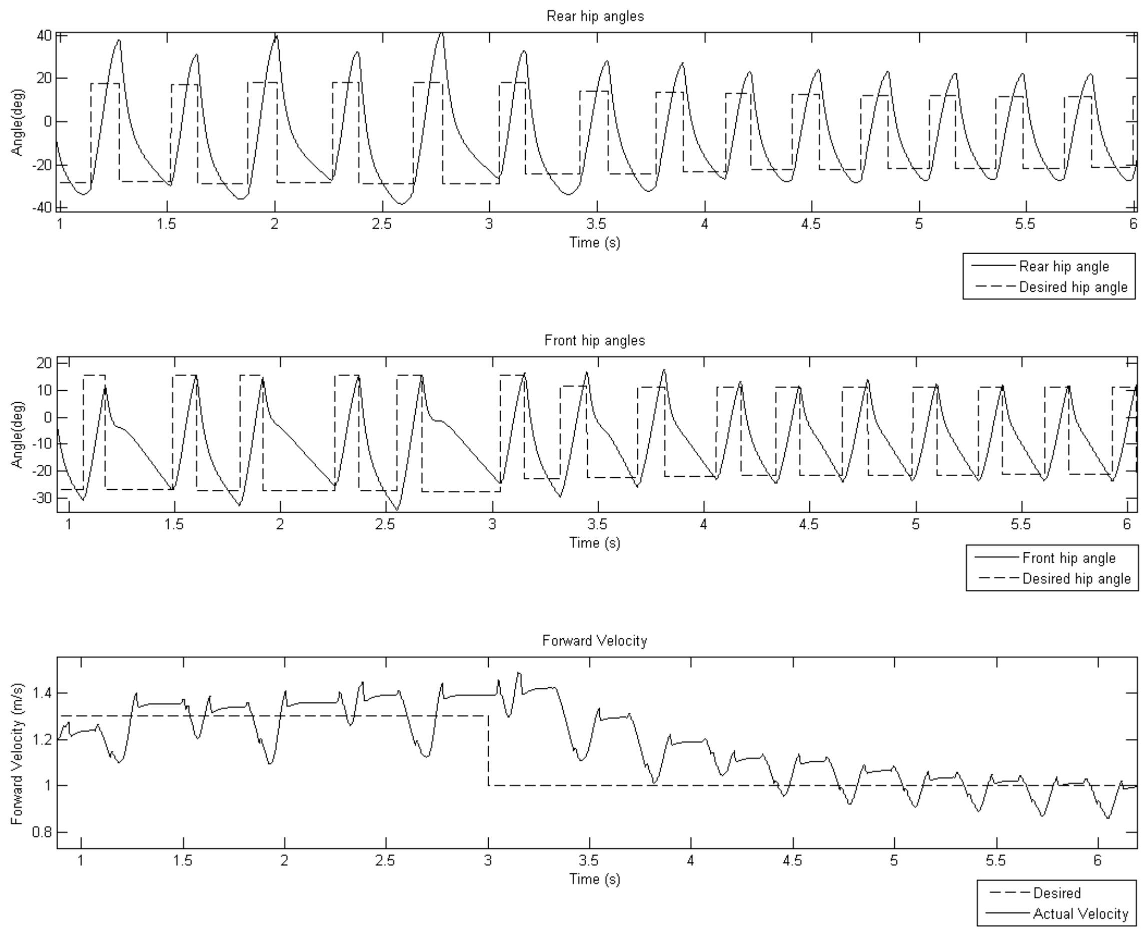


FIGURE 2.9 PAW DECELERATING FROM 1.3 TO 1.0M/S

Although there are upper and lower limits to the allowable bounding speed, as it will be discussed in section 2.3.4, commanding a desired velocity of 0 m/s successfully stops the bound sequence while maintaining stability. A more detailed plot of a stopping test is shown in Figure 2.10. From this plot, it is clear how the gradual reduction of hip angles occurs, allowing the robot to come to rest without failing. Unfortunately, due to the passive nature of the system, it is not possible to reinitiate bounding without a predefined start up sequence.

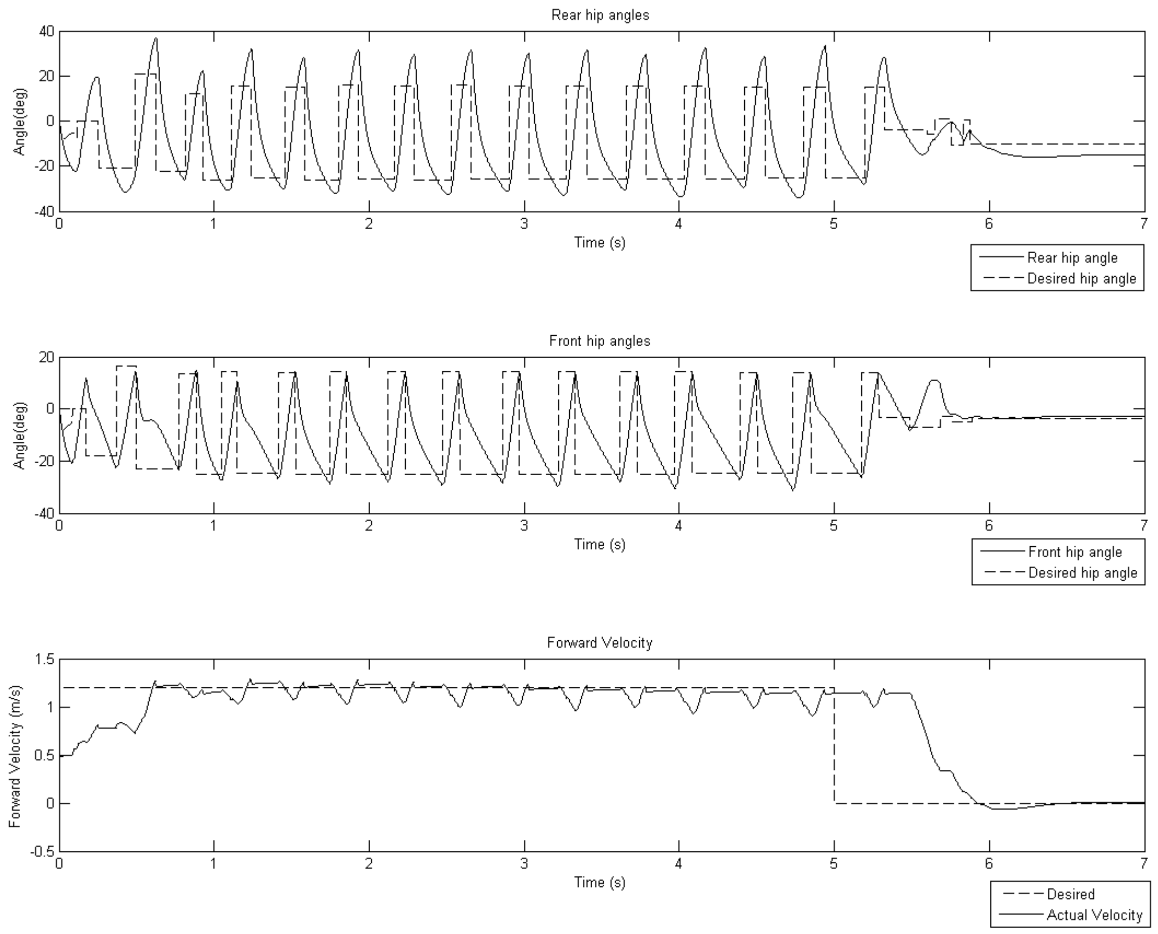


FIGURE 2.10 PAW STOPPING SEQUENCE

2.3.4 ROBUSTNESS TO DISTURBANCES

As the intelligent controller uses velocity feedback in the control loop, the controller corrects for changes in the robot's forward speed. So if the error in velocity is caused by a disturbance to the system instead of a change in the set point, it may be possible for the robot to compensate for this change. Therefore, it is worthwhile to verify the controller's robustness to disturbances. These changes to the robot's performance are primarily caused by changes in the robot-ground interaction. In the context of the MSC Adams simulation, we investigate the effects of changes in robot-ground interaction by varying the critical parameters of the contact model for leg-ground contact. Table 2.2 summarizes the robot-ground property changes considered, and the following subsections describe a performance analysis of the controllers with varying terrain properties.

Test	Stiffness	Force Exponent	Damping	Penetration Depth (m)	Static friction coefficient	Dynamic friction coefficient	Stiction tran. Vel.	Friction tran. Vel.
Original	10^7	2.2	20	10^{-4}	0.8	0.76	2.0	3.0
Friction change	10^7	2.2	20	10^{-4}	0.4	0.38	2.0	3.0
Stiffness change	10^6	2.2	20	10^{-4}	0.8	0.76	2.0	3.0

TABLE 2.2 ROBOT-TERRAIN INTERACTION PROPERTIES

2.3.3.1 TERRAIN FRICTION COEFFICIENT CHANGE

A terrain friction coefficient reduction by 50% reduces the robot's ability to track a desired velocity as seen in Figure 2.11.

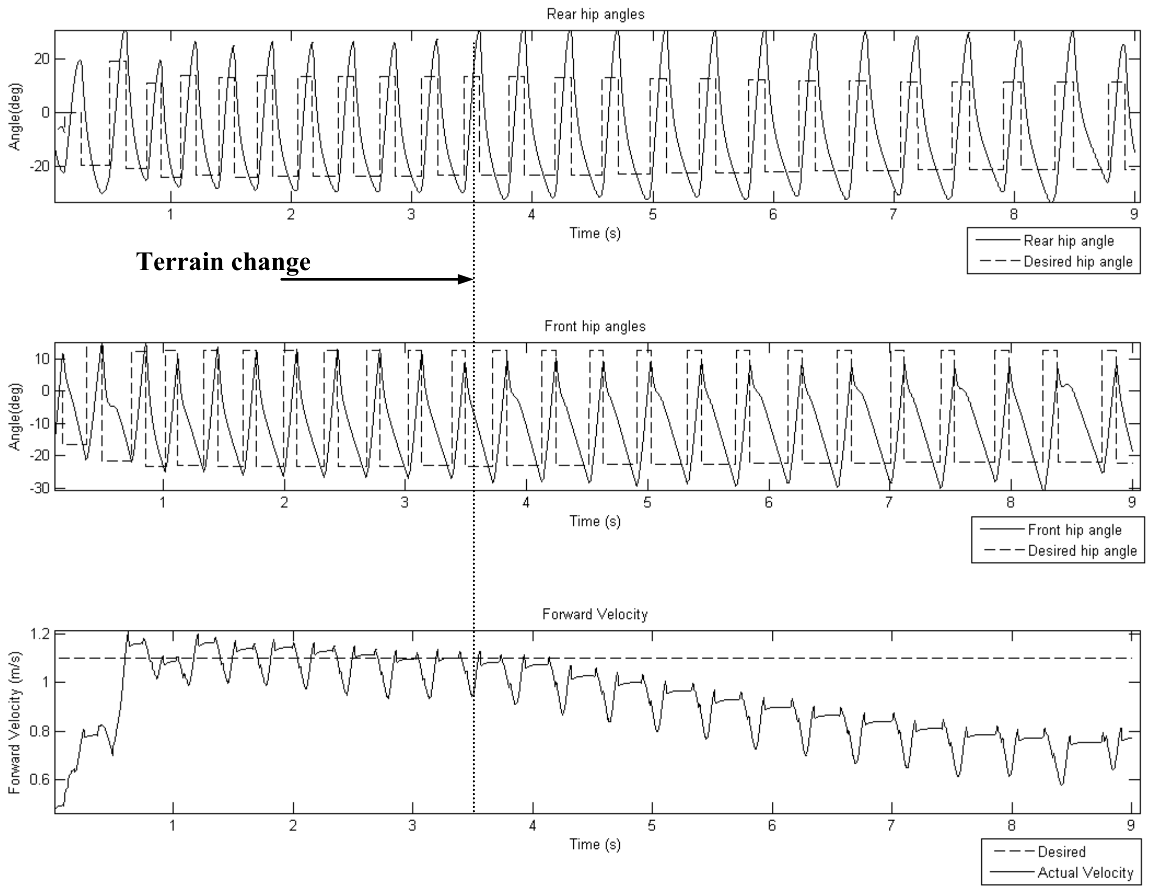


FIGURE 2.11 FRICTION TERRAIN CHANGE WITH INTELLIGENT CONTROLLER

The reduction of the friction coefficient causes some slip in the stance phase of the bound stride. Although the hip actuators are able to achieve the desired set points, the slip between the toe and the ground negates the SLIP assumption shown in Figure 1.2, causing some loss of energy in the stance phase. This has the negative effect of reducing the forward bounding velocity after the change in friction has occurred. In Figure 2.11, the actual robot forward bounding velocity has been reduced by over 20%, 5 seconds after the change in terrain. In Figure 2.12, a simulation was conducted to verify the response of the controller over a longer test area. The controller appears to compensate for the loss in velocity by slightly modifying the desired hip angles, however after over 30 s, the velocity error is still above 0.2 m/s. The LM learning algorithm modifies gains α_1 and α_2 in attempt to compensate for the increasing velocity error at the top of flight of each bound stride, but only small adjustments to these gains at each top of flight in the bound are possible. As described previously, gains α_1 and α_2 are found by reducing the error between the desired and actual top of flight forward velocities, whereas gain α_3 was determined experimentally to compensate for the passive nature of PAW's legs. The value of α_3 is decidedly a function of the leg-terrain interaction. Therefore, a change in terrain properties should be reflected by a change in the value of α_3. Ultimately, to increase the robot's ability to overcome varying terrain properties, a relationship between α_3 and terrain properties should be determined.

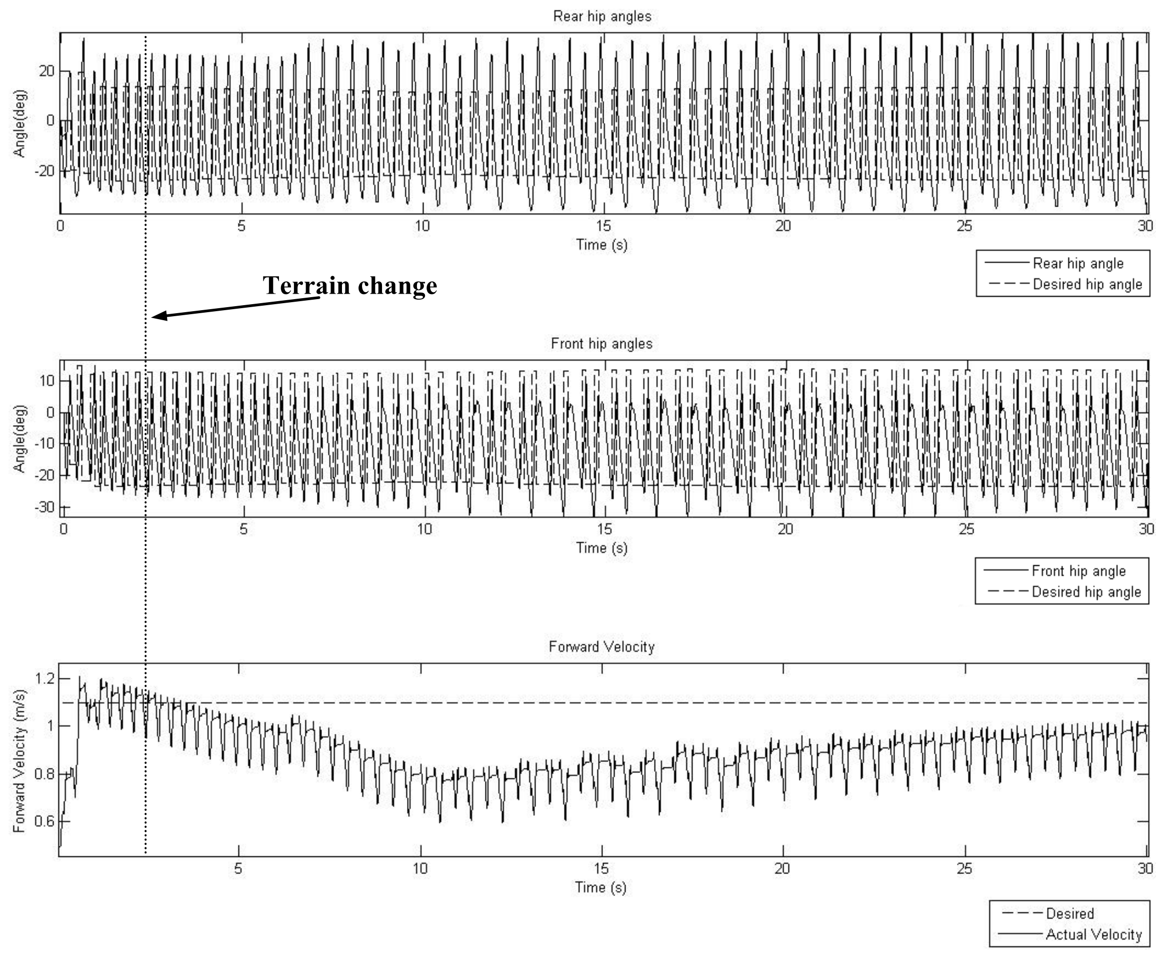


FIGURE 2.12 FRICTION TERRAIN CHANGE WITH INTELLIGENT CONTROLLER (LONG)

The terrain change simulation was tested for the fixed angles controller. This experiment is shown in Figure 2.13. The friction change causes a gradual loss of energy in the system at each stride. Without modifications to the touchdown and liftoff angles, the springs can no longer store the required energy to achieve liftoff after approximately 4.5 seconds, when failure occurs.

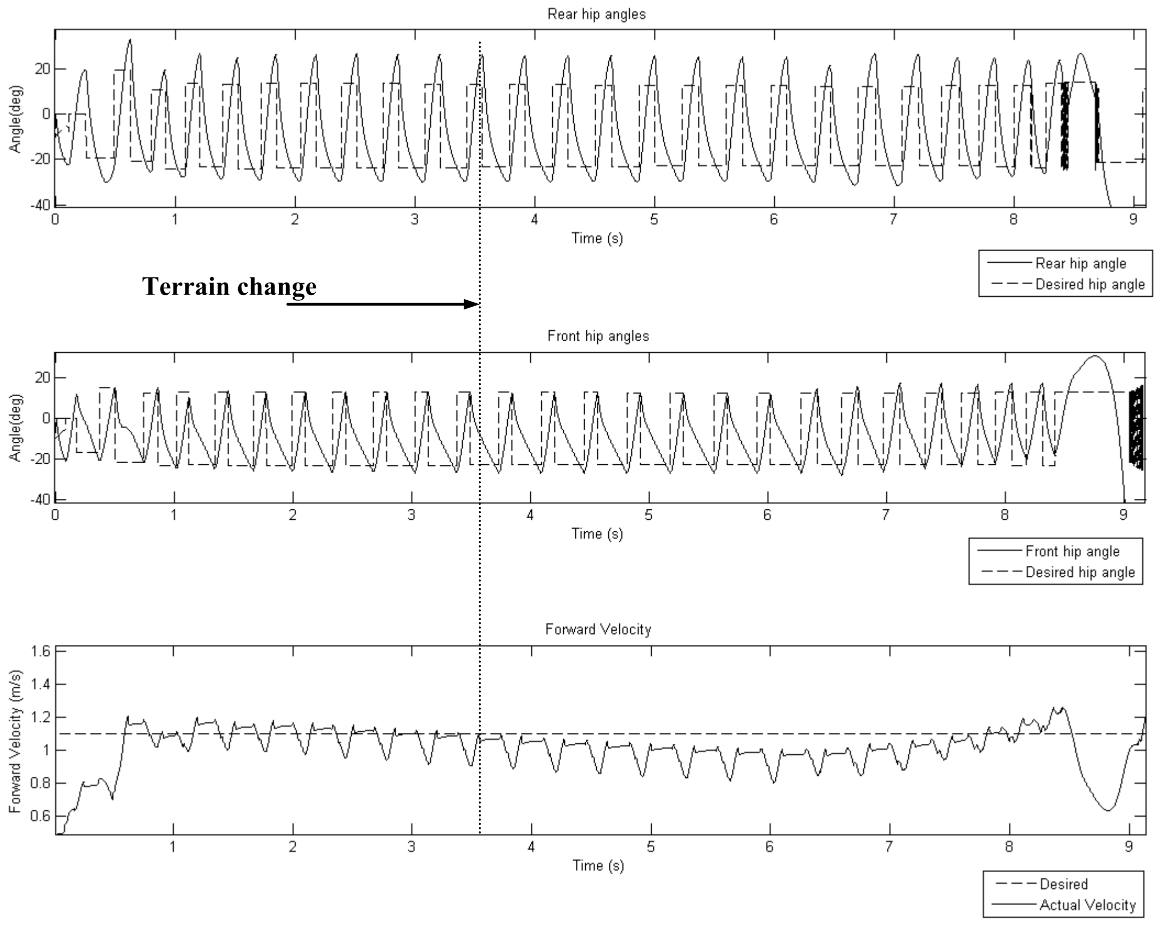


FIGURE 2.13 FRICTION TERRAIN CHANGE WITH FIXED-ANGLE CONTROLLER

2.3.3.2 TERRAIN STIFFNESS CHANGE

The intelligent controller was tested over a terrain with reduced stiffness (factor of 10 reduction) in Figure 2.14. Unlike the friction change simulations, the controller was unable to overcome the softer terrain. The lower stiffness caused a reduced height and length of the flight phase, leading to smaller bound lengths. This can be gleaned from Figure 2.14 after 2.5 s as the flight phases become shorter. The reduced flight phase causes a reduction in the ability of the hip actuators to achieve their desired angles, leading to failure by rear leg dragging at approximately 4 s.

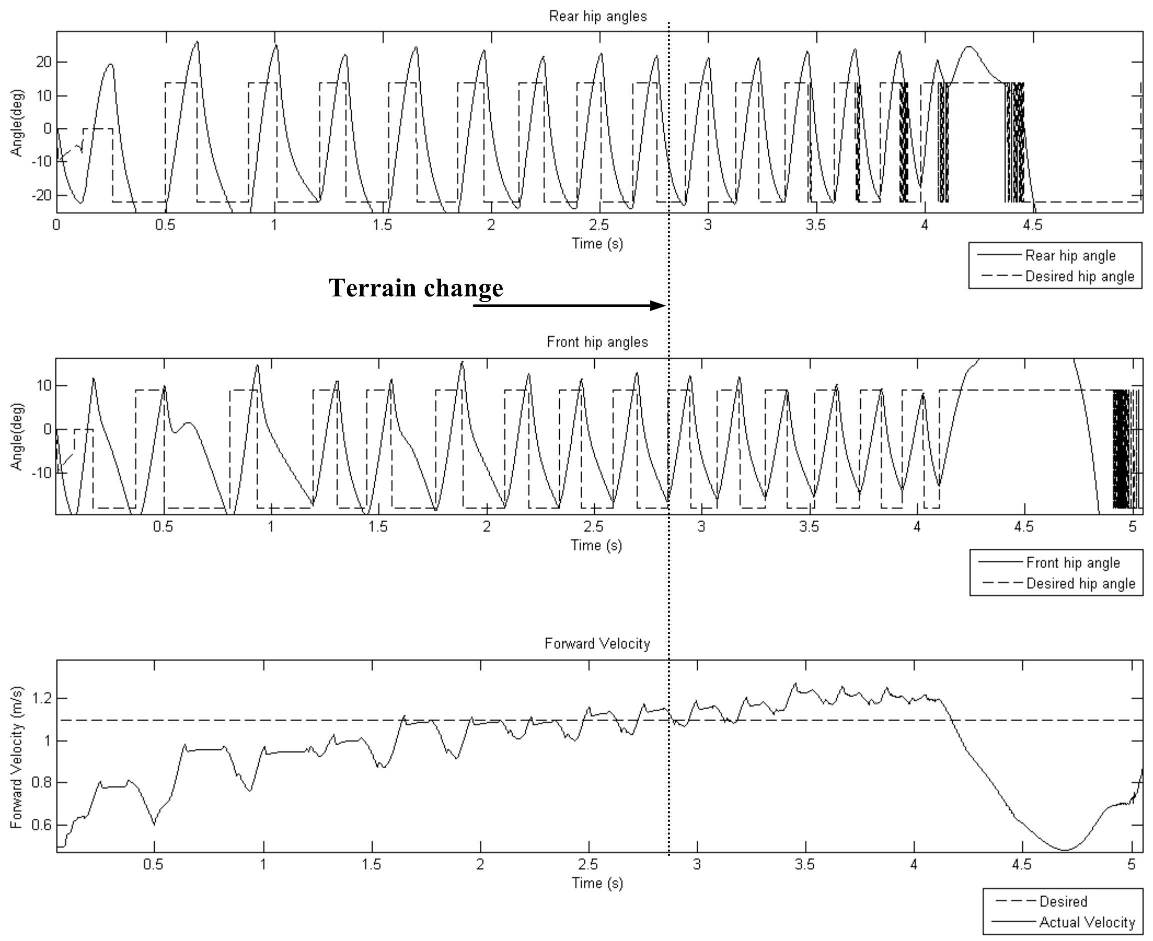


FIGURE 2.14 STIFFNESS TERRAIN CHANGE WITH INTELLIGENT CONTROLLER

The terrain stiffness is identified as a more critical parameter to the success of the bound gait than the ground friction. The stiffness directly affects the flight phase of the bound due to the passive nature of the robot's compliant legs. The failure occurs approximately 5 strides after the terrain change, and the intelligent controller is not able to overcome this stiffness change. The ground friction change, on the other hand, affects the SLIP model assumption. In this case, the intelligent controller is able to continue a stable bound gait, albeit with reduced velocity tracking capabilities. Nonetheless, this is an improvement to the previous fixed-angle controller that was unable to withstand the ground friction change.

Ultimately, the intelligent controller has limited success in withstanding significant terrain changes due to the nature of PAW's design. A reduction in ground stiffness directly affects the passive nature of the robot's compliant legs, quickly leading to failure. This failure is not specific to the proposed controller, and an increase in robustness to a change of ground stiffness would require alternative adaptation mechanisms, such as a change in the robot gait. For a change in friction, the intelligent controller proved more robust than the pre-existing non-intelligent controller.

2.3.5 Limitations

The results point to limitations at high and low speeds for stable bounding. The high and low limits were found to be 1.3m/s and 0.9m/s respectively. These limitations, however, are inherent to the robot's configuration and design parameters, and not the fault of the intelligent controller per se. If the desired velocity is set higher to 1.4 m/s the rear legs will start to drag instead of lifting off. This is a consequence of the larger liftoff angles required at higher speeds, which in turn cause lower bounding height for an already rather short-legged PAW. Figure 2.15 shows the result of accelerating from 1.0m/s to 1.4m/s. At 6 seconds, the forward pitch of the robot, prior to the front legs touching down, is diminished due to the high liftoff angles required to achieve the desired velocity. Failure occurs at 8 seconds where the rear legs do not liftoff, causing the robot to slow to a halt.

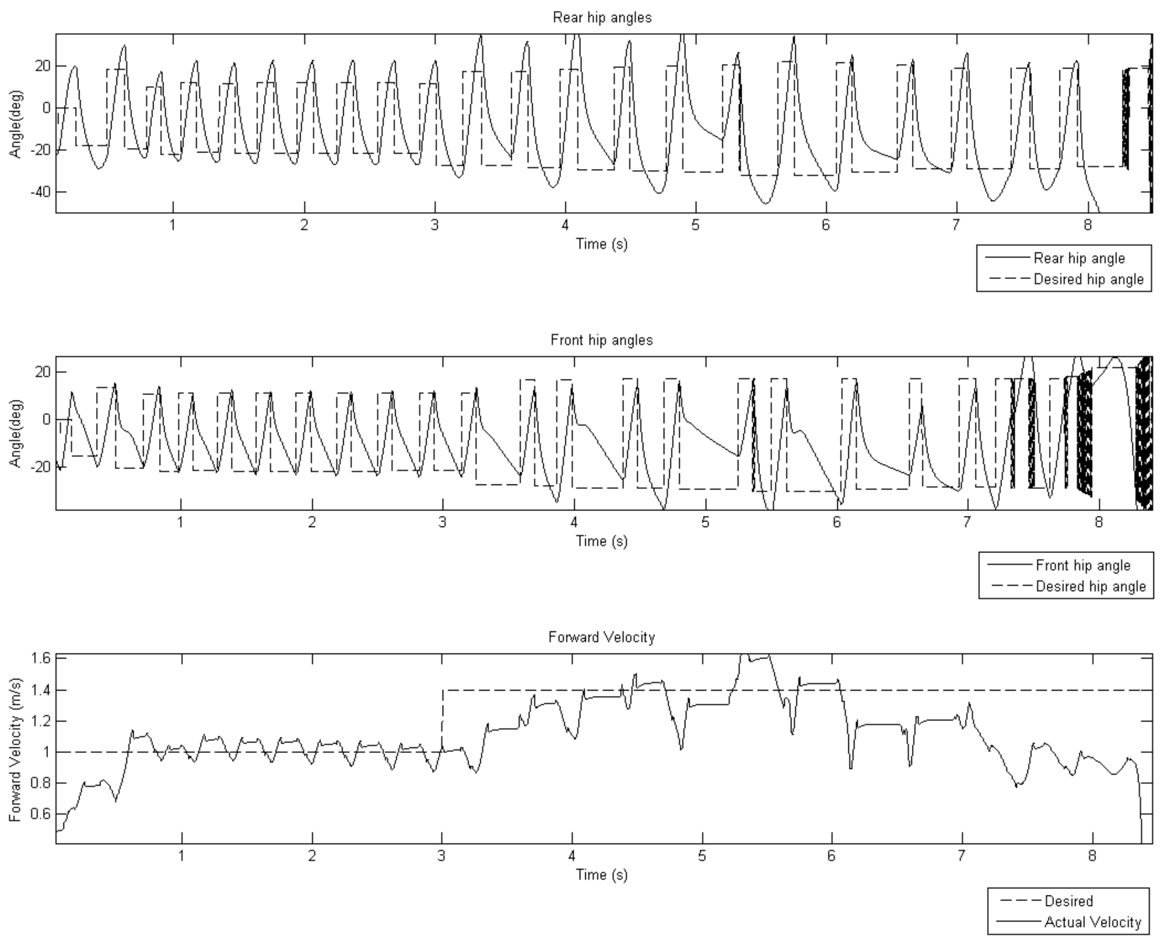


FIGURE 2.15 ACCELERATION FAILURE AT 1.4M/S

At desired velocities below 0.9 m/s, the front legs drag instead of lifting off. This occurs as PAW's rear legs take longer to touch down, and the front legs "wait" for them to land, thus dragging and creating instability. This "waiting" is due to the forward pitching motion caused by the low touchdown and liftoff angles. Figure 2.16 demonstrates the failure by decelerating from 1.3m/s to 0.8m/s. Front leg dragging can be seen by the front hip angles continuing into the positive direction instead of initiating flight when the liftoff angle is achieved. Figure 2.17 shows the robot's position at front leg touchdown, demonstrating the high forward pitch motion causing failure.

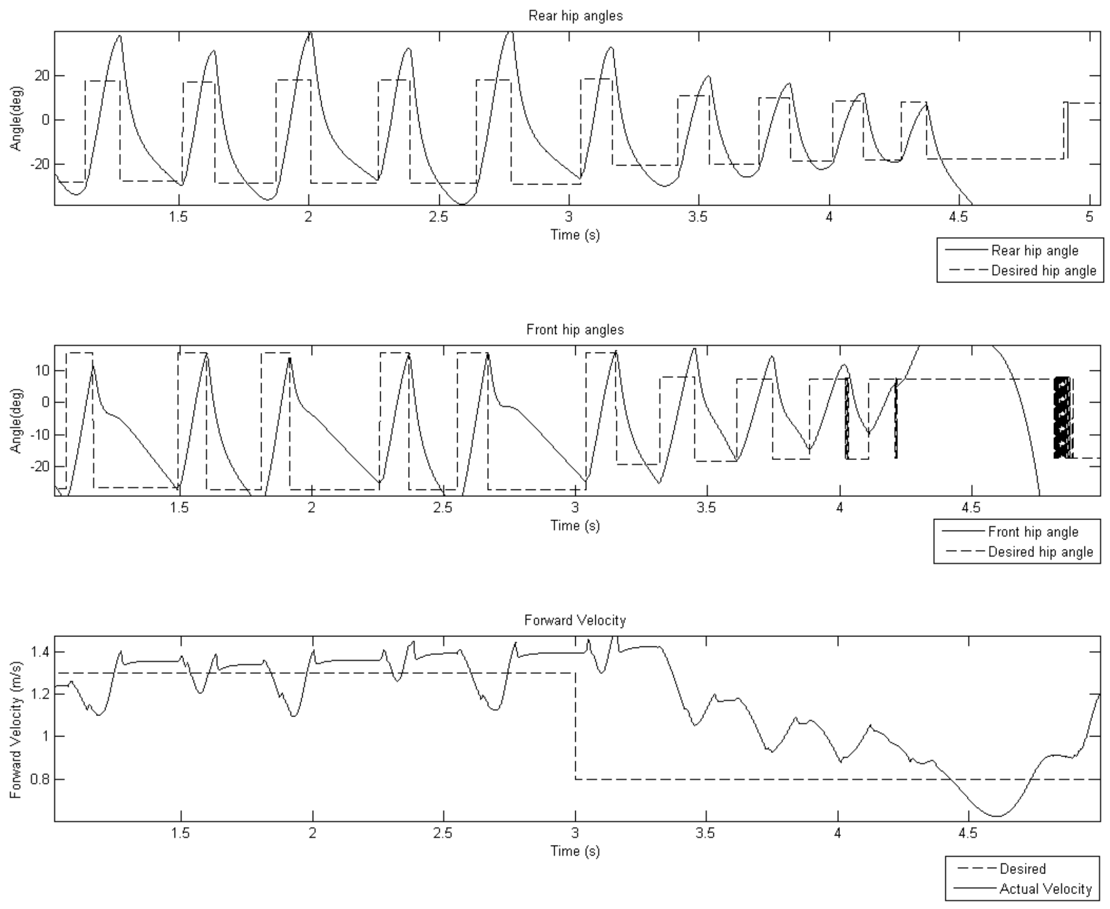


FIGURE 2.16 DECELERATION FAILURE AT 0.8M/S

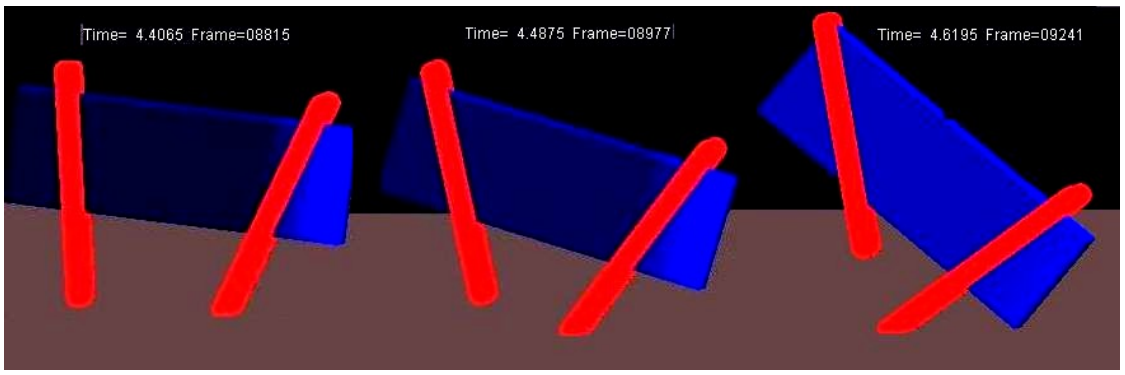


FIGURE 2.17 PAW SIMULATION IMAGE

PAW's natural dynamics limit the range of possible forward bounding velocity. By controlling PAW's leg lengths using prismatic actuators, it would be possible to inject the required energy into the system to achieve higher and lower bounding speeds. The intelligent velocity controller, however, is capable of velocity tracking within the operational range of the robot.

CHAPTER 3: VELOCITY ESTIMATION

One major advantage of the PAW platform over other robots is the simple controller and actuator design that require minimal sensing for dynamically stable locomotion. However, implementing intelligent control on the robot increases both the capabilities and requirements of the control system. In the existing non-intelligent controller described in Chapter 2, it is clear that robot touchdown detection, liftoff detection, hip and wheel position and velocity are required in the control loop. Additionally, the intelligent controller requires accurate measurement of the center of mass velocity. This chapter will describe the existing sensing capabilities of the robot, an extended Kalman filter (EKF) for estimating the forward velocity of the robot, and a validation of the proposed estimation method.

3.1 AVAILABLE SENSORS

The robot has 2000 count-per-revolution encoders for the hip and wheel motors, potentiometers with a precision of 1 mm to measure the compression in the leg springs, and a BAE SiIMU-01 inertial measurement unit (IMU). The IMU packages three gyroscopes and three accelerometers and can generate the body attitude and acceleration information. The specifications of the IMU are displayed in Table 3.1. Current and voltage sensors for the battery and a current sensor on each hip motor amplifier are also available but not used for robot control [27].

Parameter	Angular Value	Linear Value
Measurement Range	600-1000 deg/sec	50 g
Scale Factor	500 ppm 1 σ	2000 ppm 1 σ
Bias Instability	5 deg/hr 1 σ	-
Bias Repeatability	100 deg/hr 1 σ	10 mg 1σ
Random Walk	1.0 deg/ √hr	1.0 m/s/ √hr
Bandwidth	75 Hz	75 Hz
Update Rate	200 Hz	200 Hz

TABLE 3.1 BAE SIIMU-01 SPECIFICATIONS

In the existing configuration, only a poor estimate of forward velocity is possible from integrating the IMU's linear acceleration data – a process well-known to be prone to drift. A commonly attempted solution to this problem is to use an Extended Kalman Filter (EKF) to combine the IMU data with another available estimate to produce an optimal estimate of forward velocity [35]. In our case, the additional velocity estimate can be obtained from the measured leg angles and leg lengths, through the kinematics relations for the robot.

3.2 KINEMATICS ESTIMATE OF FORWARD VELOCITY

During various phases in the robot stride illustrated in Figure 2.2 of Chapter 2, the center of mass (COM) velocity can be estimated using the sensors available on the platform. To achieve accurate velocity estimation, velocity equations must be developed for each phase of the robot's gait.

For the single-leg stance phases (B and D of Figure 2.2), the COM position can be expressed relative to the toe's position on the ground. We designate the frame with the origin at the toe's position the toe fixed reference frame. For the planar robot shown in Figure (2.3), the toe fixed reference frame has an orientation identical to the inertial frame of reference. The general 3D case for the toe fixed frame can be seen in Figure (3.6). Equations (3.1) and (3.2) describe the COM position relative to the front and rear contact points respectively. In Equations (3.1) to (3.4), an average between the two front and two rear legs is used for the front and rear calculations respectively. Note that for actively controlled wheels on the physical robot, the fixed toe assumption used in simulation is no longer valid, and the wheel angular position and radius must be taken into account. [36]:

$$x_{COM\,f} = -L\cos\theta + l_f \sin(\gamma_f - \theta) + r_{wheel}\,\theta_{wheel\,f} \tag{3.1}$$

$$x_{COM\,r} = L\cos\theta + l_r \sin(\gamma_r - \theta) + r_{wheel}\,\theta_{wheel\,r} \tag{3.2}$$

The corresponding body velocity can be found, at both single-leg stance phases, by differentiating Equations (3.1) and (3.2) with respect to time, yielding Equations (3.3) and (3.4):

$$\dot{x}_{COM\,f} = L\dot{\theta}\sin\theta + \dot{l}_f \sin(\gamma_f - \theta) + l_f(\dot{\gamma}_f - \dot{\theta})\cos(\gamma_f - \theta) + r_{wheel}\omega_{wheel_f} \tag{3.3}$$

$$\dot{x}_{COM\,r} = -L\dot{\theta}\sin\theta + \dot{l}_r \sin(\gamma_r - \theta) + l_r(\dot{\gamma}_r - \dot{\theta})\cos(\gamma_r - \theta) + r_{wheel}\omega_{wheel_r} \tag{3.4}$$

Equations (3.3) and (3.4) express the COM velocity during front and rear leg stance respectively. During the double stance phase C, velocity is taken as the average of the two estimates from Equations (3.3) and (3.4), while during the flight phase A, forward velocity is assumed to remain constant from the liftoff of the rear leg at phase D. The assumption of constant flight velocity coincides with the simulation results presented in Chapter 2. Sections 3.2.1 and 3.2.2 will show the effectiveness of these equations during a Matlab/Adam co-simulation, and on the PAW platform.

3.2.1 Kinematics estimation Results and Validation

The kinematics velocity estimation Equations (3.3) and (3.4) were evaluated in both the PAW's Matlab-Adams co-simulation model, and on the physical platform.

In the simulation, the bound sequence is initiated by setting an initial pitch of 60 degrees, a zero initial forward velocity, and letting the robot drop from a predefined height. It should be noted that touchdown/liftoff detection in simulation is detected when a leg length compression threshold is met. This threshold is identical to that used on the PAW robot. The value is 0.02m less than the uncompressed leg length.

In the simulation shown in Figure 3.1, the robot is set to bound at 1.0 m/s using the intelligent controller described in Chapter 2. Figure 3.1 compares the estimated forward velocity from the kinematics equations (3.3) and (3.4) to the true forward velocity of the robot in simulation. Note that in simulation, the wheels at the ends of the legs are not modeled, thus the wheel portions of Equations (3.3) and (3.4) are neglected in the following tests.

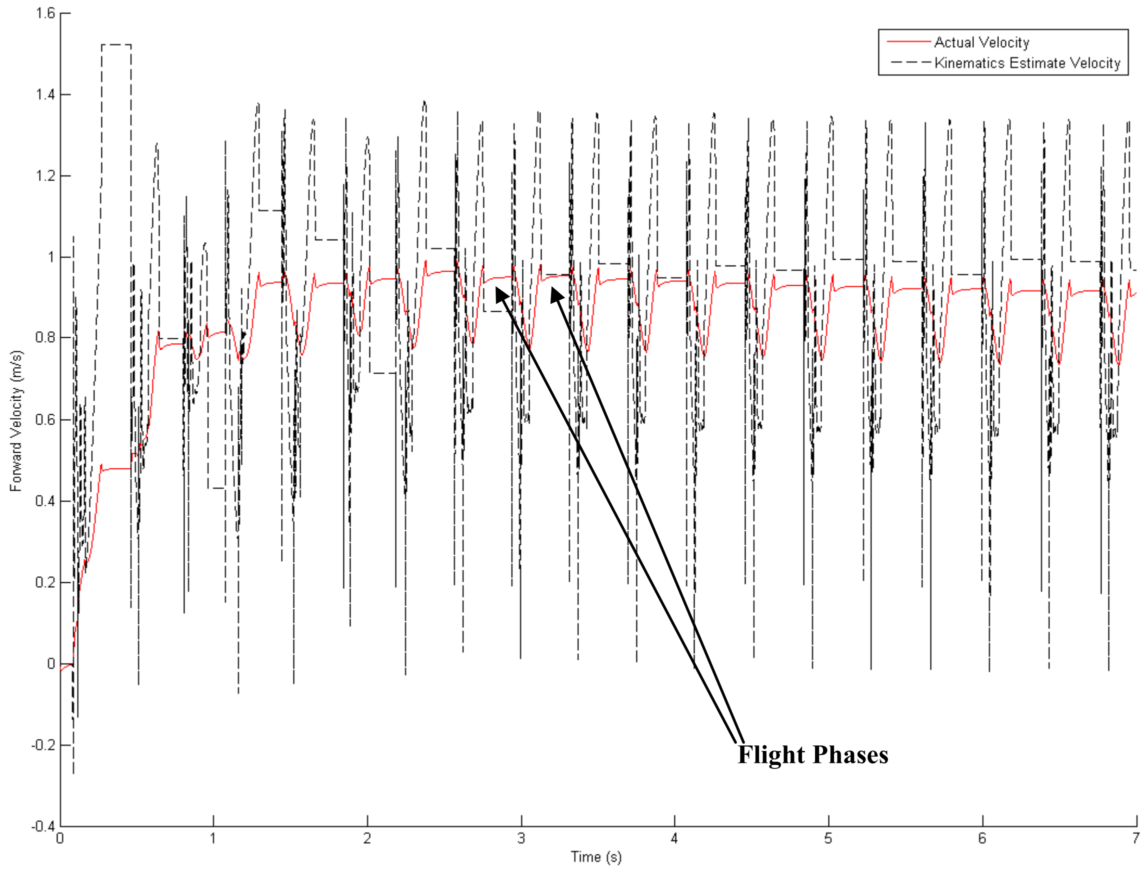


FIGURE 3.1 KINEMATICS ESTIMATE OF FORWARD VELOCITY ON SIMULATED DATA

The two velocity profiles are quite close to one another during the flight phase of the bound. The flight phase in the velocity plot is identified by the 'plateau' portion of the profile, which indicates a near constant velocity during flight. This is promising as the input to the learning controller is simply the top-of-flight bounding velocity. The major difference between the profiles is seen during the stance phase of the legs. Figure 3.2 provides a detailed comparison of the two profiles.

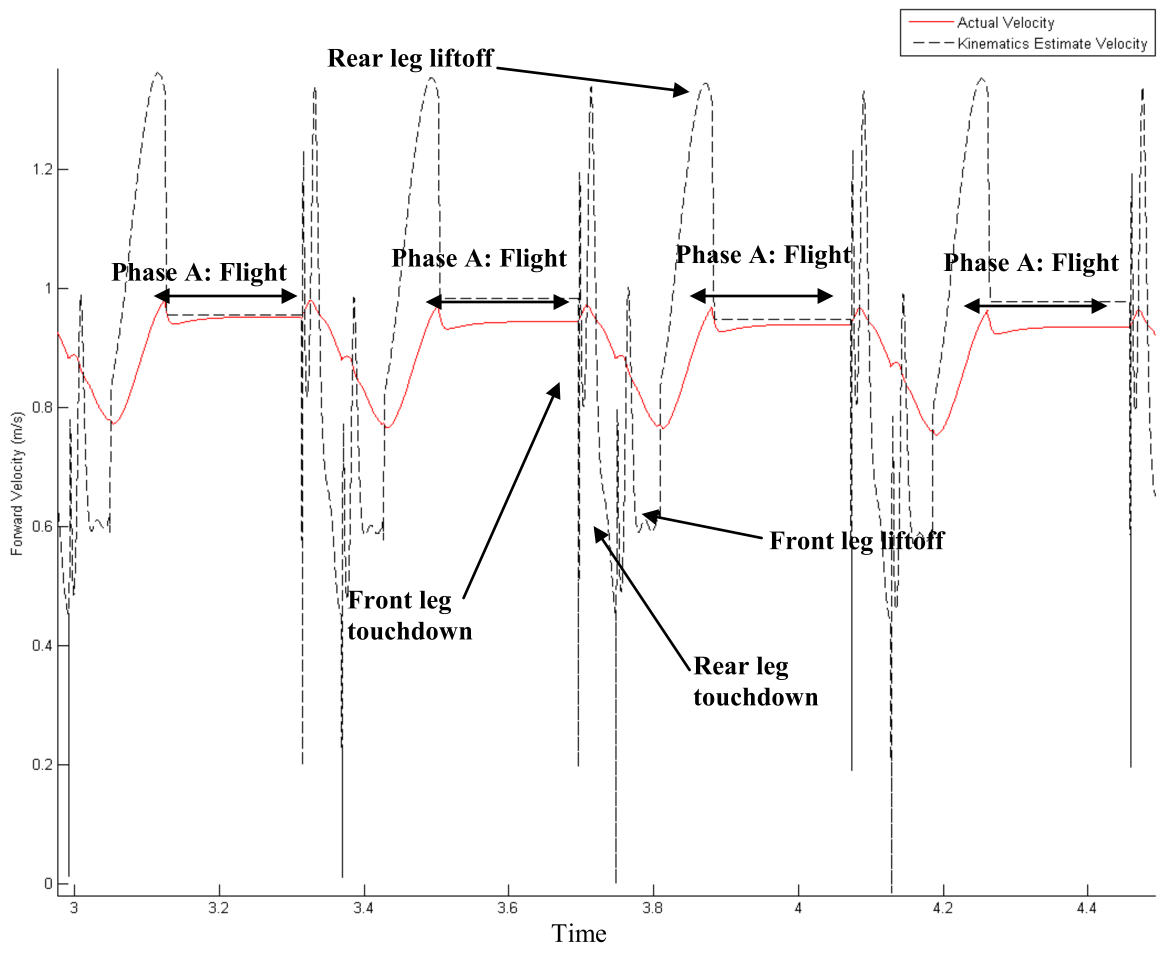


FIGURE 3.2 DETAIL OF KINEMATICS ESTIMATE OF VELOCITY ON SIMULATED DATA

The spikes in the profile occur at the touchdown of the front and rear leg pairs. At the end of the flight phase, the front legs touchdown causing a sudden change in leg lengths followed by a change in hip angular rates, causing the spikes seen in Figure 3.2. The second set of spikes occurs as the rear leg pair touchdown causes the robot to decelerate to its lowest speed in the bound stride. The rear touchdown is followed by an acceleration of the robot as the rear hips are actuated to their desired liftoff angle and the front leg pair takeoff. The change in hip angular rate and pitch, as well as the rapid rear leg extension causes Equation (3.4) to overshoot the actual forward velocity of the robot prior to settling to the flight velocity.

The kinematics estimate equations were also implemented on real PAW bounding data. That is, the velocity profiles seen in Figures 3.3 and 3.4 used data collected from the robot's sensors during a 9 second bound test conducted in the laboratory. On the real robot, the wheel portion of Equations (3.3) and (3.4) are being used.

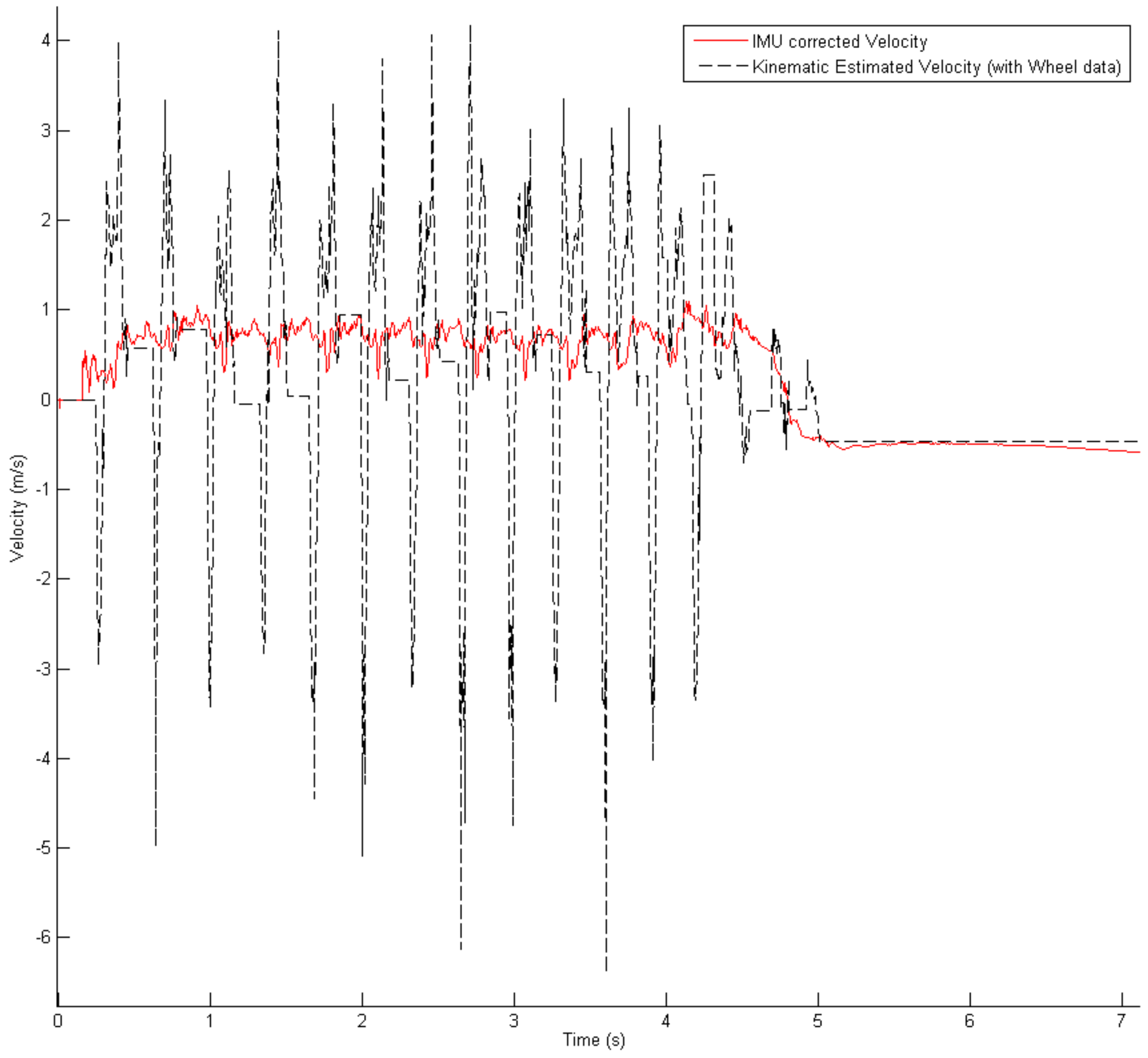


FIGURE 3.3 KINEMATICS ESTIMATE OF FORWARD VELOCITY ON ROBOT DATA

The kinematics estimate of velocity is compared to the velocity generated with the IMU measurements; the latter is corrected using an error model, which will be described in Section 3.3.1. Figure 3.4 provides a more detailed view of the bound segment of the test.

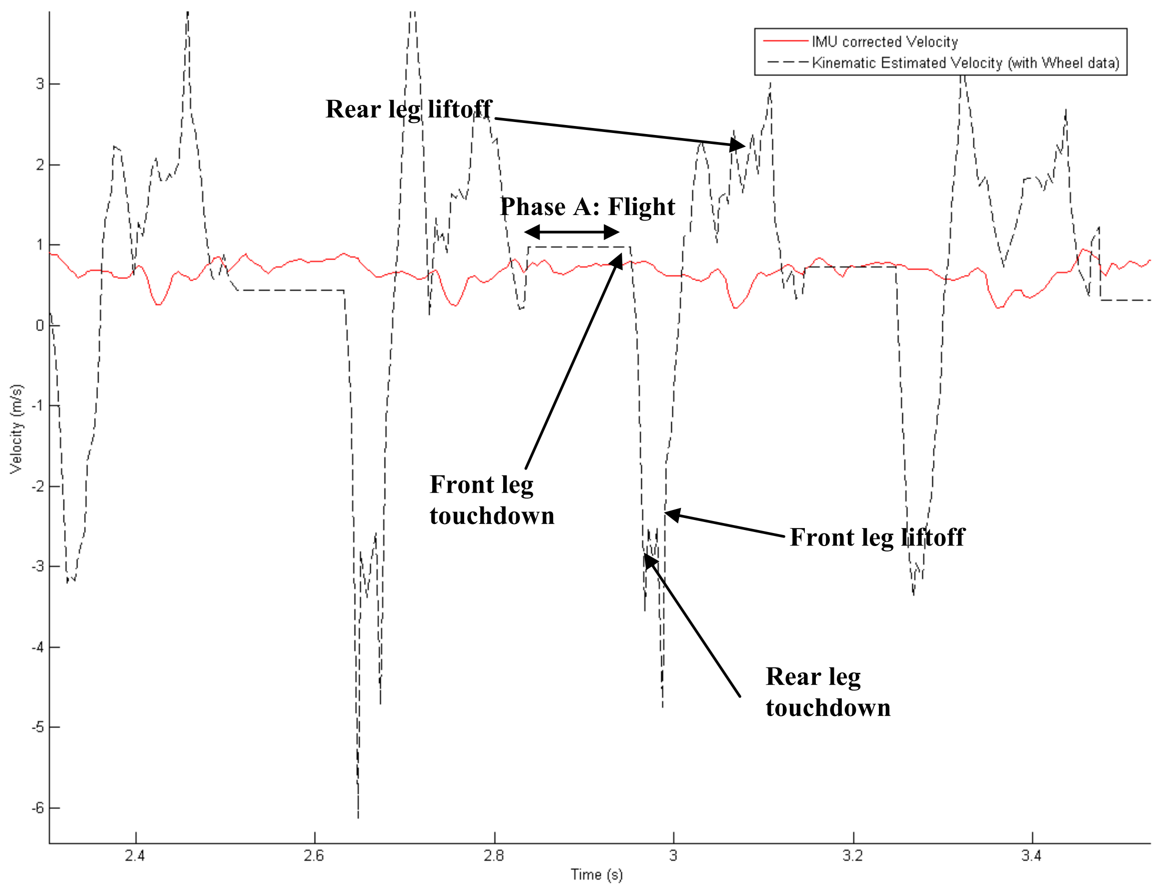


FIGURE 3.4 DETAIL OF KINEMATICS ESTIMATE OF ROBOT DATA

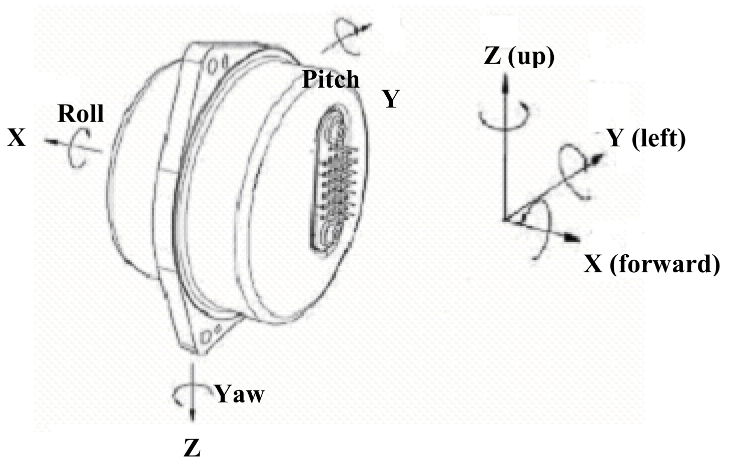
Similarly to the simulation results in Figure 3.2, the robot's kinematics velocity during stance is characterized by large spikes overshooting the actual robot velocity. It is clear that the kinematics velocity estimate performs rather poorly on the real robot data, with errors of up to 0.7 m/s during flight phases. It is believed that this occurs because the inputs to the kinematics estimator are generated by real sensors, which have noise and vibrations, which are multiplied in Equations (3.3) and (3.4). However, since the kinematics estimate is simply used to correct the IMU data primarily during the flight phase, we suggest these equations may be of use in the EKF, as during some flight sequences, the kinematics estimate is comparable to the IMU estimate.

3.3 IMU MEASUREMENTS

We now discuss in detail the generation of velocity estimates with the IMU. As noted earlier, a BAE SiIMU-01 is mounted on PAW, near the center of mass of the robot. In previous work, this IMU was used solely for measuring robot pitch data. Thus, some modifications were necessary to use the data in the control loop. The following subsections describe how the IMU is mounted and how the measurements are processed to generate the COM velocity. Additionally, an error model was developed for the IMU in order to improve the accelerometer and gyroscope data prior to use in the EKF implementation.

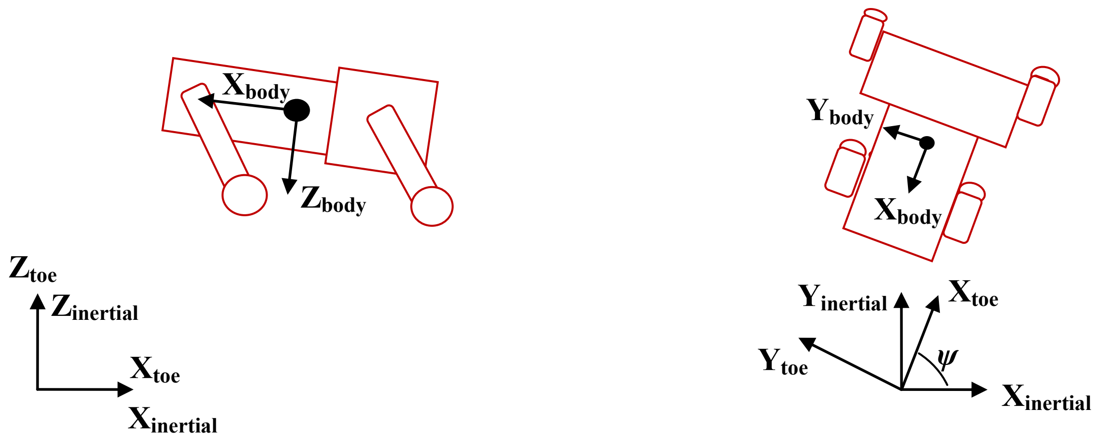
3.3.1 IMU DATA PROCESSING

Figure 3.4 shows the mounted orientation of the IMU on the robot and the inertial reference frame.



(A) IMU MOUNTED ORIENTATION (B) INERTIAL REFERENCE FRAME
FIGURE 3.5 REFERENCE FRAMES

The BAE SiIMU-01 measures the accelerations and angular rates in the body reference frame as shown in Figure 3.5(a) and it outputs incremental velocity and rotation angles in the same frame, at a rate of 200Hz. For the intelligent controller described in Chapter 2, it is necessary to obtain an estimate of center of mass forward velocity. In Equations (3.3) and (3.4), the kinematics estimate of COM is expressed in a toe fixed reference frame, which is identical to the inertial reference frame in the case of a planar robot as in Figure 2.3. However, in the presence of some yaw motion during bounding, the toe fixed frame differs from the inertial frame, as illustrated in Figure 3.6. As PAW does have some unwanted yaw motion during bound caused by an uneven mass distribution [29], the toe fixed frame forward velocity should be used as the controller input. Thus, the IMU data must be integrated and converted to the toe fixed frame of reference prior to use in the EKF.



(A) PAW BOUNDING SIDE VIEW (B) PAW BOUNDING TOP VIEW
FIGURE 3.6 PAW REFERENCE FRAMES

Quaternions are commonly used for converting acceleration and angular rate data in a body reference frame to velocity and position in an inertial frame [37]. The following presents the algorithm used on the robot to find the robot's attitude, velocity, and position in the inertial reference frame.

Prior to transforming the reference frame, the IMU is initialized by averaging the first 2000 time steps to find an offset value for each measurement. This offset value is subsequently subtracted from all measurements. As mentioned previously, the IMU outputs incremental body velocity and incremental body rotations at a rate of 200Hz so that the linear acceleration and angular velocity of the robot in the body reference frame are found via Equation (3.5), where Δt is a time step of approximately 0.005 seconds.

$$\mathbf{A}_{\text{body}} = \frac{\Delta \mathbf{V}_{\text{body}}}{\Delta t} = \begin{bmatrix} a_{x\,body} \\ a_{y\,body} \\ a_{z\,body} \end{bmatrix} \tag{3.5}$$

$$\mathbf{\Omega}_{\text{body}} = \frac{\Delta \boldsymbol{\theta}_{\text{body}}}{\Delta t} = \begin{bmatrix} \Omega_{x\,body} \\ \Omega_{y\,body} \\ \Omega_{z\,body} \end{bmatrix} \tag{3.6}$$

To initialize the algorithm, the quaternion is set to:

$$\mathbf{Q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.7}$$

In this convention [37], q_3 represents the scalar part of the quaternion. In order to find the linear acceleration and angular velocity with respect to the inertial frame, we use the incremental quaternion equation as follows.

$$\Delta \mathbf{Q} = 0.5 \begin{bmatrix} q_3 & -q_2 & q_1 \\ q_2 & q_3 & -q_0 \\ -q_1 & q_0 & q_3 \\ -q_0 & -q_1 & -q_2 \end{bmatrix} \begin{bmatrix} \Omega_{x\,body} \\ \Omega_{y\,body} \\ \Omega_{z\,body} \end{bmatrix} \tag{3.8}$$

Then, the quaternion at the i$^{\text{th}}$ time step can be found with:

$$\mathbf{Q_i} = \mathbf{Q_{i-1}} + \Delta \mathbf{Q_i} \Delta t \tag{3.9}$$

The quaternion is normalized at each time step to ensure it has a norm of one.

$$\mathbf{Q_i} = \frac{\mathbf{Q_i}}{\sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}} \tag{3.10}$$

From the normalized quaternion, it is possible to convert to Euler angles to find roll, pitch, and yaw relative to the inertial reference frame via Equations (3.11) through (3.13) respectively.

$$\phi = \tan^{-1}\left(\frac{2q_3q_0 + q_1q_2}{q_3^2 - q_0^2 - q_1^2 + q_2^2}\right) \tag{3.11}$$

$$\theta = \sin^{-1}\left(-2\left(q_0q_2 - q_3q_1\right)\right) \tag{3.12}$$

$$\psi = \tan^{-1}\left(\frac{2q_0q_1 + q_3q_2}{q_3^2 + q_0^2 - q_1^2 - q_2^2}\right) \tag{3.13}$$

Knowing the robot attitude in the inertial reference frame, it is possible to find robot velocity and acceleration in the inertial frame using the rotation matrix in Equation (3.14).

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.14}$$

First, the gravity vector must be expressed in the robot body frame. According to:

$$\mathbf{g_{body}} = \mathbf{R}\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{3.15}$$

The effects of gravity on the body frame velocity increments are accounted for in Equation (3.16).

$$\Delta\mathbf{V}_{\text{body}} = \Delta\mathbf{V}_{\text{measured}} + \mathbf{g_{body}}\,\Delta t \tag{3.16}$$

The body frame velocity increments are transformed to inertial frame velocity increments using the rotation matrix

$$\Delta\dot{\mathbf{X}}_{\mathbf{inertial}} = \mathbf{R}^{\mathrm{T}}\Delta\mathbf{V}_{\mathbf{body}} \tag{3.17}$$

and subsequently are added to find updated inertial robot velocity. These can be integrated to find position.

$$\dot{\mathbf{X}}_{\mathbf{i},\,\mathbf{inertial}} = \dot{\mathbf{X}}_{\mathbf{i-1},\,\mathbf{inertial}} + \Delta\dot{\mathbf{X}}_{\mathbf{i},\,\mathbf{inertial}} \tag{3.18}$$

$$\mathbf{X}_{\mathbf{i},\mathbf{inertial}} = \mathbf{X}_{\mathbf{i-1},\mathbf{inertial}} + \Delta\dot{\mathbf{X}}_{\mathbf{i},\mathbf{inertial}}\,\Delta t \tag{3.19}$$

Finally, the inertial frame velocity is transformed to the toe fixed velocity using the yaw rotation matrix, $\mathbf{R}_\psi$. The x-component of $\dot{\mathbf{X}}_{\mathbf{i,toe}}$ is the forward COM velocity of the robot at time **i**.

$$\dot{\mathbf{X}}_{\mathbf{i,toe}} = \mathbf{R}_\psi \dot{\mathbf{X}}_{\mathbf{i,inertial}} \tag{3.20}$$

Note that the *x* and *z* axis velocities and accelerations must change signs to account for the nominal orientation of the IMU on the robot relative to the direction of motion, as shown in Figure 3.4.

3.3.2 IMU Error Model

Subsection 3.3.1 describes the method used for obtaining attitude, velocity, and position of the robot with respect to an inertial reference frame. However, there are noise and bias errors present in the measured data from the IMU, causing significant errors and apparent drift when adding the incremental measurements and integrating them at a rate of 200Hz in Equations (3.18) and (3.19). Stationary and linear motion tests were conducted on the IMU mounted on PAW in the laboratory. The purpose was to develop an error model to correct the IMU measurements prior to handling in the EKF.

First, a 60 s stationary test was conducted to observe the drift in the measurements of the IMU. Figure 3.7 shows the inertial frame roll, pitch and yaw measurements. Figure 3.8 shows the inertial frame *x*, *y*, and *z* velocities.

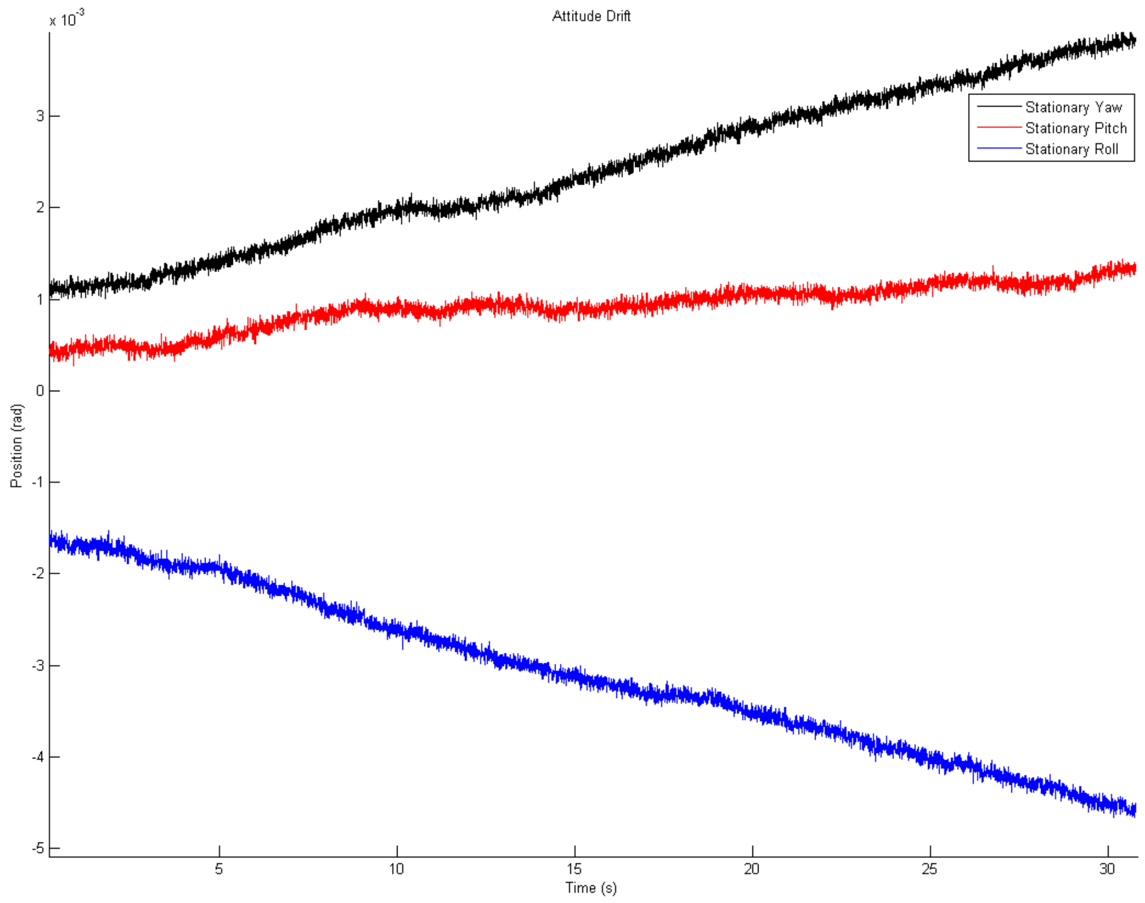


FIGURE 3.7 IMU ATTITUDE DRIFT TEST

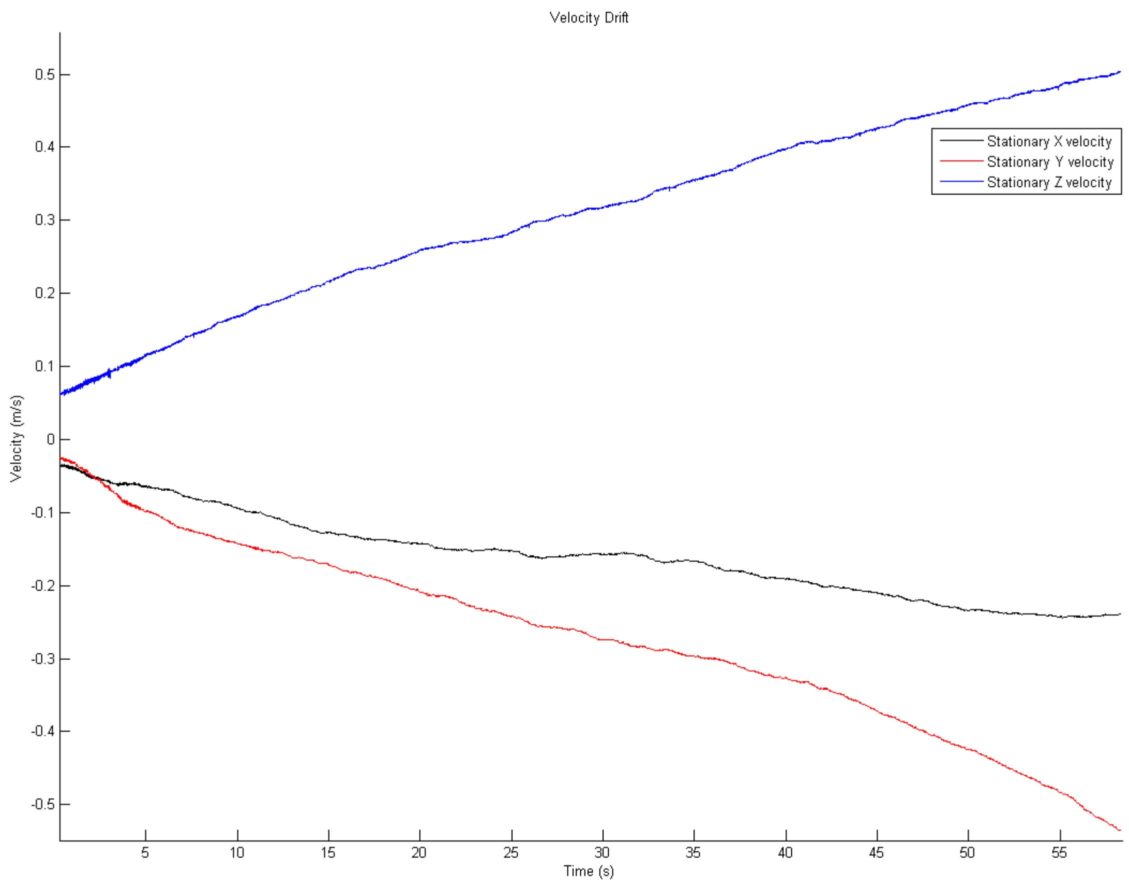


FIGURE 3.8 IMU VELOCITY DRIFT TEST

Inaccuracies and bias errors in the IMU cause incremental errors at each time step of measurements. The substantial drift in the velocity, as seen in Figure 3.8, is created by the numerical integration, which adds the incremental errors at each time step, creating a large drift of as much as 0.6m/s over 60 seconds. The attitude drift is much smaller as the quaternion method reduces the integration errors. From these data sets, a linear error model was developed to account for this drift. Equations (3.21) to (3.26) describe the models, where T represents the total elapsed time and Δt the measurement increment.

$$\theta_{corrected} = \theta_{uncorrected} - 1\times10^{-5}T \tag{3.21}$$

$$\phi_{corrected} = \phi_{uncorrected} + 7.33\times10^{-5}T \tag{3.22}$$

$$\psi_{corrected} = \psi_{uncorrected} - 5\times10^{-5}T \tag{3.23}$$

$$\Delta \dot{x}_{corrected} = \Delta \dot{x}_{uncorrected} + 0.018\,\Delta t \tag{3.24}$$

$$\Delta \dot{y}_{corrected} = \Delta \dot{y}_{uncorrected} + 0.01\,\Delta t \tag{3.25}$$

$$\Delta \dot{z}_{corrected} = \Delta \dot{z}_{uncorrected} - 0.01\,\Delta t \tag{3.26}$$

A forward rolling test, where the robot was commanded to a fixed velocity using its wheels, was performed to test the effectiveness of the error model in the *x*-direction. Figure 3.9 demonstrates the performance of the linear error model compared to the velocity data provided by the wheel encoders. This test consists of the robot starting from rest and calibrating, then standing up and accelerating forward to a velocity of 0.48m/s, followed by a stop and sit down. The error model shows a significant improvement on the raw IMU data.

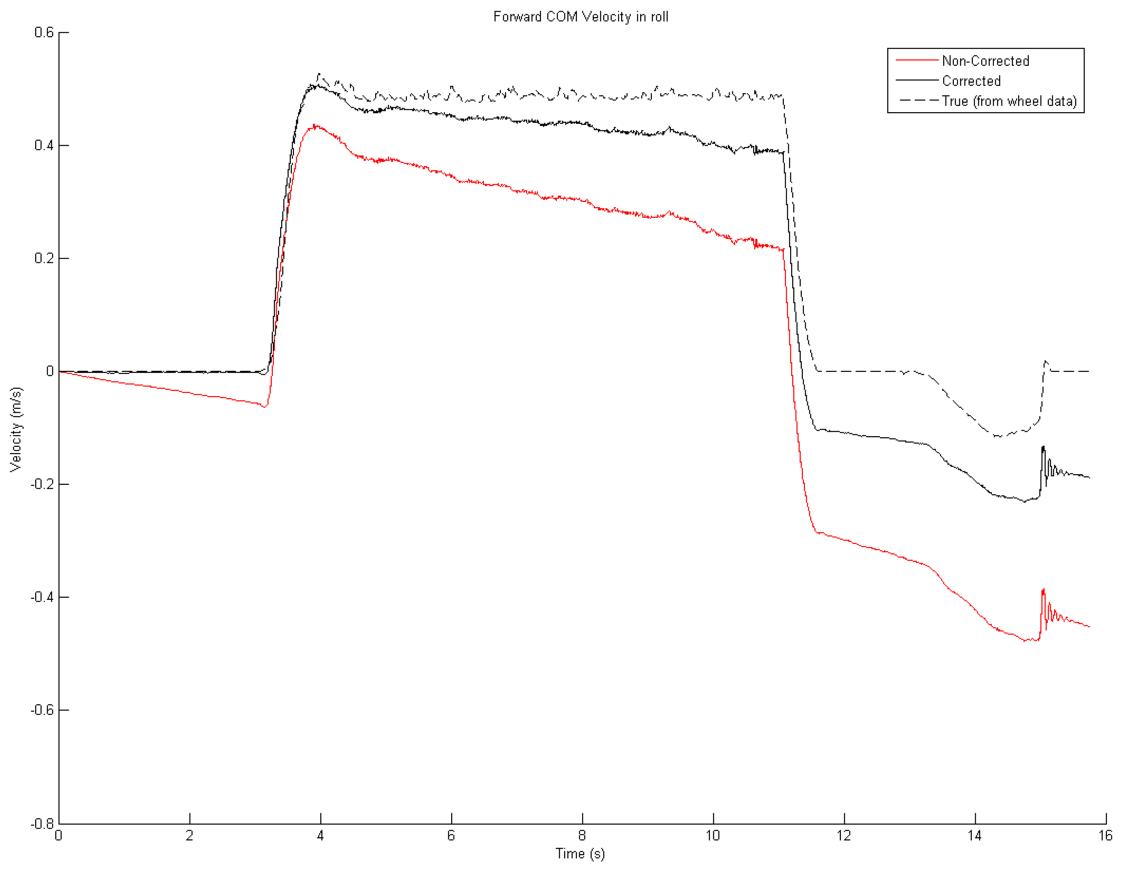


FIGURE 3.9 IMU ERROR MODEL TEST: FORWARD VELOCITY IN ROLLING

Lastly, a test was performed to verify the effectiveness of the error models to estimate the forward velocity in the bound gait as shown in Figure 3.10.

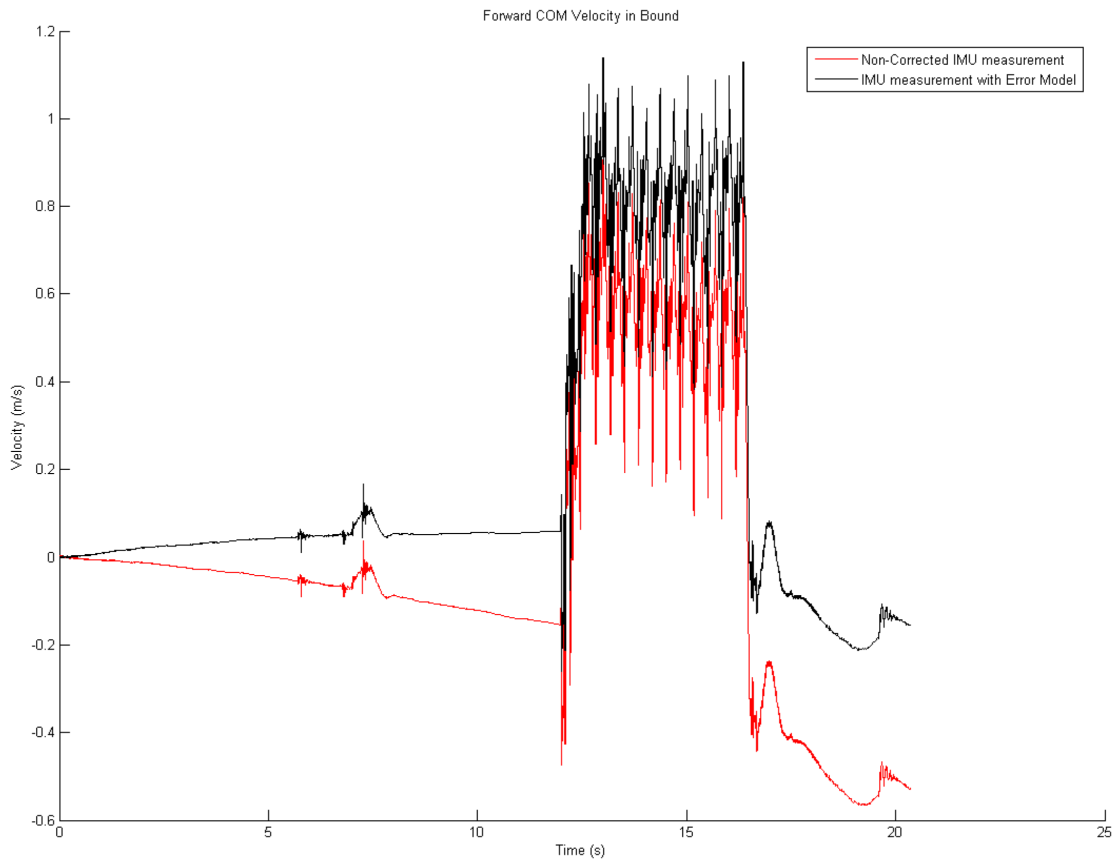


FIGURE 3.10 IMU ERROR MODEL TEST: FORWARD VELOCITY IN BOUND

It can be observed that the error model is not perfect. This is partially due to the inconsistency of the errors present on the IMU and the non-linear nature of the drift. Extensive testing of the same IMU unit was performed at Defense Research and Development Canada (DRDC) and those results indicate some inconsistencies, as seen in Table 3.2 [38]. DRDC performed a 15.5 hour stationary test on the IMU and compiled the accelerometer drift errors at every 100 seconds into 4 parts, each part consisting of almost 4 hours each. The inconsistencies of the drift error in the DRDC tests are clear from table 3.2. The last row of the table is included to compare the 60 second stationary test, shown in Figure 3.7, performed in our laboratory. The drift values found at McGill are smaller than those found during the DRDC tests. This is believed to be due to the much shorter time of McGill's stationary tests.

Test	X (mili-g/hr)	Y (mili-g/hr)	Z (mili-g/hr)
DRDC Test Part 1	-0.037	-0.231	0.140
DRDC Test Part 2	-0.530	0.713	0.094
DRDC Test Part 3	0.016	-0.300	-0.178
DRDC Test Part 4	-0.142	0.043	-0.307
McGill stationary test	-0.0733	-0.0122	0.022

TABLE 3.2 ACCELEROMETER DRIFT ANALYSIS RESULTS

Additional inconsistencies are introduced in the IMU measurements as a result of its mounting on the PAW robot and the nature of the robot's operation. PAW experiences significant ground impacts during bounding, causing vibration of its mechanical structure. Additionally, the IMU mounting plate was an afterthought in the design of the robot, which ideally should be further reinforced to reduce IMU vibrations and misalignments. However, the nature of the measurements will inevitably be noisy and inaccurate, and for the purposes of this research, the results shown for the *x*-direction bound in Figure 3.10 are deemed sufficiently accurate for use in the EKF.

3.4 EKF ALGORITHM

The Kalman Filter attempts to predict the current state of the system using the previous estimated states, combined with the noisy measurement data. The Extended Kalman Filter (EKF) enables the use of non-linear state equations to estimate the desired states [39]. In this work, the kinematics estimate of velocity is used for propagation of the robot's velocity, while the IMU measured velocity is used for the measurement update of robot velocity. Presented here is a modification of the EKF to account for the various phases of the bound gait as discussed in Chapter 2.

First the state vector of the system is defined as

$$\mathbf{x} = \begin{bmatrix} \gamma_f & \gamma_r & \theta & l_f & l_r & \dot{\gamma}_f & \dot{\gamma}_r & \dot{\theta} & \dot{l}_f & \dot{l}_r & v \end{bmatrix}^{\mathrm{T}} \tag{3.27}$$

These states are illustrated in Figure 2.3 and the dotted elements represent their time derivative. The last element in the state vector, *v,* denotes the COM forward velocity for the robot. The discrete update at time *k* of the state is defined as

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) = \mathbf{A}(\mathbf{x}_{k-1})\ \mathbf{x}_{k-1} + \mathbf{w}_{k-1} \tag{3.28}$$

where $\mathbf{w}_{k\text{-}1}$ is process noise assumed to be zero mean Gaussian white noise with covariance $\mathbf{Q}_{\mathrm{k}}$ estimated based on the accuracy of the state measurements, and

$$\mathbf{A}(\mathbf{x}_{k-1}) = \begin{bmatrix} \mathbf{\Phi}_{10\times 11} \\ \boldsymbol{\varphi}_{1\times 11} \end{bmatrix} \tag{3.29}$$

The $\mathbf{\Phi}_{10\mathrm{x}11}$ matrix is given by

$$\mathbf{\Phi}_{10x11} = \begin{bmatrix} \mathbf{I}_{5\times5} & \mathbf{\Phi}^*_{5\times5} & \mathbf{0}_{5\times1} \\ \mathbf{0}_{5\times1} & \mathbf{I}_{5\times5} & \mathbf{0}_{5\times1} \end{bmatrix} \ , \quad \mathbf{\Phi}^*_{5\times5} = \Delta T\, \mathbf{I}_{5\times5} \tag{3.30}$$

where $\mathbf{I}_{ixj}$ and $\mathbf{0}_{ixj}$ are the *i*-by-*j* identity and null matrices respectively and $\boldsymbol{\varphi}_{1\mathrm{x}11}$ is defined as:

$$\boldsymbol{\varphi}_{1\times11} = \begin{bmatrix} 0 & 0 & 0 & \phi_4 & \phi_5 & 0 & 0 & \phi_8 & \phi_9 & \phi_{10} & \phi_{11} \end{bmatrix} \tag{3.31}$$

where $\phi_4 = S_1(\dot{\gamma}_f - \dot{\theta})\cos(\gamma_f - \theta)$ (3.32)

$$\phi_5 = S_2(\dot{\gamma}_r - \dot{\theta})\cos(\gamma_r - \theta) \tag{3.33}$$

$$\phi_8 = S_1(L\sin\theta) - S_2(L\sin\theta) \tag{3.34}$$

$$\phi_9 = S_1 \sin(\gamma_f - \theta) \tag{3.35}$$

$$\phi_{10} = S_2 \sin(\gamma_r - \theta) \tag{3.36}$$

$$\phi_{11} = S_3 \tag{3.37}$$

The *S*-values in Equations (3.32) to (3.37) are defined based on the particular phase in the bound cycle as shown in Table 3.3.

Phases	S_1	S_2	S_3
A: Flight	0	0	1
B: Front stance	1	0	0
C: Double Stance	0.5	0.5	0
D: Rear Stance	0	1	0

TABLE 3.3 S-VALUES FOR EKF

Therefore, the velocity element in state vector update $\mathbf{x}_k$ is found using estimate for velocity kinematically.

Then, the measurement equation can be defined as:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k) = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3.38}$$

where $\mathbf{v}_k$ is the observation noise assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_k$ determined from the variance of measurements and

$$\mathbf{H}_k = \mathbf{I}_{11\times 11} \tag{3.39}$$

The measurement equation for $\mathbf{z}_k$ is updated directly from the sensor values and uses the IMU measurements for the velocity state update.

Covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ represent the accuracy of the state estimates and measurements during each phase of the robot's bound sequence. An analysis of each sensor was performed to determine the uncertainty in the measurement. For the elements contained in state vector $\mathbf{x}_k$ and the measurement vector $\mathbf{z}_k$, the leg lengths and the velocity estimates and measurements displayed some level of inaccuracy. The other sensors perform adequately and a very small value was used in the matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$.

The standard deviations for each state element contained in Table 3.4, were found by comparing measured values to some known value. In this case, the standard deviation is simply the square root of the covariance, which is used in $\mathbf{Q}_k$ and $\mathbf{R}_{k.}$

State Estimate	Standard Deviation
Leg length front (m)	6.123×10^{-5}
Leg length rear (m)	1.84×10^{-4}
Kinematics Velocity Phase A: Flight (m/s)	0.45
Kinematics Velocity Phase B: Front Stance (m/s)	0.35
Kinematics Velocity Phase C: Double Stance (m/s)	0.47
Kinematics Velocity Phase D: Rear Stance (m/s)	0.8
IMU Velocity (m/s)	1.67×10^{-4}

TABLE 3.4 STANDARD DEVIATION OF STATE ESTIMATES

The standard deviation of the velocity estimate for each phase of the robot's bound was found by comparing the kinematics estimate to the true velocity value in simulation, as seen in Figure 3.2. For the IMU velocity covariance, ideally, this value would be found by comparing the IMU measured velocity to the true velocity at instants through the bound. However, the most accurate measure of velocity is only available from the IMU as there is no 'true' velocity value available. Therefore, we rely on the error model described in Equations (3.21) to (3.26), in section 3.3.2, to improve the accuracy of the IMU measurement and use the covariance value shown in table 3.4 tuned via trial and error by comparing the IMU measurement to simulated values, in the EKF.

The best estimate of the robot's state at time $k, \hat{\mathbf{x}}_k$, can now be found using the EKF algorithm presented in Equations (3.40) to (3.47). In these equations, the subscripts $k|k$-1 and $k|k$ denote the a priori and a posteriori values respectively for time k, where k-1$|k$-1 denotes the a posteriori value for the previous time step, k-1.

1. Predicted state:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}(\hat{\mathbf{x}}_{k-1|k-1})\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{w}_{k-1} \tag{3.40}$$

2. Predicted estimate of covariance:

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^{\ T} + \mathbf{Q}_{k-1} \tag{3.41}$$

where,

$$\mathbf{F}_k = \left. \frac{\partial \mathbf{A}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{w}_{k-1}} \tag{3.42}$$

3. Measurement residual:

$$\widetilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}, 0) \tag{3.43}$$

4. Residual covariance:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{3.44}$$

5. Optimal Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{3.45}$$

6. Updated state estimate:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \widetilde{\mathbf{y}}_k \tag{3.46}$$

7. Updated estimate covariance:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{3.47}$$

The best estimate of forward velocity of the robot at time *k* is the last element in the state update $\hat{\mathbf{x}}_{k|k}$. The EKF algorithm was implemented on the PAW robot and the results are presented in section 3.5.

3.5 EKF Results and Validation

The EKF algorithm was tested on the robot commanded to bound over a distance of 4m in a laboratory environment, with fixed touchdown and liftoff angles. Figure 3.11 compares the IMU measured velocity, combined with the error model developed in 3.3.2, and the EKF estimated velocity.

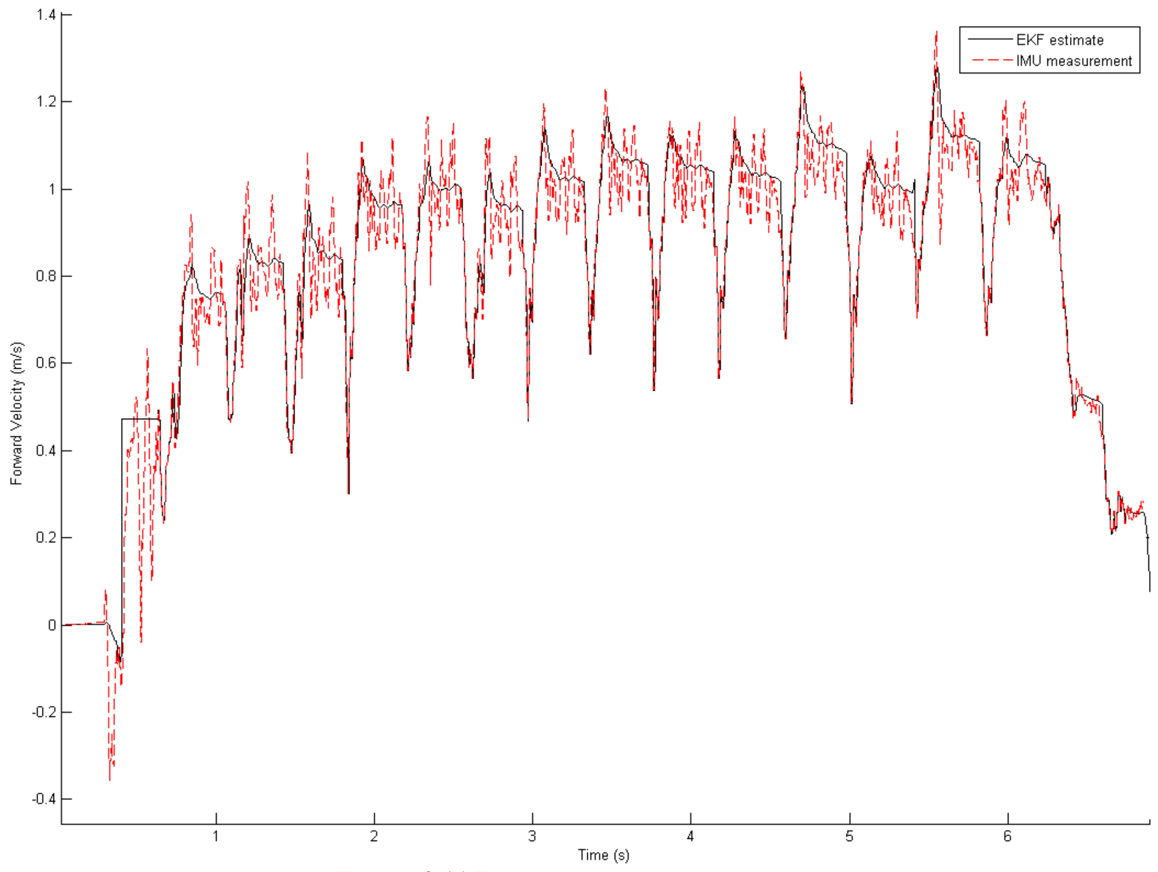


FIGURE 3.11 FORWARD VELOCITY OF BOUND

At a first glance, the EKF and the IMU estimates appear to be quite close. However, the point of interest is primarily the top-of-flight velocity, which is used as an input to the intelligent controller described in Chapter 2. Figure 3.12 displays a zoom-in of the bound sequence, and the flight velocity estimate.

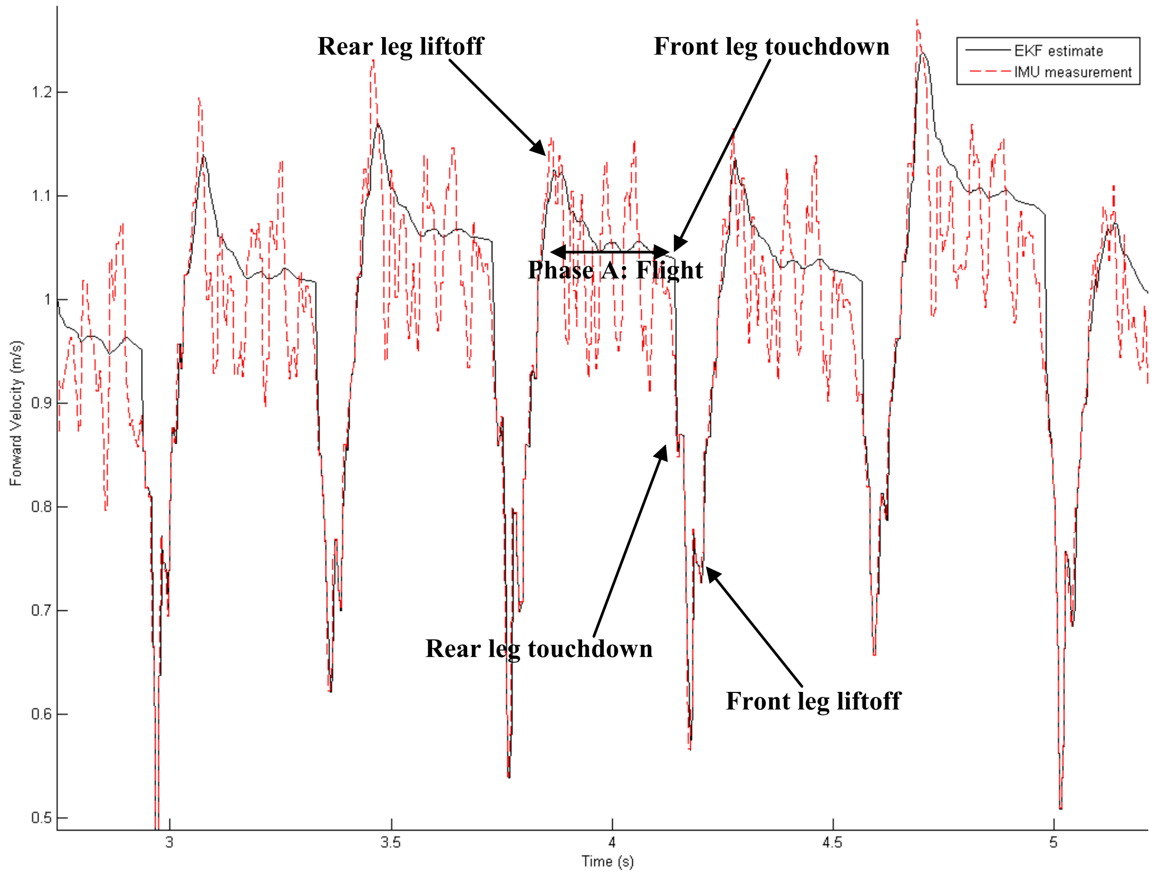


FIGURE 3.12 DETAIL OF FORWARD VELOCITY OF BOUND

The profile of the velocity estimate is characterized by large spikes during leg impacts and smoother profiles during flight. It is clear that the EKF reduces the magnitude and the number of spikes in the velocity estimate, most notably during the flight phase. Without the EKF, the flight phase velocity can vary by up to 0.3m/s, which would be unacceptable for intelligent velocity control between 0.9m/s – 1.3m/s. The EKF's flight phase velocity has a range of 0.1m/s.

A true TOF velocity of the robot is difficult to determine, but an estimate of the average velocity during the bounding sequences of Figures 3.11 and 3.12 was obtained by using a high speed camera, the Casio EX-F1, which enables 300fps video imaging. According to the video analysis, the average robot speed throughout the bound cycle is 0.98m/s ± 0.15m/s. This analysis was performed by setting markers 0.5m apart on the ground, and estimating the frame number at which the robot crosses the markers. The stated mean velocity was found by averaging the velocities at each marker, whereas the range is simply the maximum and minimum velocities found. As the test was performed for a very limited range, and the camera was stationary throughout the test, some inaccuracies will result in the visual estimate of the exact frame at which the robot crossed the markers. However, the range of possible frame numbers at which the robot crosses the marker result in discrepancies smaller than the range calculated above.

This method does not yield an estimate of the top-of-flight bound velocity but a mean velocity throughout the robot's bound. Nonetheless, the EKF estimated velocity falls in mid-range of the estimate provided by the high speed camera test. Figure 3.13 shows several still frame shots of the robot bounding across the markers during this test.

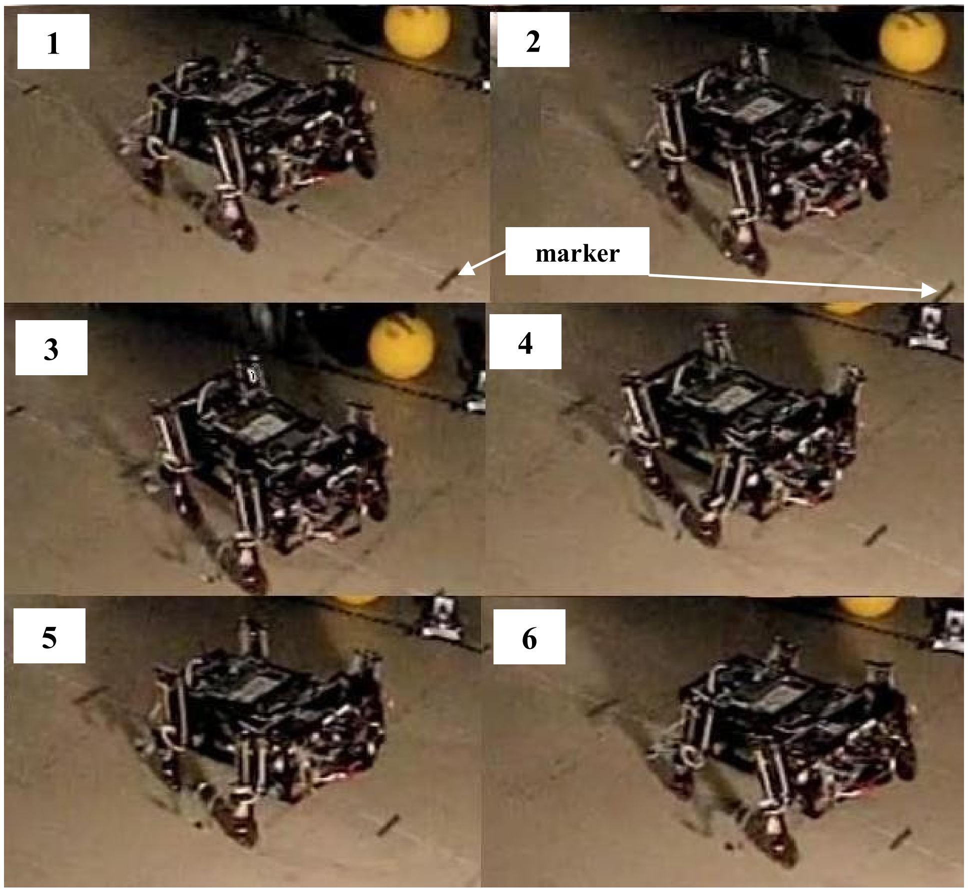


FIGURE 3.13 STILL FRAMES OF BOUNDING TEST IN LABORATORY

A final validation of the EKF velocity estimate was performed by comparing the velocity profile with the bounding velocity of the robot in simulation, shown in Figure 3.14.

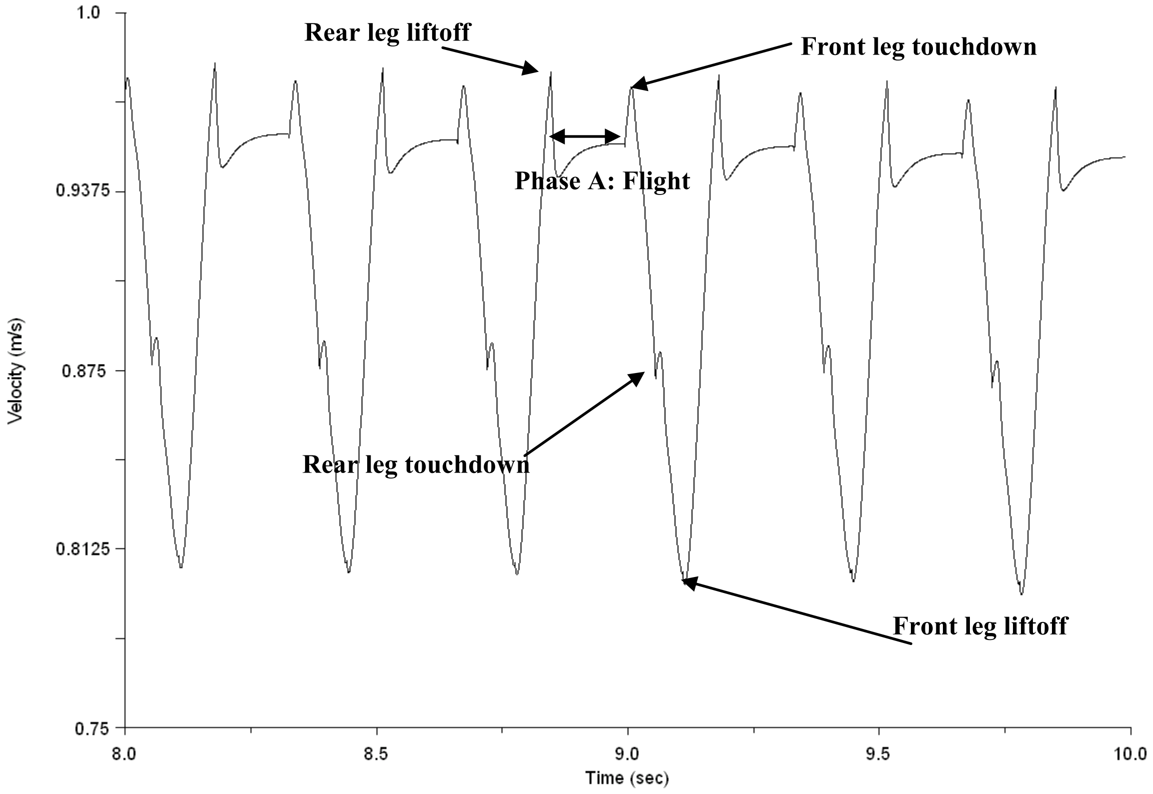


FIGURE 3.14 FORWARD VELOCITY SIMULATION OF PAW BOUNDING

The profile of the curves in Figure 3.12 and 3.14 are very similar to one another. In Figure 3.14, the spike that occurs just before the flight phase corresponds to the rear leg take off and is of the same order of magnitude for the EKF estimate as for the true simulated velocity. At the front leg touchdown instant, there is a brief acceleration where the velocity increases by approximately 0.5 m/s. This acceleration is not seen in the EKF results in Figure 3.12. The double stance phases in the EKF results; the portion of the curve contained between the rear leg touchdown and front leg liftoff, shows a larger deceleration, approximately 0.4 m/s below the flight velocity, whereas in simulation, the value is about 0.1m/s below the flight velocity. This is due to the kinematic estimate of the velocity and the covariance values used in the EKF at double stance phase. The flight phase velocity estimate of the EKF profile resembles that of the simulated results, yielding velocity variations of below 0.1 m/s during the phase.

There are inevitably more sensor errors, noise and vibration on the physical platform; as seen by the small oscillations in the velocity profile in Figure 3.12 that are not visible in Figure 3.14. Nonetheless the variability of under 0.1m/s during flight demonstrates sufficient accuracy to test the intelligent velocity control described in Chapter 2. Chapter 4 shall demonstrate the effectiveness of the intelligent controller on the physical platform, while using the EKF described here for estimating COM TOF velocity in the control loop.

CHAPTER 4: CONTROLLER IMPLEMENTATION AND RESULTS

The first section in this chapter describes the steps taken towards the implementation of the intelligent controller and the Extended Kalman filter on the PAW robot. The subsequent sections describe in detail the results of the controller testing on PAW. A detailed analysis is contained to provide the reader with a comparison to the simulation results shown in Chapter 2. Lastly, a discussion is presented of the various parameters that were identified as critical to the effectiveness of the intelligent velocity controller, including the velocity limits.

4.1 IMPLEMENTATION

Chapter 2 described the MSC Adams model combined with MATLAB used to evaluate the performance of the intelligent velocity controller. Initial steps towards physical implementation were taken in Chapter 3 where the EKF, used for estimating top-of-flight forward bounding velocity as an input to the controller, was developed. Some additional modifications were required prior to testing the intelligent controller on the robot. These modifications are described in the following subsections.

4.1.1 MECHANICALLY BLOCKED WHEELS

The simulation results of Chapter 2 were obtained without actuated wheels modeled at the distal ends of the robot's legs. However, as discussed previously, the PAW robot uses actuated wheels to combine the advantages of legged and wheeled locomotion. For the bound gait, the wheels can be actuated to prevent their rotation via the closed-loop controller, as described in Equation (2.2), or mechanically blocked.

In the case where the wheels are actuated, the wheel position changes at the touchdown instant, however, this occurs before the leg reaches the threshold at which the leg state machines transitions between flight and stance. Figure 4.1 shows the effect of touchdown on the wheel velocity and the leg length, where the * represents the point at which the leg state machine changes from flight to stance. The first dotted vertical line represents the instant at which the wheel first displaces, while the second vertical dotted line represents the instant when the leg potentiometers reach the leg compression threshold. The time separating these two instances is approximately 0.015 seconds.

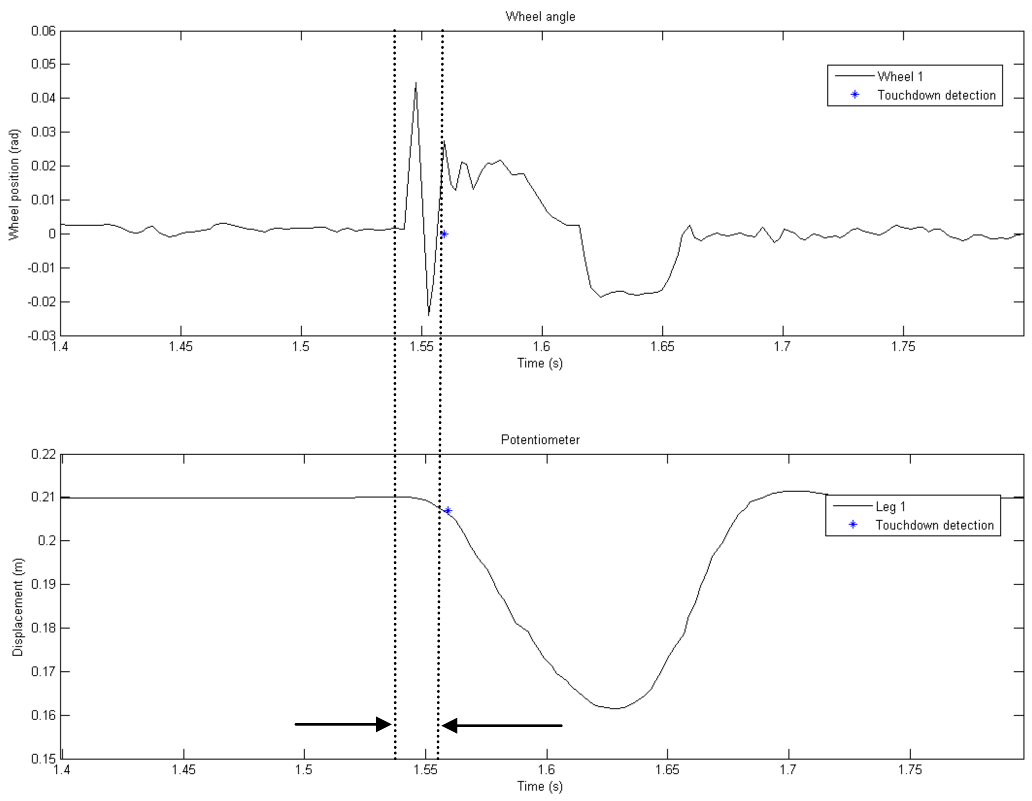
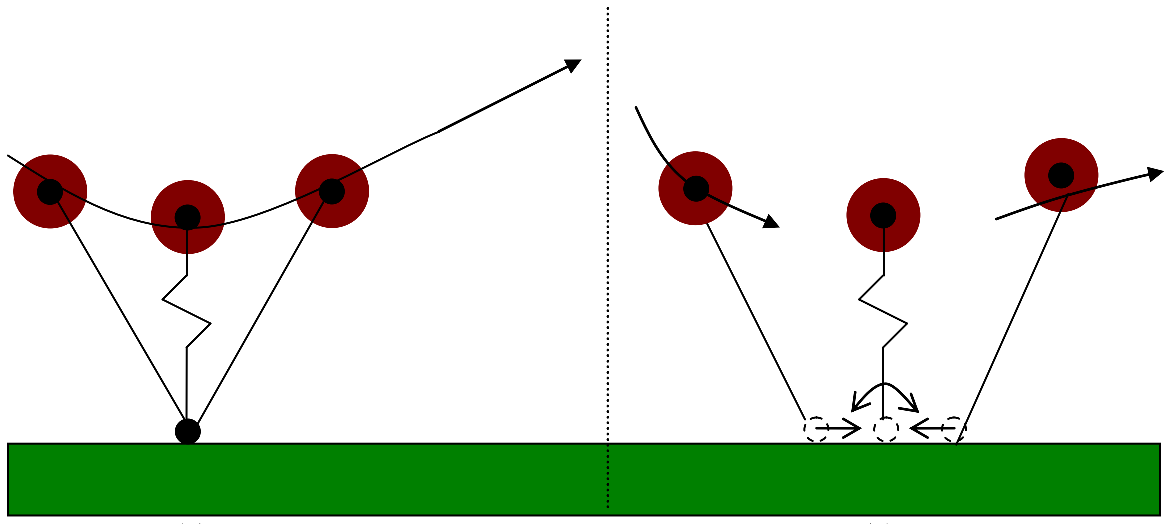


FIGURE 4.1 TOUCHDOWN DETECTION

The touchdown is identified by a change in angular position of the wheel, which leads to a counteracting actuating motion to ensure the wheel remains at rest. This ultimately negates the SLIP model assumption, which affects the stability of the stance phase, as seen in Figure 4.2 (a) and (b).



(A) BLOCKED WHEELS (B) ACTIVE WHEELS

FIGURE 4.2 SLIP MODEL WITH WHEELS

The actuated wheels lead to variability in the hip actuation, which tends to destabilize the robot during the stance phase, the phase in which stability is normally gained for the next step in the bound gait [29]. It was noted during the present work on PAW and in [29], that this correlates to the level of repeatability between tests, as actively controlled wheels during bound lead to higher failure rates. Figures 4.3 and 4.4 compare results from bounding tests with actuated wheels vs. mechanically blocked wheels, respectively, to demonstrate the variability in the velocity and pitching motion during bound. The test shown in Figure 4.3 uses the same fixed touchdown and liftoff angles as in Figure 4.4. A visual comparison of the two tests clearly shows a more regular bounding motion with the mechanically blocked wheels.

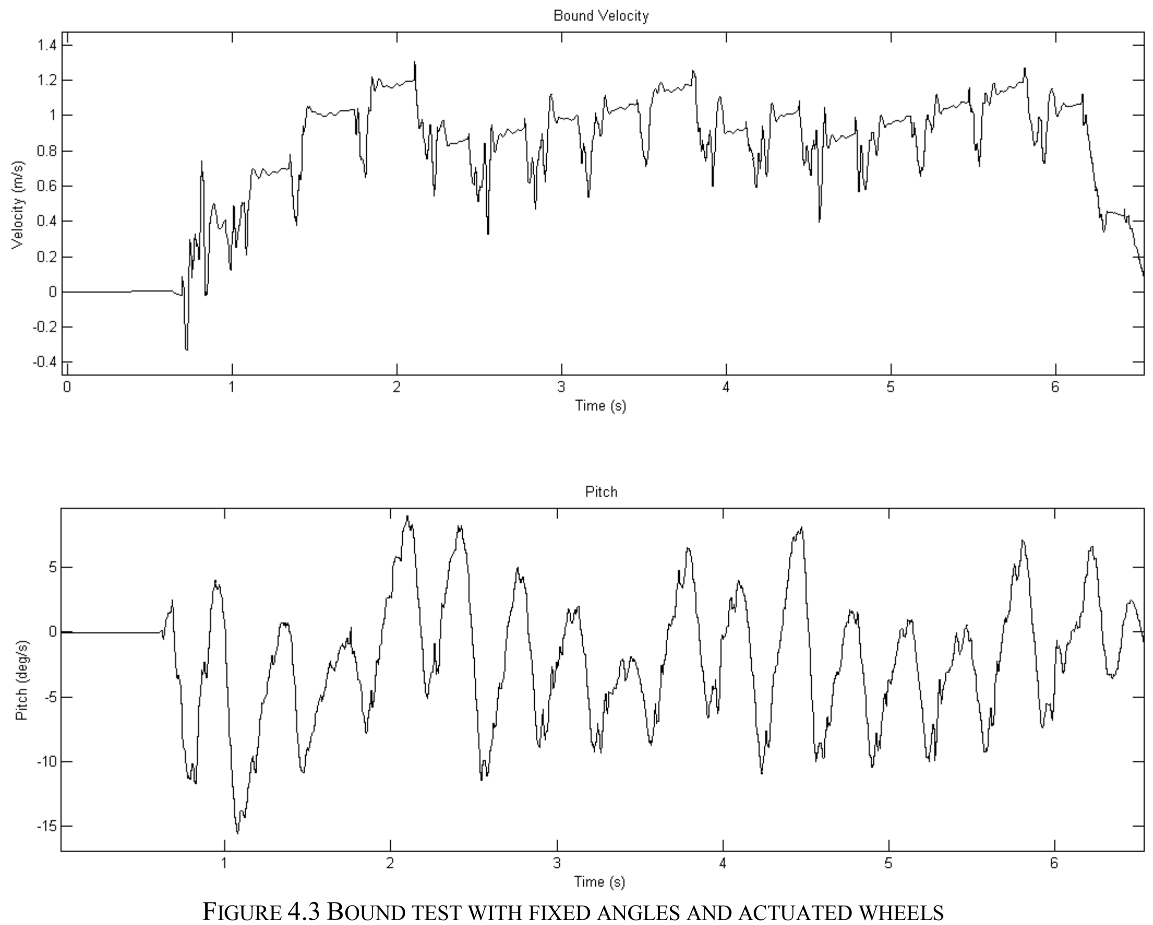
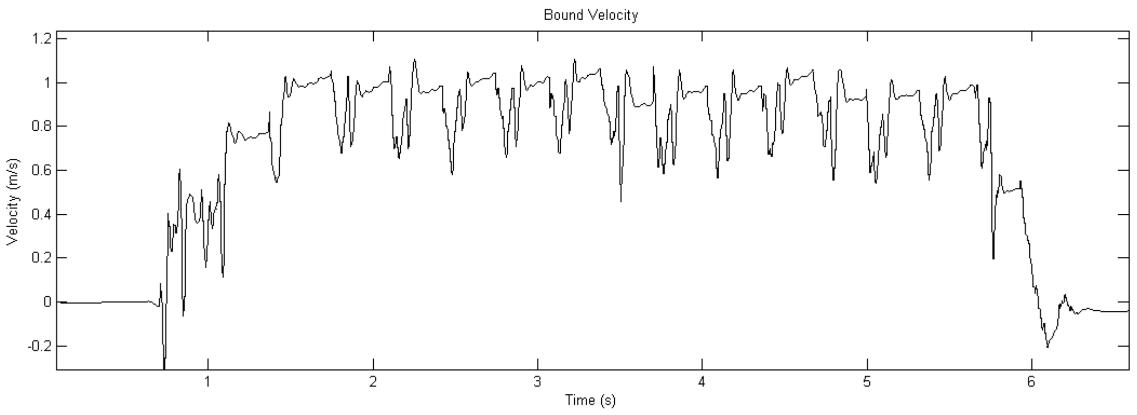


FIGURE 4.3 BOUND TEST WITH FIXED ANGLES AND ACTUATED WHEELS



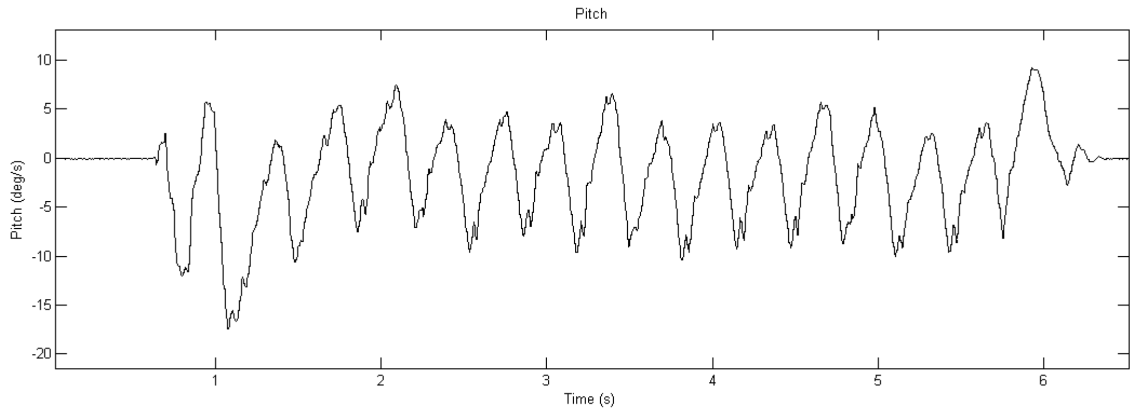


FIGURE 4.4 BOUND TEST WITH FIXED ANGLES AND MECHANICALLY BLOCKED WHEELS

In order to evaluate the intelligent controller under more favorable conditions, similar to the MATLAB/Adams co-simulation, the mechanically blocked wheels method was used to generate the results in Section 4.2. To block the wheels, hot glue is injected in the bevel gear between the wheel and the wheel motor. Additionally, electrical tape is used to reinforce the blocking mechanism, while the wheel motors are deactivated in the robot code. Figure 4.5 shows an image of the mechanically blocked wheels.

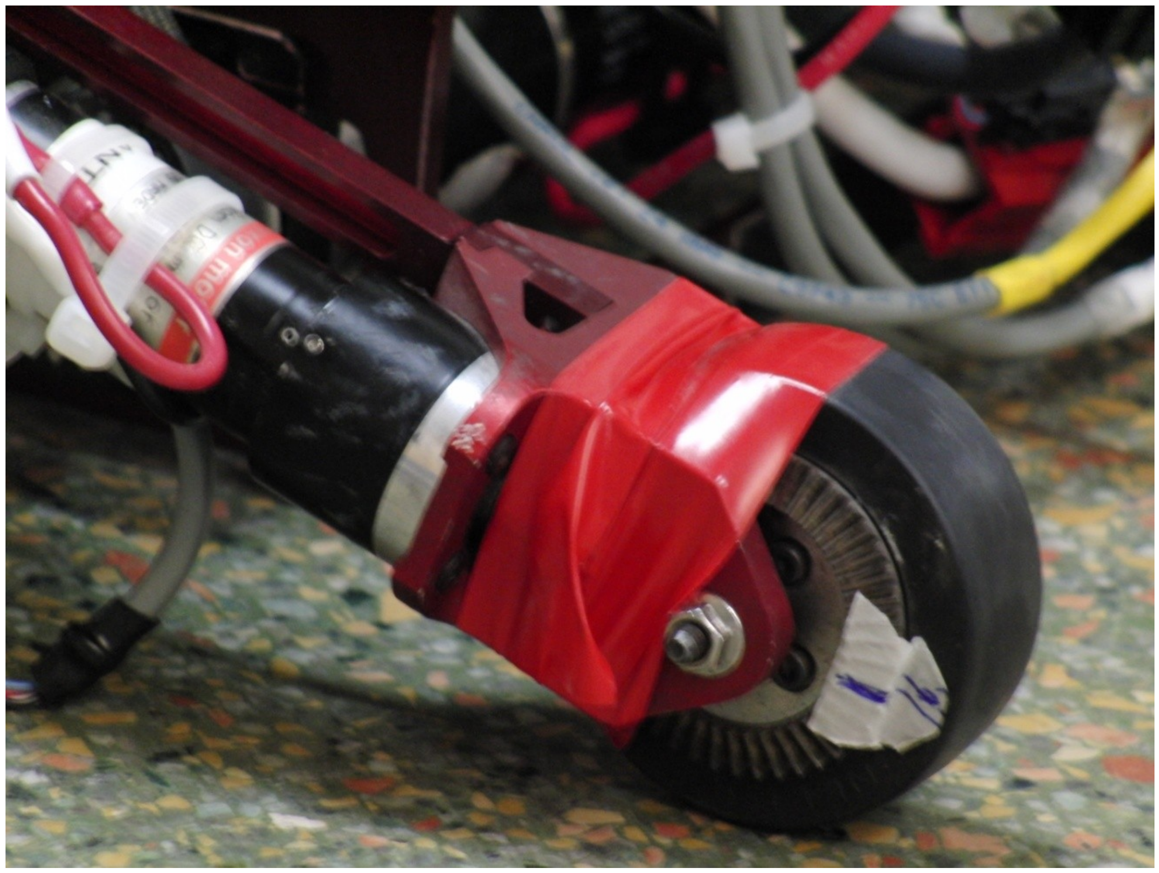


FIGURE 4.5 MECHANICALLY BLOCKED WHEEL

4.1.3 ROBOT CONTROLLER BLOCK DIAGRAM

Figure 4.6 represents the block diagram of the control scheme implemented on the robot.

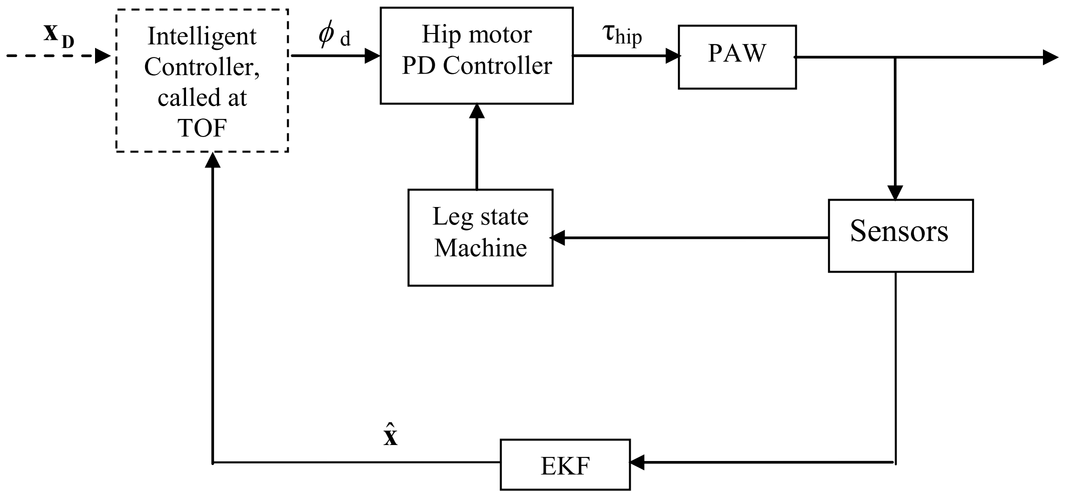


FIGURE 4.6 ROBOT CONTROLLER BLOCK DIAGRAM

4.2 CONTROLLER PERFORMANCE

This section demonstrates the effectiveness of the intelligent controller to track a desired set point, and to transition between desired set points during the bound gait.

4.2.1 CONVERGENCE

The controller's tracking ability was evaluated by testing various set points. Figure 4.7 demonstrates a test in which the desired TOF velocity was 1.0m/s.

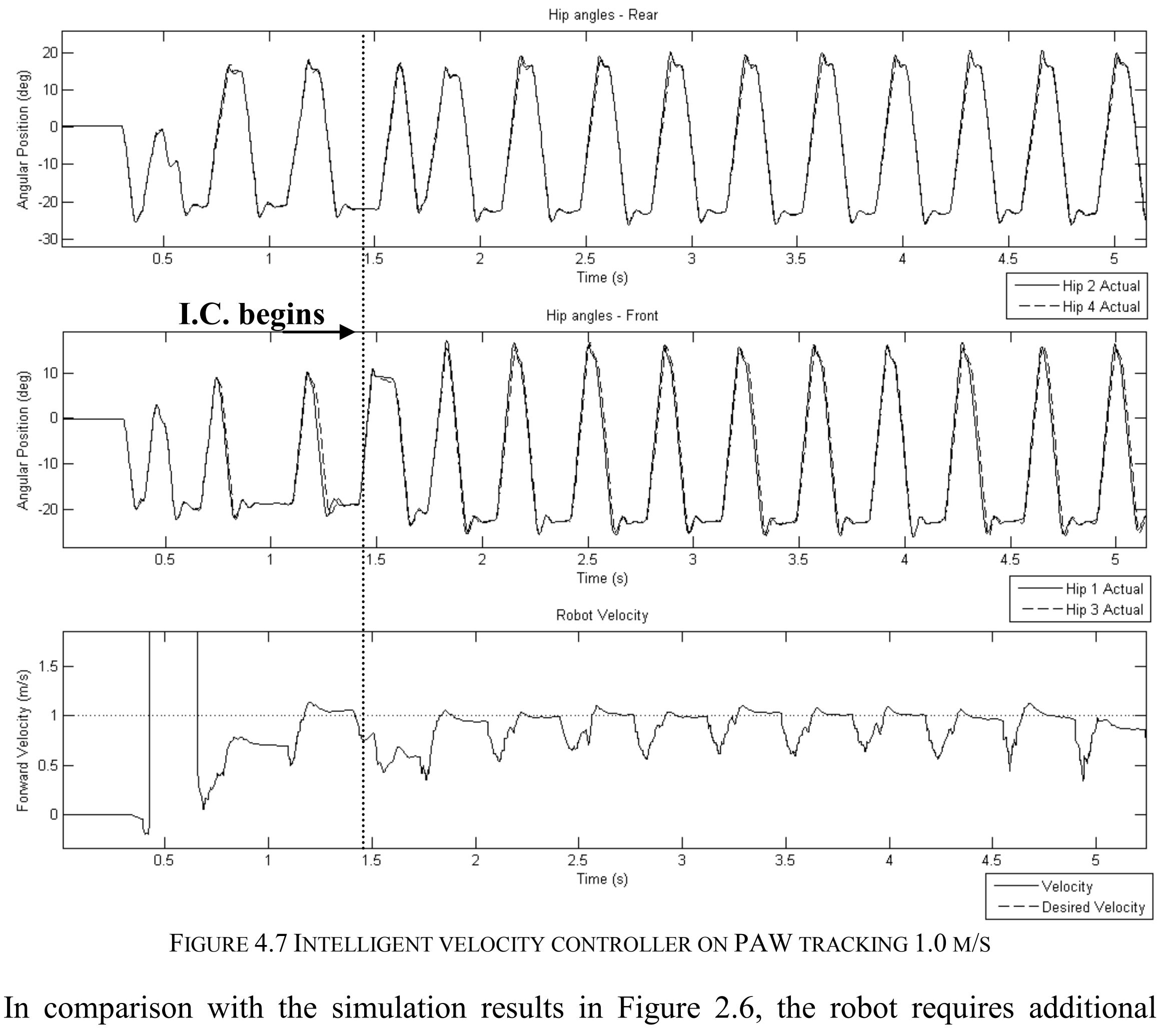


FIGURE 4.7 INTELLIGENT VELOCITY CONTROLLER ON PAW TRACKING 1.0 M/S

In comparison with the simulation results in Figure 2.6, the robot requires additional strides to converge to the desired set point. This is due to the start-up routine for the bound gait. In simulation, the robot is simply dropped from a predefined height with an initial pitch, whereas in physical testing, the robot has a kick-off routine to initiate the bound. Combined with changing hip angle set points from the intelligent controller, stable bounding takes longer to achieve and bound failure rates rise on the robot. To compensate for this reality, the first 4 strides in the bound on the physical robot are set with predefined touchdown and liftoff angles, so that stable pitching is achieved prior to the initialization of the intelligent controller. Intelligent controller starting instant is indicated by the vertical dashed line through the plot.

An analysis of the tracking ability was performed for each bound test with a fixed set point, of which the results are shown in table 4.1. The values in Table 4.1 for each set point are computed from a single trial. These trials are also illustrated in Figures 4.7 to 4.11. The repeatability between trials of the controller is high as failures only occur when there is an error reading data from the IMU to the robot's I/O board, which in turn affects the velocity estimate used in the control loop.

Velocity set point (m/s)	**σ** (at controller initialization)	# of strides till convergence	**σ** (after convergence)
0.9	0.11	3	0.116
1.0	0.12	1	0.0752
1.1	0.065	3	0.0558
1.2	0.134	3	0.0763
1.3	0.136	4	0.0749

TABLE 4.1 CONTROLLER TRACKING ABILITY

The standard deviation is computed by taking the error between the desired COM velocity and the actual COM velocity at the top-of-flight instant (the value used in the controller), for each stride. The standard deviation calculation above begins at the start of the intelligent controller. As seen in Figure 4.7 and in the results in Chapter 2, several strides are required to achieve the desired velocity. Thus, the **σ** value in Table 4.1 is high in some cases. The column entitled '# of strides till convergence' indicates the number of strides between the initialization of the intelligent controller to when the set point is achieved. The second calculation of the standard deviation serves as a measure of how well the intelligent controller maintains the desired set point after the required number of strides for convergence. In most cases, this is a significantly lower value. This simply means that although as the controller may take several strides to achieve the set point, mostly in the cases for high velocities, the velocity is maintained quite accurately. Figure 4.8 is an example of this, where a velocity of 1.3m/s is desired.

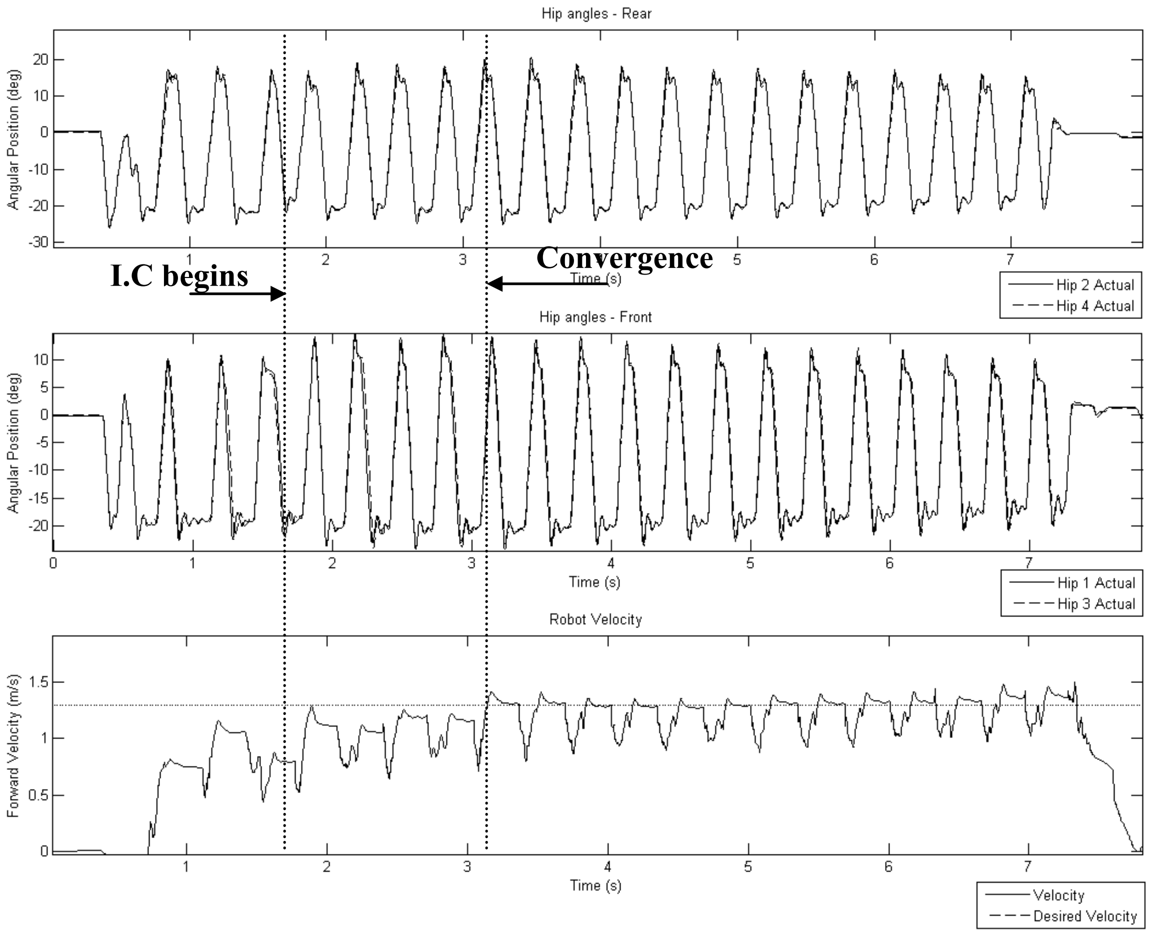


FIGURE 4.8 INTELLIGENT VELOCITY CONTROLLER ON PAW TRACKING 1.3 M/S

The results in Table 4.1 indicate that as the desired velocity moves away from 1.0m/s, more strides are needed to converge towards the desired set point. This is because prior to the intelligent controller initialization, the fixed touchdown and liftoff angles used cause the robot to bound near 1.0 m/s. As the set point deviates from 1.0m/s, there is a larger error between the robot's actual velocity and the desired velocity at the start-up of the intelligent controller.

Another interesting observation is that, at higher velocities, the robot maintains the velocity set point better than at low velocities. This is likely due to the relationship between stride length and velocity, as seen in Equation (2.3). As desired velocity increases, the robot's strides become longer, increasing both the stance time and the flight time. As indicated in [29], the stance phase of the bound allows the robot to maintain stability. At lower velocities, as stance time decreases, the robot has less time to achieve the desired hip angles, increasing the chances that hip angles are not achieved prior to lift-off. Ultimately, this leads to fluctuations in the bound velocity, hence the higher fluctuations about the set point at 0.9 m/s in Figure 4.9. As it will be discussed in section 4.3, there is a minimum velocity where the gait is able to maintain stability.

Figures 4.9 to 4.11 illustrate the controller's performance for the velocity set points of 0.9 m/s, 1.1 m/s, and 1.2 m/s respectively.

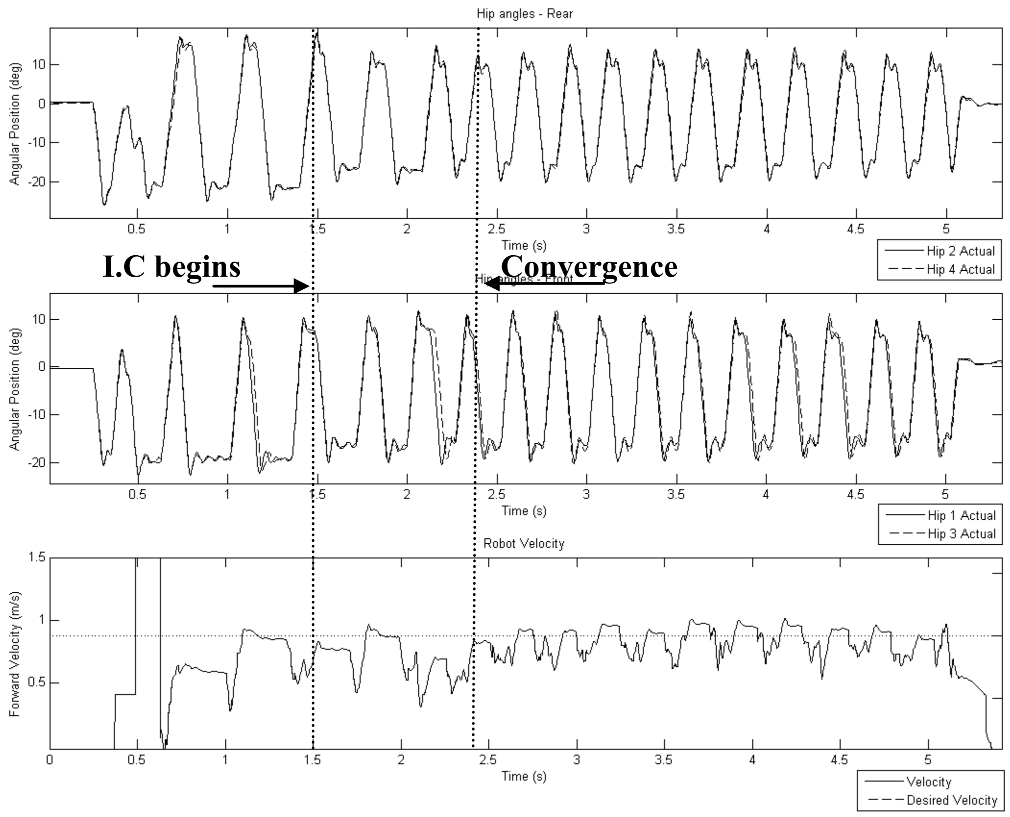


FIGURE 4.9 INTELLIGENT VELOCITY CONTROLLER ON PAW TRACKING 0.9 M/S

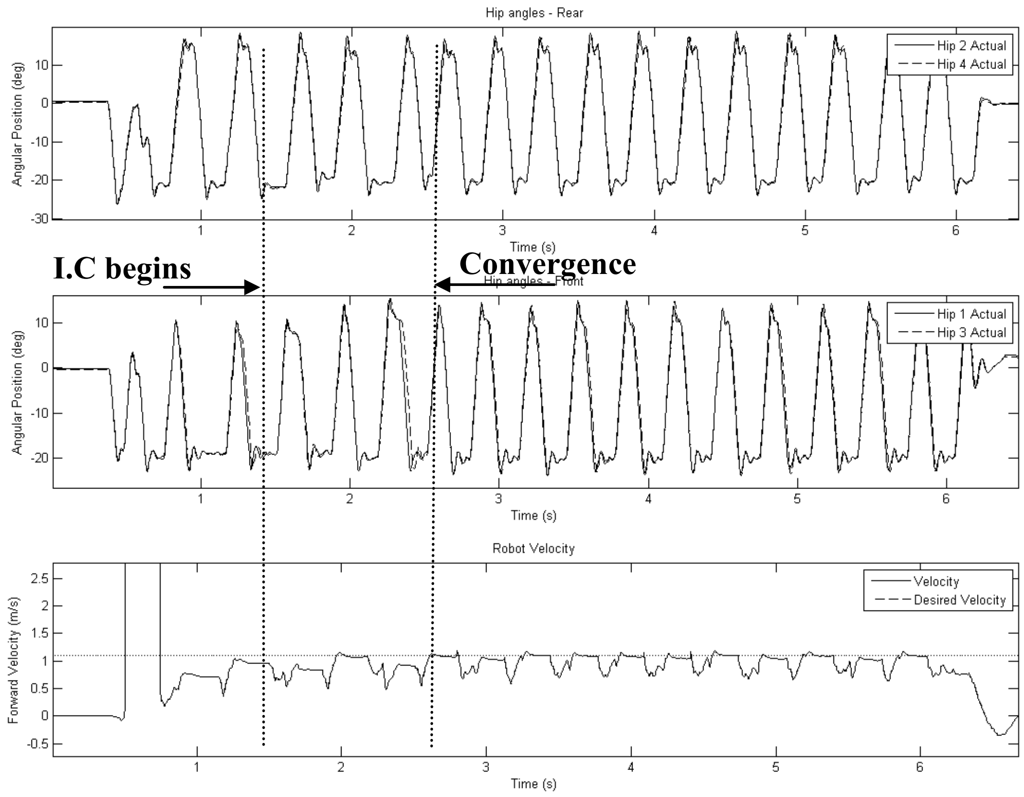


FIGURE 4.10 INTELLIGENT VELOCITY CONTROLLER ON PAW TRACKING 1.1 M/S

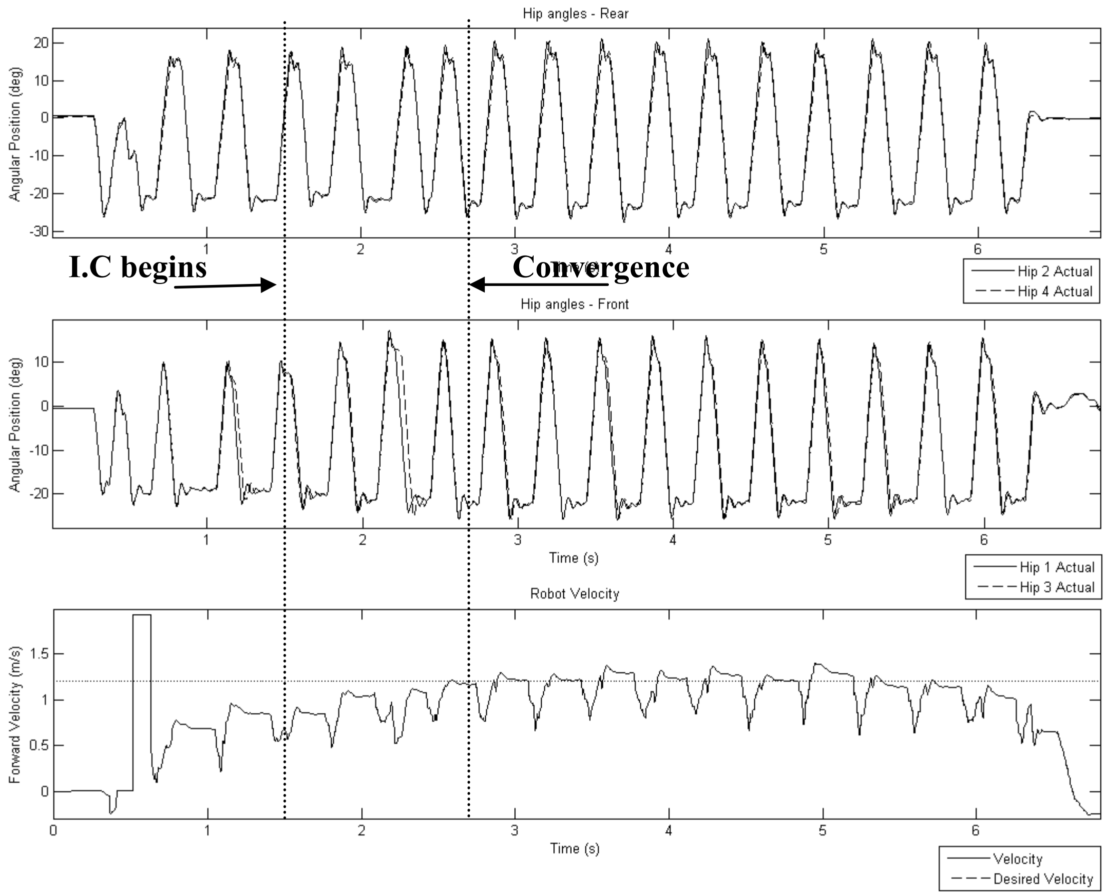


FIGURE 4.11 INTELLIGENT VELOCITY CONTROLLER ON PAW TRACKING 1.2 M/S

4.2.2 ACCELERATION

Although analyzing the ability of the controller to achieve a desired set-point is essential in evaluating its performance, the advantage of such a controller is to allow for the robot to accelerate and decelerate during the bound gait. As seen in Section 4.2.1, the controller can maintain a desired set point with a standard deviation of less than 0.1 after convergence. Thus, in the acceleration and deceleration tests shown in the following sections, it was deemed more appropriate to track large velocity transitions, as they are more statistically significant.

Figure 4.12 demonstrates an acceleration test of the robot between 0.9m/s and 1.3 m/s. The acceleration in the robot occurs smoothly, as an increase in the stance and flight phase increases stability, which leads to improved velocity tracking at the higher velocity. The new set point is achieved after 3 bounds, due to the rapid transition in hip angles, as seen in Figure 4.12.

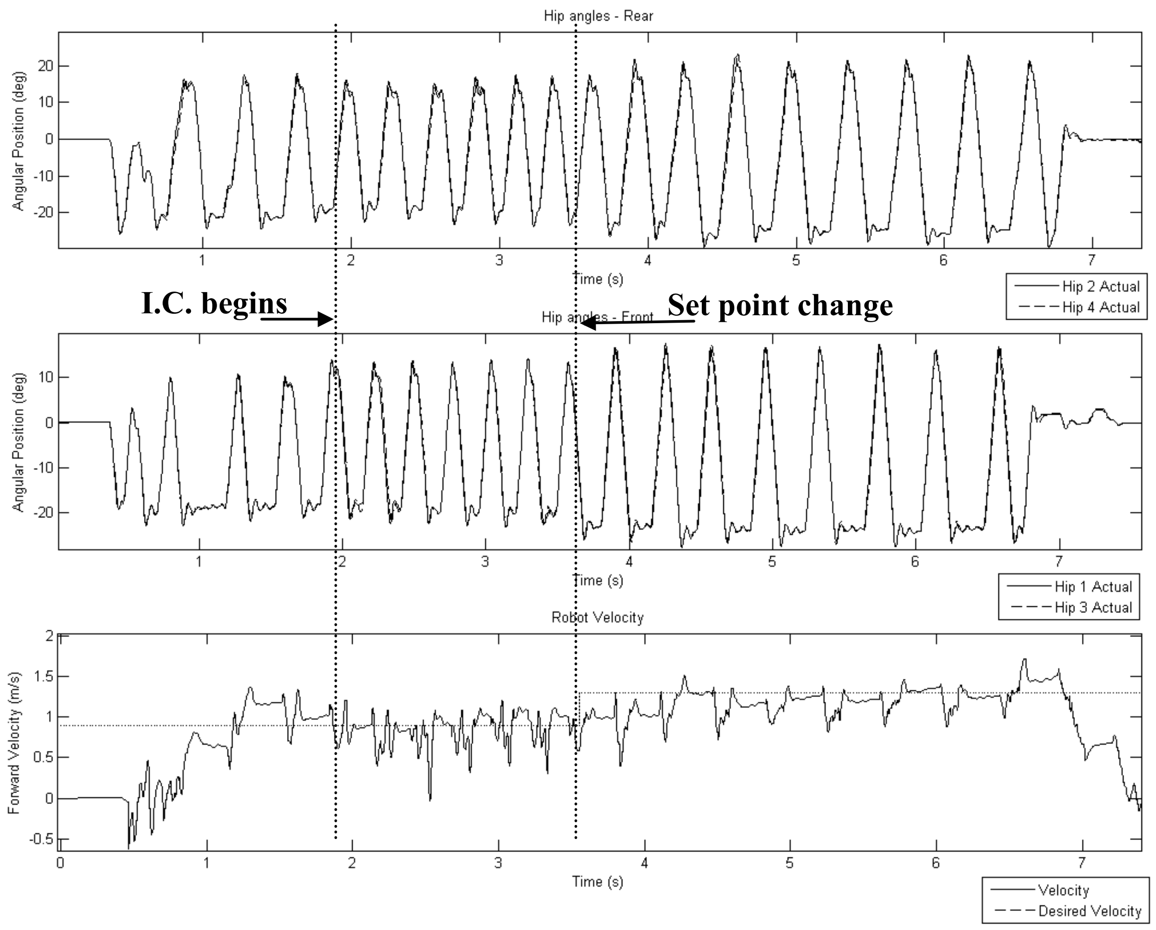


FIGURE 4.12 INTELLIGENT VELOCITY CONTROLLER ACCELERATING 0.9 M/S TO 1.3 M/S

4.2.3 DECELERATION

The deceleration test in Figure 4.13 shows a velocity transition between 1.1 m/s and 0.9 m/s. The deceleration takes 4 strides, although the shorter stance and flight times at lower velocities cause higher errors about desired set point as in Figure 4.13. Overall though, these results indicate that the intelligent controller is capable of transitioning between velocity set points.

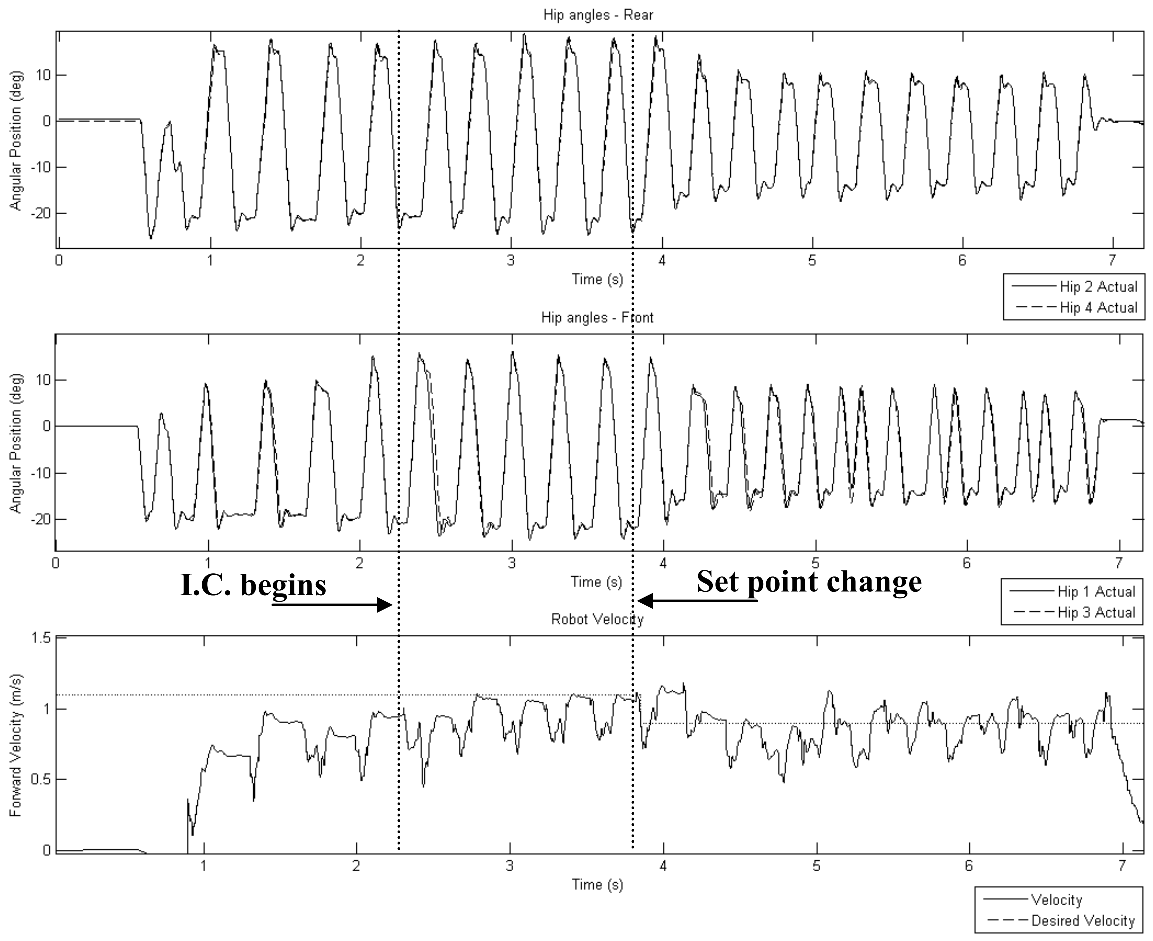


FIGURE 4.13 INTELLIGENT VELOCITY CONTROLLER DECELERATING 1.1 M/S TO 0.9 M/S

4.3 CRITICAL PARAMETERS AND LIMITATIONS

Extensive testing of the controller was performed to achieve the results shown in Section 4.2. Several parameters were identified during testing as critical to the success of the controller. Outlined in the subsequent subsections are these parameters and their effect on the robot's gait.

4.3.1 IMPORTANCE OF α_3 GAINS

The intelligent controller, as described in Chapter 2, uses a gain α_3 that must be tuned manually. As this gain is required for the front and rear leg pair controllers, for touchdown and liftoff angle computation, four values for α_3 need to be specified. The α_3

component in the controller equation acts as a correction factor when computing the required hip angle to achieve the desired bound velocity. The intelligently tuned gains α_1 and α_2 in equation (2.6) will lead to adjustments at each stride of the required hip angles, based on the feedback error. As will be shown in Section 4.3.3, if the front or rear leg hip angles are too large or small, the bound gait can become irregular and lead to gait failure. Testing showed that, in tuning the α_3 parameter, the hip angle changed up to 4° for each 0.01 increment of α_3. Thus, tuning the α_3 parameter was an important process in achieving a robust controller. Table 4.2 shows the values for the α_3 gains on both the robot and in simulation.

α_3 parameter	Simulation Value	Robot Value
Front Touchdown	0.01	0.01
Rear Touchdown	0.00	0.00
Front Liftoff	-0.01	-0.01
Rear Liftoff	-0.03	-0.02

TABLE 4.2 CONTROLLER A3 GAINS

4.3.2 START-UP ROUTINE

As mentioned in Section 4.2, the robot initiates the bound sequence differently from the simulation model. The robot's legs are actuated forward to a fixed angle, and then rapidly commanded back to a fixed take-off angle to inject energy into the springs to cause flight. Once the robot is in flight, the legs are then commanded to the bound gait touchdown angles. Prior to initiating the intelligent controller, it is important that the robot is in a regular bounding motion, or any instability in the gait may be enhanced by changes to the touchdown and liftoff angles. Therefore, in the tests presented in this chapter, the intelligent controller was commanded to begin after the fourth stride in the bound cycle, once a cyclical bound gait is achieved.

Additionally, the fixed angles used prior to starting the intelligent controller are important parameters, as they ultimately determine the actual bounding velocity in the first feedback loop of the controller. Setting the fixed angles to achieve a velocity of approximately 1.0m/s during the start-up routine allows the controller to transition well between any allowable desired velocities, without any instability.

4.3.3 VELOCITY THRESHOLDS

Similarly to what was observed in simulation, there is upper and lower limit velocity thresholds. At set points above of 1.3 m/s, the large desired front leg touchdown angles cause a decrease in the pitching motion of the robot during bound, as seen in Figure 4.14. This irregular pitching motion causes the touchdown sequence of the legs to change, so that the rear legs touchdown before the front legs. This touchdown sequence causes unpredictable robot behavior, leading to failure. In simulation, these failures consisted of the robot's rear legs' dragging until liftoff no longer occurs, as discussed in Section 2.3.4. On the robot, this dragging behavior is dangerous as it causes unpredictable output of the leg state machine. This in turn causes unpredictable hip motor actuation that could result in unpredictable motions of the robot and possible damage.

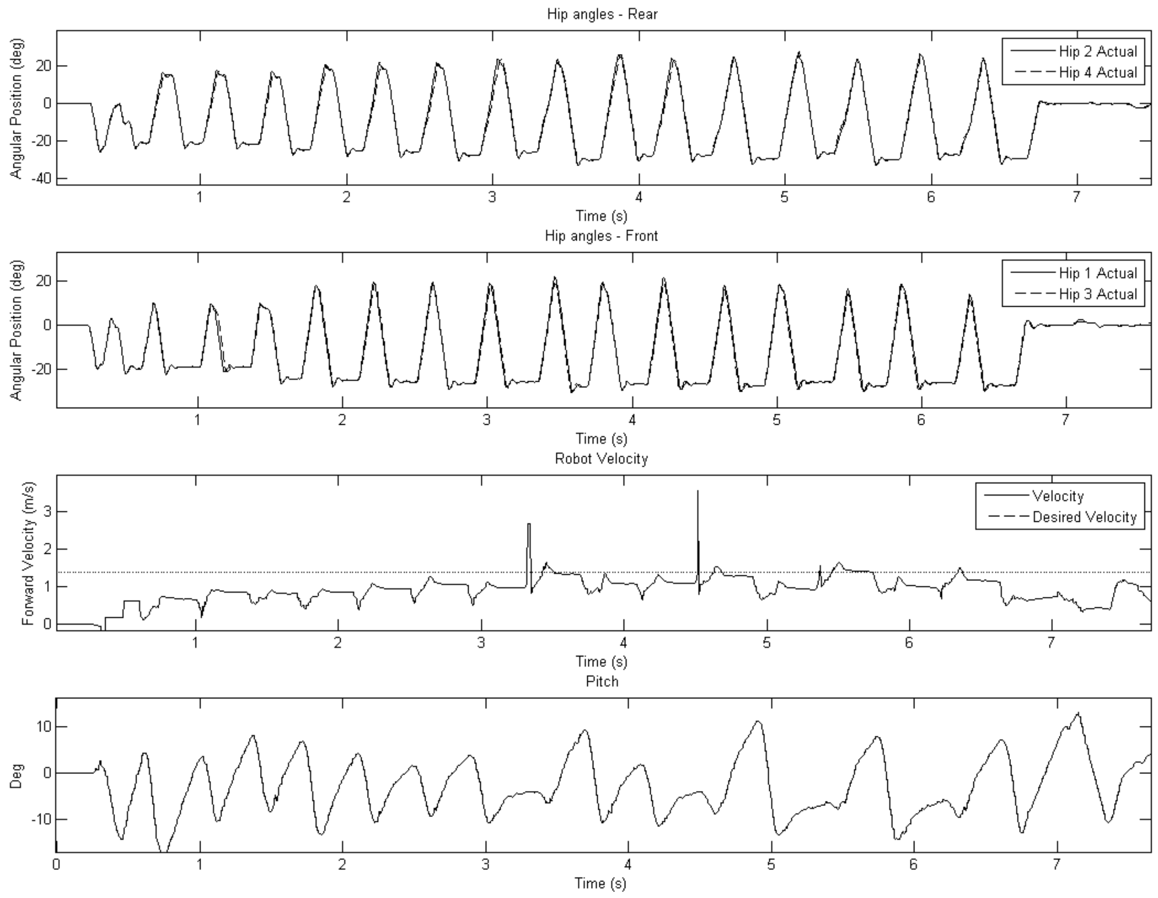


FIGURE 4.14 INTELLIGENT VELOCITY CONTROLLER ON PAW TRACKING 1.4 M/S

For the PAW robot, the controller's lower limit was found to be approximately 0.9 m/s At low velocities, the robot's stance and flight time are shorter, and the desired hip angles are much smaller. With smaller required hip actuation, less compression occurs in the front leg pair, eventually leading to the pair not lifting-off. Thus, the front leg pair drags, as the rear legs continue to be actuated as there is no controller-coupling. This behavior occurs rapidly, where the robot pivots about the front leg pair to flip forward, landing upside down. An emergency power cut-off switch was created to stop any human or robot damage. Unfortunately, when this failure occurs, the power is cut before any data can be saved to the robot's computer. Thus, the evidence of this behavior is seen in the images in Figure 4.15, which resembles the failure occurring in simulation at low velocities, as seen in Figure 2.17.

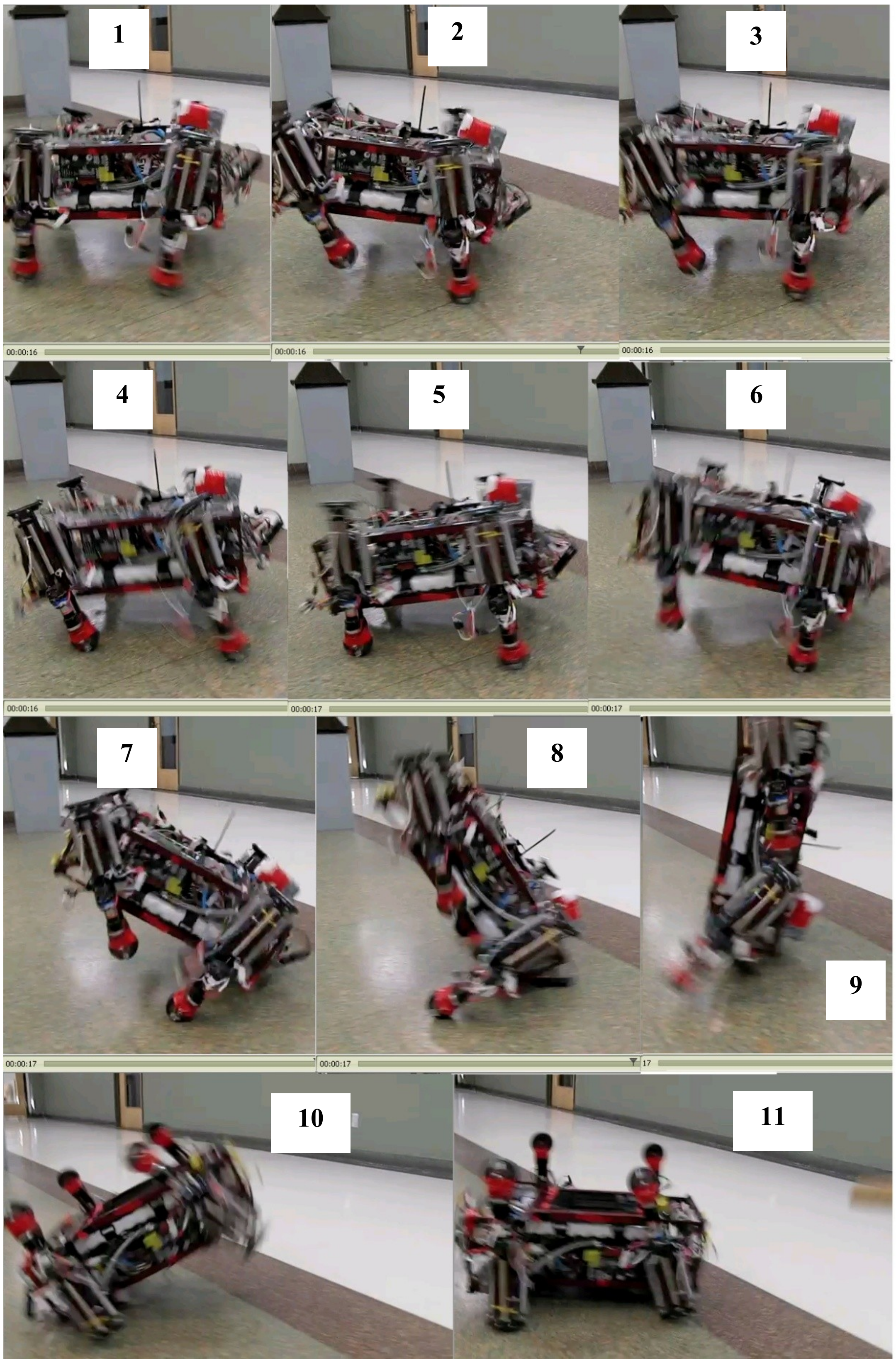


FIGURE 4.15 ROBOT FAILING AT VELOCITIES BELOW 0.8M/S

CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS

The goal of the implementation of an intelligent velocity controller was to introduce an increased level of autonomy and robustness to the PAW robot. The simulation results in Chapter 2 demonstrated the controller's performance in tracking velocity, and discussed the controller's limited ability to maintain its set point in varying terrain types. To implement this controller on the physical platform, an extended Kalman filter was developed in Chapter 3 to estimate the forward bounding velocity of PAW. Finally, Chapter 4 evaluated the controller's performance on the robot, identified some limitations and discussed the critical parameters to the success of the bound gait. This chapter will present a discussion of the contributions made to the PAW robot, as well as present some recommendations for future work.

5.1 Contributions to PAW

This research presents to the reader an intelligent controller developed and tested to allow the PAW robot to track a velocity set point. In doing so, the sensing abilities of the robot have been increased, as well as its versatility. The following subsections present an evaluation of the contributions made to the platform.

5.1.1 Intelligent Velocity Controller

In simulation, the intelligent controller demonstrated velocity tracking between 0.9m/s to 1.3m/s, with errors up to 0.1m/s. The controller adequately transitions between high and low velocity, demonstrating good acceleration and deceleration capacities. The controller was then evaluated on the PAW platform to validate the simulation results. The results from physical testing correlate quite closely with the simulation results, where tracking errors range up to 0.1m/s once convergence is met. The allowable velocity tracking with the controller on the robot is the same as in simulation, between 0.9 m/s to 1.3 m/s. Ultimately, the controller found the upper and lower limits of velocity in bound for the PAW robot.

5.1.2 Robustness to Disturbances

The controller was evaluated in simulation with changing terrain types, where the friction coefficient and stiffness of the ground was varied. It was seen that the intelligent controller attempted to adapt to the changing environment by changing the desired touchdown and liftoff angles required for tracking the velocity set point. It was shown, however, that the controller's robustness to the changing environment is limited on PAW. This is due to the passive nature of the robot's dynamics. As the stiffness or friction of the ground is reduced, the robot's ability to inject energy into the passive legs during stance is reduced, causing a change in the bound gait. A reduction of ground stiffness had more detrimental effects, as the decrease in the leg compression directly affected the flight phase of the bound gait leading directly to failure. The friction coefficient change causes some slip at the point of contact between the leg and the ground. The effect is that the bound gait remains stable; however, the performance of the velocity tracking is reduced to up to 0.3m/s errors.

Although the controller demonstrated adequate velocity tracking on the physical robot, the robustness to terrain changes was not tested. In an indoor environment, the controller performed well on both linoleum and concrete. The alternative terrains available were outdoor environment conditions, such as grass or gravel. The stiffness and friction of these outdoor conditions are uncontrolled and inconsistent, and ultimately, the controller robustness would not be evaluated, as failures under outdoor environments would not be isolated to the controller's performance but to the robot's general physical characteristics.

5.1.3 Extended Kalman Filter

The extended Kalman filter, presented in Chapter 3, was necessary for estimating the top-of-flight center of mass velocity during the bound gait. Prior to the EKF development, the inertial measurement unit provided a noisy estimate of velocity prone to drift. The accuracy of the measurement was evaluated against a high speed video analysis of the robot's motion and deemed accurate up to 0.1 m/s. In combining the IMU measurement with a kinematics' velocity estimate, the flight phase velocity of the bound was comparable to the simulation result's true velocity, confirming the accuracy of the proposed method. Ultimately, the EKF developed presents an accurate way to estimate the forward velocity of the robot during dynamically stable gaits and has the potential to expand the onboard sensing capabilities. The potential for increasing sensing will be discussed in Section 5.2, which presents avenues for future control possibilities.

5.2 RECOMMENDATIONS

The present work presents the initial steps taken towards increasing the autonomy and versatility of the PAW robot. PAW is now capable of transitioning smoothly between various velocities, without extensive parameter tuning, and has the potential to withstand some limited varying terrain conditions. Additionally, the EKF developed presents an opportunity for an expansion on the sensing capabilities of the platform. The following subsection presents some possible avenues for future work on the robot.

5.2.1 DIRECTIONAL CONTROLLER

A directional controller for the bound gait is a logical expansion of the presented work. The work in [29] discusses fixed touchdown and liftoff angles that cause a yawing motion. Similarly to velocity control, the yaw could be controlled at each top-of-flight instant to determine the required touchdown and liftoff angles to achieve the desired lateral motion. Instead of a front and rear leg pair, the controller would use a lateral leg pair combination. To combine this controller with the existing intelligent velocity controller, the directional controller could output an angular phase difference between the right and left leg pairs. This phase difference could be added to the touchdown and liftoff angles determined from the intelligent velocity controller. A phase difference in the desired hip angles could cause a change in the touchdown sequence, resembling a galloping gait, an interesting effect, as this gait would occur almost naturally, out of necessity to achieve a desired motion.

5.2.2 Hopping Height Controller

As shown in [19], hopping height and forward velocity are directly related. An evaluation of the present work could be made to verify the correlation between the two states. An extension could be made based on the findings to evaluate the height at each top-of-flight instant, to achieve a desired height. Similarly to the intelligent velocity controller, there would be upper and lower limits to the desired set point. Nonetheless, the use of such a controller would be apparent if the robot would have to overcome a known obstacle.

5.2.3 Sensing Capabilities

As mentioned in Section 5.1.3, the EKF performs quite well in estimating center of mass forward velocity. The state vector **x** used in the EKF is an 11x1 vector, but could be expanded to include additional states. These additional states could include, but are not limited to: the position in the inertial frame, attitude, and attitude rate of the robot. By filtering these additional states in the EKF, a more precise estimate becomes available, expanding the control possibilities. As it was demonstrated in this work, accurate velocity control requires an accurate estimate of velocity. Thus, the recommendations for future controllers will require additional sensing capacities. As the EKF has already been developed, an expansion on the existing framework would provide the required sensing capabilities necessary for alternate controllers.

5.2.4 INCREASED AUTONOMY

The goal of dynamically stable running robots is to provide a platform with a high level of autonomy capable of rapidly traversing rough terrain. As discussed in Section 5.1.3, PAW's physical characteristics may limit its use in various terrains. Nonetheless, this work demonstrated that PAW can be used as a platform for developing and testing various intelligent control schemes and sensing algorithms. PAW also presents itself as an excellent platform for developing increased locomotion capabilities for hybrid robots such as jumping or step climbing [40].

The recommendations made in this section, present solutions to build on the versatility of PAW. Ultimately, a higher level of control would decide on the most efficient gait for the robot to accomplish a desired task or traverse an unknown terrain. Efficiency may be evaluated by energy consumption or task completion time, via a reward function. Then, obstacles, terrain, and disturbances will determine the lower level controls: desired velocity, direction, and hopping height.

REFERENCES

[1] M.H. Raibert, “Legged Robots That Balance”. The MIT press, Cambridge, Massachusetts, 1986.

[2] Massachusetts Institute of Technology Leg Laboratory (online), Available: “http://www.ai.mit.edu/projects/leglab/robots/robots.html”, 2008.

[3] M.H. Raibert, “Trotting, Pacing, and Bounding by a Quadruped Robot”. Journal of Biomechanics, Vol. 23, pp. 79-98, 1990.

[4] J.K. Hodgins, M.H. Raibert, “Adjusting Step Length for Rough Terrain Locomotion”. IEEE Transactions on Robotics and Automation, Vol.7, No.3, pp.289-298, June 1991.

[5] J.G. Nichol, S.P.N. Singh, K.J. Waldron, L.R. Palmer III, D.E. Orin, “System Design of a Quadrupedal Galloping Machine”. The International Journal of Robotics Research, Vol. 23, No. 10-11, pp. 1013-1027, October-November 2004,.

[6] J. Hodgins, “Legged Robots on Rough Terrain: Experiments in Adjusting Step Length”. Proceedings of the 1988 IEEE International Conference on Robotics and Automation, Vol. 2, pp. 824-82, 24-29 April, 1988.

[7] L.R. Palmer III, D.E. Orin, “3D Control of a High-Speed Quadruped Trot”. Industrial Robot, Vol. 33, No. 4, pp. 298-3023, 2006.

[8] D.J. Pack, A.C. Kak, "A Simplified Forward Gait Control for a Quadruped Walking Robot". Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, Munich, Germany, Vol.2, pp. 1011-1018, Septemeber, 1994.

[9] D.P. Krasny, D.E. Orin, "Achieving Periodic Leg Trajectories to Evolve a Quadruped Gallop". Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 3842-3848, September, 2003.

[10] Z.G. Zhang, H. Kimura, K. Takase, "Adaptive Running of a Quadruped Robot Using Forced Vibration and Synchronization". Journal of Vibration and Control, Vol. 12, pp 1361-1383, 2006.

[11] S. Chernova, M. Veloso, "An Evolutionary Approach to Gait Learning for Four-Legged Robots". Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, September 28 – October 2, 2004.

[12] P. Arena, L. Fortuna, M. Frasca, G. Sicurella. "An Adaptive, Self-Organizing Dynamical System for Hierarchical Control of Bio-Inspired Locomotion". IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 34, No. 4, pp. 1823- 1837, August, 2004.

[13] Y. Fukuoka, H. Kimura, A.H. Cohen, "Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts". The International Journal of Robotics Research, Vol. 22, No. 3-4, pp. 187-202, March-April 2003.

[14] M. A. Lewis, G. A. Bekey, "Gait Adaptation in a Quadruped Robot". The Journal of Autonomous Robots, Vol. 12, No. 3, pp. 301-312, 2, May, 2002.

[15] I. Poulakakis, E. Papadopoulos, M. Buehler, "On the Stability of the Passive Dynamics of Quadrupedal Running with a Bounding Gait". The International Journal of Robotics Research, Vol. 25, No. 7, pp. 669-687, July 2006.

[16] L.R. Palmer III, D.E. Orin, "Attitude Control of a Quadruped Trot While Turning". Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, October 9-15, 2006.

[17] L.R. Palmer III, D.E. Orin, "Force Redistribution in a Quadruped Running Trot". Proceedings of 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, April 10-14, 2007.

[18] L.R. Palmer III, D.E. Orin, "3D Control of a High-Speed Quadruped Trot". Industrial Robot, Vol. 33, No. 4, pp. 298-3023, 2006.

[19] D.W. Marhefka, D.E. Orin, J.P. Scmiedeler, K.J. Waldron, “Intelligent Control of Quadruped Gallops”. IEEE/ASME Transactions on Mechatronics, Vol. 8, No. 4, December, 2003.

[20] D. P. Krasny, D. E. Orin, “Evolution of Dynamic Maneuvers in a 3D Galloping Quadruped Robot”. Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Florida, May, 2006.

[21] A. Muraro, C. Chevallereau, Y. Aoustin, “Optimal Trajectories for a Quadruped Robot with Trot, Amble, and Curvet Gaits for two Energetic Criteria”. Multibody System Dynamics, Vol. 9, pp 39-62, 2003.

[22] D.W. Marhefka and D.E. Orin, "Fuzzy Control of Quadrupedal Running". Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, pp. 3063-3069, April, 2000.

[23] D. P. Krasny, D.E. Orin, “Generating High-Speed Dynamic Running Gaits in a Quadruped Robot Using an Evolutionary Search”. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 34, No. 4, August, 2004.

[24] L.R. Palmer, D.E. Orin, D.W. Marhefka, J.P. Schmiedeler, K.J. Waldron, “Intelligent Control of an Experimental Articulated Leg for a Galloping Machine”. Proceedings of the 2003 IEEE International Conference on Robotics & Automation, Taipei, Taiwan, September 14-19, 2003.

[25] S. Skaff, G. Kantor, D. Maiwand, A.A. Rizzi, “Inertial Navigation and Visual Line Following for a Dynamical Hexapod Robot”. Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Nevada, October, 2003.

[26] I. Poulakakis, J.A. Smith, M. Buehler, “On the Dynamics of Bounding and Extensions: Towards the Half-Bound and Gallop Gaits”. Adaptive Motion of Animals and Machines, pp 79-88, Springer Tokyo, 2006.

[27] J.A. Smith, I. Sharf, M. Trentini, “PAW: a Hybrid Wheeled-Leg Robot”. Proceedings of the 2006 International Conference on Robotics and Automation, Orlando, FL, USA, May 2006

[28] J.A. Smith, I. Sharf, M. Trentini, “Bounding Gait in a Hybrid Wheeled-Leg Robot”. Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, October 2006.

[29] J.A. Smith, "Galloping, Bounding and Wheeled-Leg Modes of Locomotion on Underactuated Quadrupedal Robots". Phd Thesis, McGill University, November 2006.

[30] I. Poulakakis, J. Smith, E. Papadopoulos, M. Buehler, "Opportunities for Adaptation and Learning in Dynamically Stable Legged Robots". Yale Workshop on Adaptive and Learning Systems, Center for Systems Science, Yale University, New Haven, CT, pp. 129-134, June, 2001.

[31] A. Billard, A.J. Ijspeert, "Biologically inspired neural controllers for motor control in a quadruped robot". Proceedings of the IEEE-INNS-ENNS Joint Conference on Neural Networks, Vol. 6, pp. 637-641, 2000.

[32] R. McNeill Alexander, "Principles of Animal Locomotion". Princeton University Press, 2003.

[33] T.A. McMahon, "The Role of Compliance in Mammalian Running Gaits". Journal of Experimental Biology, Vol. 115, Issue 1, pp. 263-282, 1985.

[34] D. Papadopoulos, M. Buehler, "Stable Running in a Quadruped Robot with Compliant Legs". Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA, April 2000.

[35] B. Barshan, H.F. Durrant-Whyte, "Inertial Navigation Systems for Mobile Robots". IEEE Transactions on Robotics and Automation, Vol.11, No. 3, June 1995.

[36] D. Papadopoulos, "Stable Running for a Quadruped Robot with Compliant Legs", M. Eng Thesis, McGill University, April 2000.

[37] Kuipers, J.B., "Quaternions and Rotation Sequences", Princeton University Press, Princeton, N.J., 1999.

[38] Arden, D., "Analysis of MEMS IMU Motion Table Testing", Contract Report, Defence R&D Canada, Ottawa, Canada, 2007.

[39] S.P.N. Singh, K.J. Waldron, "Motion Estimation by Optical Flow and Inertial Measurements for Dynamic Legged Locomotion". Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Edmonton, Canada, 2005.

[40] S. Talebi, M. Buehler, E. Papadopoulos, "Towards Dynamic Step Climbing for a Quadruped Robot with Compliant Legs". Compliant Legs, 3rd International Conference on Climbing and Walking Robots, 2000.