UNIVERSITY OF CALIFORNIA SAN DIEGO

Mechanical Design, Dynamic Modeling, State Estimation, and Feedback Control of a Micro-Ball-Balancing Robot at High Yaw Rates

A dissertation submitted in partial satisfaction of the requirements
for the degree Doctor of Philosophy

in

Engineering Science (Mechanical Engineering)

by

Eric Nauli Sihite

Committee in charge:

Professor Thomas Bewley, Chair
Professor Mauricio de'Oliveira
Professor Falko Kuester
Professor Michael Tolley
Professor Michael Yip

2019

The dissertation of Eric Nauli Sihite is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California San Diego

2019

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

MIP	Mobile Inverted Pendulum
MBBR	Micro Ball-Balancing Robot
SROW	Single-Row Omniwheel
DROW	Double-Row Omniwheel
KF	Kalman Filter
EKF	Extended Kalman Filter
DC	Direct Current
LQR	Linear Quadratic Regulator
SLC	Successive Loop Closure
Mocap	Motion Capture
RMS	Root Mean Squared
RMSE	Root Mean Squared Error
FFT	Fast Fourier Transform
IMU	Inertial Measurement Unit
DMP	Digital Motion Processor (in InvenSense MPU-9250)

LIST OF SYMBOLS

x	Regular letter font for scalars.
$\boldsymbol{x}$	Bolded letter font for vectors.
X	Capital letter font for matrices and some scalars.
$\dot{x}$	Dot above the variable for time derivatives.
t	Time.
m	Mass.
$\hat{I}$	Body reference inertia.
I_n	Identity matrix of size n.
r	Radius.
l	Length.
k	Motor constant.
R	Electrical resistance.
L	Electrical inductance.
V	Electrical voltage.
T	Period.
f	Frequency.
τ	Torque.
θ, ϕ, φ	Angles.
ω, Ω	Angular velocities.
p	Position.
g	Gravity constant.
μ_c, μ_v	Friction coefficient constants.

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The work in this dissertation is possible due to the help of my Principal Investigator Prof. Thomas Bewley, my colleagues Daniel Yang, Jeffrey Friesen and James Strawson, Clark Briggs from ATA Engineering, and Wowwee Robotics together with their manufacturing facilities. I also would like to thanks everyone involved in this project, including the other lab members and Professors that helped my research and taught me how to be a better scientist.

Wowwee Robotics has contributed greatly for this research by providing the most important components for the MBBR, which are the omniwheels, motors with gearboxes, and the motor mounts. They also helped with the motor selection and design improvements.

James Strawson developed the Robotics Cape and the Beaglebone Black library, both are used extensively for controlling all of the experiments I have done. MIP, MBBR, and the coasting DC motor experiment used the Beaglebone Black and the Robotics Cape, which makes James contribution extremely important for the work I have done for this dissertation.

Clark Briggs from ATA Engineering lent us the Optitrack motion capture system which was greatly utilized in the experiments done in Chapter 3 and 6.

Chapter 2 partially uses the material as it appears in the published paper in IEEE International Conference on Intelligent Robots and Systems (IROS) 2015, D. Yang and E. Sihite and M. Friesen and T. Bewley, "Design and control of a micro ball-balancing robot (MBBR) with orthogonal midlatitude omniwheel placement". The dissertation author was the co-author and investigator of this paper, contributed to the CAD design, controller design, and experimentation of the robot.

Chapter 3, in full, is a reprint of the material as it appears in the published paper in American Control Conference (ACC) 2018, E. Sihite and T. Bewley, "Attitude estimation of a high-yaw-rate Mobile Inverted Pendulum; comparison of Extended Kalman Filtering, Complementary Filtering, and motion capture". The dissertation author was the sole investigator

and author of this paper.

Chapter 4, in full, uses the material as it may appear in the paper submitted for publication to IEEE International Conference on Automation Science and Engineering (CASE) 2019, E. Sihite, D. Yang, and T. Bewley, "Derivation of a new drive/coast motor driver model for real-time brushed DC motor control, and validation on a MIP robot". The dissertation author was the primary investigator and author of this paper, contributed to the development of the novel model and control algorithm, experimentation, and data analysis.

Chapter 5 and 6 partially uses the material as it appears in the published paper in IEEE International Conference on Robotics and Automation (ICRA) 2019, E. Sihite, D. Yang, and T. Bewley, "Modeling and state estimation of a Micro Ball-balancing Robot using a high yaw-rate dynamic model and an Extended Kalman Filter". The dissertation author was the primary investigator and author of this paper, contributed to the estimator and controller design, experimentation, and data analysis.

VITA

2008	B.S. in Mechanical Engineering, University of Michigan Ann Arbor.
2010	M.S. in Mechanical Engineering, University of Michigan Ann Arbor.
2011 - 2012	Graduate Researcher, University of Michigan Ann Arbor.
2012 - 2019	Graduate Student Researcher, University of California San Diego.
2019	Ph.D. in Engineering Science (Mechanical Engineering), University of California San Diego.

PUBLICATIONS

D. Yang, E. Sihite, J. Friesen, T. Bewley. "Design and controls of a micro ball-balancing robot (MBBR) with orthogonal midlatitude omniwheel placement." *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

E. Sihite, T. Bewley. "Modeling and state estimation of a mobile inverted pendulum robot using Extended Kalman Filter under high yaw rate." *American Control Conference (ACC)*, 2018.

E. Sihite, D. Yang, T. Bewley. 'Modeling and state estimation of a Micro Ball-balancing Robot using a high yaw-rate dynamic model and an Extended Kalman Filter." *International Conference on Robotics and Automation (ICRA)*, 2019.

E. Sihite, D. Yang, T. Bewley. 'Derivation of a new drive/coast motor driver model for real-time brushed DC motor control, and validation on a MIP robot." *International Conference on Automation Science and Engineering (CASE)*, 2019. Accepted for publication.

T. Bewley, J. Friesen, E. Sihite, D. Yang. Ball-balancing robot and drive assembly therefor. *USA Patent No. US20180022197A1*. January 29, 2019.

ABSTRACT OF THE DISSERTATION

Mechanical Design, Dynamic Modeling, State Estimation, and Feedback Control of a Micro-Ball-Balancing Robot at High Yaw Rates

by

Eric Nauli Sihite

Doctor of Philosophy in Engineering Science (Mechanical Engineering)

University of California San Diego, 2019

Professor Thomas Bewley, Chair

A ball-balancing robot (BBR) is a robot that balances itself on top of a ball, typically using three omni-directional wheels. This class of robot features a highly coupled 3D nonlinear dynamics that is capable of natural and holonomic motion. This dissertation presents a new mechanical design, a tractable dynamic model that is accurate at high yaw rates, and an effective estimation and control strategy for a micro ball-balancing robot (MBBR). The miniaturization and the low-cost components used in this design add significant control challenges, which manifest in the form of reduced durability and high amounts of noise, friction, and vibration, making the design of effective state estimation and control strategies for it very difficult, especially under high yaw rates. This motivates the design and use of a reduced nonlinear model which

captures the important high yaw-rate dynamics well, and the design of an effective model-based observer and controller based on this reduced nonlinear model. The primary contributions of this dissertation in the general area of ball-balancing robotics include:

(1) the novel (and, now, patented) midlatitude and orthogonal-omniwheel orientation of the design,

(2) a reduced (minimum complexity) nonlinear BBR dynamic model which well captures its high yaw-rate behavior, and

(3) the development of an effective model-based estimator (Extended Kalman Filter) and controller which are capable of achieving remarkable performance of this delicate system under high yaw rates.

The novel omniwheel placement minimizes coupling and increases the normal force acting on the omniwheels, which helps to reduce the slipping caused by its very light body. Another contribution of this dissertation is:

(4) the modeling and real-time implementation of drive/coast motor drivers,

which is also used in the MBBR. Drive/coast motor drivers exhibit highly nonlinear behavior, which makes using them in a model-based controller difficult, but they allow for zero torque dynamics which can be quite useful for many wheeled robots. The drive/coast motor model has been implemented in our linear feedback controller, and has achieved remarkably good position tracking even under high yaw-rates. The performance of the observer and controller were verified with a motion capture system.

Chapter 1

Introduction

The robotics field has evolved significantly with the arrival of 3D printing technologies. Small scale robots can quickly be manufactured which allows for quick assembly and rapid prototyping. One example of such robots is the eduMIP (Educational Mobile Inverted Pendulum) robot developed in our Coordinated Robotics Labs in the University of California San Diego. This mobile inverted pendulum robot (MIP) robot was designed as an educational tool to teach undergraduate students control system theories. One of the initial designs of the eduMIP used a breadboard with off-the-shelf sensor and motor driver breakout boards, Arduino microcontroller, and 3D printed chassis. All of these parts were then assembled together with injection molded plastic mounts. The latest version of eduMIP is powered by a Beaglebone Blue, or a Beaglebone Black and Robotics Cape with injected molded plastic mounts. Our lab has also built several other robot prototypes using the rapid prototyping capabilities of the 3D printer. Some examples of these robots are the stair climbing MIP robot, a MIP robot that can pickup ping-pong balls and rapidly shoot them, and a hexacopter.

The Micro Ball-Balancing Robot (MBBR) was developed as a follow-up to the MIP projects. Ball-Balancing robots (BBRs) dynamics can be simplified into two decoupled MIP dynamics in the roll and pitch directions under the assumption of slow dynamics and trivial yaw rates. This class of robot allows for a fluid and organic holonomic movements which can serve as an interesting toy product when miniaturized. At the time of the project conception, many

existing BBRs are large or extremely large. The largest BBR is almost as tall as a person (150 cm) and one of the smaller BBRs are at least taller than a knee height (> 50 cm). Therefore, the design and fabrication of a miniaturized BBR is novel and, after our first few attempts, also proves to be very challenging. Using 3D printers, several prototypes of the MBBR was built using different types of driving mechanism, omniwheels type, orientations, and placements. Our MBBR design features a novel and patented ([1]) orthogonal midlatitude omniwheel placement. The midlatitude placement serves as a method to increase the normal force acting on the omniwheels without adding weight to the robot. The orthogonal omniwheel placement allows each omniwheels to rotate independently to each other, reducing the coupling torque and allow the wheels to spin more efficiently. The MBBR mechanical design and the novel omniwheel placement paper was published in the IEEE International Conference on Intelligent Robots and Systems (IROS) 2015 [2]. Within a year, several successful prototypes has been built which can balance and be driven slowly. However, the robot has a serious problem with stability when driven fast or starts spinning. Some significant problems have surfaced due to the miniaturization of the robot, primarily in the form of faster dynamics, large vibrations, noise, and friction. In addition to this, BBRs exhibit a highly coupled and nonlinear dynamics when subjected to nontrivial yaw rates. All of these problems cause stability issues when the robot starts to drive or spin quickly. Therefore, a good controller and state estimator that can work under high yaw rates and noisy measurements are required before this robot can show its true potential.

As a type of an inverted pendulum robot, MBBR shares similar dynamic behavior as a MIP robot. Therefore, testing the high yaw-rate modeling, estimation and controls on the MIP robot first is the natural step before implementing them in the MBBR. We developed the high yaw-rate model for the MIP robot and implemented in an Extended Kalman Filter (EKF). The accuracy of the attitude state estimation by EKF was verified by using motion capture. Several other state estimators were also compared, such as linear model Kalman Filter (KF), complementary filter, complementary KF, and the Digital Motion Processor (DMP) which is the proprietary sensor fusion algorithm native to the InvenSense MPU-9250 Inertial Measurement

Unit (IMU). The motion capture experiment showed that the attitude estimation of the high yaw-rate model is more accurate than the linear model KF and other estimators under high yaw rates, except the DMP which has comparable performance. However, the DMP only estimates the orientation, which is useful if the robot only need an accurate orientation estimates. The EKF has the advantage full system states estimation which is required for using a state feedback controller. The MIP high yaw-rate estimation paper was published in the IEEE American Control Conference (ACC) 2018 [3].

There is an additional problem which became apparent when attempting to model the motors for the MBBR. The DC motors are driven by using Pulse Width Modulation (PWM) through the motor drivers. There are two common types of motor drivers for brushed DC motors depending on how the current flows during the low PWM duty: drive/coast and drive/brake. The drive/brake driver shorts the motor terminals during the low PWM signal while the drive/coast driver opens all MOSFET gates which allows the current to flow through the flyback diodes back into the battery's positive terminal. Conveniently, the motors using a drive/brake driver can be modeled with a linear model. On the other hand, the drive/coast drivers exhibits a nonlinear behavior which is heavily influenced by the PWM frequency, rotor speed, and the motor inductance. The most recent MBBR prototype uses drive/coast motor drivers, which motivated us to investigate on the drive/coast motor model and a method to compensate the nonlinearities in real-time control applications. The drive/coast drivers have some advantages compared to the drive/brake motors; this type of drivers allows for the zero-input response and free-wheeling which might be advantageous for some wheeled robots. There might be a benefit of using drive/coast drivers from the energy efficiency perspective as the current flows regeneratively back into the battery during the low PWM duty cycle. More experiments must be done to back up this claim, which can be a part of the future work in this topic. A novel model for the drive/coast drivers which can be used for bi-directional (forward and reverse) motor control and a real-time compensation algorithm were successfully developed and tested on a MIP robot. The experiment used the controller designed for the linear motor model of a drive/brake driver in

a drive/coast driver. This control command was compensated by using the algorithm in real-time and has showed that the driving performance of the compensated drive/coast system is similar to the drive/brake system with the same controller. This indicates that the model is correct and we have successfully use the PWM command in a drive/coast driver which is equivalent to the drive/brake controller. The drive/coast modeling and compensation paper has been accepted for publication in the IEEE International Conference on Automation Science and Engineering (CASE) 2019 [4].

The high yaw-rate modeling, estimations, and the drive/coast compensation has been successfully done on a MIP robot. Then the next step is to implement the same methodology on the MBBR, which is significantly more challenging due to its mechanical and dynamical complexities. A high yaw-rate MBBR dynamical model was developed, implemented in an EKF, and verified by using motion capture. The MBBR high yaw-rate modeling and estimation paper was published in the IEEE International Conference on Robotics and Automation (ICRA) 2019 [5]. The modeling and implementation of the estimator faced even more challenge with the presence of high friction, omniwheel slipping, and other vibrations which manifest during the balancing. While the attitude estimations were relatively accurate, the ball speed estimation was highly inaccurate without using friction compensation in the controller. The ball speed estimation is important for the robot's position and speed tracking, which is also the main motivation of developing this robot: organic and fluid movement that can spin and translate at the same time. More extra work has been done to improve the estimation accuracy and the position tracking performance of the controller which includes the friction parameter system identification, using the robot's non-minimum phase dynamics to determine the attitude state reference, and gain scheduling based on the desired yaw rates. Numerical simulations have also been done using the full nonlinear model, EKF and the controller used in the actual MBBR to show that the simultaneous spinning and translation can be done by using a linear feedback controller. After the development of the finalized controller algorithm, several motion capture experiments were done to prove the estimation accuracy, the controller stability, and position tracking performance.

The contributions of this dissertation in the general area of ball-balancing robotics and controls are:

(1) the novel and patented midlatitude and orthogonal-omniwheel orientation,

(2) a minimum complexity nonlinear BBR dynamic model which well captures its high yaw-rate behavior,

(3) the development of an effective model-based estimator (Extended Kalman Filter) and controller which are capable of achieving remarkable performance of this delicate system under high yaw rates, and

(4) the modeling and real-time implementation of drive/coast motor drivers.

This dissertation is outlined as follows: MBBR mechanical design, MIP high yaw-rate model and estimations, drive/coast DC motor model and compensation, MBBR high yaw-rate model, and finally the MBBR control, estimation and motion capture experiment.

References

[1] E. Sihite, D. Yang, J. Friesen, and T. Bewley, "Ball-balancing robot and drive assembly therefor," *USA Patent No. US20180022197A1*, January 29, 2019.

[2] D. Yang, E. Sihite, M. Friesen, and T. Bewley, "Design and control of a micro ball-balancing robot (mbbr) with orthogonal midlatitude omniwheel placement," *IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pp. 4098–4104, 2015.

[3] E. Sihite and T. Bewley, "Attitude estimation of a high-yaw-rate mobile inverted pendulum; comparison of extended kalman filtering, complementary filtering, and motion capture," *American Control Conference (ACC)*, pp. 5831–5836, 2018.

[4] E. Sihite, D. Yang, and T. Bewley, "Derivation of a new drive/coast motor driver model for real-time brushed dc motor control, and validation on a mip robot," *IEEE International Conference on Automation Science and Engineering (CASE)*, 2019. In press.

[5] E. Sihite, D. Yang, and T. Bewley, "Modeling and state estimation of a micro ball-balancing robot using a high yaw-rate dynamic model and an extended kalman filter," *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2019.

Chapter 2

Micro Ball-Balancing Robot (MBBR) Design

A Ball-Balancing Robot (BBR) is a robot which balances itself on top of a ball, either using omni-directional wheels (omniwheels) or an inverse mouse-ball mechanism. This class of a robot has a holonomic motion and exhibits rich 3D dynamics and is capable of fluid and graceful motion. A BBR under a small perturbation has a similar dynamics to two independent Mobile Inverted Pendulum (MIP) robot problems in both the fore/aft and left/right directions. However, BBRs has a highly nonlinear dynamics when driving under nontrivial yaw-rates which proves to be a highly challenging controls problem.

The conceptual design of the Micro Ball-Balancing Robot (MBBR) began in 2013, where the robot was proposed as a continuation of the MIP and the eduMIP projects. As mentioned above, the dynamics of the BBR is similar to the MIP's dynamics under low perturbations, making it an ideal continuation of the MIP projects. The MBBR had undergone many iterations throughout the years, with varying body designs, DC motors, motor drivers, and omniwheels. The latest iteration of MBBR weighs 700g and is 25 cm tall, making it one of the, if not the smallest, BBRs in the world. The MBBR also costs less than USD$200 to build with potentially significant cost savings when mass-produced. Since the MBBR is much cheaper to manufacture, it is viable for commercial applications in entertainment, service, education, and research. Throughout all of the MBBR iterations, it was controlled by using the Beaglebone Black and a Robotics Cape

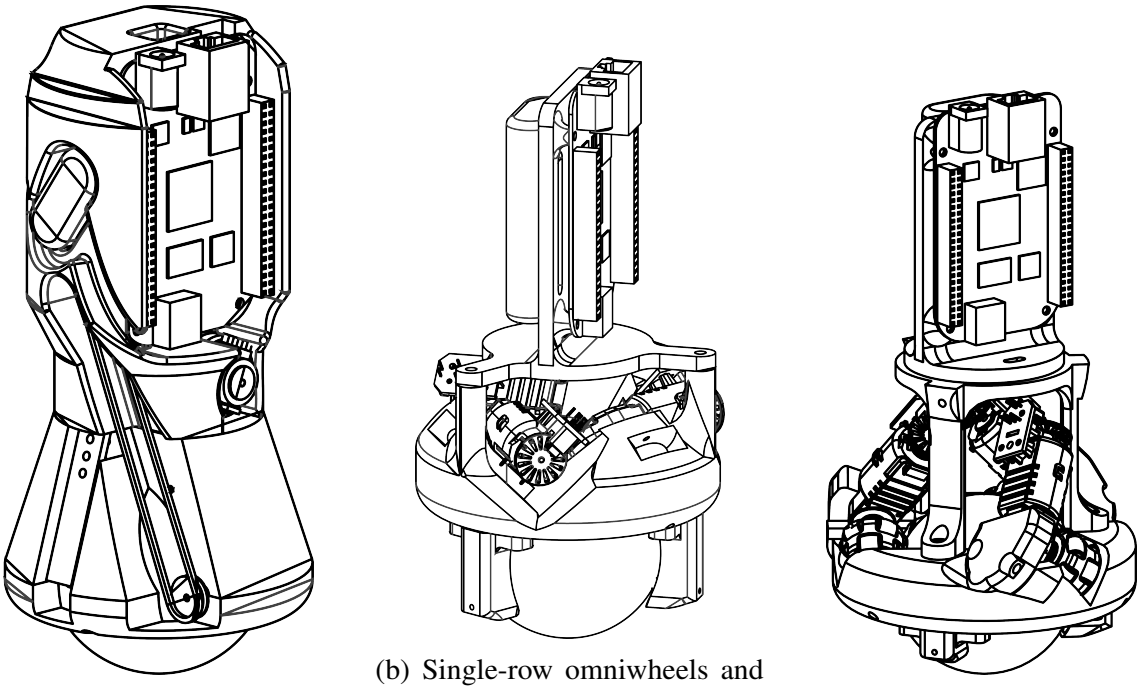
which was also developed at the same time in our Coordinated Robotics Labs. The MBBR design features a novel midlatitude placement and orthogonal omniwheels orientation. The midlatitude placement increases the normal forces applied on the omniwheels which is used to counteract the problem which arises from the robot's small weight. The orthogonal omniwheels orientation is designed to allow the omniwheels to rotate independently with each other and improve the efficiency of the actuation.

This chapter contents include the past works of the other existing BBRs, preliminary MBBR designs, the omniwheels design and placements, and the latest MBBR build which is used in the motion capture experiments in Chapter 6.

2.1 Past Works and Preliminary MBBR Designs

Early research on BBRs appeared around 2006 (see [1]). To date, most BBR research has focused on human-scale designs ([1]–[3]), ball-balancing transportation vehicles ([4]–[6]), and knee-high to waist-high designs ([7], [8]). These larger vehicles weigh between 8.7kg and 45kg, and are characterized by significantly higher build cost and slower dynamics than the designs considered in our MBBR.

There are significant challenges and limitations to contend with when attempting to miniaturize a BBR. Tolerances become more stringent and cross-sectional areas decrease, lowering the yield strength of the mechanical components. Scaling down the characteristic length scale, l, of a given design generally reduces the volume and mass of the design by l^3; on a BBR, this significantly reduces the normal force between the omniwheels and the ball, creating problems with slip. Simultaneously, the time scale of the nonminimum-phase inverted-pendulum dynamics of a BBR is proportional to $\sqrt{l}$, so as l is reduced, the time scale decreases and the actuators must respond faster, further exacerbating the slip problem. In extreme cases, the drive wheels may even lose contact with the ball completely for short periods of time. The preliminary designs in the MBBR attempted to address these issues, such as manufacturing tolerances, drivewheel



(a) Belt-driven inverse mouseball mechanism.

(b) Single-row omniwheels and conventional omniwheel placement.

(c) Double-row omniwheel and orthogonal omniwheel placement.

Figure 2.1: Three MBBR mechanisms considered during the preliminary design process.

slippage, and fast dynamics.

To begin the design process, we compiled an ordered list of design requirements: size, robustness, cost, and energy efficiency. With such an open design space, we began by examining existing large-scale designs which seemed fit for miniaturization. Previous work on BBRs fall into two main categories: those driven by an inverse mouseball mechanism, and those driven by three omniwheels. At first, neither category presented obvious barriers to miniaturization. The many design choices available, the inaccuracy of analytical friction models, and various unresolved questions related to manufacturing led to the conclusion that rapid test and iteration were essential to converge quickly to the best practical solution. We thus decided to pursue both the inverse mouseball mechanism and the omniwheel-driven mechanism in parallel. Using low-cost additive manufacturing techniques, we iterated quickly on various design changes, and built more than 10 different working prototypes, three of which are shown in Fig. 2.1, in a span of three months. We successfully balanced both archetypes of the MBBR, eventually

Figure 2.2: The inverse mouse-ball mechanism MBBR.

downselecting to the omniwheel-driven design, such as in Fig.2.1c.

2.2 Inverse Mouseball Mechanism MBBR

The inverse mouseball mechanism (Fig. 2.1a and 2.2) relies on two perpendicular rollers along the equator of the ball. A small, low-friction bearing is used at the top of the ball to support the weight of the upper body. Spring-loaded idler wheels at opposite points along the equator press the rollers against the ball to create enough friction to eliminate slip between the rollers and the ball. This allows the rollers to actuate the ball in the two horizontal directions independently. The inverse mouse-ball MBBR shown in Fig. 2.1a uses timing belts to actuate the rollers on the equator which add some complexity in the robot's actuation (e.g. belt tension, friction).

There were two key problems with MBBRs driven by this mechanism. The first was contamination. During testing, dirt and rubber particles would build up on the support bearing, idler wheels, and drive rollers. This would increase friction and degrade performance to the

point of failure. For large BBRs, the torques and forces involved are greater in magnitude, and tolerances are relaxed, so a thin layer of dirt has little effect on performance. However, for an MBBR, dirt buildup on the rotating components proved to be a critical point of failure. The second problem encountered was the ball being pushed out of the socket during maneuvers. The rollers transmit torque to the ball by applying a friction force along the ball's equator. Depending on the direction of rotation, this force tends either to push the ball farther into the socket, or to pull the ball away from the socket. The only force preventing the ball from leaving the socket is the weight of the robot itself. Since MBBRs have reduced mass, their weight is generally insufficient to keep the ball in the socket. Even if the ball does not leave the socket, the design suffers from asymmetric friction. As the roller actuates the ball in one direction, the ball is forced into the socket, increasing both the normal force and the friction at the top support bearing; when actuating the ball in the opposite direction, the friction at the top support bearing is reduced. Additionally, the inverse mouseball mechanism does not control the yaw of the robot about its vertical axis, so additional actuators would be needed to make the robot face in a desired direction. Weighing all of these considerations, we ultimately concluded that the inverse mouseball mechanism, though feasible, was the lesser of the two available choices.

2.3 Omniwheel-Driven MBBR

Several omniwheel-driven MBBR prototypes were developed using varying omniwheel types and configurations. The omniwheel can be categorized into two types depending on the number of rows of the rollers: single-row omniwheel (SROW) and double-row omniwheel (DROW). Varying omniwheel midlatitude placements and orthogonal orientations were tested and compared to the traditional omniwheel placements. After some experimentation on these on the omniwheels type and configurations, we finally settled on the MBBR prototype seen in Fig. 2.3 for IROS 2015 [9]. The design was further improved in the latest prototype by using a more optimized DROW, appropriate motor, motor driver and gearbox selections.

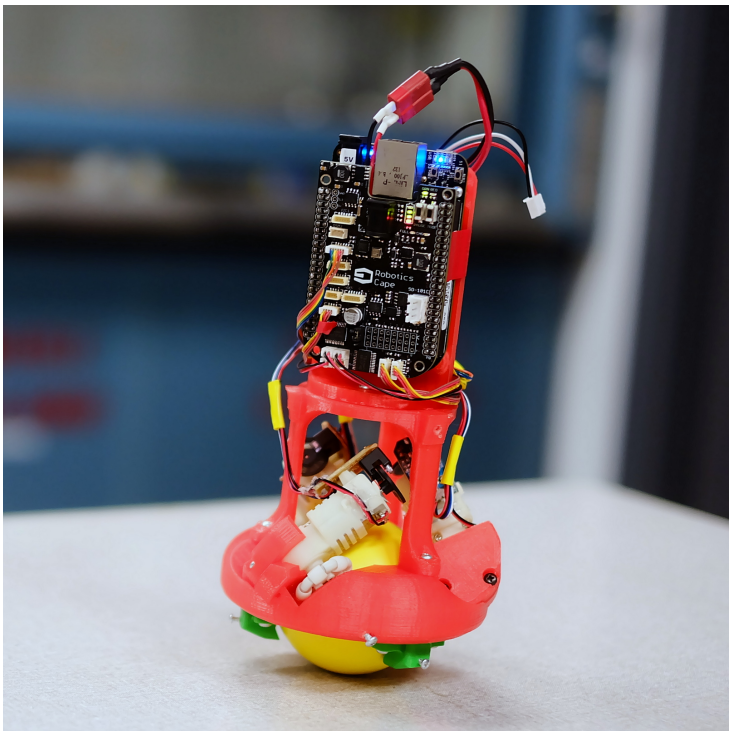


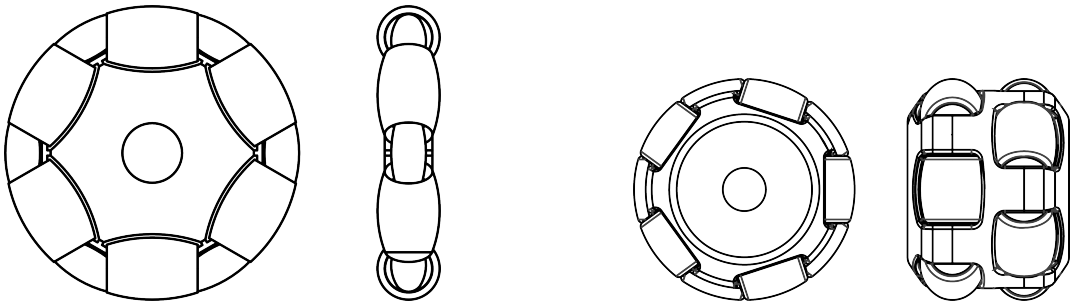
Figure 2.3: SROW MBBR prototype for IROS 2015.

This section discusses the following topics: SROW and DROW omniwheel design, midlatitude and orthogonal omniwheel placements, motor selections, and the latest optimized DROW MBBR prototypes. The discussion is structured to show the progress from the 2015 to the latest 2018 MBBR prototype.

2.3.1 Omniwheel Design

The most limiting factor for the omniwheel-driven MBBR is the omniwheels themselves. An ideal omniwheel (see Fig. 2.4) has:

- Zero friction in the direction of the wheel axis, which is achieved using rollers around its circumference,
- Zero friction resisting yaw about the contact patch, and
- Zero slip (infinite stiction) in its direction of rotation.



(a) Single-row omniwheel (SROW). (b) Double-row omniwheel (DROW).

Figure 2.4: Two primary types of omniwheels.

The smallest off-the-shelf omniwheels found were too large, at 4cm in diameter. We thus custom fabricated smaller omniwheels. Due to manufacturing tolerances and material strength limitations, the smallest omniwheels we could reliably manufacture were 2.5cm in diameter. Two omniwheel designs were selected for testing (see Fig. 2.4): the single-row omniwheel (SROW) design found in [8], [10], and the more conventional double-row omniwheel (DROW) design.

The SROW design considered (Fig. 2.4a) consists of 12 rollers lying in the same plane. The rollers alternate between large and small, allowing the smaller rollers to nest within the larger rollers to form an nearly circular profile [11]. In practice, SROWs allowed for very smooth actuation of the ball, due to the absence of gaps between rollers and the single contact point on the ball. Several drawbacks were also encountered. First, the design complexity is high, with 37 parts per wheel, which increases the manufacturing cost significantly. Additionally, parts of the hub had to be made as thin as 1mm, which were prone to mechanical failures.

The DROW design considered (Fig. 2.4b) consists of 12 equally-sized rollers in two parallel planes, which greatly simplifies construction. Although the part count is still relatively high, the individual pieces are larger and more robust. The drawback of the DROW design lies in the differences in the engagement of the two different rows on the curved surface of the ball. On flat surfaces, a DROW behaves much like a SROW. However, in MBBRs, as the ratio between the ball diameter and the distance between the two rows of rollers becomes smaller, the omniwheels induce a characteristic “wobble” into the dynamics, as derived below, that is nearly

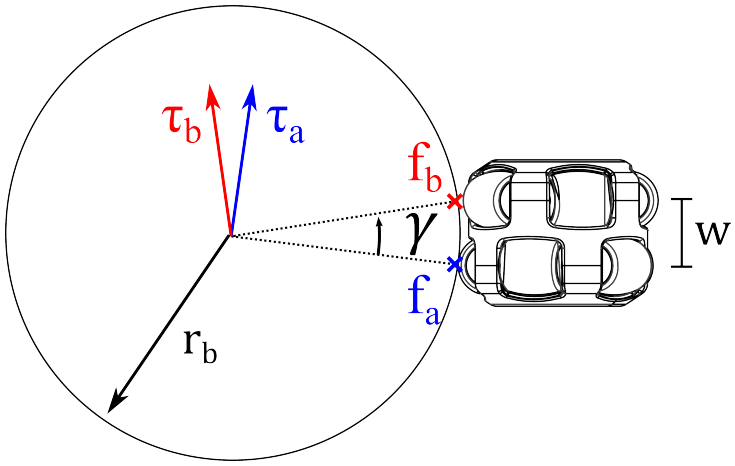


Figure 2.5: Double row omniwheel and ball contact interaction.

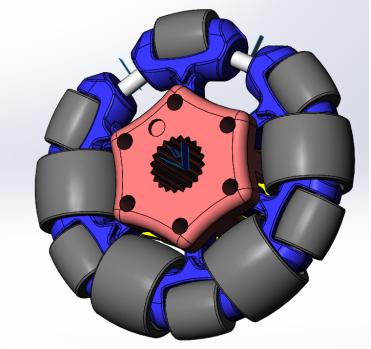
impossible to eliminate.

As illustrated in Fig. 2.5, as the DROW rotates, the normal force f_a generated by the DROW on the ball creates (via stiction) a torque on the ball, τ_a, in a direction perpendicular to both f_a and the azimuthal direction of the omniwheel. As the omniwheel continues to rotate, the second roller contacts the ball, resulting in a normal force f_b and concomitant torque on the ball, τ_b. It is thus seen that the torque that the DROW generates on the ball switches back and forth between between τ_a and τ_b. This switching induces a periodic disturbance which leads to the undesirable "wobble". The angle between these two axes of rotation is given by

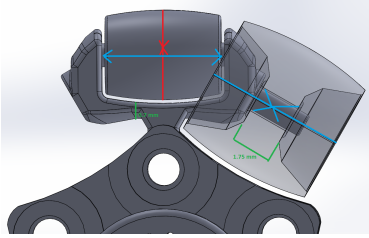
$$\sin(\gamma) = w/(2r_b). \tag{2.1}$$

As the ratio between the radius of the ball and the distance between the two rows of rollers in the DROW increases, the wobble diminishes. In practice, we found both the SROW and DROW designs viable for miniaturization, though neither was perfect.

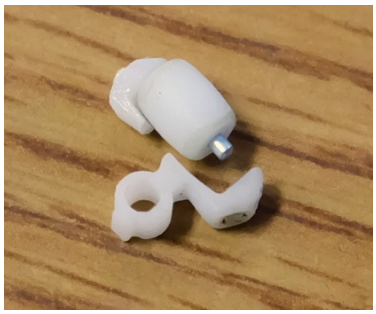
The SROW that is used in the MBBR can be seen in Fig. 2.6a. The rollers are all placed in one plane, using six small and six large rollers alternating between each other. How the rollers are setup can be seen in more detail in Fig. 2.6b. Both rollers are setup to minimize the gap between the rollers, making the ball transition between the rollers very smoothly. However, this



(a) SROW CAD model.



(b) SROW roller details.



(c) SROW mechanical failure.

Figure 2.6: The SROW CAD model, details, and mechanical failure.

design has a significant mechanical weakness in the part that's holding the small roller. This part is very prone to break under high torques at its smallest cross-section, as shown in Fig. 2.6c. The mechanical failure is very likely to occur when the SROW MBBR is being driven aggressively. Figure 2.7 shows the FEA analysis on the most vulnerable part, with the largest von Mises stress of 35 MPa under 0.05 N.m torque. This stress is very close to the tensile strength of the SROW material (PVC), which is 52 MPa. As the robot is being driven around, we believe that plastic deformation occurs and this lead to failure after several high stress cycles. This is torque is chosen because it is the static torque of the robot tipping at around 30 degrees, which is generally the nominal torque during balancing. The motor itself has a maximum stall torque of 0.176 N.m and can apply torques more than 0.05 N.m during aggressive movements. Ultimately, despite the SROW's advantages of lower vibrations, the part is not reliable for aggressive maneuvering. The SROW is much more feasible when it's scaled up in size, however our robot's demand of small omniwheel size does not allow that option.

As SROW can't be improved further for MBBR, we put our attention into optimizing the DROW to minimize slips and vibrations from the ball's transition across the alternating rows of wheels. Similarly to the SROW design, the rollers in the optimized DROW, shown in Fig. 2.8, are designed to minimize the gap between the rollers. Unlike the SROW design, this optimized DROW design does not have a structural weakness and the smoother roller transition results in

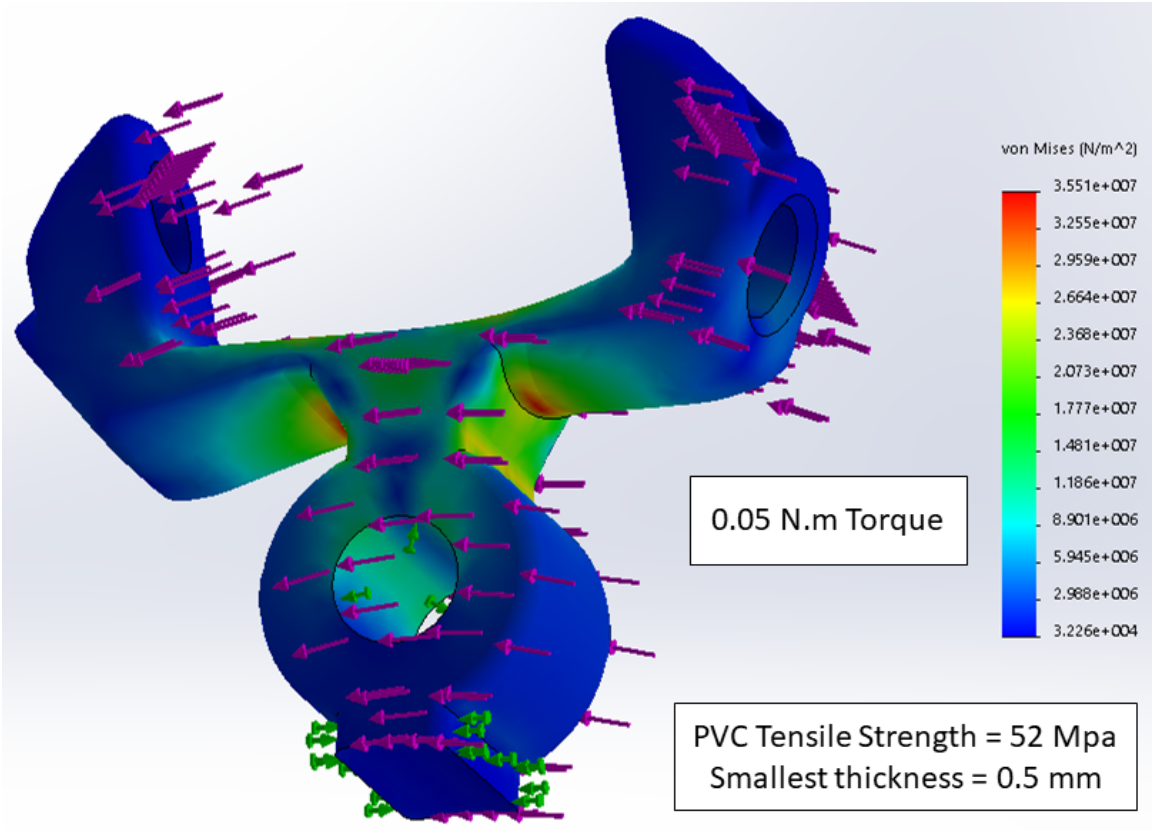
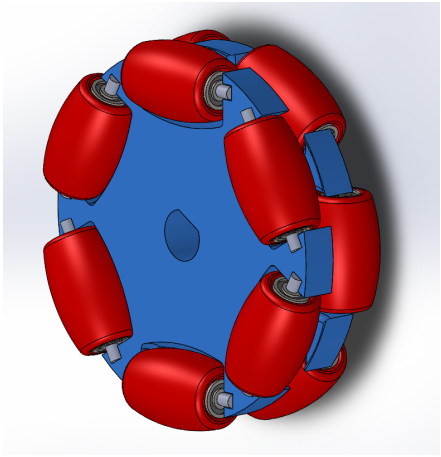
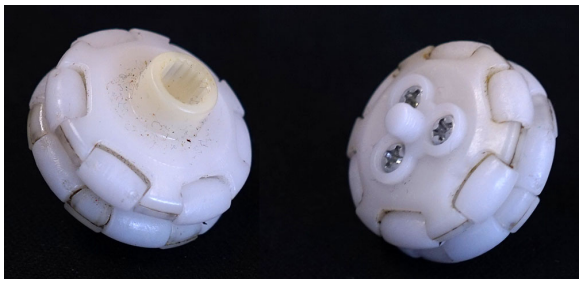


Figure 2.7: Stress analysis on the SROW most vulnerable component.

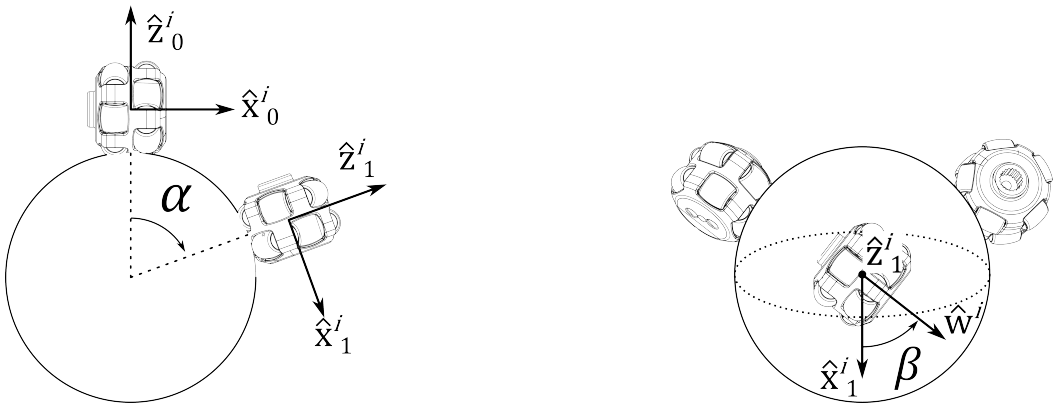


(a) DROW CAD model.



(b) Photo of the DROW.

Figure 2.8: Optimized DROW design.



(a) Angle from the North Pole

(b) Angle of twist

Figure 2.9: Omniwheel placement and orientation.

less vibrations in the system during the balancing. This optimized DROW is the omniwheel of choice for our latest MBBR build purely due to its robustness.

Both omniwheel designs have advantages and disadvantages. The SROW design has smoother operation but more fragile than its DROW counterpart. We have not yet found a good solution for mass producing both omniwheel designs, but a mass produced DROW design seems more likely to be found compared to SROW's. Mass production and cost cutting can be the topics for future research and development.

2.3.2 Omniwheel Midlatitude and Orthogonal Placement

Following the work of others (see §2.1), our early omniwheel-driven MBBR prototypes, as seen in Fig. 2.1b, placed the omniwheels between $\alpha = 30°$ and $\alpha = 45°$, and took $\beta = 0°$ (see Fig. 2.9). With the omniwheels in such a position, we found that slippage between DROW and the ball was significant. To mitigate this problem, the friction between the wheel and the ball must be increased. There are two ways to accomplish this: increase the coefficient of friction between the ball and rollers, or increase the normal force between the omniwheels and the ball. Since there are limited options in materials, and it is undesirable to increase normal force by adding mass to the main body, we sought other solutions to this problem. We found that placing

omniwheels closer to the equator of the ball (that is, in the "midlatitudes"), could greatly increase the normal force between the ball and omniwheels. It is easily seen that the normal force is equal to:

$$N = mg/[3\cos(\alpha)] \tag{2.2}$$

The normal force monotonically increases as α increases from near 0° to near 90°. Thus, by moving the omniwheels from $\alpha = 45^\circ$ to 70°, we can double the normal force between the omniwheels without adding any mass. This simple geometric change significantly mitigated the omniwheel slippage issues. Note that, if α becomes too large, the amplified normal forces increase friction in the drivetrain, and degrades performance. We built several prototypes, with $\alpha = \{45^\circ, 60^\circ, 70^\circ, 80^\circ\}$, and found that, for our design, $\alpha = 60^\circ$ to 70° represented a good compromise.

The lower placement of the omniwheels also effectively solved the problem of the omniwheels losing contact with the ball during quick maneuvers. Lowering the omniwheels result in larger normal forces, which allows the omniwheels to grip the ball better, and the contact points upon which the upper body rests are spaced farther apart, requiring a larger moment to tip the upper body off the ball.

There is a significant drawback to lowering the omniwheel placement if the orientation of the omniwheels is left in the conventional orientation with $\beta = 0$. The magnitude of the component of the torque which contributes to the yaw of the ball about the vertical axis scales with $\sin(\alpha)$. Conversely, the magnitude of the components of torque which contribute to translational movement of the ball scales with $\cos(\alpha)$. Thus, as α approaches 90°, the actuator input corresponding to balancing the vehicle approaches 0, and balancing becomes impossible. A natural solution to this problem is achieved by varying β, as discussed below.

We now consider the issue of the omniwheel orientation β. Our solution of mounting the omniwheels in mutually-orthogonal planes increases the overall efficiency of the power transfer and, as discussed above, is desirable when used in conjunction with midlatitude omniwheel

placement. Traditionally (see, e.g., [12]), omniwheels are spaced evenly apart, with 120° separation, and oriented in a radially-symmetric fashion about the z-axis, with $\beta = 0°$. By aligning the three omniwheels in mutually-orthogonal planes, we effectively decouple their effects.

If the omniwheels are *not* in mutually orthogonal planes, actuating one while holding the other two fixed results in both rolling of the ball about an axis τ parallel to that of the actuated omniwheel, and rolling about the other two axes, which gives no slip in the direction of rotation of the other two omniwheels. This is achieved by spinning the rollers of all three omniwheels, including the actuated omniwheel.

If, on the other hand, the omniwheels *are* in mutually orthogonal planes, actuating one while holding the other two fixed results in pure rolling of the ball about an axis parallel to that of the actuated omniwheel, and zero rolling about the other two axes. This is more direct and efficient, as it doesn't result in the spinning of the roller of the actuated omniwheel.

Our method to enforce orthogonal omniwheel orientation follows: let the rotational axis of the omniwheel $i \in \{1, 2, 3\}$ be represented by the unit vector $\hat{\boldsymbol{w}}^i \in \mathbb{R}^3$. The omniwheels are orthogonal to each other if the vectors $\hat{\boldsymbol{w}}^1$, $\hat{\boldsymbol{w}}^2$, $\hat{\boldsymbol{w}}^3$ are mutually orthogonal. This may be achieved by calculating the $\hat{\boldsymbol{w}}^i$ given the α and β, then checking if $\hat{\boldsymbol{w}}^i \cdot \hat{\boldsymbol{w}}^j = 0$ for $i \neq j$, $i, j \in \{1, 2, 3\}$. The $\hat{\boldsymbol{w}}^i$ can be determined using the following equations:

$$\hat{\boldsymbol{z}}_1^1 = \mathrm{Rot}(\hat{\boldsymbol{y}}_0^1, \alpha)\, \hat{\boldsymbol{z}}_0^1, \tag{2.3}$$

$$\hat{\boldsymbol{w}}^1 = \mathrm{Rot}(\hat{\boldsymbol{z}}_1^1, \beta)\, \mathrm{Rot}(\hat{\boldsymbol{y}}_0^1, \alpha)\, \hat{\boldsymbol{x}}_0^1, \tag{2.4}$$

$$\hat{\boldsymbol{w}}^i = \mathrm{Rot}(\hat{\boldsymbol{e}}^3, 2\pi/3)^{i-1}\, \hat{\boldsymbol{w}}^1, \quad i \in \{2, 3\}, \tag{2.5}$$

where $\mathrm{Rot}(\hat{\boldsymbol{x}}, \alpha)$ is the Euler rotation matrix about $\hat{\boldsymbol{x}}$, and $\{\hat{\boldsymbol{x}}_0^i, \hat{\boldsymbol{y}}_0^i, \hat{\boldsymbol{z}}_0^i\}$ are the initial wheel i axis coordinates, which are set up as shown in Fig. 2.9. In particular, our prototype is set up such that the initial coordinates for the first wheel are the same as lab coordinates $\{\hat{\boldsymbol{e}}^1, \hat{\boldsymbol{e}}^2, \hat{\boldsymbol{e}}^3\}$, where $\hat{\boldsymbol{e}}^1 = [1, 0, 0]^T$, $\hat{\boldsymbol{e}}^2 = [0, 1, 0]^T$, and $\hat{\boldsymbol{e}}^3 = [0, 0, 1]^T$. Given a particular value for α, we can

Table 2.1: Some orthogonal omniwheel orientations.

α	$45°$	$60°$	$70°$	$80°$
β	$35.237°$	$48.186°$	$52.082°$	$54.145°$

determine the β which gives us $\hat{\boldsymbol{w}}^i \cdot \hat{\boldsymbol{w}}^j = 0$ for $i \neq j$ using an iterative method. Some of the $\{\alpha, \beta\}$ pairs resulting in an orthogonal omniwheel orientation are given in Table 2.1.

As discussed previously, there are multiple benefits of such an orthogonal omniwheel orientation. First, the conventional omniwheel orientation at large α angles leads to diminished torque available to translate the ball. Further, the orthogonal omniwheel orientation leads directly to rotation of the ball in the actuated direction; actuation of a single omniwheel doesn't result in a force applied against the other two omniwheels in a manner that results in the spinning of the rollers of the actuated omniwheel, therefore providing a smoother application of torque.

2.3.3 Motor and Gearbox Selection

The motor and gearbox combination of the MBBR need to be carefully selected in order to achieve the best performance under our operating conditions. The MBBR is designed with the goal to develop a low-cost balancing robot that can drive and spinning fast at the same time. Small brushed DC motors are used due to their low cost and ease of use. An angled gearbox is used for the motors because the angled gearbox can be swiveled in such a way that it minimizes the robot's width. This makes the robot "slimmer", reducing the inertia about the robot's $\hat{\boldsymbol{z}}$ axis in addition to some aesthetics benefits. It is necessary for the motors to provide adequate torque under high rotor speed to achieve adequate performance. If the gear ratio is too high, the motor lose its maximum torque and may not have enough torque when the motor is spinning at high speed. Therefore, it is necessary to do some preliminary analysis in order to select the correct gearbox for our motor. The motor and gear ratio combination used in the IROS 2015 prototype are not adequate for high speed maneuvers, and this problem has been resolved in the latest MBBR prototype.

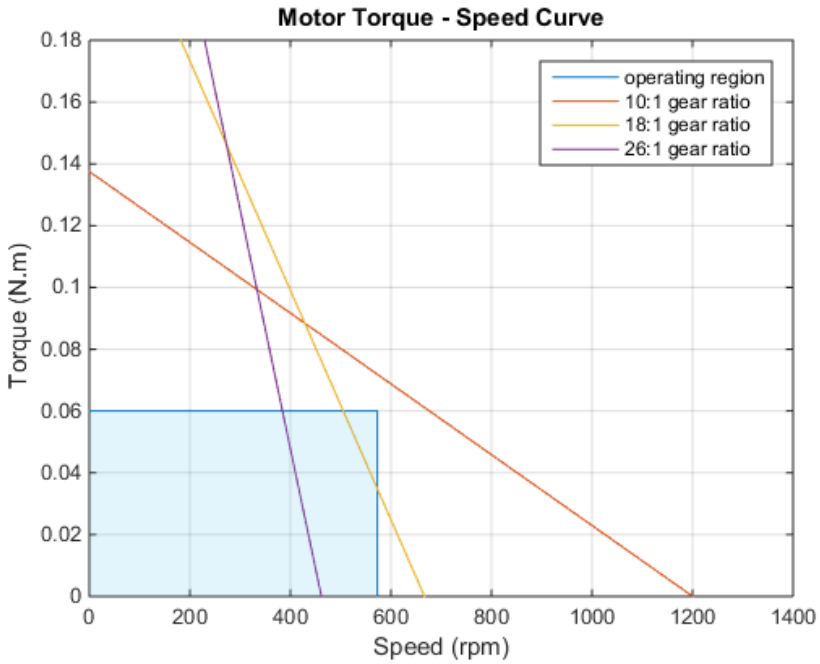


Figure 2.10: The motor torque vs. speed curve of the MBBR motor.

The torque vs. speed curve of the motor at nominal battery voltage of 7.4V with several gear ratios is shown in Fig. 2.10. The bottom left square in Fig. 2.10 is the estimated motor operating region, with a torque of 0.06 N.m and motor speed of 600 rpm. The torque of 0.06 N.m is generously estimated from the torque required to counter the gravity when the robot is tilted at 30 degrees. The 600 rpm speed is the estimated motor speed to drive the ball forward at the speed of 4 rotation per second, or approximately linear translational speed of 1.5 m/s. These numbers are rough approximation and we simply assume that the MBBR primarily operates under these torque and speed range. Using Fig. 2.10 as a guide, we finally decided to use gearbox with 12:1 gear ratio for our latest MBBR prototypes. The experiments under high yaw-rates and translational speed has shown that this motor and gearbox combination works very well and can achieve our target performance for the MBBR.

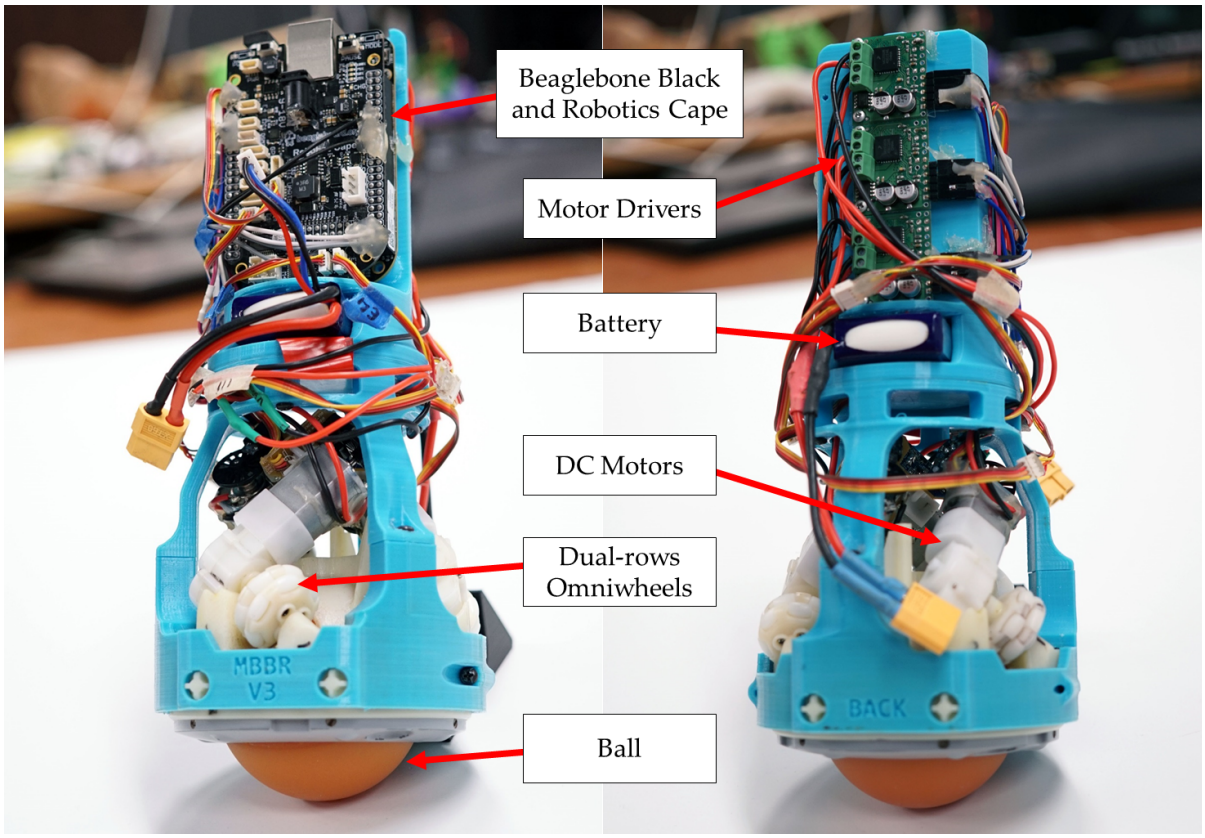


Figure 2.11: The latest MBBR prototype (2018) and its components.

2.3.4 The Most Recent MBBR Build

The MBBR has gone through several iterations from the first design paper in 2015. The latest build which was completed in 2018, as shown in Fig. 2.11, is the MBBR used in the high yaw-rate model and experiments in Chapter 5 and 6. This design has been proven to fulfill our design criterion of a low-cost BBR that can translate and spin fast at the same time. The optimized DROWs and improved construction tolerance have reduced some of the vibration and friction in the system, but these problems are still nontrivial which require robust controller and estimator to balance well. Design improvements which address the vibration issue can be a part of the future work.

This design uses $\alpha = 60°$ and the corresponding β angle shown in Table 2.1 for the orthogonal midlatitude placement. As mentioned in §2.3.1 and §2.3.3, the latest MBBR prototype

used optimized DROWs and high powered brushed DC motors with 12:1 gear ratio. The robot is also controlled by using a Linux embedded systems using Beaglebone Black and the Robotics Cape like the previous iterations. However, it used external motor drivers due to the increase in the motor current demands. This robot uses a different motor driver than the one used in the IROS 2015 prototype: the drive/coast motor drivers. This motor driver exhibits a nonlinear behavior which will be explained in further details in Chapter 4. The motors, gearboxes, the motor housing, and the ball were provided by Wowwee Robotics. The bill of material of the latest MBBR build is listed below:

- Three motors, CL-FF180SH-1885V-50, rated 6V.
- 2 Channel optical encoders (15 slits) on each motor.
- Beaglebone Black.
- Robotics Cape.
- MC33926 coasting motor drivers for each motor.
- 2.5 inch Ball.
- Six support balls inside the bottom cover.
- 3D printed chassis and PVC motor mounts.
- 2 cells LiPo battery, 1300 mAh 50C.
- JY-MCU HC-06 Bluetooth dongle.

Acknowledgments

Wowwee Robotics has contributed greatly for this research by providing the most important components for the MBBR, which are the omniwheels, motors with gearboxes, and the motor mounts. They also helped with the motor selection and design improvements.

James Strawson developed the Robotics Cape and the Beaglebone Black library, both are used extensively for controlling all of the experiments I have done. MIP, MBBR, and the coasting DC motor experiment used the Beaglebone Black and the Robotics Cape, which makes James contribution extremely important for the work I have done for this dissertation.

Chapter 2 partially uses the material as it appears in the published paper in IEEE International Conference on Intelligent Robots and Systems (IROS) 2015, D. Yang and E. Sihite and M. Friesen and T. Bewley, "Design and control of a micro ball-balancing robot (MBBR) with orthogonal midlatitude omniwheel placement". The dissertation author was the co-author and investigator of this paper, contributed to the CAD design, controller design, and experimentation of the robot.

References

[1] T. B. Lauwers, G. A. Kantor, and R. Hollis, “A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive,” *IEEE Int’l Conf. on Robotics and Automation*, pp. 2884–2889, 2006.

[2] U. Nagarajan, G. Kantor, and R. Hollis, “The ballbot: An omnidirectional balancing mobile robot,” *The Int’l Journal of Robotics Research*, 2013.

[3] A. Lotfiani, M. Keshmiri, and M. Danesh, “Dynamic analysis and control synthesis of a spherical wheeled robot (ballbot),” *RSI/ISM Int’l Conf. on Robotics and Mechatronics*, 2013.

[4] C. Tsai, C. Chan, and L. Kuo, “Lqr motion control of a ball-riding robot,” *IEEE / ASME Int’l Conf. on Advanced Intelligent Mechatronics*, 2012.

[5] T. Hoshino, S. Yokota, and T.Chino, “Omniride: A personal vehicle with 3 dof mobility,” *Int’l Conf. on Control, Automation, Robotics and Embedded Systems*, 2013.

[6] C. Souleymane, C. Tsai, and Y. Chiu, “Self-balancing control using wavelet fuzzy cmac for uncertain omnidirectional ball-driven vehicles,” *IEEE Int’l Conf. on Fuzzy Theory and Its Applications*, 2013.

[7] M. Akbari, H. Z. Kheibari, and A. S. M. Nejad, “Timing belt gearbox in ballbot robot,” *RSI/ISM Int’l Conf. on Robotics and Mechatronics*, 2013.

[8] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, "Unified state estimation for a ballbot," *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pp. 2471–2476, 2013.

[9] D. Yang, E. Sihite, M. Friesen, and T. Bewley, "Design and control of a micro ball-balancing robot (mbbr) with orthogonal midlatitude omniwheel placement," *IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pp. 4098–4104, 2015.

[10] M. Kumagai and T. Ochiai, "Development of a robot balanced on a ball - first report, implementation of the robot and basic control," *Journal of Robotics and Mechatronics*, vol. 22, no. 3, pp. 348–355, 2010.

[11] K.-S. Byun and J.-B. Song, "Design and construction of continuous alternate wheels for an omnidirectional mobile robot," *Journal of Robotic Systems 20.9*, pp. 569–579, 2003.

[12] M. J. Bjärenstam and M. Lennartsson, "Development of a ball balancing robot with omni wheels," Master's thesis, Lund University, Lund, Sweden, 2012.

Chapter 3

MIP Dynamic Modeling and State Estimation

The dynamics of a BBR is similar to a MIP under trivial yaw rates. However, as the yaw rate increases, the dynamics of a BBR becomes significantly nonlinear and coupled. A similar effect can also be observed in the MIP dynamics, where its dynamics also becomes nonlinear as the yaw rate increases. Compared to a BBR, a MIP robot has a simpler dynamics even under high yaw rates, and a simple actuation mechanism that has less noise and points of failure. Therefore, a comprehensive study on the effect of high yaw rates on a MIP robot is a great stepping stone before attempting to control the BBR under a similar condition. Developing a nonlinear controller is very difficult compared to nonlinear estimators. The latter can be done by implementing an Extended Kalman Filter (EKF), and can be designed separately from the controller itself. By intuition, a well designed EKF using a more accurate dynamic model will have a better estimation accuracy compared to the estimators using the simplified linear model. This motivated us to study on the accuracy of commonly used MIP estimators compared to an EKF with a high yaw-rate MIP model. By showing that the high yaw-rate model EKF has a better accuracy than the other linear estimators, we might be able to use a similar methodology on the BBR to improve its estimation accuracy as well. This will be discussed in more detail in Chapter 5.

This chapter explores the accuracy of several state estimators used in MIP robots. Accu-

rate state estimation is essential for effective feedback stabilization of such vehicles, especially at high yaw-rates. The MIP estimation techniques compared in this section are the Complementary Filter, the Complementary Kalman Filter, the (proprietary) Digital Motion Processor (DMP) from the (common) TDK InvenSense MPU-9250, and a dynamically modelled EKF. The dynamics of MIPs undergoing high yaw rates, as used by the EKF, are derived by using a Lagrangian dynamics formulation. The MIP was then controlled at several different yaw rate setpoints, and the tilt angle estimates were compared with the ("ground truth") measurements obtained via motion capture. Our test results indicate that the high yaw rate dynamic EKF and DMP are significantly more accurate than the usual Complementary Filter and planar dynamic EKF. The inaccuracy of the Complementary Filter is likely caused by the IMU not being aligned with the body's center of mass, creating a significant centrifugal force while spinning quickly.

3.1 Past Works

A MIP robot is a feedback-stabilized inverted pendulum that is rigidly mounted to two individually-controllable coaxial wheels. Many groups have designed and stabilized MIP robots, using techniques ranging from PID to State-Space Control. In general, performance of the feedback stabilization algorithm implemented is limited by the estimator, which is very challenging in a small embedded systems with low-cost processors and sensors. In addition, small light robots can accelerate quickly and operate on fast timescales, further complicating their estimation.

In order to overcome these challenges, groups have employed a variety of estimation approaches, such as complementary filters and Kalman Filters (KF) [1][2]. However, many of these solutions obtain body angle estimates by treating the accelerometer and gyroscope as a simple inclinometer [1][2]. While a few references have considered body dynamics during the state estimation process, they only used the gyroscope and encoders as measurements. Further, all prior models for estimation that we could find in the literature were based on planar

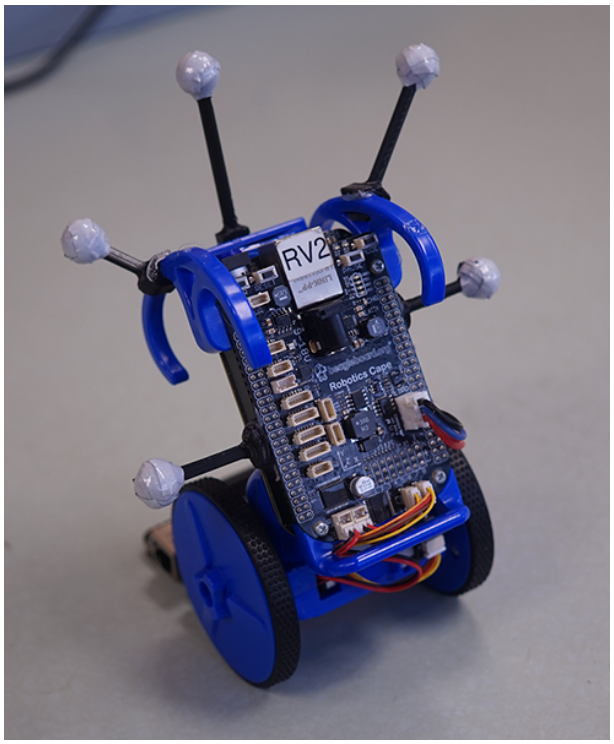


Figure 3.1: The eduMIP with motion capture markers attached.

MIP dynamics, under the assumption that any yaw motions of the MIP were decoupled from the longitudinal dynamics. High-performance MIP (and, ultimately, ball-balancing) robots [3][4], undergoing high yaw rates, are not accurately modeled under this assumption. Smooth stabilization of a very aggressive maneuvers of small MIP like this, or a small ball-balancing robot with complex dynamics, will likely be dependant on an accurate situational awareness provided by such state estimators. This motivates the present investigation, which aims to develop improved state estimates by reconciling with the raw sensor measurements with the dynamic equation of motion of the vehicle itself.

In this chapter, we present a 3D model for a MIP undergoing fast yaw dynamics, and introduce an EKF for state estimation using both the accelerometer and gyroscope measurements. In addition, using the (commercially-available) Renaissance Robotics eduMIP depicted in Fig. 3.1 as a test platform and a motion capture system to capture "ground truth", we compared our new yaw model and estimator to three existing methods: the complementary filter, the

complementary Kalman Filter, and the planar MIP dynamic Kalman Filter. The most difficult and important state to estimate on a MIP is the body tilt angle θ. The following subsections describe and derive the existing state estimators for θ.

3.1.1 Complementary Filter with Kalman Filter

Complementary Filtering can also be done using Kalman Filter [5]. This filter only uses the measurement from the IMU into the Kalman Filter in order to estimate θ [1][2]. This Complementary Kalman Filter setup is shown in Eq. 3.1, where $y_k^g = \omega_k + b_k$ is the gyro measurement, ω is the body's true angular velocity and b is the sensor bias. $\theta_k = \theta_{k-1} + \omega_k\, dt$ and y_k^a is the inclinometer angle estimate from the accelerometer measurements, $\boldsymbol{v}_k$ and w_k are zero mean white noise.

$$\begin{aligned} \begin{bmatrix} \theta_k \\ b_k \end{bmatrix} &= \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_{k-1} \\ b_{k-1} \end{bmatrix} + \begin{bmatrix} dt \\ 0 \end{bmatrix} y_{k-1}^g + \boldsymbol{v}_k \\ y_k^a &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_k \\ b_k \end{bmatrix} + w_k \end{aligned} \tag{3.1}$$

There are several similar estimation methods that do not use a dynamic model. For example, the state can be estimated using an Indirect Kalman Filter with a kinematic model, which was used on the high performance ball-balancing robot Rezero[4]. There are also other IMU sensor fusion algorithm that are used to estimate the orientation, such as for UAV and wearable sensors [6][7][8]. In addition, the TDK InvenSense MPU-9250, which is the IMU used in our MIP, has a proprietary Digital Motion Processing (DMP) algorithm which calculates a very accurate angle estimate without a dynamic model.

3.1.2 Dynamic Kalman Filter

The Extended Kalman Filter using the system dynamic model is shown in Eq. 3.2 with the $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$ are the dynamically modelled system and measurement of the robot. The accuracy of the Kalman Filter's estimate relies heavily on the model.

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{v}_k \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \end{aligned} \tag{3.2}$$

State estimation using Kalman Filter with the MIP dynamic equation has been done, but mostly in simulations where it is assumed that the θ can be measured directly [9]. In addition, Shimizu et al. developed a MIP Dynamic Kalman Filter using only the gyroscope and encoder measurements [10].

The Extended Kalman Filter which uses a 3D high yaw rate MIP dynamic model has not been explored yet. We hypothesized that by implementing this model with the Extended Kalman Filter can yield a significantly more accurate estimates compared to the Complementary filter and the Kalman Filter with planar MIP dynamics. The next section describes the dynamic modeling for MIP robots with high yaw rate.

3.2 MIP Dynamic Modeling

The high yaw-rate MIP dynamic model is derived in this section by using Lagrangian dynamics formulation. The methodology can be described as follows: derive the full nonlinear MIP model, and then apply simplification which allows for high yaw rates. The linear acceleration of the body is also derived in order to determine the measurement model for the IMU's accelerometer for the KF and EKF. Table 3.1 lists the parameters used in this chapter.

The linearized planar MIP dynamic model in Eq. 3.3 as derived in [11] is well known

Table 3.1: Parameter and Time Varying Variable List. Abbreviations: CoM = Center of Mass, CoR = Center of Rotation (origin of the body frame).

Parameter List	**Time Varying Variable List**, $i = \{1, 2\}$
m_b = body mass.	θ = pitch angle.
m_w = wheel mass.	ψ = yaw angle.
$\hat{I}_b$ = body inertia about CoM.	ϕ_i = wheel i rotation angle.
$\hat{I}_w$ = wheel inertia about CoM.	φ_i = wheel i encoder rotation angle.
r = wheel radius.	τ_i = wheel i torque.
l = length of body's CoM to CoR.	u_i = motor i PWM command $\in [-1, 1]$.
d = distance between wheels.	$\boldsymbol{p}$ = body CoM position vector.
g = gravity constant.	$\boldsymbol{p}_{wi}$ = wheel CoM position vector.
k_1 = motor torque gain.	$\boldsymbol{p}_w$ = body frame's CoR position vector.
k_2 = motor back EMF gain.	$\boldsymbol{l}_b$ = length vector from CoR to body CoM.
	$\boldsymbol{l}_r$ = length vector of wheel CoM to the floor.

and used in MIP robots where the yaw rate is trivial or ignored.

$$\begin{aligned} \left(\hat{I}_{b1} + m_b L^2\right)\, \ddot{\theta} + m_b r l\, \ddot{\phi} - g\, m_b\, \theta &= -\tau \\ m_b r l\, \ddot{\theta} + \left(\hat{I}_{w1} + (m_w + m_b)\, r^2\right)\, \ddot{\phi} &= \tau. \end{aligned} \tag{3.3}$$

$\hat{I}_{b1}$ and $\hat{I}_{w1}$ are the component of the matrices $\hat{I}_b^B$ and $\hat{I}_w^B$ respectively, as shown in Eq. 3.4. $\hat{I}_b^B$ and $\hat{I}_w^B$ in Eq. 3.4 are the body frame moment of inertia of the body and the wheel respectively.

$$\hat{I}_b^B = \begin{bmatrix} I_{b1} & 0 & 0 \\ 0 & I_{b2} & 0 \\ 0 & 0 & I_{b3} \end{bmatrix} \qquad \hat{I}_w^B = \begin{bmatrix} I_{w1} & 0 & 0 \\ 0 & I_{w2} & 0 \\ 0 & 0 & I_{w3} \end{bmatrix}. \tag{3.4}$$

In order to develop a model which allows for high yaw rates, the simplification conditions used in Eq. 3.3 can be relaxed to allow nontrivial yaw rates. The MIP kinematics are defined using two separate coordinate frames: the inertial frame and the body frame about the center of rotation as shown in Fig. 3.2. The body reference frame's axis $\{\hat{\boldsymbol{e}}^1, \hat{\boldsymbol{e}}^2, \hat{\boldsymbol{e}}^3\}$ is pointed to the body's left, back, and top respectively and the variables defined in the body frame are represented with a

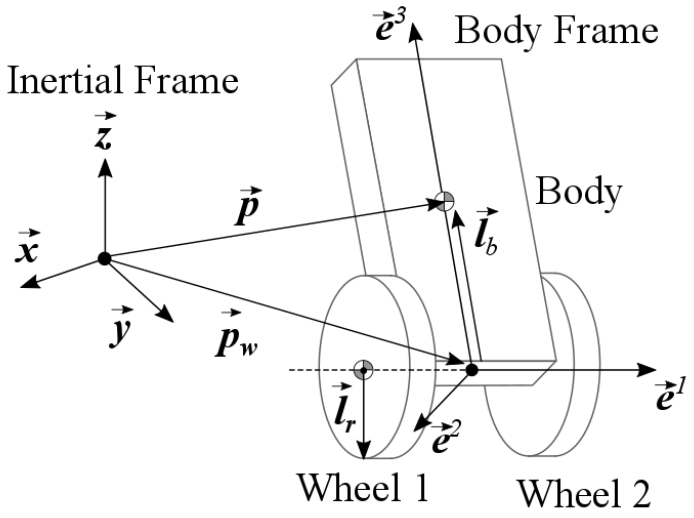


Figure 3.2: Coordinate frames of the kinematic model. The body frame origin is the center of rotation for both the body and the wheels.

superscript B, e.g. $\boldsymbol{x}^B$. The rotation from the body into the inertial frame is defined in Eq. 3.5 below.

$$\begin{aligned} \boldsymbol{x} &= R_B \, \boldsymbol{x}^B \\ R_B &= R_z(\psi) \, R_x(\theta). \end{aligned} \tag{3.5}$$

R_x and R_z are the Euler rotation about the inertial frame's x-axis and z-axis respectively. The linear position of the body frame's origin about the inertial frame can be defined with the vector $\boldsymbol{p}_w$ which is the average position of both wheels $\boldsymbol{p}_{w1}$ and $\boldsymbol{p}_{w2}$.

3.2.1 Kinematic Formulation

The kinematic equations derived in this subsection are used to determine the kinetic and potential energy of the system. Some of the position and length vectors are defined below:

$$\boldsymbol{p}_w = (\boldsymbol{p}_{w1} + \boldsymbol{p}_{w2})/2 \tag{3.6}$$

$$\boldsymbol{l}_b^B = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^T, \qquad \boldsymbol{l}_r = \begin{bmatrix} 0 & 0 & -r \end{bmatrix}^T. \tag{3.7}$$

The upper body rotational speed $\boldsymbol{\Omega}$ and the wheel $i \in \{1, 2\}$ rotational speed $\boldsymbol{\omega}_i$ are:

$$\boldsymbol{\Omega} = R_z(\psi) \begin{bmatrix} \dot{\theta} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}, \qquad \boldsymbol{\omega}_i = R_z(\psi) \begin{bmatrix} \dot{\varphi}_i \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{\Omega}. \tag{3.8}$$

Note that the $\boldsymbol{\omega}_i$ in Eq. 3.8 is defined by using the encoder measurements φ_i. The no slip conditions between the wheel and the ground is:

$$\frac{d\boldsymbol{p}_{wi}}{dt} = \boldsymbol{r} \times \boldsymbol{\omega}_i, \quad i = \{1, 2\} \tag{3.9}$$

$$\psi = (\varphi_1 - \varphi_2)\, r/d. \tag{3.10}$$

Finally, the body linear velocity is:

$$\frac{d\boldsymbol{p}}{dt} = \frac{d}{dt}\left(R_B \boldsymbol{l}_b^B\right) + \frac{1}{2}\left(\frac{d\boldsymbol{p}_{w1}}{dt} + \frac{d\boldsymbol{p}_{w2}}{dt}\right). \tag{3.11}$$

3.2.2 Lagrangian Dynamics Formulation

The MIP equations of motion are derived using Lagrangian dynamics and transformed into the state space form to be used with the Kalman Filter. The kinetic and potential energy into

the system are:

$$\begin{aligned} K_b &= \frac{1}{2}(R_B^T\,\boldsymbol{\Omega})^T\,\hat{I}_b\,(R_B^T\,\boldsymbol{\Omega}) + \frac{m_b}{2}\frac{d\boldsymbol{p}}{dt}^T\frac{d\boldsymbol{p}}{dt} \\ K_{wi} &= \frac{1}{2}(R_B^T\,\boldsymbol{\omega}_i)^T\,\hat{I}_w\,(R_B^T\,\boldsymbol{\omega}_i) + \frac{m_w}{2}\frac{d\boldsymbol{p}_{wi}}{dt}^T\frac{d\boldsymbol{p}_{wi}}{dt} \\ U_b &= -m_b\begin{bmatrix}0 & 0 & -g\end{bmatrix}\boldsymbol{p} \\ U_{wi} &= 0, \qquad i = \{1,2\}. \end{aligned} \tag{3.12}$$

Then the Lagrangian of the system is $L(\boldsymbol{q},\dot{\boldsymbol{q}}) = K_b + K_{w1} + K_{w2} - U_b$, which is a function of the time varying variable $\boldsymbol{q}(t)$:

$$\boldsymbol{q}(t) = \begin{bmatrix}\theta(t) & \varphi_1(t) & \varphi_2(t)\end{bmatrix}^T. \tag{3.13}$$

The system dynamic equations is solved by using the Lagrange's equation as shown below:

$$\begin{bmatrix}L_1\\L_2\\L_3\end{bmatrix} = \frac{d}{dt}\frac{\partial L}{\partial\dot{\boldsymbol{q}}} - \frac{\partial L}{\partial\boldsymbol{q}} - \boldsymbol{\tau} = \boldsymbol{0}. \tag{3.14}$$

There is no Lagrange Multiplier in Eq. 3.14 because the constraints are already applied into the kinematic equation from Eq. 3.9 and 3.10. The force acting onto the system is applied through the wheel's motor. The force $\boldsymbol{\tau}$ and the motor i torque τ_i are:

$$\boldsymbol{\tau} = \begin{bmatrix}0 & \tau_1 & \tau_2\end{bmatrix}^T \tag{3.15}$$

$$\tau_i = k_1 u_i - k_2\dot{\varphi}_i, \quad i = \{1,2\}. \tag{3.16}$$

The wheel angles φ_1 and φ_2 and motor command u_1 and u_2 are not useful for controlling the robot. To simplify the states, modify the Eq. 3.14 as follows:

$$\begin{bmatrix} L_1^* \\ L_2^* \\ L_3^* \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 + L_3 \\ L_2 - L_3 \end{bmatrix} = \mathbf{0}. \tag{3.17}$$

Next, do the following change of variables:

$$\begin{aligned} u_1 &= u_x + u_z \quad \varphi_1 = \varphi + \psi d/(2r) \\ u_2 &= u_x - u_z \quad \varphi_2 = \varphi - \psi d/(2r). \end{aligned} \tag{3.18}$$

The equations of motion in Eq. 3.17 are now a function of $\boldsymbol{q}^* = [\theta, \varphi, \psi]$, $\dot{\boldsymbol{q}}^*$ and $[u_x, u_z]$ where φ is the average encoder angles from both wheels, ψ is the yaw rate, u_x and u_z are the motor command forward and spin respectively.

The full nonlinear dynamic model for the MIP can be seen in Appendix A.1. The dynamic equations are simplified by assuming the body angle θ and its derivatives are small, similar to the linearized planar MIP equations in Eq. 3.3. This approximates $\sin\theta \approx \theta$, $\cos\theta \approx 1$, $\{\theta^2, \theta\dot{\theta}, \dot{\theta}^2\} \approx 0$. However, the small yaw rates assumption is not used in the high yaw-rate model derivations. Then rearrange Eq. 3.17 into the general form:

$$M(\boldsymbol{q}^*)\,\ddot{\boldsymbol{q}}^* + \boldsymbol{h}(\boldsymbol{q}^*, \dot{\boldsymbol{q}}^*) = \boldsymbol{\tau}^*(u_x, u_z) \tag{3.19}$$

$$M\,\ddot{\boldsymbol{q}}^* = \boldsymbol{b} \tag{3.20}$$

$$M = \begin{bmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{3.21}$$

$$
\begin{aligned}
m_{11} &= I_{b1} + 2I_{w1} + 2m_w r^2 + m_b(l+r)^2 \\
m_{12} &= m_{21} = 2I_{w1} + 2m_w r^2 + m_b r(l+r) \\
m_{22} &= 2I_{w1} + (m_b + 2m_w) r^2 \\
m_{33} &= I_{w1} d/r + r(dm_w + 2(I_{t3} + 2I_{w3})/d) \\
b_1 &= glm_b\theta + (I_{b2} - I_{b3} + 2(I_{w2} - I_{w3}) + l^2 m_b)\theta\dot{\psi}^2 \\
b_2 &= 2(k_1 u_x - k_2\dot{\varphi}) \\
b_3 &= 2k_1 u_z - k_2 d\dot{\psi}/r.
\end{aligned} \tag{3.22}
$$

Finally, derive the high yaw-rate model which will be used in the EKF:

$$
\boldsymbol{x} = \begin{bmatrix} \theta & \varphi & \psi & \dot{\theta} & \dot{\varphi} & \dot{\psi} \end{bmatrix}^T \tag{3.23}
$$

$$
\boldsymbol{u} = \begin{bmatrix} u_x & u_z \end{bmatrix}^T \tag{3.24}
$$

$$
\frac{d\boldsymbol{x}}{dt} = \begin{bmatrix} \dot{\boldsymbol{q}}^* \\ \ddot{\boldsymbol{q}}^* \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}}^* \\ M^{-1}\,\boldsymbol{b} \end{bmatrix} = \boldsymbol{f}_c(\boldsymbol{x}, \boldsymbol{u}). \tag{3.25}
$$

The equations above can be simplified down into the linear planar MIP dynamic in Eq. 3.3 by setting $\dot{\psi} = 0$ and doing a change of variable $\phi = \varphi + \theta$.

3.2.3 Sensor Dynamics

The body's linear acceleration can be solved as a function of $\boldsymbol{x}$ using the angular acceleration $\ddot{\boldsymbol{q}}^*$ derived from Eq. 3.25. Then the linear acceleration as measured by the accelerometer is:

$$
\boldsymbol{y}_a^B(\boldsymbol{x}) = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R_B^T \left(\begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} - \frac{d^2\boldsymbol{p}}{dt^2} \right). \tag{3.26}
$$

The full nonlinear derivations of the sensor model can be seen in Appendix A. The $\boldsymbol{y}_a^B$ can be simplified by using the same small θ and $\dot{\theta}$ assumptions as the equation of motion above. Only a_y (body forward/back direction) is used in the EKF because the other acceleration measurements were very noisy even when idling. The a_y is expressed below:

$$a_y = -l\theta\dot{\psi}^2 + (l+r)\ddot{\theta} + r\ddot{\varphi} - g\theta. \tag{3.27}$$

3.3 Estimators Design

This section lists the actual equations used by the estimations. The EKF algorithm follows the standard discrete time EKF algorithm. The controller loop is assumed to run at a constant frequency of 200 Hz ($dt = 0.005$ s).

Complementary Filter

The Complementary Filter was setup with the cutoff frequency of 4 rad/s, which gave us $c = 0.98$ and the following equations:

$$\theta_k^{accel} = \operatorname{atan2}(-a_{yk}, -a_{zk}) \tag{3.28}$$

$$\hat{\theta}_k = 0.98\,\hat{\theta}_{k-1} + 0.005\,\omega_{k-1}^{gyro} + 0.02\,\theta_{k-1}^{accel} \tag{3.29}$$

Complementary Kalman Filter

The Complementary Kalman Filter is setup the same way as the Eq. 3.1, using the following values:

$$y_k^a = \theta_k^{accel} = \operatorname{atan2}(-a_{yk}, -a_{zk}) \tag{3.30}$$

$$Q_c = \operatorname{diag}([10^{-8}, 10^{-10}]), \qquad R_c = 10^{-4} \tag{3.31}$$

where the Q_c and R_c are the process and measurement noise covariance matrices respectively.

Planar Dynamic Kalman Filter

The dynamic equation for the planar MIP dynamic can be calculated from the fast yaw dynamic equation in Eq. 3.37 by setting $\dot{\psi} = 0$. The dynamic equation will become linear time-invariant and is defined with only 4 states: $\boldsymbol{x} = [\theta, \varphi, \dot{\theta}, \dot{\varphi}]$ and 1 input $u = u_x$. The measurements used in the planar KF is:

$$\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} \varphi_k & \omega_k^{gyro} & a_{yk} \end{bmatrix}^T \tag{3.32}$$

The process and measurement noise covariance matrices values are:

$$Q_p = \mathrm{diag}([10^{-11}, 10^{-7}, 10^{-5}, 10^{-7}]) \tag{3.33}$$

$$R_p = \mathrm{diag}([10^{-5}, 3.07\,10^{-6}, 3.04\,10^{-3}]) \tag{3.34}$$

The equation of motion derived in Eq. 3.25 is the continuous time dynamic equation, so we need to transform it into a discrete time equation before we can use it in the EKF. The discrete time equation of motion was estimated using the Explicit Euler method. Then the difference equation for the EKF becomes:

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{v}_k \tag{3.35}$$

$$\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) = \boldsymbol{x}_{k-1} + dt\boldsymbol{f}_c(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \tag{3.36}$$

$$\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \end{bmatrix}^T \tag{3.37}$$

Plugging in the parameters values in Table 3.2 into the equations above yields:

$$f_1 = \theta_k + 0.005\,\dot{\theta}_k \tag{3.38}$$

$$f_2 = \varphi_k + 0.005\,\dot{\varphi}_k \tag{3.39}$$

$$f_3 = \psi_k + 0.005\,\dot{\psi}_k \tag{3.40}$$

$$f_4 = \dot{\theta}_k + 0.629\,\theta_k + 0.068\,\dot{\varphi}_k + 0.009\,\theta_k\,\dot{\psi}_k^2 - 2.72 u_{xk} \tag{3.41}$$

$$f_5 = 0.813\,\dot{\varphi}_k - 1.12\,\theta_k - 0.016\,\theta_k\,\dot{\psi}_k^2 + 7.47\,u_{xk} \tag{3.42}$$

$$f_6 = 0.939\,\dot{\psi}_k + 2.44\,u_{zk} \tag{3.43}$$

and the measurements used in the yaw dynamic EKF is:

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \tag{3.44}$$

$$\boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \begin{bmatrix} \varphi_k & \omega_k^{gyro} & a_{yk} & \psi_k \end{bmatrix}^T \tag{3.45}$$

$$\omega_k^{gyro} = \dot{\theta}_k \tag{3.46}$$

$$a_y = 8.17\,u_x - 7.41\,\theta - 0.204\,\dot{\varphi} - 0.0346\,\theta\,\dot{\psi}^2 \tag{3.47}$$

The process and measurement noise covariance matrices values used for the EKF are:

$$Q_y = \mathrm{diag}([10^{-11}, 10^{-7}, 10^{-7}, 10^{-5}, 10^{-7}, 10^{-7}]) \tag{3.48}$$

$$R_y = \mathrm{diag}([10^{-5}, 2.62\,10^{-7}, 5.44\,10^{-3}, 10^{-5}]) \tag{3.49}$$

3.4 Motion Capture Experiment

The motion capture system comprised of four Optitrack 13 cameras placed in four different corners at varying heights around the testing platform. The camera have a sampling rate of 120 Hz. During testing, we simultaneously recorded the motion capture data on a laptop

Table 3.2: Edu MIP Parameter Values.

Parameter	Value	Parameter	Value
m_b	218 g	I_{b1}	5.91 10^{-4} kg.m^2
m_w	24 g	I_{b2}	2.99 10^{-4} kg.m^2
r	35 mm	I_{b3}	2.91 10^{-4} kg.m^2
d	70 mm	I_{w1}	6.42 10^{-5} kg.m^2
l	46 mm	I_{w2}	7.48 10^{-6} kg.m^2
k_1	0.12 N.m	I_{w3}	7.48 10^{-6} kg.m^2
k_2	0.003 N.m.s/rad	g	9.8 m/s^2
dt	0.005 s		

and state estimates on board MIP's Beaglebone Black. The data was synchronized in post by matching the measurements when the robot begins balancing from its resting position.

3.4.1 Experimental Setup

We used motion capture to obtain a ground truth model to test the validity of our various state estimators. We were able to simultaneously we record all the θ estimates on board the robot and the camera data from motion capture. To test the performance of our estimators, we ran our MIP test under three different experiments: idling balancing, 5 rad/s and 10 rad/s spinning while balancing in place.

The MIP used in this work is the eduMIP, available from Renaissance Robotics, as shown in Fig. 3.1. The Edu MIP is controlled using the Beaglebone Black and a robotics cape, which contains the 9-axis IMU, breakout for encoder counting and motor drivers. Beaglebone can store data into its hard drive at a sampling rate of 50 Hz. The parameter values for this Edu MIP can be seen in Table 3.2, which were estimated using system identification by Zhuo [12].

In order to balance the MIP, we used a Successive Loop Closure (shown in Fig. 3.3) for controlling θ and a simple PD controller for controlling ψ. The controller's discrete time transfer

Table 3.3: Root Mean Squared (RMS) Error between the estimates and the motion captured θ.

Estimator	RMS Error (rad)		
	No Spin	5 rad/s spin	10 rad/s spin
DMP	**0.0148**	**0.0115**	**0.0129**
Planar EKF	0.0133	0.0256	0.1028
High Yaw Rate EKF	**0.0133**	**0.0107**	**0.0249**
Complementary Filter	0.0228	0.0424	0.1605
Complementary KF	0.0199	0.0432	0.1636

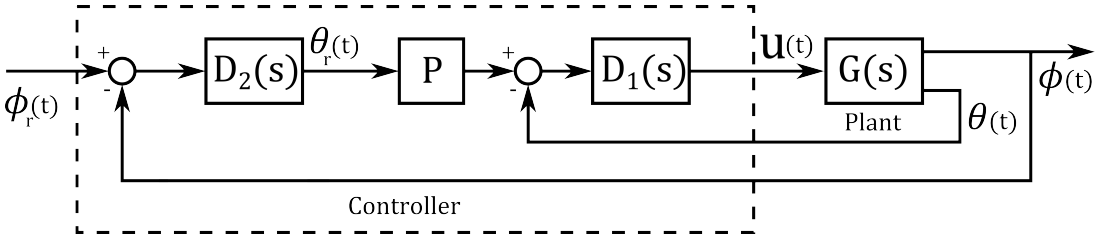


Figure 3.3: Successive Loop Closure Block Diagram

functions are:

$$D_1(z) = \frac{-4.95\,z^2 + 8.86\,z - 3.97}{1.000\,z^2 - 1.48\,z + 0.481} \tag{3.50}$$

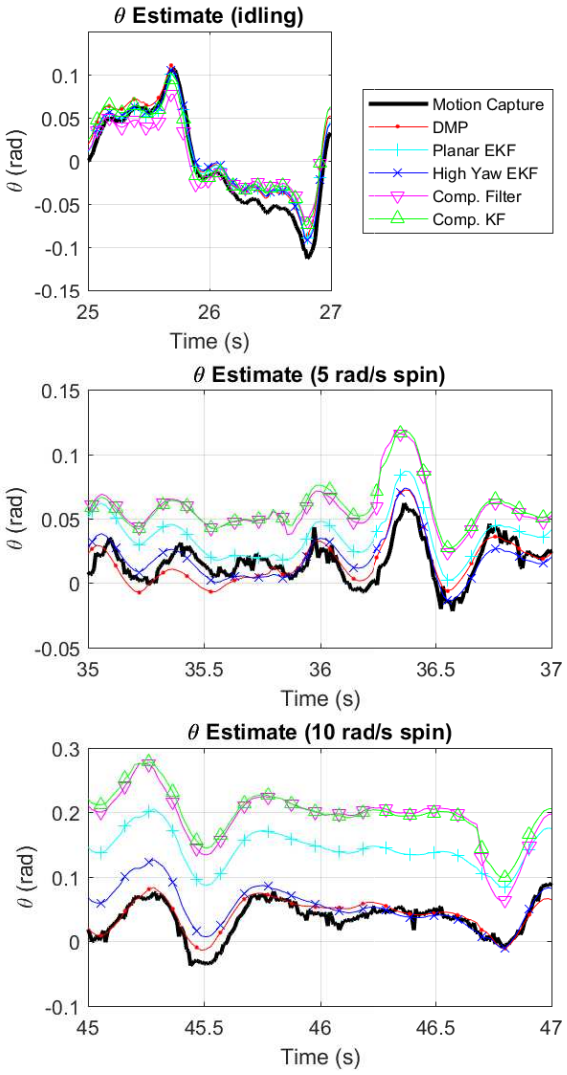
$$D_2(z)P = \frac{-0.189\,z^2 + 0.372\,z - 0.184}{1.000\,z^2 - 1.86\,z + 0.86} \tag{3.51}$$

$$D_z = 1.0(\psi_r - \psi) + 0.05(\dot{\psi}_r - \dot{\psi}) \tag{3.52}$$

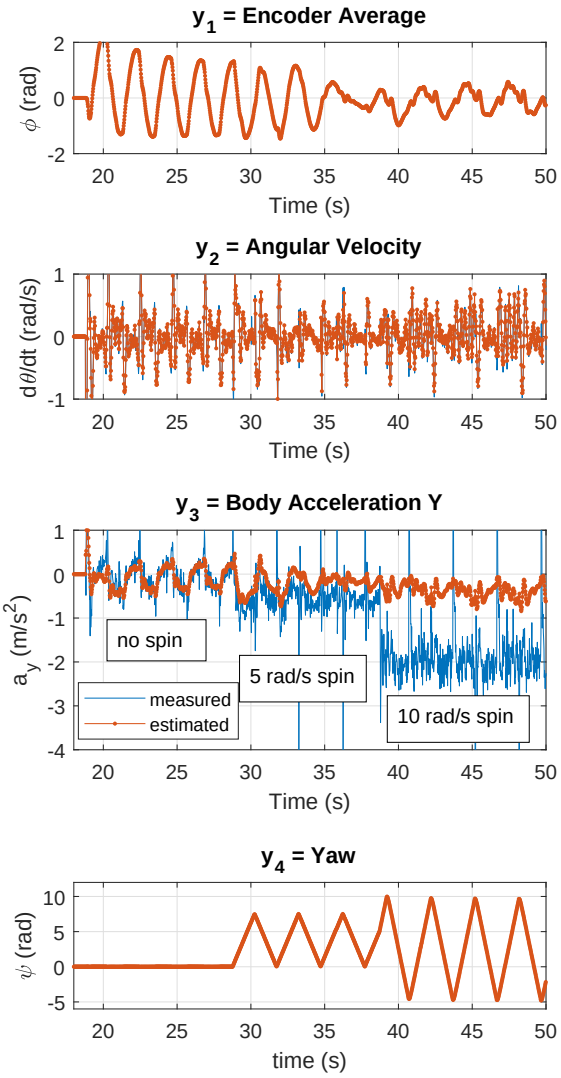
where D_z is the yaw controller which outputs u_z, ψ_r and $\dot{\psi}_r$ are the reference yaw values. This controller uses the DMP measurement from the IMU and the raw encoder values as the state into the controller. Since we want to compare the accuracy of the estimators relative to each other, we want to use the same controller for all of the test and this controller worked very well.

3.4.2 Experimental Results

The θ estimates under several yaw rates can be seen in Figure 3.4a and their respective Root Mean Squared errors are listed in Table 3.3. The error from Table 3.3 shows that the DMP



(a) Estimated θ and the motion captured θ.



(b) Measured $\boldsymbol{y}$ vs estimated $\boldsymbol{y}$ plot for the high yaw-rate model EKF during the 0 rad/s, 5 rad/s and 10 rad/s yaw-rates experiments.

Figure 3.4: MIP high yaw rates motion capture experimental results.

and yaw dynamic EKF have the overall best performance with better accuracy under high yaw rates compared to the other estimates, like what we expected. The DMP's estimate is especially good because it does not use the system's dynamic equation to get such an accurate estimate. So, it is very likely that we can use the DMP's estimate without the need of using an observer if the orientation is the only important state to estimate.

Looking at the acceleration data in Figure 3.4b shows an interesting situation. The yaw dynamic EKF's estimated y_3 value did not match the measured value during the 10 rad/s spinning but the estimated θ is still more accurate than the others. We believe that there is an offset on the acceleration data from the IMU placement being not on the top body's center of mass, causing the acceleration data to receive a bias as a function of yaw rate due to the centripetal force. The complementary filter used the atan2 function to determine θ, but the estimation during high yaw rate is heavily skewed due to the accelerator bias and this caused the estimates to be inaccurate during fast yaw movements.

In this chapter we surveyed several different state estimators used on MIP robots and compared their performance using a ground truth established by a motion capture system. In addition, we presented our novel high yaw rate dynamic model and it's corresponding extended Kalman Filter which proved to provide better estimates as the robot is performing dynamic maneuvers. Also, we found that the Complementary Filter and the Complementary Kalman Filter have almost identical performance under all tested situations. Remarkably, the proprietary DMP estimate from the MPU-9250 appears to very accurate over the conditions tested, even under high yaw rates, even though it is not based on the dynamic model of the physical system. The next step of this project is to implement a similar high yaw rate dynamic modeled Kalman Filter into our ball-balancing robot [3] to help us control the robot during aggressive and fast yaw maneuvers. Compared to the simpler MIP robot, the BBR is highly nonlinear and the dynamics are heavily coupled under high yaw rate, which should prove to be a challenging topic.

Acknowledgements

Clark Briggs from ATA Engineering lend us the Optitrack motion capture system which was greatly utilized in the experiments done in Chapter 3 and 6.

Chapter 3, in full, is a reprint of the material as it appears in the published paper in American Control Conference (ACC) 2018, E. Sihite and T. Bewley, "Attitude estimation of a high-yaw-rate Mobile Inverted Pendulum; comparison of Extended Kalman Filtering, Complementary Filtering, and motion capture". The dissertation author was the sole investigator and author of this paper.

References

[1] H. Lee and S. Jung, "Balancing and navigation control of a mobile inverted pendulum robot using sensor fusion of low cost sensors," *Mechatronics 22.1*, pp. 95–105, 2012.

[2] J. Akesson, A. Blomdell, and R. Braun, "Design and control of yaip — an inverted pendulum on two wheels robot," *IEEE International Conference on Control Applications*, 2006.

[3] D. Yang, E. Sihite, M. Friesen, and T. Bewley, "Design and control of a micro ball-balancing robot (mbbr) with orthogonal midlatitude omniwheel placement," *IEEE Int'l Conf. on Intelligent Robots and Systems (IROS)*, pp. 4098–4104, 2015.

[4] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, "Unified state estimation for a ballbot," *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pp. 2471–2476, 2013.

[5] W. T. Higgins, "A comparison of complementary and kalman filtering," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 321–325, 1975.

[6] A. M. Sabatini, "Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Transactions on Biomedical Engineering 53.7*, pp. 1346–1356, 2006.

[7] Y. S. Young, "Orientation estimation using a quaternion-based indirect kalman filter with adaptive estimation of external acceleration," *IEEE Transactions on Instrumentation and Measurement 59.12*, pp. 3296–3305, 2010.

[8] G. Ligorio and A. M. Sabatini, "A novel kalman filter for human motion tracking with an inertial-based dynamic inclinometer," *IEEE Transactions on Biomedical Engineering 62.8*, pp. 2033–2043, 2015.

[9] R. Eide, P. M. Egelid, A. Stamsø, and H. R. Karimi, "Lqg control design for balancing an inverted pendulum mobile robot," *Intelligent Control and Automation 2*, pp. 160–166, 2011.

[10] Y. Shimizu and A. Shimada, "Direct tilt angle control on inverted pendulum mobile robots," *IEEE International Workshop on Advanced Motion Control*, pp. 160–166, 2010.

[11] T. Bewley, *Numerical Renaissance: simulation, optimization, and control*. Renaissance Press, San Diego, 2015.

[12] Z. Zhuo, "Lqg controller design of the mobile inverted pendulum," Master's thesis, University of California San Diego, San Diego, USA, 2017.

Chapter 4

Coasting Brushed DC Motor Driver Model

Brushed DC motors are generally driven by using PWM signal which operates in one of two modes: drive/brake or drive/coast. That is, at the low state of the PWM forcing profile, the motor driver will either "brake" the motor with its own back EMF, or allow the motor to "coast" (i.e., spin freely). Drive/brake motor drivers, which are by far the most common, may be represented by a Multilevel Four-Quadrant DC Chopper model, while drive/coast motor drivers may be represented by two independent Bipolar Two-Quadrant DC Chopper models. Conveniently, when averaged over the PWM duty cycle, drive/brake motor drivers are accurately modeled as linear systems over their entire operational range. On the other hand, drive/coast motor drivers, when averaged over the PWM duty cycle, exhibit significant nonlinear behaviors that are dependent on factors such as inductance, PWM frequency, and rotor speed. Though there are some existing partial derivations of drive/coast motor driver models, no comprehensive, experimentally-validated modeling approaches appropriate for feedback control applications over the full dynamic range of the motor could be readily found in the literature.

The most recent MBBR prototype uses a motor driver which operates in the drive/coast configuration. The drive/coast motor drivers were chosen because the limitation of the PWM pins in the Beaglebone Black and the electrical current requirements of the motors used. The Beaglebone only has four PWM capable pins and most drive/brake drivers that fulfills our current requirements requires two PWM pins for each motor to operate. Since we have three motors, we

can't use these drive/brake drivers with the Beaglebone Black. On the other hand, drive/coast drivers can be driven with one PWM pin and two GPIO pins which. However, the nonlinear behavior in the drive/coast drivers must be compensated so that we can use a linear torque model in our dynamic model. In this chapter, we derive a practical nonlinear model of a drive/coast motor driver, validate this model using a motor dynamometer, and demonstrate a real-time implementation of this model on a Mobile Inverted Pendulum (MIP) robot.

4.1 Past Works on DC Motor Drivers

Agile dynamic UxVs require accurate physical models to inform mechanical designs and to develop controllers that achieve maximum performance. Mass distributions and other physical parameters can generally be obtained via CAD and simple experiments, but accurate dynamic motor models are often much more difficult to develop. Least-square fits to experiments can be used to identify the parameters of simple linear dynamic models of motors if they are of the correct structure, but getting the (nonlinear) structure of these models correct in the drive/coast case is delicate.

Brushed DC motors are commonly driven by modulating the input voltage via pulse width modulation (PWM), at frequencies from 500Hz to 20kHz, together with one or two logic signals to indicate the forward or reverse direction. The MOSFETs in an H-Bridge are then opened or closed in pairs to allow current to pass through the circuit in the appropriate direction. There are two types of motor driver modes discussed in this chapter, depending on the MOSFET settings during the low state of the PWM forcing profile: braking or coasting. Braking occurs when both of the upper (or, lower) MOSFETs are closed, and the other MOSFETs are open; in this case, current circulates in the upper (or, lower) part of the H-bridge circuit, and the back EMF of the motor itself drives the current in the opposite direction of the rotor speed, slowing the motor. Coasting occurs when all four MOSFETs in the H-bridge are open, and current flows from ground to Vcc through two of the flyback diodes when necessary. The coasting steady state

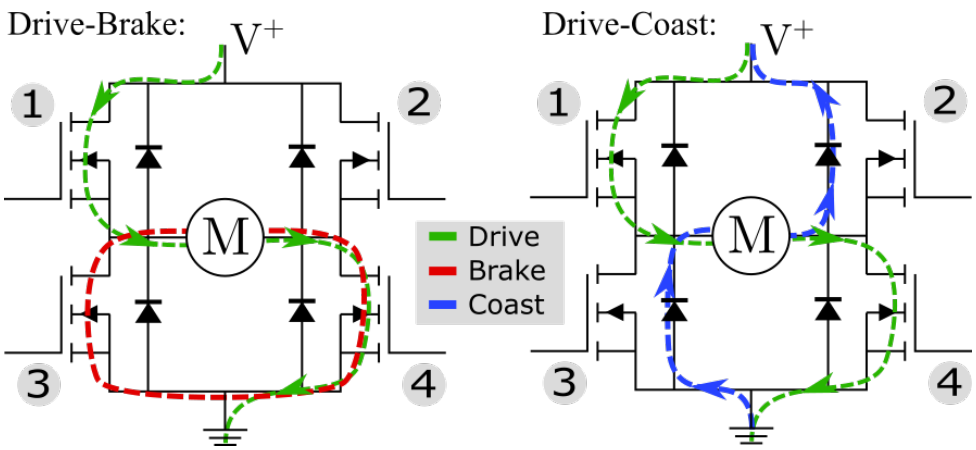


Figure 4.1: Full H-bridge circuit demonstrating the current paths during forward drive, brake, and coast.

current applies a zero torque on the motor, allowing the motor to spin freely. Figure 4.1 shows the driving, braking and coasting states of an H-bridge. To drive the motor forward (green), MOSFETs 1 and 4 are closed. To brake (red), either MOSFETs 3 and 4 are closed (low side) or 1 and 2 (high side) which shorts the motor and allows the current to circulate and brake the motor. Finally, to coast (blue), all of the MOSFETs are left open, allowing the current to flow through the diodes as necessary, and causing the motor to coast when this current decays to zero.

Drive/brake motor drivers may be represented by a Multilevel Four-Quadrant DC Chopper, and drive/coast motor drivers may be represented by two independent Bipolar Two-Quadrant DC Chopper [1][2]. Drive/coast motor drivers in constant forward drive use the I-IV quadrants, and in reverse drive use the II-III quadrants [1]. Averaged over the PWM duty cycle, such drive/brake motor drivers may be accurately modeled with a simple linear model [3][4][5][6]. Averaged over the PWM duty cycle, the drive/coast motor driver, on the other hand, exhibits a significant nonlinear behavior especially near zero duty cycle. Partial derivations of electrical current models for such drivers exist [1]; however, this work does not extend to a full range of forward and reverse driving, and no real-time implementation suitable for control applications is provided.

There are several advantages of using drive/coast motor drivers. Coasting engages the

natural (unforced) dynamic behavior of the vehicle, which is at times preferred. There might also be a good incentive to use drive/coast instead of drive/brake drivers from an energy efficiency standpoint. There has been much research on regenerative braking with brushless DC motors in, e.g., hybrid/electical vehicles [7][8][9][10]. However, to the best of our knowledge, the energy efficiency of drive/coast vs. drive/brake systems for vehicles that often operate near zero speed and/or zero torque has not been extensively studied. A drive/brake system can produce regenerative current during the braking sequence, transforming the kinetic energy into electrical energy [7]. A drive/coast system, on the other hand, regenerates the battery during coasting by flowing current back into the Vcc through the flyback diodes [1][9]. In a motor that operates near zero torque and/or zero speed, the drive/brake regenerative method is ineffective due to the lack of kinetic energy. The drive/coast approach might have better efficiency in this range, but conclusively establishing this would require further study. Recent advanced TI motor drivers, such as the DRV8881 in the so-called "fast decay" mode, detect the regenerative current during coasting and close the respective gates such that the current flows through the MOSFETs instead of the flyback diodes, resulting in reduced voltage drop (and, thus preventing the flyback diodes from overheating). If this is utilized well, drive/coast systems might show better energy efficiency than drive/brake systems near zero torque in speed control applications.

Brushed DC motors are cheap and often used in low cost robots which generally have a high amount of noise. The challenge of accurately modeling such motors is exacerbated in a drive/coast implementation due to the nonlinearities near zero torque and speed, which is a primary mode of operation for a balancing robot such as a Mobile Inverted Pendulum (MIP). Modeling the system dynamics as accurately as possible can increase the performance of the controller. For these reasons, it is important to develop an accurate drive/coast model that can be used in a real-time feedback controller.

In this chapter, we present a new dynamic model for a system with a brushed DC motor using a drive/coast motor driver, validate the model using a low cost dynamometer, propose a method of real-time implementation of the drive/coast model, and demonstrate the performance

of our model on a educational Mobile Inverted Pendulum (eduMIP) robot.

4.2 H-Bridge and Drive Modes

An H-bridge circuit allows for directional control of a brushed DC motor. It typically consists of four MOSFETs and diodes arranged as letter H with a load at the center [11]. By switching the diagonally opposite MOSFETs on and off, the motor driving direction (forward or reverse) can be controlled. During the off duty of the PWM signal, the motor will either brake or coast based on the states of the four MOSFETs, as shown in Fig. 4.1. The choice of coast or brake mode has a significant effect on motor's dynamics, especially at low duty cycles. A quick survey of common motor drivers used in robotics shows that the drivers are either drive/coast only (DRV8881E, MC33926), drive/brake only (TB6612FNG), or configurable in either mode (DRV8881P, DRV8871). In practice, there is no "standard" mode of operation for the motor drivers and it is often left to the user's discretion. Table 4.1 lists the parameters and the variables used in the equations and derivations in this chapter. The following subsections derive and discuss the electrical equation model for the drive/brake and drive/coast motor drivers.

4.2.1 Drive-Brake

In drive/brake mode, the current path through the H-bridge can be seen in Fig. 4.1. During a high PWM signal, the appropriate MOSFETs pair is closed to drive the motor forward or reverse [11]. During low PWM signal, either the high side or low side MOSFETs are closed, shorting the motor terminals and allowing the back EMF to circulate and brake the motor [12]. The drive/brake system is very well understood and commonly modeled as follows: let u be the motor command, where $v = |u|$ is the PWM duty cycle and $\mathrm{sign}(u)$ is the motor driving direction where $u > 0$ and $u < 0$ are forward and reverse driving respectively. The electrical

Table 4.1: List of parameters and time varying variables.

Constant Parameters, all positive values.		**Time Varying Variables**	
k	torque constant.	i	electrical current.
V	supply voltage.	u	motor command $\in [-1, 1]$.
R	motor resistance.	τ	motor torque.
L	motor inductance.	v	PWM duty cycle, $v = \|u\| \in [0, 1]$.
i_s	stall current at $u = 1, \omega = 0$.	ω	rotor velocity.
ω_{nl}	maximum rotor no load speed.	ω_r	scaled rotor velocity $\in [-1, 1]$.
T_e	electrical time constant.		
T_r	ratio of T_{pwm} over T_e.		
T_{pwm}	PWM period.		
f_{pwm}	PWM frequency.		

circuit equation during the high and low PWM signals can be seen in (4.1) and (4.2).

$$\text{PWM high:} \qquad \operatorname{sign}(u)\, V = R\, i + L\, \tfrac{di}{dt} + k\, \omega \tag{4.1}$$

$$\text{PWM low:} \qquad 0 = R\, i + L\, \tfrac{di}{dt} + k\, \omega \tag{4.2}$$

$$u\, V = R\, i_{avg} + k\, \omega. \tag{4.3}$$

Both models can be combined into the linear system shown in (4.3) where i_{avg} is the average current of one PWM pulse [6]. The inductance can be ignored because the controller and measurement update period are significantly longer than the electrical time constant. This linear behavior and simplicity is a significant advantage for controller design and is the most common motor model used for PWM based motor controls.

4.2.2 Drive-Coast

The drive/coast model is much more complex. The current path for drive/coast mode can be seen in Fig. 4.1. During the low PWM signal, all of the MOSFETs are opened, forcing the current of the spinning motor to pass through the diodes into the battery positive terminal. The current quickly decays to zero but does not change direction, allowing the motor to spin

unimpeded. The electrical circuit equation during the high and low PWM signals can be seen in (4.4) and (4.5).

$$\text{PWM high:} \qquad \text{sign}(u)\, V = R\, i + L\, \tfrac{di}{dt} + k\, \omega \tag{4.4}$$

$$\text{PWM low:} \qquad -\text{sign}(i)\, (V + 2\, V_d) = R\, i + L\, \tfrac{di}{dt} + k\, \omega, \tag{4.5}$$

where V_d is the voltage drop across the diodes during the low PWM signal. As the motor coasts, the current drains down to zero and stays at zero for as long as the MOSFETs are opened, resulting in nonlinear dynamics. Additionally, di/dt is nonzero at $i = 0$, so this system must be described as a hybrid system, making it difficult to determine an averaged system model for a single PWM pulse.

4.3 Drive-Coast Model Derivations

Although the drive/coast system is a hybrid system, $i(t)$ can be solved from both ODEs in (4.4) and (4.5), as done by [1] on a First-Quadrant and a Bipolar Two-Quadrant DC Choppers. The averaged system model such as in (4.3) can't be derived. However, the average current i_{avg} during one PWM pulse can then be derived from the solution of $i(t)$. Deriving the i_{avg} is useful for controls application because the applied motor torque is a linear function of the motor current ($\tau = k\, i$). The following assumptions are used in order to simplify the $i(t)$ derivations:

- Electrical time constant is much smaller than mechanical time constant, controller update and sampling time.
- Time-periodic PWM pulse, as shown in Fig. 4.2, where the starting and final current in one PWM pulse are equal. t_0, t_1, and t_2 are the pulse start time, low PWM signal start time and the final pulse time respectively. The initial current during each time-periodic PWM pulse is assumed to be constant ($i(t_0) = i(t_2)$).

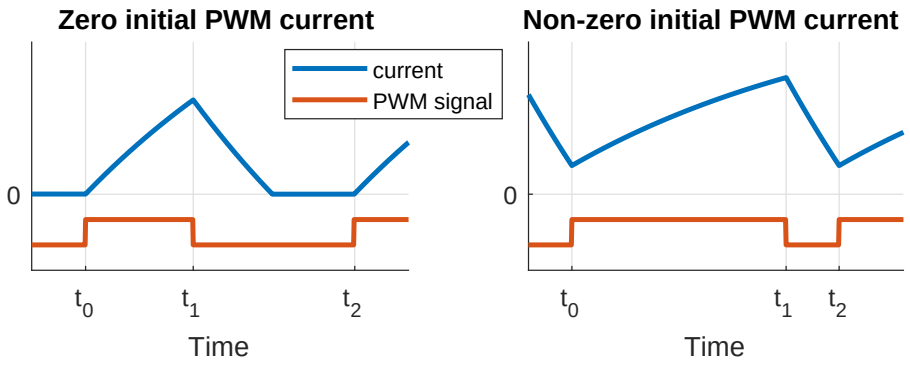


Figure 4.2: Plot of the current and PWM signal (shown at an offset) during one time-periodic PWM pulse in a drive/coast motor driver.

- Voltage drop across the diodes is negligible ($V_d \approx 0$).
- Instantaneous MOSFETs switching time.
- No external force to the mechanical system that can push the rotor speed above ω_{nl}.

The first assumption simplifies the ODEs in (4.4) and (4.5) by setting u and ω approximately constant compared to the i within one PWM pulse. The second assumption simplifies the problem further by constraining the initial and final current in one PWM pulse. The equations can be simplified even further by applying change of variables below:

$$\begin{aligned} i_s &= V/R & T_e &= L/R & \omega_{nl} &= V/k \\ T_r &= T_{pwm}/T_e & \omega_r &= \omega/\omega_{nl}, \end{aligned} \tag{4.6}$$

where ω_r is the ratio of between the rotor velocity and the maximum no load speed while T_r is the ratio between the PWM period and the electrical time constant. The final assumption above guarantees that $\omega_r \in [-1, 1]$. Using (4.6) and the assumptions above into (4.4) and (4.5) results

in the following simplified equations:

$$\text{PWM high:} \qquad \frac{di}{dt}(t) = \left(-i(t) + i_s(\text{sign}(u) - \omega_r)\right)/T_e \tag{4.7}$$

$$\text{PWM low:} \qquad \frac{di}{dt}(t) = \left(-i(t) - i_s(\text{sign}(i(t)) + \omega_r)\right)/T_e. \tag{4.8}$$

Let t_0 be the time at the beginning of the PWM pulse, $t_1 = t_0 + v\,T_{pwm}$ is the starting time of the low PWM signal, and $t_2 = t_0 + T_{pwm}$ is the end time of the pulse, as shown in Fig. 4.2. There are several statements that can be made from using $\omega_r \in [-1, 1]$, (4.7) and (4.8) that will be useful during the derivation of $i(t)$.

Lemma 1. *The current is always being driven to* $i_{hi} = i_s(\text{sign(u)} - \omega_r)$ *during high PWM signal. Also,* $i_{hi} \geq 0$ *if* $u > 0$ *and* $i_{hi} \leq 0$ *if* $u < 0$.

Proof. From (4.7), it can be easily shown that $\frac{di}{dt} < 0$ if $i > i_{hi}$ and $\frac{di}{dt} > 0$ if $i < i_{hi}$. Using $\omega_r \in [-1, 1]$, it can also be shown that if $\text{sign}(u) > 0$, then $i_{hi} \geq 0$ and if $\text{sign}(u) < 0$, then $i_{hi} \leq 0$. □ □

Lemma 2. *The current is always being drained to zero during low PWM signal.*

Proof. Use $\omega_r \in [-1, 1]$ in the (4.8), then it can be shown that $\frac{di}{dt} < 0$ if $i > 0$ and $\frac{di}{dt} > 0$ if $i < 0$. □ □

Proposition 1. *Assuming time-periodic PWM pulse, if* $u > 0$*, then* $i(t) \geq 0$ *within one pulse* $(t \in [t_0, t_0 + T_{pwm}])$*. Conversely, if* $u < 0$*, then* $i(t) \leq 0$.

Proof. During high PWM signal, the current is being driven to $i_{hi} \geq 0$ if $u > 0$ using Lemma 1. Then during low PWM signal, the current is being driven to zero using Lemma 2. Then that means the current during both high and low PWM signal, or one PWM pulse, is always ≥ 0 for $u > 0$. Case $u < 0$ can be proven the same way. □ □

Solve the ODE in (4.7) and (4.8) for $i(t)$ using the assumptions and statements listed

above. There are two different cases that need to be explored: $i(t_0) \neq 0$ and $i(t_0) = 0$, which is also shown in Fig. 4.2.

4.3.1 Case $i(t_0) \neq 0$

Solve the ODE in (4.7) and (4.8) for $u > 0$ and $i(t_0) > 0$. Using Proposition 1 and time-periodic PWM signal, $\text{sign}(i)$ is inferred to be constant in (4.8). This transforms (4.7) and (4.8) into easier to solve linear equations:

$$\text{PWM high:} \qquad \tfrac{di}{dt}(t) = -\left(i(t) + i_s(-1+\omega_r)\right)/T_e \tag{4.9}$$

$$\text{PWM low:} \qquad \tfrac{di}{dt}(t) = -\left(i(t) + i_s(1+\omega_r)\right)/T_e. \tag{4.10}$$

Solve the ODE in (4.9) and (4.10) for $i(t)$:

$$i_H(t_H) = e^{-t_H/T_e}\left(i(t_0) + i_s(1-e^{t_H/T_e})(\omega_r - 1)\right) \tag{4.11}$$

$$i_L(t_L) = e^{-t_L/T_e}\left(i(t_1) + i_s(1-e^{t_L/T_e})(\omega_r + 1)\right), \tag{4.12}$$

where $i_H(t_H)$ and $i_L(t_L)$ are the current equation during high and low PWM signal respectively. $t_H = t - t_0$ and $t_L = t - t_1$ are the time shift such that t_H and t_L are zero at the start of the high and low PWM signal respectively. Using (4.11) and (4.12), solve for $i(t_2)$ which should be equal to $i(t_0)$ by using the time-periodic PWM pulse assumption as shown below:

$$i(t_1) = i_H(v\,T_{pwm}) = e^{-v\,T_r}\left[i(t_0) + i_s(1-e^{v\,T_r})(\omega_r - 1)\right] \tag{4.13}$$

$$i(t_2) = i_L((1-v)\,T_{pwm}) = i(t_0) \tag{4.14}$$

$$i(t_0) = e^{-T_r}\left[i(t_0) - i_s(1-\omega_r - 2e^{v\,T_r} + e^{T_r}(1+\omega_r))\right]. \tag{4.15}$$

Then solve for $i(t_0)$ from (4.15):

$$i(t_0) = \frac{-i_s\left(1-\omega_r-2e^{vT_r}+e^{T_r}(1+\omega_r)\right)}{e^{T_r}-1} > 0. \tag{4.16}$$

Using the knowledge that $T_r > 0$ and $e^{T_r} > 1$, derive the domain for $i(t_0) > 0$:

$$e^{vT_r} > (e^{T_r}(1+\omega_r)+1-\omega_r)/2. \tag{4.17}$$

By plugging in the initial current $i(t_0)$ in (4.16) into (4.11), (4.13) and (4.12), the average current during one time-periodic PWM pulse can be solved for $u > 0$ and $i(t_0) > 0$ case:

$$\begin{aligned} i^+_{avg,i\neq 0} &= \left(\int_0^{vT_{pwm}} i_H(t_H)dt_H + \int_0^{(1-v)T_{pwm}} i_L(t_L)dt_L\right)/T_{pwm} \\ &= i_s(2v-1-\omega_r). \end{aligned} \tag{4.18}$$

Solving the equation for $u < 0$ and $i(t_0) < 0$ case using the same methodology yields the following average current:

$$i^-_{avg,i\neq 0} = -i_s(2v-1+\omega_r), \tag{4.19}$$

with the domain for where $i(t_0) < 0$ being:

$$e^{vT_r} > (e^{T_r}(1-\omega_r)+1+\omega_r)/2. \tag{4.20}$$

4.3.2 Case $i(t_0) = 0$

In this case, the current is fully drained during the low PWM signal, and the draining time must be derived in order to solve for the $i_L(t)$. Solve for the $u > 0$ case first by using (4.11),

(4.12) and (4.13) with $i(t_0) = 0$. Then solve for the time where the current is fully drained (t_d):

$$i_L(t_d) = i_s\, e^{-t_d/T_e}\left(2 + e^{-v\,T_r}(\omega_r - 1)\right) - i_s(1+\omega_r) = 0 \tag{4.21}$$

$$t_d = -T_e \log\left(\frac{e^{v\,T_r}\,(1+\omega_r)}{2e^{v\,T_r} - 1 + \omega_r}\right). \tag{4.22}$$

This log function must have a real solution for the domain where $i(t_0) = 0$. Using $v \in (0, 1]$, $\omega_r \in [-1, 1]$ and $T_r > 0$, the log function is valid everywhere except for $\omega_r = -1$. However $\omega_r = -1$ is always within the domain for $i(t_0) > 0$, so the log function is always valid. The current is fully drained during the low PWM signal, so $t_d \leq (1-v)T_{pwm}$ is set as a constraint. This constraint can be solved further into:

$$e^{v\,T_r} \leq (e^{T_r}(1+\omega_r) + 1 - \omega_r)/2, \tag{4.23}$$

which is the complete opposite of the domain for $i(t_0) > 0$ in (4.17). This means that the initial current $i(t_0) \geq 0$ covers for all $u > 0$ and $\omega_r \in [-1, 1]$. Finally, solve for the average current in one time-periodic PWM pulse for the $u > 0$ and $i(t_0) = 0$ case:

$$\begin{aligned} i^{+}_{avg,i_0=0} &= \left(\int_0^{vT_{pwm}} i_H(t_H)dt_H + \int_0^{t_d} i_L(t_L)dt_L\right)/T_{pwm} \\ &= \frac{i_s}{T_r}\left(v\,T_r(1-\omega_r) + (1+\omega_r)\log\left(\frac{e^{v\,T_r}(1+\omega_r)}{2e^{v\,T_r} - 1 + \omega_r}\right)\right). \end{aligned} \tag{4.24}$$

Using the same methodology to solve for the $u < 0$ and $i(t_0) = 0$ case yields the following average current:

$$\begin{aligned} i^{-}_{avg,i_0=0} = -\frac{i_s}{T_r}(&v\,T_r(1+\omega_r) \\ &+ (1-\omega_r)\log\left(\frac{e^{v\,T_r}(1-\omega_r)}{2e^{v\,T_r} - 1 - \omega_r}\right)), \end{aligned} \tag{4.25}$$

and the following domain where $i(t_0) = 0$:

$$e^{v\,T_r} \leq (e^{T_r}(1-\omega_r) + 1 + \omega_r)/2. \tag{4.26}$$

4.3.3 Summary

Let $v = |u| \in [0,1]$, $\omega_s = \text{sign}(u)\,\omega_r$, $\omega_r \in [-1,1]$, and the the domain for $i(t_0) \neq 0$ is:

$$\mathbf{V} = \left\{ v \,|\, v > \frac{1}{T_r} \log\left(\frac{1}{2}(e^{T_r}(1+\omega_s) + 1 - \omega_s)\right)\right\}. \tag{4.27}$$

Then average current i_{avg} of the time-periodic PWM pulse as a function of motor command u, scaled rotor speed ω_r, and a constant T_r can be calculated with the following algorithm:

if $v = 0$ **then**

$$i_{avg,v=0} = 0 \tag{4.28}$$

else if $v \in \mathbf{V}$ **then**

$$i_{avg,i_0\neq 0} = i_s\,(2u - \text{sign}(u) - \omega_r) \tag{4.29}$$

else

$$\begin{aligned} i_{avg,i_0=0} &= i_s\,(u\,(1-\omega_s) \\ &+ \left(\frac{\text{sign}(u)+\omega_r}{T_r}\right) \log\left(\frac{e^{v\,T_r}(1+\omega_s)}{2e^{v\,T_r} - 1 + \omega_s}\right)) \end{aligned} \tag{4.30}$$

end if

As shown above, the average current is nonlinear especially when $i(t_0) = 0$. The average current equations are defined using the i_s and unitless parameters ω_r, T_r, and u.

4.4 Real-Time Implementation Strategy

In order to be used in a controller, the motor command u must be solved given the target average current i_t and the measured rotor speed ω. Solving for u in the $i_0 \neq 0$ case from (4.29) is easy. However, as shown in the (4.30), the drive/coast model is nonlinear in the $i_0 = 0$ case, making it difficult to solve for $u(i_t, \omega_r)$ directly. Solving for $v = |u|$ in the $i(t_0) = 0$ case can be done by using scalar iterative methods such as the Newton-Raphson method [13]. Our tests have shown that this algorithm converges within 3 to 5 iterations which is quick enough to be used in real-time computations. The equations to be used in the Newton-Raphson method can be seen below:

$$f(v) = i_{avg,i_0=0} - i_t = 0, \tag{4.31}$$

$$f'(v) = \frac{2\, i_s\, (e^{v\, T_r} - 1)(\text{sign}(i_t) - \omega_r)}{2\, e^{v\, T_r} + \omega_s - 1} \tag{4.32}$$

$$v_{n+1} = v_n - f(v_n)/f'(v_n). \tag{4.33}$$

Then the method to solve for v can be outlined below:

1. $\text{sign}(u) = \text{sign}(i_t)$. If $i_t = 0$, then $v = 0$.
2. else, assume $i(t_0) \neq 0$ and solve for v using (4.29) then check if $v \in \mathbf{V}$ and $v \in (0, 1]$.
3. If $v \notin \mathbf{V}$ or $v \notin (0, 1]$, then we have the $i(t_0) = 0$ case and solve for v with Newton-Raphson method using (4.30) to (4.33). Set the initial guess v_0 to be the middle of the range of v for $i(t_0) = 0$:

$$v_0 = \tfrac{1}{2\, T_r} \log\left(\tfrac{1}{2}(e^{T_r}(1 + \omega_s) + 1 - \omega_s)\right). \tag{4.34}$$

In addition, there are two edge cases for the model in (4.27) to (4.30): the case of lim

Table 4.2: Experiment Motor and Motor Driver List

No.	Motors	Motor Drivers
1	Maxon A-Max 22 110160, 14:1	DRV8871 (3A i_{max})
2	Maxon 273688, 13mm, 17:1	DRV8881P (2.5A i_{max})
3	eduMIP motor, 6.6:1	MC33926 (5A i_{max})

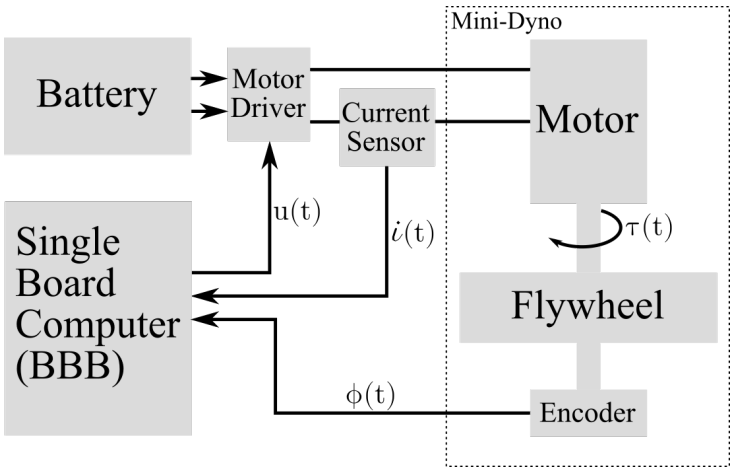


Figure 4.3: Experimental setup diagram of the dynamometer for the drive/coast model validation.

$T_r \to 0$ and lim $T_r \to \infty$ which represent very high and very low PWM frequency respectively. Solving the model using the limits and L'Hospital Rule yields the following results:

$$\text{Case lim } T_r \to 0: \qquad u = (i_t/i_s + \text{sign}(i_t) + \omega_r)/2 \tag{4.35}$$

$$\text{Case lim } T_r \to \infty: \qquad u = (i_t/i_s)/(1 - \text{sign}(i_t)\,\omega_r). \tag{4.36}$$

4.5 Model Validation Experiment

To validate our drive/coast model, we compared the estimated and actual current measurements of a motor-flywheel system (Mini-Dyno [14]). First we identify the motor parameters, shown in Table 4.3, using the procedure outlined in [14]. Using these parameters and a known

Table 4.3: Identified Motor Parameters

Parameters	**Motor No.**		
	1	2	3
Resistance, R (Ω)	6.49	15.4	9.06
Inductance, L (mH)	0.362	0.0494	2.36
Motor gain, k (N.m/A)	0.133	0.161	0.127

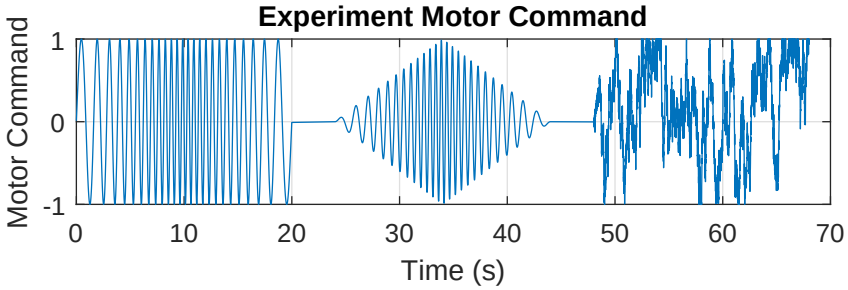


Figure 4.4: Plot of the motor command u signal for the experiment in the following sequence: sine chirp, ramping sine chirp, and random walk.

input signal we were able to estimate the drive/coast model and compare it to the actual output data.

4.5.1 Experimental Setup

The Mini-Dyno [14] is a low cost motor dynamometer that consists of a weighted flywheel of known inertia attached to a motor and an optical encoder (US Digital E6-2500-250). The motor is attached to a motor driver, a current sensor (INA219), and driven by a single board computer (Beaglebone Black), as shown in Fig. 4.3.

We conducted this experiment with three different motors and motor drivers which are listed in Table 4.2. The motor drivers were selected because they are commonly used in small ground robots, have off-the-shelf breakout boards, and feature drive/coast operating modes. Two different high quality Maxon motors with datasheets were selected to provide a benchmark of performance. The last motor is a low-cost toy motor used in the eduMIP robots with unknown

parameters. We selected this motor to demonstrate that the drive/coast model is accurate with low quality motors that have higher inductance and friction. The motors' resistance and torque constant were identified using the methodology outlined in [14] but with one important note: the identification must be done with a drive/brake motor driver to avoid the nonlinear drive/coast dynamics. We estimated the motor's inductance by measuring the rise time of the voltage across the shunt resistor on the current sensor with an oscilloscope. The identified motor parameters are shown in Table 4.3.

To excite the motor across a range of frequencies and amplitudes, the motor was driven by a sequence of open loop input signals: sine chirp, ramping sine chirp, and random walk, as shown in Fig. 4.4. The sine chirp is a sine signal with frequency ramping up from 0.5 Hz to 2 Hz and back down to 0.5 Hz. The ramping sine chirp is a sine chirp with varying amplitude. The random walk is a sum of a uniform random variable, $u_{k+1} = u_k + \mathrm{unif}(-0.3, 0.3)$, and the same random walk signal was used for each experiment. Each motor and motor driver pairs were tested at PWM frequencies of 500 Hz, 1 kHz, 5 kHz, 10 kHz and 20 kHz. The low range was chosen because 500 Hz is the default PWM frequency of certain microcontrollers (e.g. Arduino) and 20 kHz is ultrasonic and inaudible. During this experiment, the motor command u, current i, battery voltage V and the flywheel angle ϕ were recorded. From the input and output data, we estimated the current of the drive/coast system using the model from (4.27) to (4.30). In order to show the inaccuracy of using the drive/brake model with a coasting motor driver, we also estimated the current by using the linear model in (4.3) for motors with driver 1, such that:

$$i_{avg,lin} = (u\,V - k\,\omega)/R. \tag{4.37}$$

R^2 and RMSE values of the estimated drive/coast current with respect to the current measurement are used to compare model accuracy. The RMSE values are represented as a percentage of the motor stall current i_s.

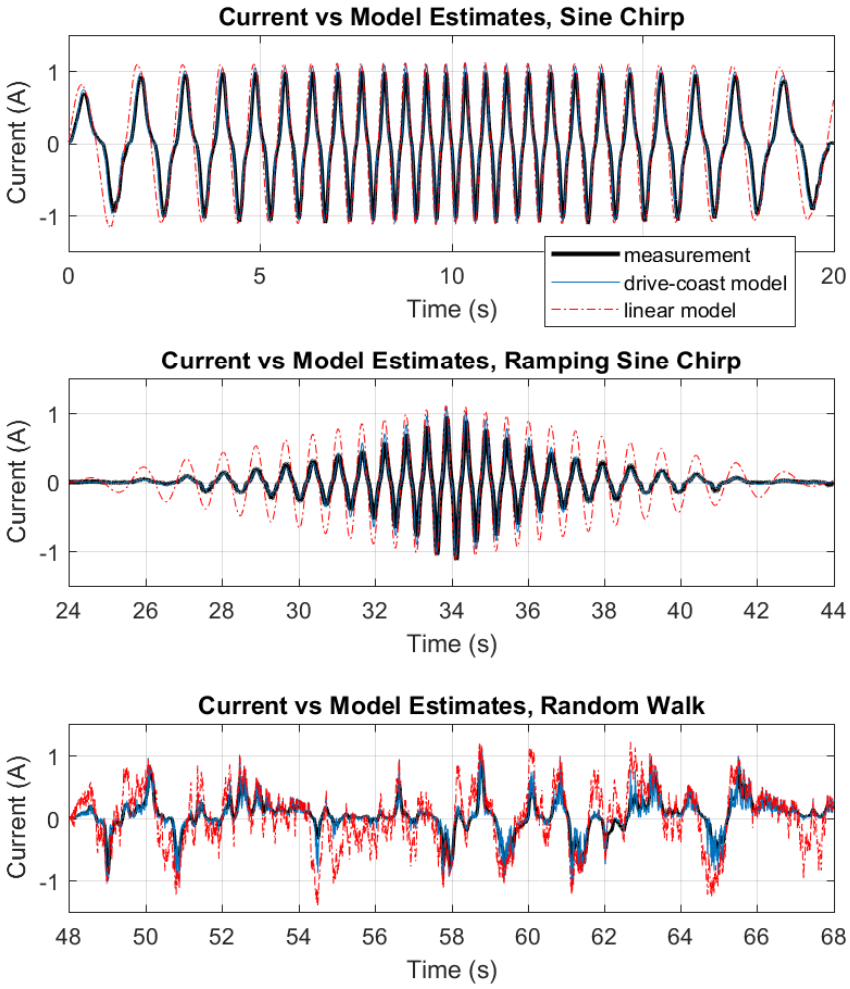


Figure 4.5: Plot of measured current vs. current estimate from the drive/coast model and the linear motor model. The data shown is of motor 1 & driver 1 at 20kHz PWM frequency.

4.5.2 Model Validation Results

The drive/coast model has high R^2 values and low RMSE values, as shown in and Table 4.4 and 4.5 respectively, which indicates an accurate model. The drive/coast model has an average RMSE error of 6.5% i_s while the linear model has an average RMSE error of 22.5% i_s. From Fig. 4.5, we can see how much more accurate the drive/coast model is compared to the linear model especially at low duty cycles. As shown in Table 4.4 and 4.5, the drive/coast model is accurate and consistent for all PWM frequencies, motor, and motor driver combinations.

Table 4.4: R^2 Values of the Current Measurement vs. Model Estimate

Motor & Mtr. Driver	**PWM Frequency (Hz)**				
	500	1000	5000	10000	20000
vs. coast model:					
Motor 1 & driver 1	0.964	0.969	0.980	0.964	0.962
Motor 1 & driver 2	0.971	0.973	0.973	0.650	0.937
Motor 1 & driver 3	0.987	0.986	0.984	0.972	0.964
Motor 2 & driver 1	0.982	0.985	0.982	0.980	0.973
Motor 2 & driver 2	0.979	0.979	0.973	0.968	0.959
Motor 2 & driver 3	0.986	0.985	0.984	0.982	0.973
Motor 3 & driver 1	0.962	0.967	0.949	0.944	0.929
Motor 3 & driver 2	0.965	0.959	0.929	0.902	0.865
Motor 3 & driver 3	0.982	0.974	0.965	0.954	0.913
vs. linear model:					
Motor 1 & driver 1	0.753	0.746	0.642	0.470	0.413
Motor 2 & driver 1	0.841	0.844	0.740	0.628	0.516
Motor 3 & driver 1	0.783	0.713	0.358	0.315	0.244

However, the low quality eduMIP motor (motor 3) has an increased error as PWM frequency increases. We believe that this might have been caused by inaccuracies in some parameter values, or nonlinearities such as static friction and backlash which are commonly seen problems in low cost motors. However, the drive/coast model still yielded a significantly more accurate estimates than the linear model.

4.6 Real-Time Implementation Experiment

We used an eduMIP to demonstrate the viability of our drive/coast model on an unstable, real-time system. A MIP balancing about it's equilibrium point requires small amount of torque to stabilize the system. Since a drive/coast system is highly nonlinear about low duty cycles, a MIP robot is a perfect test platform.

Table 4.5: RMSE Values of the Current Measurement vs. Model Estimate as a Percentage of Stall Current i_s

Motor & Mtr. Drv.	**PWM Frequency (Hz)**				
	500	1000	5000	10000	20000
vs. coast model:					
Motor 1 & driver 1	7.72%	6.70%	5.12%	5.44%	6.48%
Motor 1 & driver 2	6.95%	6.61%	5.76%	6.16%	7.69%
Motor 1 & driver 3	4.58%	4.65%	4.38%	5.47%	5.98%
Motor 2 & driver 1	6.16%	5.56%	5.39%	5.33%	5.93%
Motor 2 & driver 2	6.33%	6.41%	6.46%	6.50%	6.92%
Motor 2 & driver 3	5.16%	5.07%	4.85%	4.75%	5.65%
Motor 3 & driver 1	7.69%	6.73%	7.57%	7.89%	8.78%
Motor 3 & driver 2	5.25%	5.70%	6.27%	7.62%	11.4%
Motor 3 & driver 3	6.88%	6.82%	7.88%	8.89%	10.1%
vs. linear model:					
Motor 1 & driver 1	20.3%	20.1%	21.9%	23.9%	25.6%
Motor 2 & driver 1	18.5%	17.9%	20.6%	22.9%	25.2%
Motor 3 & driver 1	18.4%	20.0%	26.7%	27.6%	28.6%

4.6.1 Experimental Setup

We fitted an eduMIP with two DRV8871 motor drivers which can toggle between drive/brake and drive/coast mode. We used a state feedback controller designed for MIP with drive/brake motor drivers for this experiment. Using the same balancing controller, we allowed the robot to balance in place under three modes: drive/brake, and drive/coast with and without coasting model compensation. The coasting model compensation adjusts the PWM duty cycle given by the drive/brake controller to the equivalent duty cycle for drive/coast systems, as discussed in Section 4.4. Assuming that the coasting model is correct, then the control torque of the drive/coast with compensation should be the same as the drive/brake, resulting in a similar balancing performance. The balancing performance under each mode was evaluated by

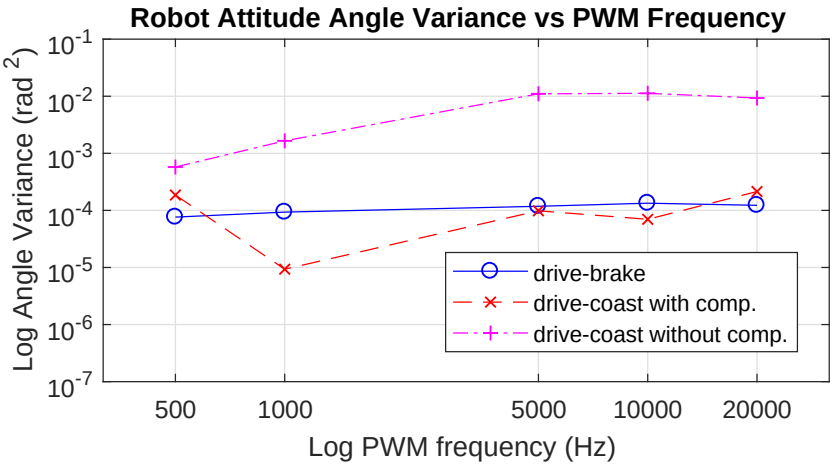


Figure 4.6: Plot of the robot attitude angle variance between drive/brake mode, and drive/coast mode with and without compensation.

calculating the variance of the robot's attitude angle during the position hold.

4.6.2 Experimental Results

The drive/coast system performance is significantly improved by implementing the coasting compensation as shown in Fig. 4.6. With compensation, the drive/coast system has approximately the same performance as the drive/brake system across all tested PWM frequencies, except at PWM frequency of 1000 Hz. We concluded that the compensated PWM duty cycle in this particular frequency hit the sweet spot for small amplitude but high frequency limit cycle for the robot. Without any compensation, the drive/coast system is significantly less stable especially as the PWM frequency increases. This demonstrates that our drive/coast model is accurate and we can effectively compensate the control signal of a drive/coast system in real-time.

In this chapter, we mathematically derived a novel model for drive/coast systems and demonstrated its accuracy with real-world experiments. In addition, we proposed a real-time implementation method of the model by using the target current and rotor speed to calculate a motor command which compensates for the nonlinearies of a drive/coast system. Finally, we

demonstrated the performance of the drive/coast model and coasting compensation in real time on an eduMIP which showed a significant improvement in stability. This new model opens up the possibility of designing more accurate system dynamic models when using drive/coast motor drivers to offer different system behaviors and advantages compared to its drive/brake counterpart.

Although we showed the effectiveness of the model on a MIP robot, we suggest to experiment further with different types of robots or controls applications that can take advantage of free-spinning motors. In addition, although coasting motor drivers can recharge the battery, we believe that more experimentation should be done to compare overall energetic efficiency between drive/brake and drive/coast motor drivers in low torque and speed applications. Regardless of efficiency, in practice, some robotic systems are bound to use drive/coast motor drivers, and our work provides an accurate method of modeling the dynamics of such drivers.

Acknowledgements

Chapter 4, in full, uses the material as it may appear in the paper submitted for publication to IEEE International Conference on Automation Science and Engineering (CASE) 2019, E. Sihite, D. Yang, and T. Bewley, "Derivation of a new drive/coast motor driver model for real-time brushed DC motor control, and validation on a MIP robot". The dissertation author was the primary investigator and author of this paper, contributed to the development of the novel model and control algorithm, experimentation, and data analysis.

References

[1] B. Williams, *Power Electronics: Devices, Drivers, Applications, and Passive Components*. University of Strathclyde, Glasgow: Online, 2013, pp. 348–411.

[2] G. C. Ioannidis, C. S. Psomopoulos, and et. al., “Ac-dc & dc-dc converters for dc motor drives,” *International Conference on Electronics and Communication Systems*, 2013.

[3] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning and Control*. Cambridge University Press, 2017.

[4] J. Braun, *Formulae Handbook*. Maxon Academy, Sachseln, 2012, pp. 39–46.

[5] U. Kafader, *The selection of high-precision microdrives*. Maxon Academy, Sachseln, 2012.

[6] R. J. Schilling, *Fundamental of Robotics: Analysis and Controls*. Prentice Hall, 1990, pp. 268–271.

[7] X. Nian, F. Peng, and H. Zhang, "Regenerative braking system of electric vehicle driven by brushless dc motor," *IEEE Transactions on Industrial Electronics 61.10*, pp. 5798–5808, 2014.

[8] H. Wu, S. Cheng, and S. Cui, "A controller of brushless dc motor for electric vehicle," *IEEE Transactions on magnetics 41.1*, pp. 509–513, 2005.

[9] M. K. Yoong and et al., "Studies of regenerative braking in electric vehicle," *IEEE Conference on Sustainable Utilization and Development in Engineering and Technology*, 2010.

[10] M. Yang, H. Jhou, B. Ma, and K. Shyu, "A cost-effective method of electric brake with energy regeneration for electric vehicles," *IEEE Transactions on Industrial Electronics 56.6*, pp. 2203–2212, 2009.

[11] N. C. Braga, *Robotics, Mechatronics, and Artificial Intelligence: Experimental Circuit Blocks for Engineers*. Newnes, 2002.

[12] J. E. Carryer, R. M. Ohline, and T. W. Kenny, *Introduction to Mechatronic Design*. Pearson, 2011.

[13] T. Bewley, *Numerical Renaissance: simulation, optimization, and control*. Renaissance Press, San Diego, 2015.

[14] N. Morozovsky, R. Moroto, and T. Bewley, "Rapid: An inexpensive open source dynamometer for robotics applications," *IEEE/ASME Transactions on Mechatronics 18.6*, pp. 1855–1860, 2013.

Chapter 5

MBBR High Yaw-Rate Dynamic Modeling

The state estimation and control of a ball-balancing robot under high yaw rate is a challenging problem due to its highly nonlinear 3D dynamic. The small size and low-cost components in our Micro Ball-Balancing Robot makes the system inherently very noisy and has nontrivial friction, which further increases the complexity of the problem. In particular, the double-rows omniwheels can have microslips when transitioning between the two rows of the wheel's rollers, adding vibrations and noise into the measurements. In order to drive the robot more aggressively such as translating and spinning at the same time, a good state estimator which works well under high yaw rates and noisy system is required. This motivated us to develop an EKF which uses a high yaw-rate dynamic model, which if worked well can improve the state estimation accuracy and stability under high yaw-rates.

Past works in this topic include the indirect Kalman Filtering used by Rezero [1]. In their work, the states were estimated using Kalman Filter (KF) from the kinematic relationship between the IMU and the encoder measurements. The formulation of a BBR 3D dynamic model has been done by [2] and [3]. However, [2] did not use the model in an estimator or controller. The KF used in [3] was not explained in detail and they assumed that robot's attitude angles can be directly measured. The linearized BBR dynamic model was used by most of the existing BBRs where the model was simplified down into two decoupled Mobile Inverted Pendulum (MIP) problems in the x-y plane (roll and pitch). The MIP dynamic model is well known and

can easily be linearized [4], and used in linear controllers and estimators. This was also the controller used in the initial MBBR prototypes and this controller has poor stability when the robot starts spinning. As shown in Chapter 3 , the attitude estimation accuracy can be improved by using a high yaw-rate dynamic model in an Extended Kalman Filter (EKF) [5]. Implementing the same methodology for the MBBR might also improve the state estimation accuracy which means better stability under nontrivial yaw-rates. The implementation of a high yaw-rate BBR dynamic model in an EKF using on board raw measurement data is novel and the methodology is extensible to other lightweight low-cost robots. The EKF is chosen instead of the other nonlinear KFs, such as Unscented KF, because it is simple to implement. The Jacobian of the high yaw-rate model is also easy to calculate which makes the implementation of the EKF easier than other nonlinear KFs.

This chapter presents the derivation of a high yaw-rate Ball-Balancing Robot dynamic model and the simplified model which will be used in an Extended Kalman Filter (EKF). The controller and estimator design, together with the motion capture experiment to verify the estimator and controller performance are shown in Chapter 6. This chapter is outlined as follows: MBBR frame of references, kinematic modeling, Lagrangian dynamic modeling, and the simplified model which will be used for controller and estimator design.

5.1 Ball-Balancing Robot Model Derivations

This section contains the derivations of the BBR dynamic equation of motion using Lagrangian Dynamics, which follows similar derivations done by Hoshino [6]. The equation of motion is derived symbolically in 'Wolfram Mathematica 10.0', which then is simplified into the high yaw-rate model used in the EKF. The list of parameters and variables used in the derivations can be seen in Table 5.1.

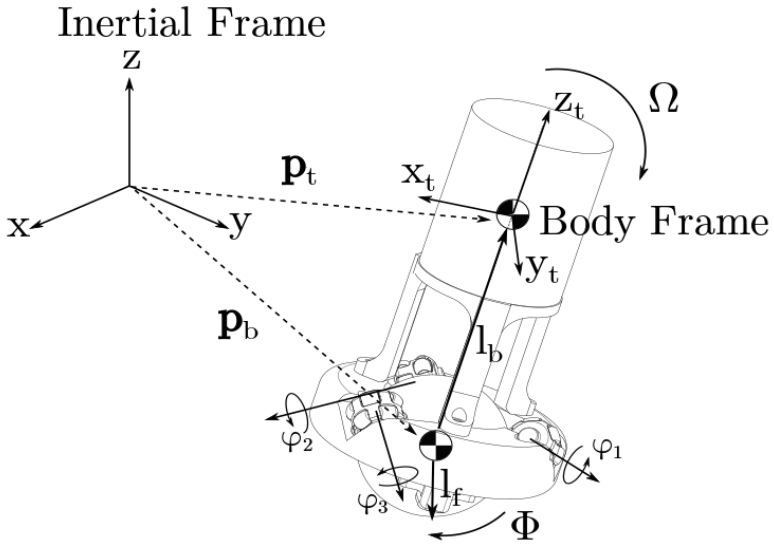


Figure 5.1: Diagram of the frame of references, some position vectors, and angular speed.

5.1.1 Frames of Reference and Wheel Transformation

The frames of reference used in the derivations are the inertial and the body frames, as shown in Fig. 5.1. The variables defined in the body frame are represented by using the superscript B. The transformation from body to inertial frame is shown in (5.1) below:

$$\begin{gathered} \boldsymbol{x} = R_B(\boldsymbol{\theta})\, \boldsymbol{x}^B, \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_x & \theta_y & \theta_z \end{bmatrix}^T \\ R_B(\boldsymbol{\theta}) = R_z(\theta_z)\, R_y(\theta_y)\, R_x(\theta_x), \end{gathered} \tag{5.1}$$

where $R_x(\theta_x)$, $R_y(\theta_y)$ and $R_z(\theta_z)$ are the rotation matrices of the Euler rotations about the inertial frame's x, y and z axis respectively. Then $R_B(\boldsymbol{\theta})$ is the body's intrinsic rotation in the body's z-y'-x'' axis (intrinsic yaw-pitch-roll), which is one of the standard Tait-Bryan angles.

The omniwheels in the MBBR are aligned perpendicularly as shown in Chapter 2, Fig. 2.9. α is the omniwheel contact angle from the ball's north pole about the body frame and β is the omniwheel's tilt angle about the wheel's axis perpendicular to the surface of the ball. The normalized vectors $\hat{\boldsymbol{w}}^i$, $i = \{1, 2, 3\}$, represent the omniwheel's spinning axis and the direction

Table 5.1: List of parameters and time varying variables. CoM = Center of Mass, CoR = Center of Rotation, which is also the ball's CoM.

Parameter List	**Time Varying Variable List**, $i = \{1, 2, 3\}$
m_t = top body mass.	θ_x = roll angle.
m_b = ball mass.	θ_y = pitch angle.
$\hat{I}_t$ = top body inertia about CoM.	θ_z = yaw angle.
$\hat{I}_b$ = ball inertia about CoM.	ϕ_x, ϕ_y, ϕ_z = ball rotation angles.
I_w = wheel inertia about its CoR.	u_x, u_y, u_z = control commands in body frame.
r = ball radius.	φ_i = encoder i rotation.
r_w = omniwheel radius.	u_i = motor i PWM command $\in [-1, 1]$.
l = length of body's CoM from CoR.	τ_i = motor i torque.
k_1 = motor torque gain.	$\boldsymbol{p}_t$ = top body CoM position vector.
k_2 = motor back EMF gain.	$\boldsymbol{p}_b$ = ball CoM position vector.
g = gravity constant.	$\boldsymbol{l}_b$ = length vector from CoR to body's CoM.
	$\boldsymbol{l}_f$ = length vector from CoR to the ground.
	$\boldsymbol{l}_a$ = length vector from CoR to the IMU.

of the torque τ_i applied by the motor i. The optical encoder attached on the motor i measures the angle φ_i which is the ball rotation angle relative to the body. This measurement can be used to determine the ball inertial rotation angle $\boldsymbol{\phi}$. For clarity, the variable φ is used for the rotation angles from the encoder measurement while ϕ is used for the ball rotation angles about the inertial frame. Assuming that there is no slip between the omniwheels and the ball, the applied torque and encoder measurement at omniwheel i in body coordinates are $(r/r_w)\tau_i\hat{\boldsymbol{w}}_i$ and $(r_w/r)\varphi_i\hat{\boldsymbol{w}}_i$ respectively. Let $\boldsymbol{\varphi}^B$ be the ball rotation angle as measured by the encoders and $\boldsymbol{\tau}^B$ be the total torque applied to the ball. Then $\boldsymbol{\varphi}^B$ and $\boldsymbol{\tau}^B$ can be calculated from τ_i and φ_i as shown below:

$$\boldsymbol{\varphi}_w = \begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 \end{bmatrix}^T, \quad \boldsymbol{\tau}_w = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 \end{bmatrix}^T \tag{5.2}$$

$$\boldsymbol{\varphi}^B = (r_w/r) \sum_{i=1}^{3} \varphi_i \hat{\boldsymbol{w}}^i = (r_w/r)\, T_{ob}\, \boldsymbol{\varphi}_w \tag{5.3}$$

$$\boldsymbol{\tau}^B = (r/r_w) \sum_{i=1}^{3} \tau_i \hat{\boldsymbol{w}}^i = (r/r_w)\, T_{ob}\, \boldsymbol{\tau}_w, \tag{5.4}$$

where the matrix $T_{ob} = \left[\hat{\boldsymbol{w}}^1|\hat{\boldsymbol{w}}^2|\hat{\boldsymbol{w}}^3\right]$ is the transformation matrix from the omniwheel axis to the body frame. The omniwheels are placed perpendicularly from each other, therefore $(T_{ob})^{-1} = (T_{ob})^T$.

5.1.2 Kinematic Formulation

This section derives the body and the ball kinematic equations which will be used for deriving the kinetic and potential energy of the system. The inertia of the body and the ball, represented by the matrix $\hat{I}_t$ and $\hat{I}_b$ respectively, are shown below:

$$\hat{I}_t^B = \mathrm{diag}(I_{t1}, I_{t2}, I_{t3}) \qquad\qquad \hat{I}_b = I_b\, I_{3\times 3}. \tag{5.5}$$

The length vectors used in the derivation are defined below:

$$\boldsymbol{l}_b^B = \begin{bmatrix} 0 & 0 & l \end{bmatrix}^T, \qquad\qquad \boldsymbol{l}_f = \begin{bmatrix} 0 & 0 & -r \end{bmatrix}^T. \tag{5.6}$$

The body rotational speed $\boldsymbol{\Omega}$ about the inertial frame:

$$\boldsymbol{\Omega} = R_z(\theta_z)\, R_y(\theta_y) \begin{bmatrix} \dot{\theta}_x \\ 0 \\ 0 \end{bmatrix} + R_z(\theta_z) \begin{bmatrix} 0 \\ \dot{\theta}_y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_z \end{bmatrix}. \tag{5.7}$$

The ball rotational speed $\boldsymbol{\omega}$ about the inertial frame:

$$\boldsymbol{\omega} = \begin{bmatrix} \dot{\phi}_x & \dot{\phi}_y & \dot{\phi}_z \end{bmatrix}^T = R_B(\boldsymbol{\theta})\, \dot{\boldsymbol{\varphi}}^B + \boldsymbol{\Omega}. \tag{5.8}$$

The ball linear velocity $\dot{\boldsymbol{p}}_b$ can be derived by using the no slip conditions between the ball and

the ground, as shown below:

$$\dot{\boldsymbol{p}}_b = \boldsymbol{l}_f \times \boldsymbol{\omega}, \qquad \dot{\phi}_z = 0. \tag{5.9}$$

This constraint is applied here in order to avoid the use of Lagrange Multiplier in the dynamic equation. This no slip condition also constrains the $\dot{\phi}_z$ to zero, which affects the encoder measurement $\dot{\boldsymbol{\varphi}}$ in (5.8). Then $\dot{\boldsymbol{\varphi}}$ can be expressed in terms of $\dot{\phi}_x$ and $\dot{\phi}_y$ as shown below:

$$\dot{\boldsymbol{\varphi}}^B = R_B(\boldsymbol{\theta})^T \left(\begin{bmatrix} \dot{\phi}_x & \dot{\phi}_y & 0 \end{bmatrix}^T - \boldsymbol{\Omega} \right). \tag{5.10}$$

Finally, we have the body linear velocity:

$$\dot{\boldsymbol{p}}_t = \tfrac{d}{dt} \left(R_B\left(\boldsymbol{\theta}\right) \boldsymbol{l}_b^B \right) + \dot{\boldsymbol{p}}_b. \tag{5.11}$$

5.1.3 Motor Dynamics

The back electromotive force (EMF) from the DC brushed motors used in the robot also contributes to the system dynamic. The torque applied by each motor is:

$$\tau_i = k_1 u_i - k_2 \dot{\varphi}_i, \qquad i = 1, 2, 3, \tag{5.12}$$

where $u_i = [-1, 1]$ is the PWM command into the motor. Using the transformations in (5.3) and (5.4) into (5.12) yields:

$$\boldsymbol{\tau}^B = k_1 \boldsymbol{u} - k_2 \left(r/r_w\right)^2 \dot{\boldsymbol{\varphi}}^B \tag{5.13}$$

$$\boldsymbol{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T = \left(r/r_w\right) T_{ob} \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T. \tag{5.14}$$

5.1.4 Lagrangian Dynamics Formulation

The BBR dynamic equation is derived using Lagrangian Dynamics which then simplified and transformed into the state space form. The energy equations for the Lagrangian are derived for the top body, the ball and the wheels. The potential and the linear kinetic energy of the wheels are assumed to be negligible compared to the body and the ball due to their small mass. However, the no slip constraint between the ball and the wheels creates a coupling. This coupling and the fast wheel speed may cause a nontrivial increase in the rotational energy. In order to keep the equations simple and prevent cross terms between the $\boldsymbol{\Omega}$ and $\boldsymbol{\omega}$, we assume that $\boldsymbol{\Omega}$ is much smaller than the wheel speed $\dot{\varphi}_i$. Then the wheel i's angular velocity vector $\boldsymbol{\omega}_{wi}$ is shown below:

$$\boldsymbol{\omega}_{wi} = \boldsymbol{\Omega} + \dot{\varphi}_i \, \hat{\boldsymbol{w}}^i \approx \dot{\varphi}_i \, \hat{\boldsymbol{w}}^i. \tag{5.15}$$

Let I_w be the wheel inertia about its axis of rotation. Then the total angular kinetic energy of the wheels K_w is:

$$\begin{aligned} K_w &= \sum_{i=1}^{3} \frac{I_w}{2} \boldsymbol{\omega}_{wi}^T \, \boldsymbol{\omega}_{wi} = \frac{I_w}{2} \dot{\boldsymbol{\varphi}}_w^T \, \dot{\boldsymbol{\varphi}}_w \\ &= (I_w/2)(r/r_w)^2 (\dot{\boldsymbol{\varphi}}^B)^T \dot{\boldsymbol{\varphi}}^B. \end{aligned} \tag{5.16}$$

The kinetic and potential energy of the body and the ball are:

$$\begin{aligned} K_t &= \tfrac{1}{2} (R_B^T \, \boldsymbol{\Omega})^T \, \hat{I}_t^B \, (R_B^T \, \boldsymbol{\Omega}) + \tfrac{1}{2} \, m_t \, \dot{\boldsymbol{p}}_t^T \, \dot{\boldsymbol{p}}_t \\ K_b &= \tfrac{1}{2} \, \boldsymbol{\omega}^T \, \hat{I}_b \, \boldsymbol{\omega} + \tfrac{1}{2} \, m_b \, \dot{\boldsymbol{p}}_b^T \, \dot{\boldsymbol{p}}_b \\ U_t &= -m_t \begin{bmatrix} 0 & 0 & -g \end{bmatrix} \boldsymbol{p}_t, \qquad U_b = 0. \end{aligned} \tag{5.17}$$

The following states $\boldsymbol{x}(t)$ are used in the dynamic equation:

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}, \qquad \boldsymbol{q} = \begin{bmatrix} \theta_x & \theta_y & \theta_z & \phi_x^* & \phi_y^* \end{bmatrix}^T, \tag{5.18}$$

where $\dot{\phi}_x^*$ and $\dot{\phi}_y^*$ are the inertial frame ball rotation speeds rotated about the inertial z-axis by θ_z as shown below:

$$\dot{\phi}_x^* = \cos(\theta_z)\,\dot{\phi}_x + \sin(\theta_z)\,\dot{\phi}_y \tag{5.19}$$

$$\dot{\phi}_y^* = -\sin(\theta_z)\,\dot{\phi}_x + \cos(\theta_z)\,\dot{\phi}_y. \tag{5.20}$$

By choosing ϕ_x^* and ϕ_y^* as the states, all $\sin(\theta_z)$ and $\cos(\theta_z)$ terms are eliminated after the simplification in Section 5.1.6. Then the Lagrangian of the system is:

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = K_t + K_b + K_w - U_t. \tag{5.21}$$

The system dynamic equation can be solved by using the Lagrange's Equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} - \frac{\partial L}{\partial \boldsymbol{q}} - \boldsymbol{\tau}_L = \boldsymbol{0}, \tag{5.22}$$

where $\boldsymbol{\tau}_L$ is the total generalized force applied by all the motors through the omniwheels. $\boldsymbol{\tau}_L$ is defined as follows:

$$\tau_{Lj} = (\boldsymbol{\tau}^B)^T \left(\frac{\partial \dot{\boldsymbol{\varphi}}^B}{\partial \dot{q}_j}\right), \qquad j = \{1, 2, 3, 4, 5\}, \tag{5.23}$$

where τ_{Lj} and $\dot{q}_j$ are the j-th component of $\boldsymbol{\tau}_L$ and $\dot{\boldsymbol{q}}$ respectively. The no slip constraints are already applied during the kinematic formulations, so there is no Lagrange Multiplier due to system constraints. The acceleration states $\ddot{\boldsymbol{q}}$ can be derived by forming (5.22) into the standard

form:

$$M(\boldsymbol{q})\,\ddot{\boldsymbol{q}} = \boldsymbol{b}(\dot{\boldsymbol{q}}, \boldsymbol{q}, \boldsymbol{u}) \tag{5.24}$$

The system dynamic equation $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ can then be calculated by solving for $\ddot{\boldsymbol{q}} = M(\boldsymbol{q})^{-1}\,\boldsymbol{b}(\dot{\boldsymbol{q}}, \boldsymbol{q}, \boldsymbol{u})$. The full symbolic form of the (5.24) can be seen in the Appendix A.2. This equation needs to be simplified before being used in the EKF because of its length and nonlinearities.

5.1.5 Sensor Dynamics

The MBBR uses the following sensors to estimate the states: the optical encoders and IMU gyrometer and accelerometer measurements. In particular, the accelerometer is greatly affected by the dynamic of the robot. Therefore, we need to derive the sensor dynamics before they can be used in the EKF. The encoders measure the ball rotation angle with respect to the body frame ($\boldsymbol{y}_{en}^{B} = \boldsymbol{\varphi}^{B}$), where $\dot{\boldsymbol{\varphi}}^{B}$ was derived in (5.10). The gyrometer measures the body angular velocity ($\boldsymbol{y}_{gy}^{B} = \boldsymbol{\Omega}^{B}$) which was derived in (5.7). The accelerometer measures the linear acceleration at the IMU's position about the body frame. Let $\boldsymbol{p}_a$ be the position of the IMU in the inertial frame and $\boldsymbol{l}_a^B = [l_{ax}, l_{ay}, l_{az}]^T$ is the length vector from ball's center of mass to the IMU. Then using a similar kinematics derivation to (5.11), we can derive the accelerometer measurement dynamics below:

$$\dot{\boldsymbol{p}}_a = \tfrac{d}{dt}\left(R_B(\boldsymbol{\theta})\,\boldsymbol{l}_a^B\right) + \dot{\boldsymbol{p}}_b \tag{5.25}$$

$$\boldsymbol{y}_{ac}^{B} = R_B(\boldsymbol{\theta})^T\left(\begin{bmatrix}0 & 0 & -g\end{bmatrix}^T - \ddot{\boldsymbol{p}}_a\right). \tag{5.26}$$

The acceleration components of $\ddot{\boldsymbol{p}}_a$ can be derived from the system dynamic equation $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ solved in the Lagrangian Dynamics Section above. Nonzero l_{ax} and l_{ay} can cause a bias in the accelerometer measurement due to the centripetal force. However, we assume that the l_{ax} and l_{ay} values are zero in order to keep the equations simple.

5.1.6 Simplifications

The dynamic equation and sensor dynamics must be simplified before they can be used in the EKF due to the sheer size and nonlinearities in the equations. We can use the linear BBR model's assumptions without the trivial yaw rate assumption. Therefore, we assume that θ_x, θ_y, $\dot{\theta}_x$, and $\dot{\theta}_y$ are small. This assumption works under the knowledge that a stable BBR system has small perturbations on these variables. Then we can use the small angle approximation for the sine and cosine functions such that $\sin(\theta) \approx \theta$, $\cos(\theta) \approx 1$ for both θ_x and θ_y. Also, all of the multiplications between θ_x, θ_y, $\dot{\theta}_x$, and $\dot{\theta}_y$, or with themselves are approximately equal to zero (e.g. $\theta_x\,\theta_x \approx 0, \dot{\theta}_x\,\theta_y \approx 0$). If ϕ_x^* and ϕ_y^* are used as the states, then there is no $\sin(\theta_z)$ and $\cos(\theta_z)$ left in the dynamic equation. The high yaw-rate model can't be simplified further, but the linear model can be derived from here by using $\dot{\theta}_z = 0$ and $u_z = 0$.

5.2 MBBR Numerical Model

This section shows the MBBR numerical model by plugging in all the corresponding parameter values shown in Table 5.2 into the simplified model. This will be the model used for developing the controller and the state estimators. The high yaw-rate model is the linear model with some additional nonlinear terms, so the linear model is derived first in the following subsection.

5.2.1 Linear MBBR Model

The linear MBBR model can be simplified into two decoupled MIP systems about the roll and pitch. The yaw dynamics can't be simplified into a linear equation just from the small yaw-rate assumption, so all of the yaw dynamic components are ignored in this model. The yaw rate can be measured by the gyrometer while the robot's heading must be estimated by using other means. A simple gyrometer integration or magnetometer measurement can be used to

Table 5.2: MBBR Parameter Values.

Parameter	**Value**	**Parameter**	**Value**
m_t	550 g	I_{t1}	2.42 10^{-3} kg.m^2
m_b	150 g	I_{t2}	2.42 10^{-3} kg.m^2
r	32 mm	I_{t3}	0.79 10^{-3} kg.m^2
r_w	12.5 mm	I_b	8.46 10^{-5} kg.m^2
l	100 mm	I_w	1.20 10^{-5} kg.m^2
l_{ax}	0 mm	k_1	0.176 N.m
l_{ay}	0 mm	$k_2(r/r_w)^2$	0.011 N.m.s/rad
l_{az}	130 mm	g	9.8 m/s^2
dt	0.005 s		

estimate the robot's heading. In our experiements, we used the IMU's DMP heading estimate for controlling the linear model Kalman Filter's heading.

The linear model uses the following states, input and output vectors respectively:

$$\begin{aligned} \boldsymbol{x}^L &= [\theta_x, \theta_y, \phi_x^*, \phi_y^*, \dot{\theta}_x, \dot{\theta}_y, \dot{\phi}_x^*, \dot{\phi}_y^*]^T \\ \boldsymbol{u}^L &= [u_x, u_y]^T \\ \boldsymbol{y}^L &= [\Omega_x^B, \Omega_y^B, \varphi_x^B, \varphi_y^B, y_{ac1}^B, y_{ac2}^B]^T. \end{aligned} \tag{5.27}$$

Then by using the parameter values listed in Table 5.2, the continuous time linear dynamic model for the MBBR is:

$$\dot{\boldsymbol{x}}^L = \boldsymbol{f}^L(\boldsymbol{x}^L, \boldsymbol{u}^L)$$

$$\begin{aligned} f_1^L &= \dot{\theta}_x, \quad f_2^L = \dot{\theta}_y, \quad f_3^L = \dot{\phi}_x^*, \quad f_4^L = \dot{\phi}_y^* \\ f_5^L &= 130\,\theta_x - 8.9\,\dot{\theta}_x + 8.9\,\dot{\phi}_x^* - 130\,u_x \\ f_6^L &= 130\,\theta_y - 8.9\,\dot{\theta}_y + 8.9\,\dot{\phi}_y^* - 130\,u_y \\ f_7^L &= -280\,\theta_x + 34\,\dot{\theta}_x - 34\,\dot{\phi}_x^* + 410\,u_x \\ f_8^L &= -280\,\theta_y + 34\,\dot{\theta}_y - 34\,\dot{\phi}_y^* + 410\,u_y, \end{aligned} \tag{5.28}$$

where f_i^L is the i-th component of the vector $\boldsymbol{f}^L$. The sensor dynamic equation for this model is:

$$\begin{aligned}
\boldsymbol{y}^L &= \boldsymbol{h}^L(\boldsymbol{x}^L, \boldsymbol{u}^L) \\
h_1^L &= \dot{\theta}_x, \qquad h_2^L = \dot{\theta}_y \\
h_3^L &= \phi_x^* - \theta_x, \quad h_4^L = \phi_y^* - \theta_y \\
h_5^L &= -4.6\,\theta_y + 0.5\,\dot{\theta}_y - 0.5\,\dot{\phi}_y^* + 7.8\,u_y \\
h_6^L &= 4.6\,\theta_x - 0.5\,\dot{\theta}_x + 0.5\,\dot{\phi}_x^* - 7.8\,u_x,
\end{aligned} \tag{5.29}$$

where h_i^L is the i-th component of the vector $\boldsymbol{h}^L$.

5.2.2 High Yaw-Rate MBBR Model

The high yaw-rate model uses the following states, input and output vectors respectively:

$$\begin{aligned}
\boldsymbol{x}^N &= [\theta_x, \theta_y, \theta_z, \phi_x^*, \phi_y^*, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z, \dot{\phi}_x^*, \dot{\phi}_y^*, x_{11}^N, x_{12}^N, x_{13}^N]^T \\
\boldsymbol{u}^N &= [u_x, u_y, u_z]^T \\
\boldsymbol{y}^N &= [\Omega_x^B, \Omega_y^B, \Omega_z^B, \varphi_x^B, \varphi_y^B, \varphi_z^B, y_{ac1}^B, y_{ac2}^B]^T.
\end{aligned} \tag{5.30}$$

The additional states x_{11}^N through x_{13}^N in (5.30) are the augmented states used for the encoder measurement dynamics. The $\dot{\boldsymbol{\varphi}}^B$ defined in (5.8) needs to be integrated to derive the encoder measurement $\boldsymbol{\varphi}^B$ which is represented by these additional states. The encoder measurement dynamics after the simplification are shown below:

$$\boldsymbol{\varphi}^B = \begin{bmatrix} \varphi_x^B \\ \varphi_y^B \\ \varphi_z^B \end{bmatrix} = \begin{bmatrix} \phi_x^* - \theta_x + \int(\theta_y\,\dot{\theta}_z)dt \\ \phi_y^* - \theta_y - \int(\theta_x\,\dot{\theta}_z)dt \\ -\theta_z + \int(\theta_y\,\dot{\phi}_x^* - \theta_x\,\dot{\phi}_y^*)dt \end{bmatrix} = \begin{bmatrix} \phi_x^* - \theta_x + x_{11}^N \\ \phi_y^* + x_{12}^N \\ -\theta_z + x_{13}^N \end{bmatrix} \tag{5.31}$$

$$x_{11}^N = \int(\theta_y\,\dot{\theta}_z)dt, \qquad x_{12}^N = -\int(\theta_x\,\dot{\theta}_z)dt, \qquad x_{13}^N = \int(\theta_y\,\dot{\phi}_x^* - \theta_x\,\dot{\phi}_y^*)dt.$$

Then the continuous time high yaw-rate dynamic model is:

$$\dot{\boldsymbol{x}}^N = \boldsymbol{f}^N(\boldsymbol{x}^N, \boldsymbol{u}^N)$$

$$f_1^N = \dot{\theta}_x, \quad f_2^N = \dot{\theta}_y, \quad f_3^N = \dot{\theta}_z, \quad f_4^N = \dot{\phi}_x^*, \quad f_5^N = \dot{\phi}_y^*$$

$$\begin{aligned}
f_6^N &= f_5^L + \dot{\theta}_z(-12\,\theta_y + 1.8\,\dot{\theta}_y + 0.0029\,\dot{\phi}_y^*) + 0.80\,\theta_x\,\dot{\theta}_z^2 - 310\,\theta_y\,u_z \\
f_7^N &= f_6^L + \dot{\theta}_z(12\,\theta_x - 1.8\,\dot{\theta}_x - 0.0029\,\dot{\phi}_x^*) + 0.80\,\theta_y\,\dot{\theta}_z^2 + 310\,\theta_x\,u_z \\
f_8^N &= -15\dot{\theta}_z - 220\,u_z + \dot{\phi}_x^*\,(14\,\theta_y + 0.015\,\dot{\theta}_y) - \dot{\phi}_y^*\,(5.2\,\theta_x + 0.015\,\dot{\theta}_x) \\
&\quad + \dot{\theta}_z\,(0.012\,\theta_x\dot{\phi}_x^* + 0.015\,\theta_y\dot{\phi}_y^*) + 7.5\,\theta_y\,u_x - 140\,\theta_x\,u_y \\
f_9^N &= f_7^L + \dot{\theta}_z(-6.5\,\theta_y + 0.44\,\dot{\theta}_y - 0.0062\,\dot{\phi}_y^*) + 0.44\,\theta_x\,\dot{\theta}_z^2 + 400\,\theta_y\,u_z \\
f_{10}^N &= f_8^L + \dot{\theta}_z(6.5\,\theta_x - 0.44\,\dot{\theta}_x + 0.0062\,\dot{\phi}_x^*) + 0.44\,\theta_y\,\dot{\theta}_z^2 - 400\,\theta_x\,u_z \\
f_{11}^N &= \theta_y\,\dot{\theta}_z, \qquad f_{12}^N = -\theta_x\,\dot{\theta}_z, \qquad f_{13}^N = \theta_y\,\dot{\phi}_x^* - \theta_x\,\dot{\phi}_y^*,
\end{aligned} \tag{5.32}$$

where f_i^N is the i-th component of the vector $\boldsymbol{f}^N$. The sensor dynamic model is:

$$\boldsymbol{y}^N = \boldsymbol{h}^N(\boldsymbol{x}^N, \boldsymbol{u}^N)$$

$$\begin{aligned}
h_1^N &= \dot{\theta}_x - \theta_y\,\dot{\theta}_z, \qquad h_2^N = \dot{\theta}_y + \theta_y\,\dot{\theta}_z \qquad h_3^N = \dot{\theta}_z \\
h_4^N &= \phi_x^* - \theta_x + x_{11}^N, \qquad h_5^N = \phi_y^* - \theta_y + x_{12}^N, \qquad h_6^N = -\theta_z + x_{13}^N \\
h_7^N &= h_5^L + \dot{\theta}_z\,(0.32\,\theta_x - 0.022\,\dot{\theta}_x + 0.00032\,\dot{\phi}_x^*) + 0.022\theta_y\,\dot{\theta}_z^2 - 3.1\,\theta_x\,u_z \\
h_8^N &= h_6^L + \dot{\theta}_z\,(0.32\,\theta_y - 0.022\,\dot{\theta}_y + 0.00032\,\dot{\phi}_y^*) - 0.022\theta_x\,\dot{\theta}_z^2 - 3.1\,\theta_y\,u_z,
\end{aligned} \tag{5.33}$$

where h_i^N is the i-th component of the vector $\boldsymbol{h}^N$. The discrete time equation for the EKF can be derived by using the same methodology as in 6.1.

5.3 Friction Model Identification

The linear and high yaw-rate models shown in §5.2.1 and §5.2.2 were used to design some preliminary KF/EKF and linear state feedback controller. Numerical simulations has shown some success with the estimator and controller designed this way and can achieve a stable and good position tracking performance under high yaw-rates. Chapter 6 will show more details on the controller, estimator design, simulations, and experiment done on the MBBR. The MBBR were successfully balanced and spun in place using these controllers, however the ball position and speed estimations were about 2.5 times larger compared to what the encoders were measuring. These error posed a serious problem to achieve a good ball position control and must be resolved. The encoder measurements might have some noise due to slipping, backlash, and motor imbalance; however, this error is way too large to be attributed to these problems. This kind of error is very likely to caused by friction, one of the major issue caused by miniaturization, which is not modeled in the dynamics formulation above. Adding a Coulomb and viscous friction model to the simulation caused a similar result, which makes friction compensation a critical element for achieving good state estimation and control performance. This will be explained in more detail in §6.3 and 6.4.

A test setup shown in Fig. 5.2 was made in order to determine the MBBR friction model as it is being driven in the x and y directions. Unfortunately, the setup is incapable of identifying the z direction, which should be a part of future work. The friction in the z direction is likely to be more complex than the other two orientations. The omniwheels placement and orientation causes the motor to push the ball into or away from the motors depending on the direction of the spin, which may cause different friction for each spinning direction.

The experiment followed the same methodology as the motor system identification in [7] which is also done in Chapter 4. A sine sweep signal was sent to the controller to spin the ball forward and backwards, which also spun the weighted flywheel as we measured the flywheel

rotation angle using the optical encoder. Unlike the motor identification experiment, the torque was estimated by using the motor model identified in Chapter 4. This was done by assuming that the motor model used in the estimator was correct.

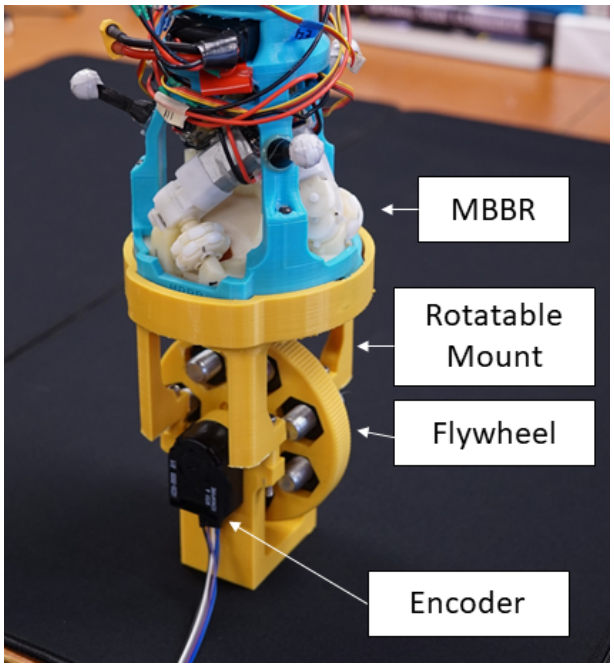


Figure 5.2: Friction identification setup, which allows identification in x and y plane.

The experiment identified the friction model by assuming that the friction can be modeled by using a simple Coulomb and viscous friction model. Let μ_c and μ_v be the Coulomb and viscous friction coefficient respectively. The friction model of the MBBR is shown below:

$$\begin{aligned} f_{cx} &= -\mu_c \operatorname{sign}(\dot{\varphi}_x) - \mu_v \dot{\varphi}_x \\ f_{cy} &= -\mu_c \operatorname{sign}(\dot{\varphi}_y) - \mu_v \dot{\varphi}_y. \end{aligned} \tag{5.34}$$

f_{cx} and f_{cy} were identified separately as the experiment was done in x and y directions. Assuming that there is no slip in between the ball and the flywheel, then ϕ_f and φ_x or φ_y have the following kinematic relationship:

$$\begin{aligned} \phi_f/\varphi_x &= r/r_f \\ \phi_f/\varphi_y &= r/r_f \end{aligned} \tag{5.35}$$

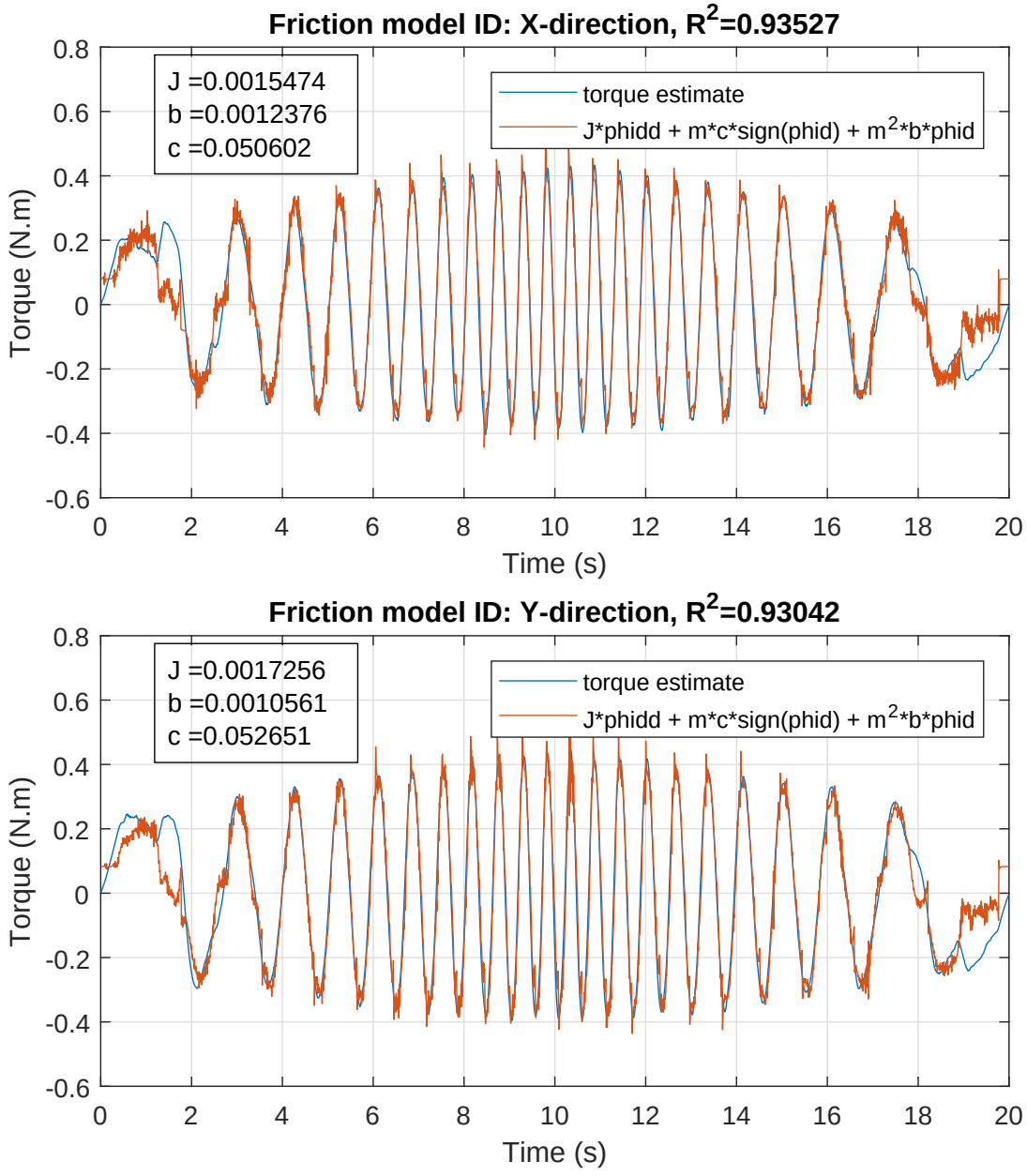


Figure 5.3: System identification regression fit for the friction model, $m = r_f/r$, $c = \mu_c$, $b = \mu_v$.

Using the motor model in (5.13), the dynamics equation for the flywheel is:

$$\begin{aligned} J_f\ddot{\phi}_f &= k_1 u_x - k_2 \left(r/r_w\right)^2 \dot{\varphi}_x + f_{cx}\left(r_f/r\right) \\ J_f\ddot{\phi}_f &= k_1 u_y - k_2 \left(r/r_w\right)^2 \dot{\varphi}_y + f_{cy}\left(r_f/r\right), \end{aligned} \tag{5.36}$$

where J_f and ϕ_f are the flywheel inertia and rotation angle respectively. Then the linear regression model for the system identification in the x direction is:

$$\begin{bmatrix} J_f & \mu_c & \mu_v \end{bmatrix} \begin{bmatrix} \ddot{\phi}_f \\ \text{sign}(\dot{\phi}_f)\left(r_f/r\right) \\ \dot{\phi}_f \left(r_f/r\right)^2 \end{bmatrix} = k_1 u_x - k_2 \left(r/r_w\right)^2 \dot{\varphi}_x. \tag{5.37}$$

The input signal u_x and u_y used the sine chirp signal, with amplitude of 3, run time of 20 seconds, initial and peak frequency of 0.1 Hz and 2 Hz respectively. The experimental data and regression model model fitting can be seen in Fig. 5.3. The identified friction parameters fits the experimental data fairly well, with the averaged R^2 value of 0.93. The data does not fit very well near the start and end of the experiment, which indicates that there might be an unmodeled dynamic when the ball starts or stops spinning. This could be caused by static friction, but identifying the static friction behavior of the robot can be very difficult. More work can be done to fully identify the friction model, but it might not be necessary.

The estimated inertia of the flywheel J_f should also be approximately the same value between the two experiments, but the regression parameter for the y direction is 11.5% larger than the x direction. This is very likely caused by a torque imbalance between the three motors which, despite the relatively small difference, might cause some issue during the control. Unfortunately, we don't have the friction parameter model for the z direction. As mentioned earlier, the identification of the z friction model can be a part of the future work. In order to keep the model simple, we used the values $\mu_c = 0.05265$ N.m and $\mu_v = 0.001056$ N.m.rad/s for all of the x, y, and z directions. The viscous friction model is linear and can be added to the dynamic model for

the estimators while the Coulomb model is discontinuous at zero speed. The methodology for the friction modeling and compensation is explained in more detail in §6.1 and §6.2.2.

Acknowledgements

Chapter 5 and 6 partially uses the material as it appears in the published paper in IEEE International Conference on Robotics and Automation (ICRA) 2019, E. Sihite, D. Yang, and T. Bewley, "Modeling and state estimation of a Micro Ball-balancing Robot using a high yaw-rate dynamic model and an Extended Kalman Filter". The dissertation author was the primary investigator and author of this paper, contributed to the estimator and controller design, experimentation, and data analysis.

References

[1] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, "Unified state estimation for a ballbot," *IEEE Int'l Conf. on Robotics and Automation (ICRA)*, pp. 2471–2476, 2013.

[2] A. Bonci, "New dynamic model for a ballbot system," *Mechatronic and Embedded Systems and Applications (MESA)*, 2016.

[3] P. Fankhauser and C. Gwerder, "Modeling and control of a ballbot," Bachelor's Thesis, Eidgenössische Technische Hochschule, Zürich, 2010.

[4] T. Bewley, *Numerical Renaissance: simulation, optimization, and control*. Renaissance Press, San Diego, 2015.

[5] E. Sihite and T. Bewley, "Attitude estimation of a high-yaw-rate mobile inverted pendulum; comparison of extended kalman filtering, complementary filtering, and motion capture," *American Control Conference (ACC)*, pp. 5831–5836, 2018.

[6] T. Hoshino, S. Yokota, and T.Chino, "Omniride: A personal vehicle with 3 dof mobility," *Int'l Conf. on Control, Automation, Robotics and Embedded Systems*, 2013.

[7] N. Morozovsky, R. Moroto, and T. Bewley, "Rapid: An inexpensive open source dynamometer for robotics applications," *IEEE/ASME Transactions on Mechatronics 18.6*, pp. 1855–1860, 2013.

Chapter 6

MBBR Controller and Estimation

This chapter discusses the MBBR estimator and controller design, numerical simulation, and the motion capture (mocap) experiments. Several preliminary controllers and estimators were designed by using the simplified the MBBR high yaw-rate model derived in Chapter 5. The feedback stabilizing controllers were designed by using a simple linear feedback controller, which were shown to be adequate and can achieve good position tracking under high yaw rates in both the simulation and the actual robot. The simulation results will be shown in more detail in §6.3. Linear state feedback controller is also the controller of choice in most of the existing BBRs [1] [2] [3]. A simulated adaptive model predictive control framework for controlling a BBR has been presented by [4]. Using a more complex or nonlinear controller which compensates for the nonlinearities caused by nontrivial yaw rates might improve the robustness of the controller. Designing a nonlinear controller is a highly challenging problem and can be a part of the future work. This chapter is outlined as follows: the estimator design, controller design, numerical simulations, and the mocap experiments.

The mocap experiments were done to show the estimator's accuracy and the controller's ball position and velocity tracking performance. The estimation accuracy of both the linear model KF and the high yaw-rate EKF were compared under several yaw rates and driving conditions. In addition to this, the InvenSense MPU-9250 DMP estimation is also compared using the mocap at the same time. Our previous work (see Chapter 3) has shown that the DMP estimations is

fairly accurate even under high yaw rates dynamics. The mocap experiments can show whether this result is also can be reproduced in the MBBR and also verify the estimation accuracy of the state estimations.

6.1 Estimators Design

This section describes the linear model KF and the high yaw-rate model EKF which were used in the motion capture experiments. The discrete time dynamic models used in KF and EKF are derived by using a simple Explicit Euler integration as shown below:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + dt\, \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{v}_k \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k, \end{aligned} \tag{6.1}$$

where dt is the sampling period, $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are the process and the measurement noise vectors respectively. The length of the vectors depends on the model used as shown in §5.2.1 and §5.2.2. The high yaw-rate model has 13 states, 3 inputs, and 8 outputs while the linear model has 8 states, 2 inputs, and 6 outputs. Let $Q = E(\boldsymbol{v}_k^T\, \boldsymbol{v}_k)$ and $R = E(\boldsymbol{w}_k^T\, \boldsymbol{w}_k)$ be the process and measurement noise covariance matrix respectively. The following Q and R matrices were used in the motion capture experiment:

$$\begin{aligned} Q^L &= \mathrm{diag}(q_1^L, q_2^L, \ldots, q_8^L) \qquad & R^L &= \mathrm{diag}(r_1^L, r_2^L, \ldots, r_6^L) \\ Q^N &= \mathrm{diag}(q_1^N, q_2^N, \ldots, q_{13}^N) \qquad & R^N &= \mathrm{diag}(r_1^N, r_2^N, \ldots, r_8^N), \end{aligned} \tag{6.2}$$

where the superscript L and N represents the linear and high yaw-rate model respectively. The values for Q and R matrices are listed for each motion capture experiment in §6.4. The covariance matrix is chosen to be diagonal to make designing the KF and EKF simpler.

The friction model or the nonlinearities caused by the drive/coast motors are not included in either estimators. The linear model can't model any of the nonlinearities, so it must use the

inputs before the friction and drive/coast compensations for the KF updates and assume that the compensation has successfully linearized the model. On the other hand, the EKF can model these nonlinearities, but using the same assumption as the KF case works just as well which can be seen in the motion capture experiments. The setup and results of the friction compensation will be discussed in more detail in §6.4.2. In order to keep similar controller structure between the two controller, both estimators use the uncompensated inputs for the KF and EKF updates. Of course, it is likely that there are modeling errors and significant unmodeled dynamics, such as the static friction, motor backlash, omniwheel slips, etc. These errors are assumed to be a part of the process noise which makes selecting the appropriate Q matrix important to achieve the best estimation accuracy.

6.1.1 Inertial Ball Angle and Velocity Estimation

Both estimators do not estimate the inertial ball angle and velocity directly. Instead, the yaw-normalized ball velocities states are rotated back into the inertial frame and then integrated to estimate the inertial ball angles. The inertial frame ball angular velocities are:

$$\begin{aligned} \dot{\phi}_x &= \cos(\theta_z)\,\dot{\phi}_x^* - \sin(\theta_z)\,\dot{\phi}_y^* \\ \dot{\phi}_y &= \sin(\theta_z)\,\dot{\phi}_x^* + \cos(\theta_z)\,\dot{\phi}_y^*, \end{aligned} \tag{6.3}$$

which are integrated using the Explicit Euler scheme to estimate the inertial ball angles:

$$\begin{aligned} \phi_{x,k+1} &= \phi_{x,k} + \dot{\phi}_{x,k}\,dt \\ \phi_{y,k+1} &= \phi_{y,k} + \dot{\phi}_{y,k}\,dt. \end{aligned} \tag{6.4}$$

Augmenting the inertial ball angle states into the EKF is possible but has no particular benefit. The system dynamics is not affected by the ball angles and the encoders only directly measure the yaw-normalized ball angles. Therefore, calculating the inertial frame states can be done outside of the EKF. The inertial ball position must be integrated, so it is highly prone to integration

error which will build up over time. Therefore, this method can only be used to estimate the "approximate" inertial position. Inertial position tracking under high yaw rates is already a difficult problem, so this approximate position works well enough for us. This can be a reason to use the yaw-normalized states under trivial yaw rates as it is less prone to the integration build up. However, in order to keep the controller simple, we only used the inertial ball position states and references in the controller.

6.2 Controller Design

The MBBR was controlled by using a linear state feedback controller and the estimated states from either the KF or the EKF. The controller gains were determined from the linear model's LQR controller gains which then were tuned by hand afterwards. The same controller was used to test both the KF and the EKF in the §6.4. The full control algorithm for the MBBR is shown in the block diagram in Fig. 6.1. The "reference and state error calculator" block uses the EKF state estimation and the Bluetooth remote control commands to calculate the state errors for the linear feedback controller. The block diagram for the reference and state error calculator can be seen in Fig. 6.2. The system inputs are defined with respect to the robot's body frame of reference, which means that the inertial ball angle and velocity state errors must be rotated back into the yaw-normalized frame. This process can be seen in the lower left part of Fig. 6.2. The MBBR has the following omniwheels axis to body frame transformation matrix:

$$T_{ob} = \begin{bmatrix} -0.7344 & 0.0583 & 0.6761 \\ 0.3567 & -0.8144 & 0.4577 \\ 0.5744 & 0.5744 & 0.5774 \end{bmatrix}, \tag{6.5}$$

which corresponds to $\alpha = 60°$, $\beta = 48.2°$, rotated about the yaw axis by $-140°$. The α and β angles corresponds to the omniwheel midlatitude and orthogonal placement as described in §2.3.2.

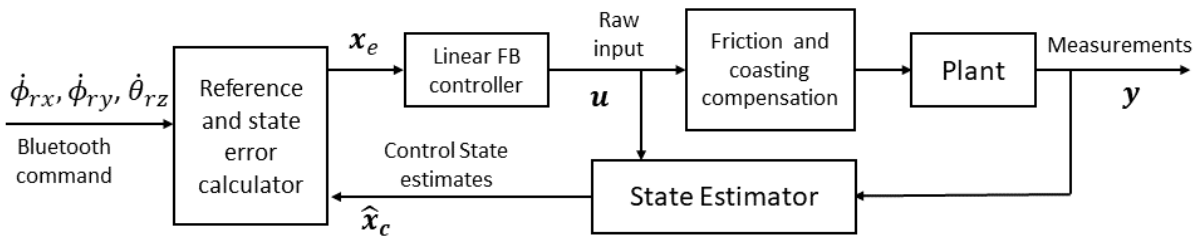


Figure 6.1: The control block diagram of the MBBR.

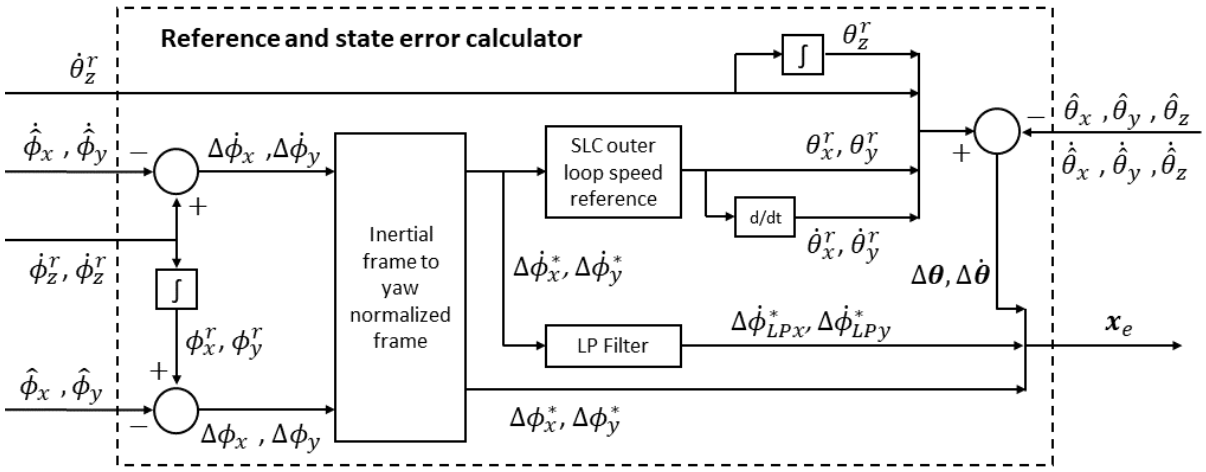


Figure 6.2: The block diagram of the reference and state error calculator.

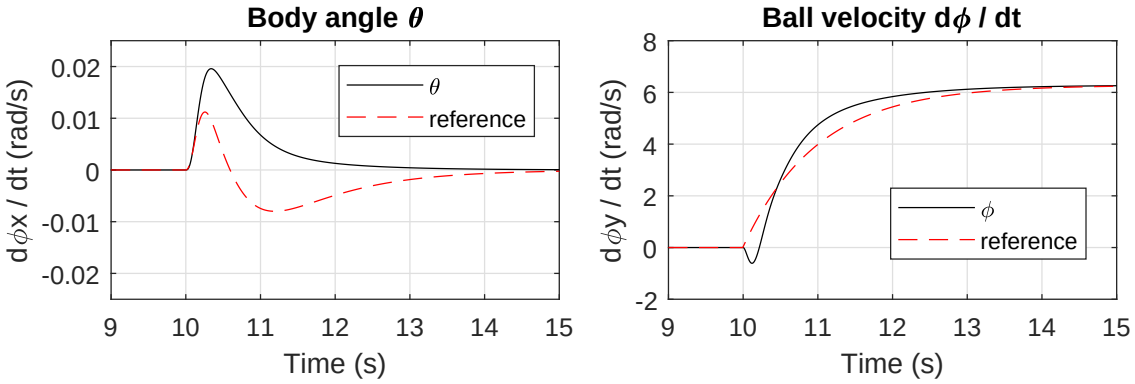


Figure 6.3: The non-minimum phase behavior of an inverted pendulum robot using a SLC outer loop and ball velocity reference.

As a type of an inverted pendulum system, the MBBR also has the non-minimum phase dynamics which is caused by an unstable zero in the system. This non-minimum phase dynamics affects the relationship between the upper body attitude angles (θ_x and θ_y) and the ball position or velocity (ϕ_x and ϕ_y). Applying torque to the ball in order to spin the ball forward will also apply counter torque into the upper body in the reverse direction and away from the equilibrium. This means that the torque controlling the ball and the upper body act in the opposite directions to each other. The Successive Loop Closure (SLC) controller which was used in the MIP in Fig. 3.3 calculates the upper body attitude angles required to move the ball position or velocity according to the non-minimum phase dynamics. An example of this can be seen in Fig. 6.3, where the outer loop controller $D_2(s) = \Theta(s)/\Phi(s)$ calculates the reference value for the upper body angle θ. The θ must tilt forward in order to move the ball (or the robot itself) forward, which results in the initial reverse ball movement as shown in Fig. 6.3. The SLC controller used in the MBBR will be explained in more detail in §6.2.1.

The friction and drive/coast compensation is done after the KF and EKF prediction updates as discussed in §6.1. A low-pass filter is used on the KF or EKF $\dot{\phi}_x$ and $\dot{\phi}_y$ state errors before the linear feedback controller because of the highly noisy estimation from these two states. Not using the low-pass filter makes the MBBR extremely jittery and very difficult to control while spinning fast. These vibrations also prevent the robot from using the control gains derived from the LQR, to the point that it have can only use around 20% of the suggested ball velocity gains. Unfortunately, we have not yet found the covariance matrices values that can eliminate this noise while also having accurate θ estimates. Discovering a way to reduce the noise in the ball velocity estimations can be a part of the future work.

Let $\boldsymbol{x}_e \in \mathcal{R}^{10}$ be state error with respect to the reference. The elements of $\boldsymbol{x}_e$ are:

$$\boldsymbol{x}_e = \begin{bmatrix} \Delta\theta_x & \Delta\theta_y & \Delta\theta_z & \Delta\phi_x^* & \Delta\phi_y^* & \Delta\dot{\theta}_x & \Delta\dot{\theta}_y & \Delta\dot{\theta}_z & \Delta\dot{\phi}_{LPx}^* & \Delta\dot{\phi}_{LPy}^* \end{bmatrix}^T \tag{6.6}$$

$$\begin{aligned} \Delta\theta_x &= \theta_x^r - \hat{\theta}_x, \qquad \Delta\theta_y = \theta_y^r - \hat{\theta}_y, \qquad \Delta\theta_z = \theta_z^r - \hat{\theta}_z \\ \Delta\dot{\theta}_x &= \dot{\theta}_x^r - \dot{\hat{\theta}}_x, \qquad \Delta\dot{\theta}_y = \dot{\theta}_y^r - \dot{\hat{\theta}}_y, \qquad \Delta\dot{\theta}_z = \dot{\theta}_z^r - \dot{\hat{\theta}}_z, \end{aligned} \tag{6.7}$$

where the hat symbol and the superscript r represents the state estimations and references respectively. The KF does not estimate θ_z and $\dot{\theta}_z$, so the IMU's DMP yaw angle estimates and raw gyro measurements are used instead. The errors $\Delta\phi_x^*$, $\Delta\phi_y^*$, $\Delta\dot{\phi}_{LPx}^*$, and $\Delta\dot{\phi}_{LPy}^*$ requires some additional steps to compute. $\Delta\phi_x^*$ and $\Delta\phi_y^*$ are the yaw-normalized ball angle errors, as shown below:

$$\begin{aligned} \Delta\phi_x^* &= \cos(\hat{\theta}_z)\,(\phi_x^r - \hat{\phi}_x) + \sin(\hat{\theta}_z)\,(\phi_y^r - \hat{\phi}_y) \\ \Delta\phi_y^* &= -\sin(\hat{\theta}_z)\,(\phi_x^r - \hat{\phi}_x) + \cos(\hat{\theta}_z)\,(\phi_y^r - \hat{\phi}_y) \end{aligned} \tag{6.8}$$

Similarly, $\Delta\dot{\phi}_x^*$ and $\Delta\dot{\phi}_y^*$ are the yaw-normalized ball angular velocity errors, as shown below:

$$\begin{aligned} \Delta\dot{\phi}_x^* &= \cos(\hat{\theta}_z)\,(\dot{\phi}_x^r - \dot{\hat{\phi}}_x) + \sin(\hat{\theta}_z)\,(\dot{\phi}_y^r - \dot{\hat{\phi}}_y) \\ \Delta\dot{\phi}_y^* &= -\sin(\hat{\theta}_z)\,(\dot{\phi}_x^r - \dot{\hat{\phi}}_x) + \cos(\hat{\theta}_z)\,(\dot{\phi}_y^r - \dot{\hat{\phi}}_y). \end{aligned} \tag{6.9}$$

$\Delta\dot{\phi}_{LPx}^*$ and $\Delta\dot{\phi}_{LPy}^*$ are the low-pass filtered signal of $\Delta\dot{\phi}_x^*$ and $\Delta\dot{\phi}_y^*$ respectively. The details of the reference states calculation and the low-pass filter can be seen in §6.2.1. Finally, the following linear state feedback controller is used to stabilize the robot:

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = -K_c\,\boldsymbol{x}_e, \qquad K_c = \begin{bmatrix} c_1 & 0 & 0 & c_2 & 0 & c_3 & 0 & 0 & c_4 & 0 \\ 0 & c_1 & 0 & 0 & c_2 & 0 & c_3 & 0 & 0 & c_4 \\ 0 & 0 & c_5 & 0 & 0 & 0 & 0 & c_6 & 0 & 0 \end{bmatrix}. \tag{6.10}$$

The values for the gain K_c differ in some of the mocap experiments, so these values will be listed in each of the experiments in §6.4. The controller gains were also gain scheduled based on the reference yaw rates, which vary in between each experiments as well. The reference states, SLC outer loop controller, friction and drive/coast compensations will be discussed in the following subsections.

6.2.1 Reference States Calculations

The reference states $\dot{\phi}_x^r$, $\dot{\phi}_y^r$, and $\dot{\theta}_z^r$ came from the Bluetooth dongle (JY-MCU HC-06) which is connected to the user by using an Android Bluetooth remote control app called "4joy Remote". This remote control feeds in the reference for the yaw rate and ball inertial velocity, which are integrated to calculate the heading and ball inertial position references, as shown below:

$$\begin{aligned} \phi_{x,k+1}^r &= \phi_{x,k}^r + \dot{\phi}_{x,k}^r \, dt \\ \phi_{y,k+1}^r &= \phi_{y,k}^r + \dot{\phi}_{y,k}^r \, dt \\ \theta_{z,k+1}^r &= \theta_{x,k}^r + \dot{\theta}_{x,k}^r \, dt. \end{aligned} \tag{6.11}$$

The remote control's inertial frame "forward" direction is set when the MBBR starts balancing or when the "heading reset button" is pressed in the remote control. A discrete set of yaw rates reference were tested in the motion capture experiment, which consists of the yaw rates 0.25 Hz, 0.5 Hz, 0.75 Hz, 1 Hz, and 1.5 Hz in the clockwise and counter-clockwise directions.

The ball velocity estimation is highly noisy which is also reflected in the encoder measurements. There are several possible cause of noise, such as omniwheels slipping, DROW roller transition, static friction, and backlash. These noises are part of the process noise and their effects are also reflected in the measurements. The encoder measurements are the most affected by these noises, which means very noisy ball velocity estimations. Even though it might be not ideal, filtering the EKF ball velocity estimates can be done to reduce the noise caused by these problems. Preliminary motion capture experiments without filtering the ball velocity estimates has shown that the estimated states were relatively accurate but was riddled with vibration which was dominant in the frequency range of 18 to 23 rad/s. This experiment and the data will be discussed in §6.4.3. Both $\Delta\dot{\phi}_x^*$ and $\Delta\dot{\phi}_y^*$ are low-pass filtered by using a discrete time low-pass filter with cutoff frequency of 10 rad/s. This cutoff frequency is relatively low for spinning at 1 Hz and 1.5 Hz which are some of the yaw rates tested in the mocap experiments. Therefore, the low-pass filter cutoff frequency is increased to 15 and 20 rad/s for the yaw rates 1 Hz and 1.5

Hz respectively. The experiments done in §6.4.5 and §6.4.6 have shown that the robot can spin relatively well under these cutoff frequencies.

The reference states for the θ_x, θ_y, $\dot{\theta}_x$, and $\dot{\theta}_y$ need to be calculated by using a filter which predicts the non-minimum phase dynamics of an inverted pendulum robot. This filter uses the same principle as an outer loop of a Successive Loop Closure (SLC) controller which is used in the MIP robot in Chapter 3. The SLC control algorithm seen in the Fig. 3.3 was used on a MIP robot, but the same principles can also be applied here. The outer loop controller of the SLC can be used to determine the state reference values θ_x^r and θ_y^r. These can then be differentiated by using a simple backwards difference scheme to determine the $\dot{\theta}_x^r$ and $\dot{\theta}_y^r$, as shown below:

$$\begin{aligned} \dot{\theta}_{x,k} &= (\theta_{x,k} - \theta_{x,k-1})/dt \\ \dot{\theta}_{y,k} &= (\theta_{y,k} - \theta_{y,k-1})/dt \end{aligned} \tag{6.12}$$

These reference angles for the upper body angles follows the robot's natural non-minimum phase trajectory, which allows for a faster ball position and velocity tracking with less overshoot. The outer loop controller used in the MIP outputs the reference θ for the inner loop controller which is not a state space controller. The state space controller also has ball position and velocity gains which have the same purpose as the outer loop controller: achieving ball position and velocity tracking. Therefore, we don't need to use high gains for the outer loop controller like the MIP case. It is enough for the outer loop to help pushing the upper body to tilt into the right direction which allows robot move smoothly from one position into another.

The SLC outer loop controller was designed by using the linear MBBR numerical model in (5.32). The controller takes in the yaw-normalized ball velocity state error $\Delta\dot{\phi}^*$ as an input and outputs the corresponding non-minimum phase θ reference angles. The following SLC outer

loop controller is used in the MBBR controller:

$$
\begin{aligned}
D_x(z) &= \frac{\Theta_x^r(z)}{\Delta\dot{\Phi}_x^*(z)} = c_{SLC}\,\frac{-0.000985}{z-\omega_{SLC}} \\
D_y(z) &= \frac{\Theta_y^r(z)}{\Delta\dot{\Phi}_y^*(z)} = c_{SLC}\,\frac{-0.000985}{z-\omega_{SLC}}
\end{aligned}
\tag{6.13}
$$

The outer loop controller pole (ω_{SLC}) and gain (c_{SLC}) are updated as the depending on the Bluetooth yaw rate command. The poles and gains may vary in between the experiments done in the §6.4, which will be listed in each of the experiment subsections.

6.2.2 Friction Compensation

The friction model is defined in the body frame of reference while the drive/coast compensation is applied on each motor commands. Therefore, the friction compensation must be done before the drive/coast compensations. The total torque applied in the x direction can be expressed as:

$$
\begin{aligned}
\Sigma\tau_x &= k_1\,u_x - k_2(r/r_w)^2\dot{\varphi}_x - \mu_c\,\mathrm{sign}(\dot{\varphi}) - \mu_v\,\dot{\varphi}_x \\
&= k_1\,(u_x - (\mu_c/k_1)\,\mathrm{sign}(\dot{\varphi}_x)) - (k_2(r/r_w)^2 + \mu_v)\,\dot{\varphi}_x.
\end{aligned}
\tag{6.14}
$$

The viscous friction model is linear and can simply be added into the system dynamic model. Therefore, the only friction being compensated is the Coulomb friction which is nonlinear and, as mentioned in §6.1, is not modeled in either KF or EKF. The compensation in the x direction can be done by adding $(\mu_c/k_1)\,\mathrm{sign}(\dot{\varphi}_x)$ into the u_x. However, the sign function can add vibrations near zero speed because of the discontinuity. The MBBR uses the smoothed sign function shown below:

$$
\mathrm{sign}(x) \approx \begin{cases} 1 - e^{-2\,x} & \text{if } x > 0 \\ -1 + e^{2\,x} & \text{if } x < 0. \end{cases}
\tag{6.15}
$$

This function is smooth near zero speed and reach 95% value at approximately $x = 3$. Using k_1, μ_c, and μ_v values listed in Chapter 5, (μ_c/k_1) and μ_v have a value of 0.30 (unitless) and 0.001056 N.m.rad/s respectively. As mentioned in §5.3, the friction is assumed to be the same in all directions which means that the friction in the y and z directions can be derived in the same way.

6.2.3 Drive/Coast Compensation

The drive/coast compensation follows the same algorithm as shown in §4.4 and is applied on each motors. The motor parameters have been identified, with motor resistance of 3.385 Ω, motor gain post 12:1 gear reduction of 0.0742 N.m/A, and inductance of 0.585 mH. Using these parameters, 20 kHz PWM frequency, and the nominal voltage of 8 V, we have the following drive/coast parameters:

$$i_s = 2.36\,\mathrm{A}, \qquad T_r = 0.289, \qquad \omega_{nl} = 108\,\mathrm{rad/s}. \tag{6.16}$$

The inputs before the drive/coast compensation must be scaled with respect to the nominal voltage of 8V. This is done by measuring the current battery voltage V_b and then scaling the inputs by $8\mathrm{V}/V_b$. The rotor speed of the motor must be estimated for this algorithm to work, so a simple filtering algorithm [5] is used to estimate the rotor i speed directly from the encoder i measurements φ_i, $i = \{1, 2, 3\}$. Let f_k be the noisy measurement at time step t_k. We may fit the 5 previous time steps tap delayed measurements into a 1st order polynomial:

$$f^{(k)}(t) = c_0^{(k)} + c_1^{(k)} t. \tag{6.17}$$

Using the time step of 0.005 s, we need to solve the following equation:

$$\begin{pmatrix} 1 & 0 \\ 1 & -0.005 \\ 1 & -0.010 \\ 1 & -0.015 \\ 1 & -0.020 \\ 1 & -0.025 \end{pmatrix} \begin{bmatrix} c_0^{(k)} \\ c_1^{(k)} \end{bmatrix} = \begin{bmatrix} f_k \\ f_{k-1} \\ f_{k-2} \\ f_{k-3} \\ f_{k-4} \\ f_{k-5} \end{bmatrix} \qquad \leftrightarrow \qquad A\,\boldsymbol{c}^{(k)} = \boldsymbol{f}^{(k)} \tag{6.18}$$

The time derivative of the measurement can be estimated as:

$$\left(\frac{df}{dt}\right)_{t=t_k} = \begin{bmatrix} 0 & 1 \end{bmatrix} A^{+}\,\boldsymbol{f}^{(k)} \tag{6.19}$$

where A^{+} is the pseudo-inverse of A. Then we can solve for the time derivative by using the following equation:

$$\left(\frac{df}{dt}\right)_{t=t_k} = \begin{bmatrix} 28.6 & 17.1 & 5.71 & -5.71 & -17.1 & -28.6 \end{bmatrix} \boldsymbol{f}^{(k)} \tag{6.20}$$

6.3 Numerical Simulation

The full MBBR dynamic model shown in the Appendix A is used in the simulation together with the high yaw-rate model EKF and controller derived above. The friction and its compensation were modeled in the simulation. However, the drive/coast dynamics were not modeled in the simulation due to the complexity of simulating each individual wheels/motors. The simulation was done using the 4th order Explicit Runge-Kutta scheme [5], which is simple and easy to use. The simulation also simulates the estimator and controller for the high yaw-rate EKF, using the same sampling rate as the MBBR (200 Hz). The simulation time step is 1 ms (1/5 of the sampling period) and used the same process and measurement noise as the EKF's Q

and R matrices. Using this process noise in the simulation is very likely not comparable with the actual noise in the system, but we do not have the means of identifying the actual process noise data. The actual robot has a highly noticeable noise in the frequency range of 20-30 rad/s which is very likely caused by the roller transitions in the DROWs. This noise seems to vary proportionally with respect to the drive speed, which adds the complexity of using this noise in the simulation. The implementation of a more realistic noise as it appears in the actual robot can be done as a part of the future work.

The simulation used the same controller and estimator structure as the one used in the actual robot. However, this simulation does not use low-pass filter and gain scheduling in the controller. The ball velocity estimate is not as noisy as the actual robot and the controller can use a single gain for every yaw rates. The simulation used the following process and measurement noise covariance matrices for the EKF:

$$\begin{aligned} q_1^N = q_2^N = 10^{-5}, \quad & q_3^N = q_4^N = q_5^N = 10^{-4}, \quad & q_6^N = q_7^N = 0.1, \\ q_8^N = q_9^N = q_{10}^N = 0.1, \quad & q_{11}^N = q_{12}^N = q_{13}^N = 10^{-3}, & \\ r_1^N = r_2^N = r_3^N = 2.08 \cdot 10^{-6}, \quad & r_4^N = r_5^N = r_6^N = 7.62 \cdot 10^{-5}, \quad & r_7^N = r_8^N = 1.41. \end{aligned} \tag{6.21}$$

The simulation uses the following controller gains:

$$\begin{aligned} c_1 =& 14, \quad & c_2 =& 2, \quad & c_3 =& 0.6, \quad & c_4 =& 1.2, \\ c_5 =& 0.4, \quad & c_6 =& 1.2, \quad & c_{SLC} =& 0.25, \quad & \omega_{SLC} =& 0.961 \end{aligned} \tag{6.22}$$

which are constant throughout the simulation. The following two subsections show the simulation for friction compensations and the position tracking under high yaw rates.

6.3.1 Friction Model Simulations

Several simulations has been done to show the effect of the friction and determine the best method to implement the friction compensation into the controller and estimators. The

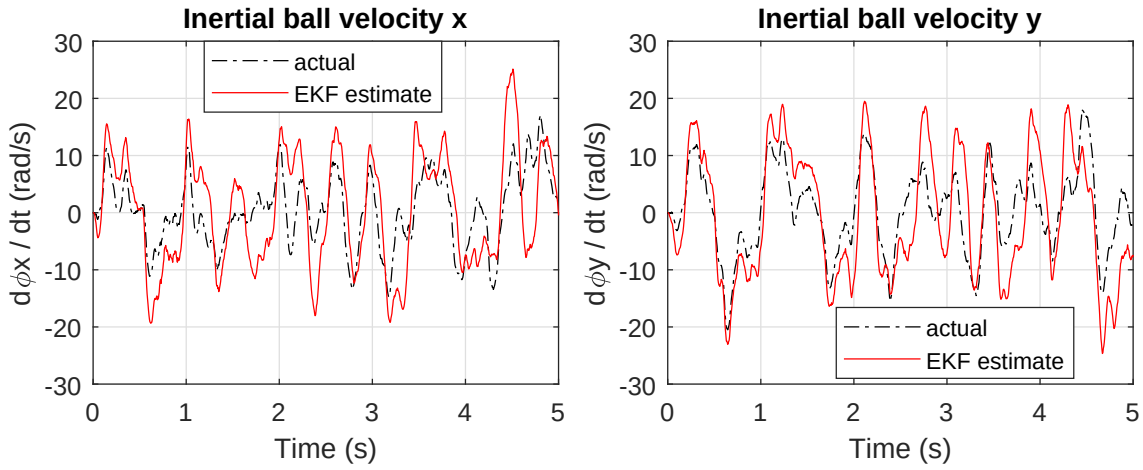


Figure 6.4: Simulated ball speed estimation without friction compensation or model.

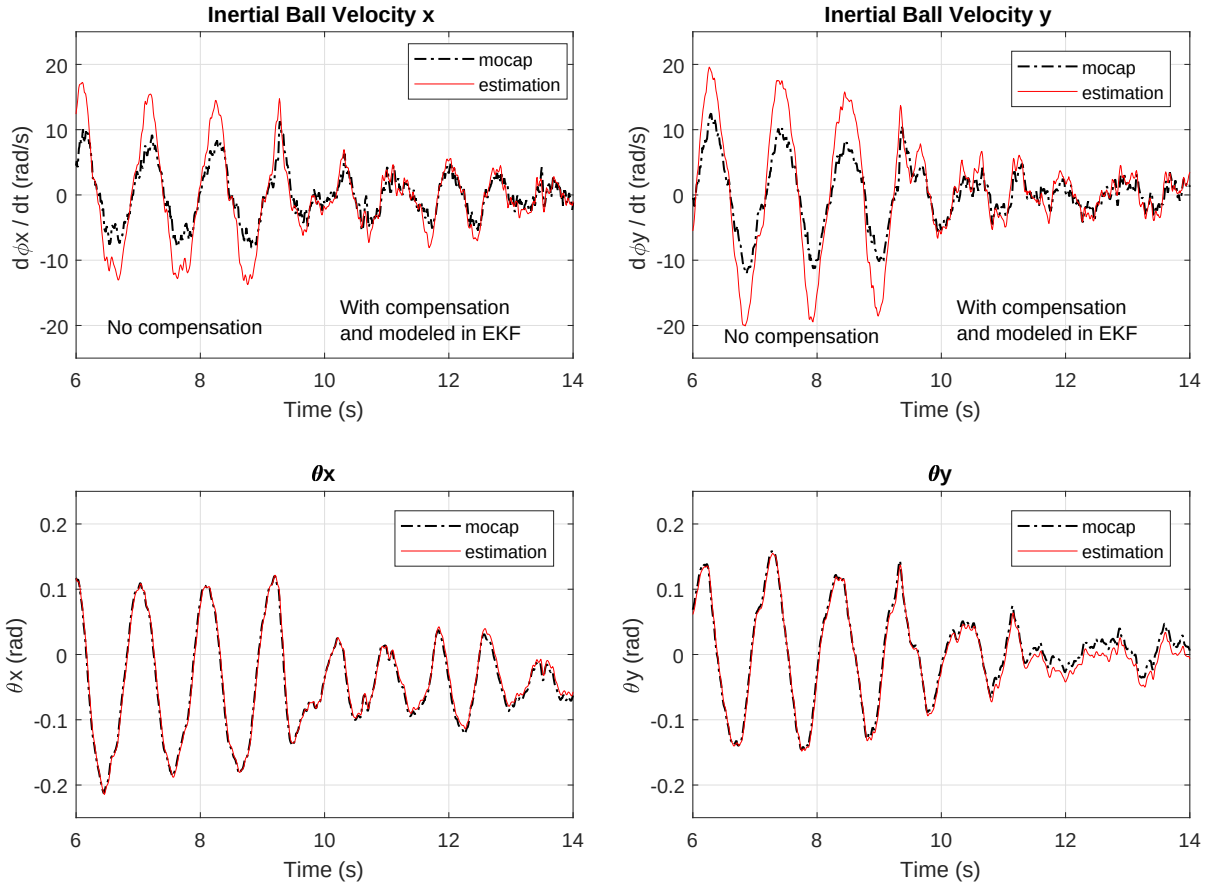


Figure 6.5: Motion capture measurements vs EKF estimation for the θ and $\dot{\phi}$ with and without friction compensation.

Coulomb and viscous friction model in §5.3 were used to simulate the friction.

Adding friction into the system and using no compensation caused the estimated ball velocity states ($\dot{\phi}_x$ and $\dot{\phi}_y$) to become too large compared to the true values. This phenomenon happened to both in the simulation and in the MBBR, as shown in Fig. 6.4 and 6.5 respectively. The $\dot{\phi}$ estimations of the MBBR were compared with the speed estimated from motion capture experiment which will be discussed in more detail in §6.4.2. Both figures showed similar ball velocity errors, which is approximately 1.5 to 2 times larger than what they should be. The errors in the simulation seemed more random than in the actual robot. This is likely caused by the simulated process noise in the simulation does not match the actual robot's noise. The error is mostly multiplicative while idling, so the MBBR can stabilize in place by using the correct gain values. However, this become a huge problem for ball position and speed tracking due to the incorrect estimations. Obviously, this problem must be resolved before we can even attempt translation and spinning at the same time because it requires a more accurate inertial ball position and velocity estimations.

The controller compensates the Coulomb friction by adding a constant value depending on the sign of the encoder velocities. The estimator can either model the friction and use the input post compensation for the prediction update, or omit the friction model and use the input before the compensation. The former method can work as well as the latter, but the predicted states tend to have huge noise near zero velocity. This is very likely caused by the nonlinearity in the Coulomb friction model. So, we assumed that the compensation fully counteract the friction and all of the errors become a part of the process noise. This is the method that works well in both the simulation and the actual robot. The same principle is also applied to the drive/coast compensation and it worked well in the real robot.

6.3.2 High Yaw-Rate Simulations

This simulation is done to show that the inertial position tracking under high yaw rates is possible by using a linear controller. The MBBR was idle and then simultaneously spinning and translating starting from $t = 10$s to $t = 20$s. The yaw rate and ball velocity references were set at 1 Hz and 20 cm/s respectively. The simulation result can be seen in Fig. 6.6. The inertial ball position estimation slowly drifts away from the actual position, which can be seen in the inertial ball position x plot. This is caused by the integration error build up which can happen due to noise and estimation error.

As shown in the simulation result, a linear controller is sufficient to control the MBBR for simultaneous translation and rotation at nontrivial yaw rates. The problem is that this simulation does not model the problematic mechanical issue in the actual robot, which includes the high frequency vibrations from using DROW, static friction, and backlash. The process noise covariance used in the simulation was the similar to the one in the actual robot's EKF, but the true process noise is unknown. This simulation only shows that the linear controller can work under ideal conditions, which might also work in the actual robot if the state estimation is accurate enough.

6.4 Motion Capture Experiment

In this section, the accuracy of the EKF and the KF were evaluated by comparing the estimated attitude angles and inertial ball velocities with the motion captured measurements. Several experiments were designed to verify the estimation accuracy, controller stability, noise frequency, and tracking performance.

The measurement noise covariance matrix remained constant throughout the experiments. On the other hand, the process noise covariance matrix may change slightly in between the experiments. The values for the process noise covariance matrix is listed in each motion capture

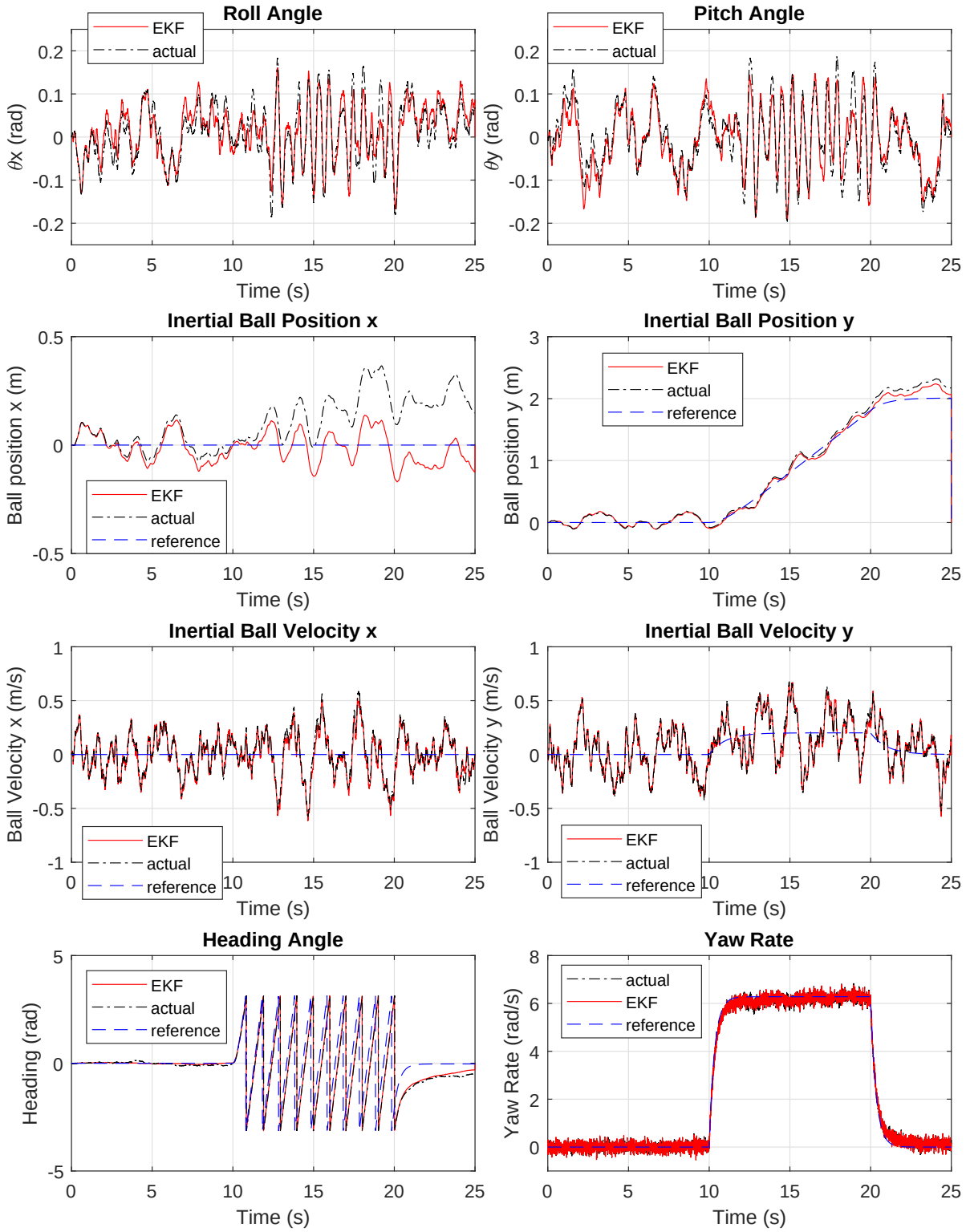


Figure 6.6: MBBR simulation plot, 1 Hz yaw rate and 20 cm/s drive rate.

experiment subsection. The following measurement noise covariance matrices values were used in all motion capture experiments:

$$\begin{aligned} r_1^N = r_2^N = r_3^N = r_1^L = r_2^L &= 2.08 \cdot 10^{-6} \\ r_4^N = r_5^N = r_6^N = r_3^L = r_4^L &= 7.62 \cdot 10^{-5} \\ r_7^N = r_8^N = r_5^L = r_6^L &= 1.41 \end{aligned} \tag{6.23}$$

The controller in the experiments were all run by using the EKF estimation states, except for the last experiment where it used the DMP and KF estimations instead. The DMP is the proprietary orientation algorithm native in the InvenSense MPU-9250 IMU which performance was shown in §3.4.2. The accuracy of its orientation estimation is also shown in this section.

6.4.1 Motion Capture Setup and Calibration

A motion capture (mocap) system using four Optitrack Prime 13 cameras was used to verify the accuracy of the estimated θ angles and the inertial frame ball angular velocities $\dot{\phi}$. The MBBR was attached with 8 mocap markers, as shown in Fig. 6.7, and was balanced with the controller in §6.2 using the estimated states from either the EKF or the KF.

The markers can be grouped up to form a rigid body model by using the mocap tracking software. The mocap can measure the rigid body centroid position and orientation in quaternion. This measurement can be transformed into the Euler roll-pitch-yaw angles and the ball velocity after some calibrations and calculations. The Euler angles can be calculated by aligning the mocap quaternion with the body's axis and then transforming the quaternion into the corresponding Euler angles. Once the Euler angles have been calculated, the ball speed can be estimated by using kinematics relationship. The MBBR data was also recorded in the Beaglebone Black internal storage, which must be calibrated to match the mocap data as much as possible post recording.

The initial quaternion of the rigid body formed by the markers was set when the rigid

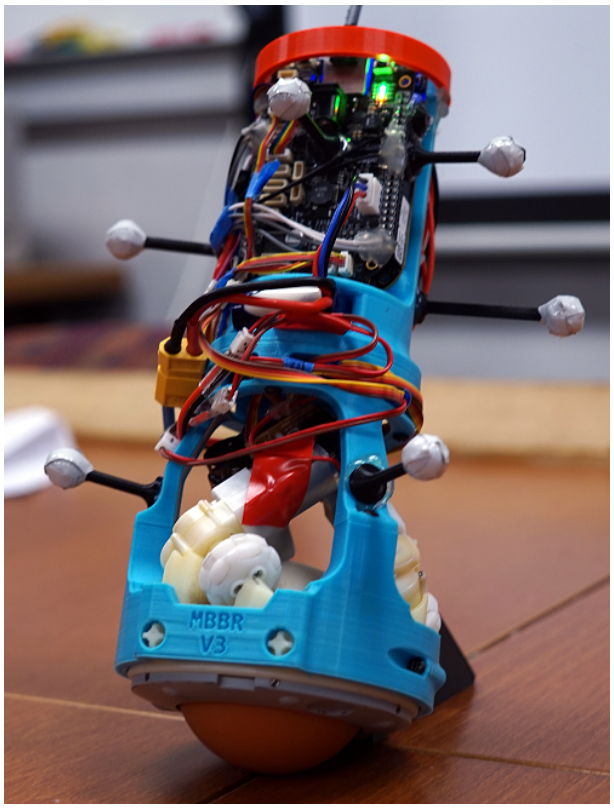


Figure 6.7: MBBR with 8 motion capture markers attached.

body was defined. Therefore, the MBBR was set as upright as possible during this part of the setup. Then, let the MBBR rest at an angle only in the x direction before starting the recording. After the mocap starts recording, upright the MBBR as quickly as possible so that there is a huge spike in the recorded data. This helps matching up the starting time between the mocap and the data recorded in the Beaglebone Black. After matching up the time between both data, the orientation is matched by rotating the quarternion angles and transform it into the roll-pitch-yaw Euler angles. The resulting angles is matched to the IMU's DMP measurements because the EKF and KF are not active until the robot starts balancing. If there are some offsets, the angles are adjusted further manually until the initial DMP orientation signals match the mocap.

After the Euler angles has been calibrated, they can then be used to estimate the ball velocity. The markers centroid is defined as the rigid body position in the mocap data. The z axis data while the MBBR is idling can be used to estimate the centroid's height from the ball's

centroid. Let the vector $\boldsymbol{p}_m$ be the markers centroid position, $\boldsymbol{p}_{bm}$ be the mocap's ball centroid position, $\boldsymbol{\theta}_m$ be the mocap Euler angles, and $\boldsymbol{l}_m$ be the length vector from the ball to the marker centroid. Then the ball position can be calculated as follows:

$$\boldsymbol{p}_{bm} = \boldsymbol{p}_m - R_b(\boldsymbol{\theta}_m)\,\boldsymbol{l}_m, \qquad \boldsymbol{l}_m = \begin{bmatrix} 0 & 0 & 0.164 \end{bmatrix}^T. \tag{6.24}$$

Then the mocap's ball velocity can be estimated by using the 1st order central difference formula:

$$\dot{\boldsymbol{p}}_{bm,k} = (\dot{\boldsymbol{p}}_{bm,k+1} - \dot{\boldsymbol{p}}_{bm,k-1})/dt_m, \tag{6.25}$$

where dt_m is the mocap sampling period (8.3 ms, or 120 Hz). This ball velocity is not measured directly which makes it not as accurate as the mocap's orientation measurements. However, it certainly can capture the vibrations caused by the mechanical problems in the MBBR which is primarily in the 20-25 rad/s frequency range.

6.4.2 Friction Compensation Experiment

This experiment is done to show that not compensating or modeling the friction resulted in a much larger ball speed estimations than the actual value. This experiment were also done to show that using the friction compensation can achieve a much more accurate ball speed estimations. The following estimator process noise values were used in this experiment:

$$\begin{aligned} q_1^N = q_2^N &= 10^{-5}, & q_3^N &= 10^{-5}, & q_4^N = q_5^N = q_6^N = q_7^N &= 0.1 \\ q_8^N &= 2, & q_9^N = q_{10}^N &= 10^{-4}, & q_{11}^N = q_{12}^N = q_{13}^N &= 0.001. \end{aligned} \tag{6.26}$$

The following controller values were used in this experiment:

$$\begin{aligned} c_1 &= 8.5, & c_2 &= 0.3, & c_3 &= 0.4, & c_4 &= 0.65 \\ c_5 &= 0.4, & c_6 &= 0.9, & c_{SLC} &= 0.2, & \omega_{SLC} &= 0.961 \end{aligned} \tag{6.27}$$

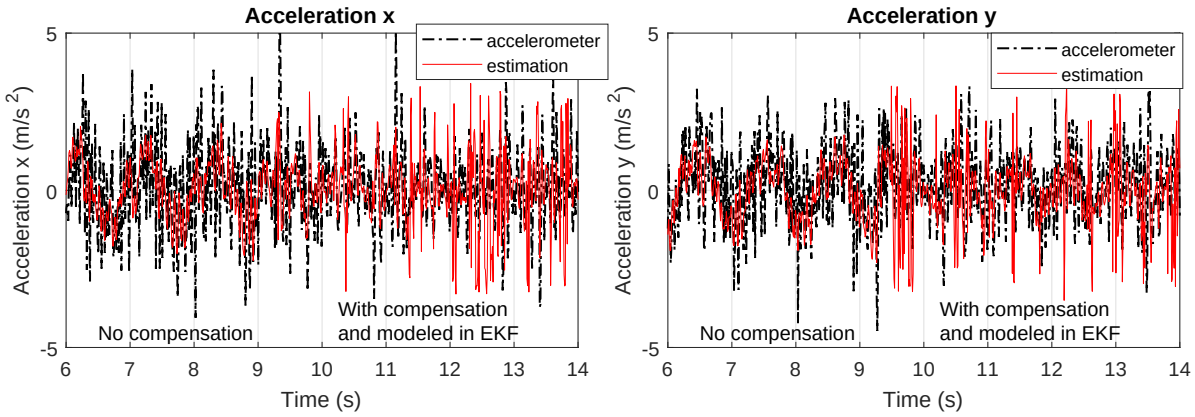


Figure 6.8: Accelerometer measurement vs EKF estimated sensor value with and without friction compensation.

The experiment were done with zero yaw rate reference, so only one set of controller gains were used. The EKF estimations of the θ_x, θ_y, $\dot{\phi}_x$, and $\dot{\phi}_y$ for the case with and without friction compensations can be seen in Fig. 6.5. The discussion of the result in Fig. 6.5 has already been done in §6.3.1. The acceleration measurement and EKF estimated value are shown in Fig. 6.8. This plot shows that by modeling the Coulomb friction in the EKF, the predicted acceleration measurement has large spikes near zero speed due to the nonlinearity in the Coulomb friction model. This might affect the state estimations near zero speed and produce a more noisy estimation. Even though this might help fighting against the static friction, we decided to not model the Coulomb friction in the EKF as mentioned in the §6.3.1.

6.4.3 Preliminary Motion Capture Experiment

This experiment is one of the preliminary experiments using the controller with SLC θ references. At this point of time, we did not use gain scheduling and low-pass filtering on the ball velocity estimates, which made the robot highly jittery and difficult to balance. The following

estimator process noise values were used in this experiment:

$$\begin{aligned} q_1^N = q_2^N = &10^{-5}, \qquad q_3^N = 10^{-5}, \qquad q_4^N = q_5^N = q_6^N = q_7^N = 0.1 \\ q_8^N = &2, \qquad q_9^N = q_{10}^N = 10^{-4}, \qquad q_{11}^N = q_{12}^N = q_{13}^N = 0.001. \end{aligned} \tag{6.28}$$

In this experiment, all controller gains are fixed which is shown below:

$$\begin{aligned} c_1 = &6.4, \quad c_2 = 0.19, \quad c_3 = 0.13, \quad c_4 = 0.0125 \\ c_5 = &0.4, \quad c_6 = 0.9, \quad c_{SLC} = 0.39, \quad \omega_{SLC} = 0.961 \end{aligned} \tag{6.29}$$

These gains are significantly different than the one used in the simulation (see §6.3.2, (6.22)), particularly the extremely small c_4 which is the ball speed gain. At the time of this experiment, the control of the robot was extremely difficult which was likely caused by the highly noisy ball speed estimates. Using c_4 gain larger than 0.05 caused extreme vibrations which made controlling the robot impossible. The c_{SLC} value is also larger than the one used in the simulation which was done to compensate the small c_4. The SLC controller has a build in low-pass filter which was the reason why the MBBR balanced well despite the large noise in the ball speed estimation. The list of the RMS values of the estimation error vs mocap under the tested driving conditions can be seen in Table 6.1. There is a trend where the estimation error increases as the yaw rate and drive rate increases. This implies that driving at higher yaw rates and drive rates is more unstable due to the increased estimation errors, which was the case in all of our experiments. This controller was stable up to approximately 0.75 Hz yaw rate. At yaw rates of 1 Hz and higher, the controller struggles to balance and often can't maintain position tracking for an extended period of time. The RSME increases more with respect to the yaw rate than the drive rate, which makes yaw rate the limiting factor in achieving simultaneous translation and rotation.

The plots of the mocap measurement vs. EKF estimations in different driving conditions can be seen in Fig. 6.9 to 6.12. The EKF θ_x estimation at the beginning of Fig. 6.9 has an offset

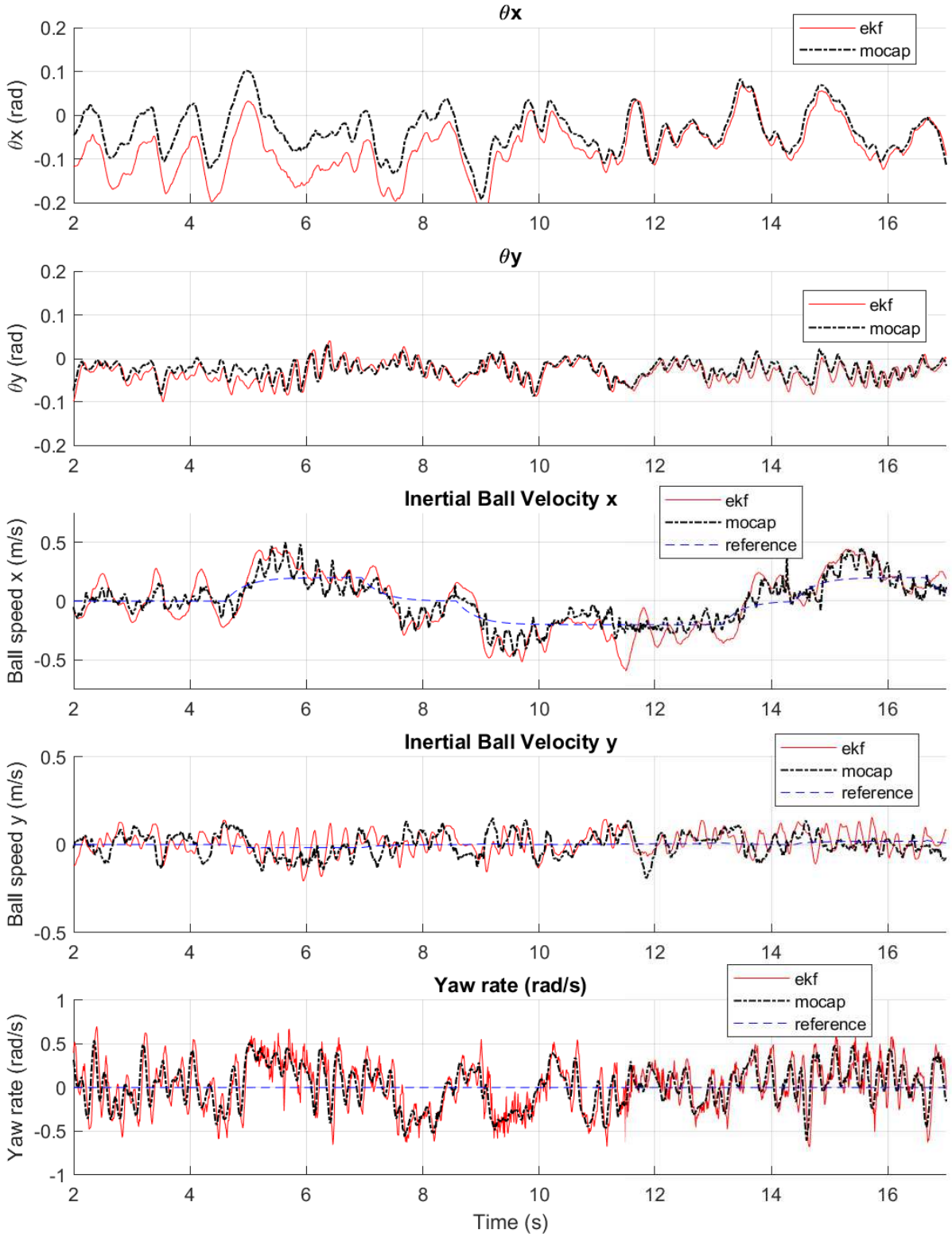


Figure 6.9: EKF estimates vs mocap plots under 0 yaw rate and 20 cm/s drive rate.

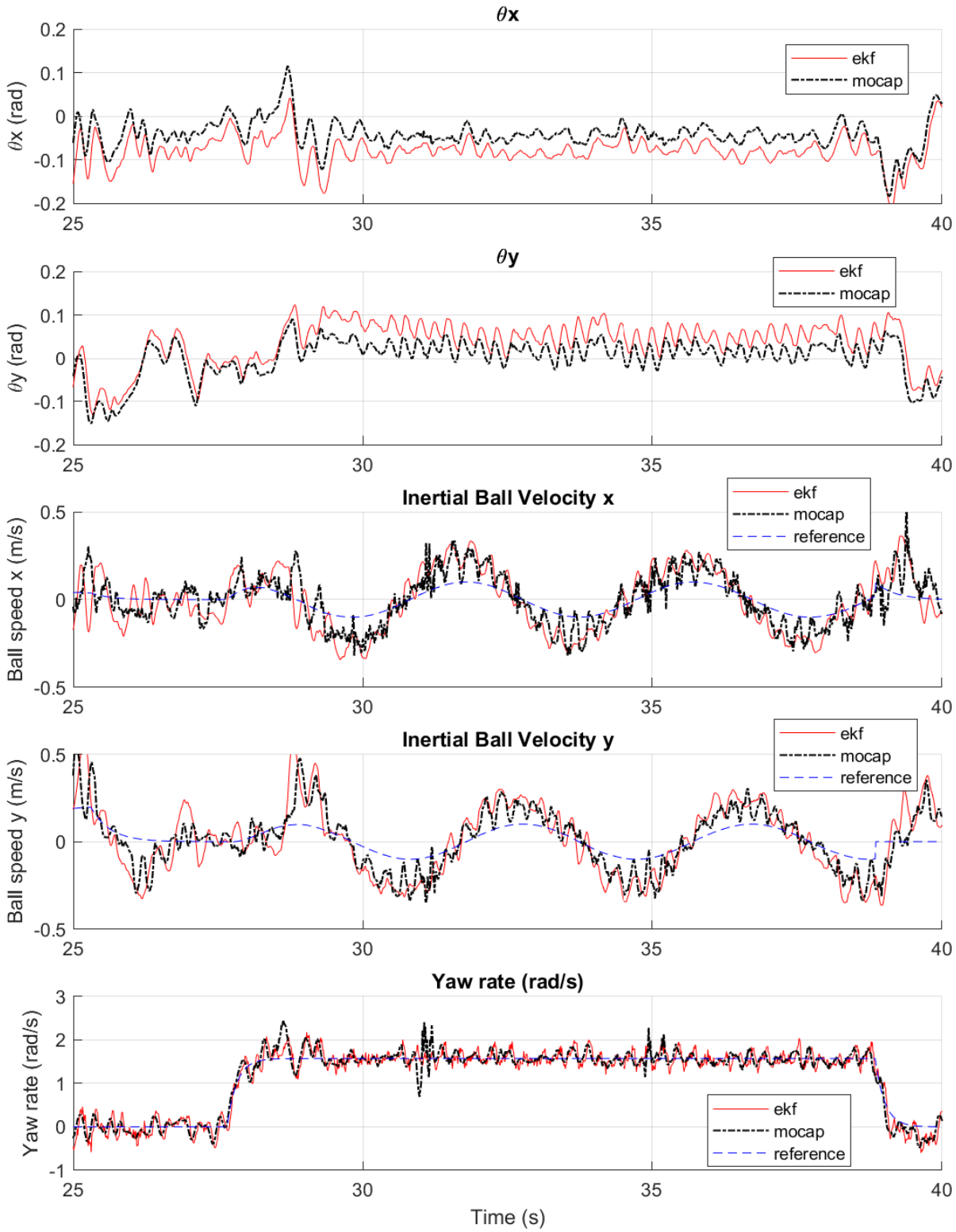


Figure 6.10: EKF estimates vs mocap plots under 0.25 Hz yaw rate and 10 cm/s drive speed.

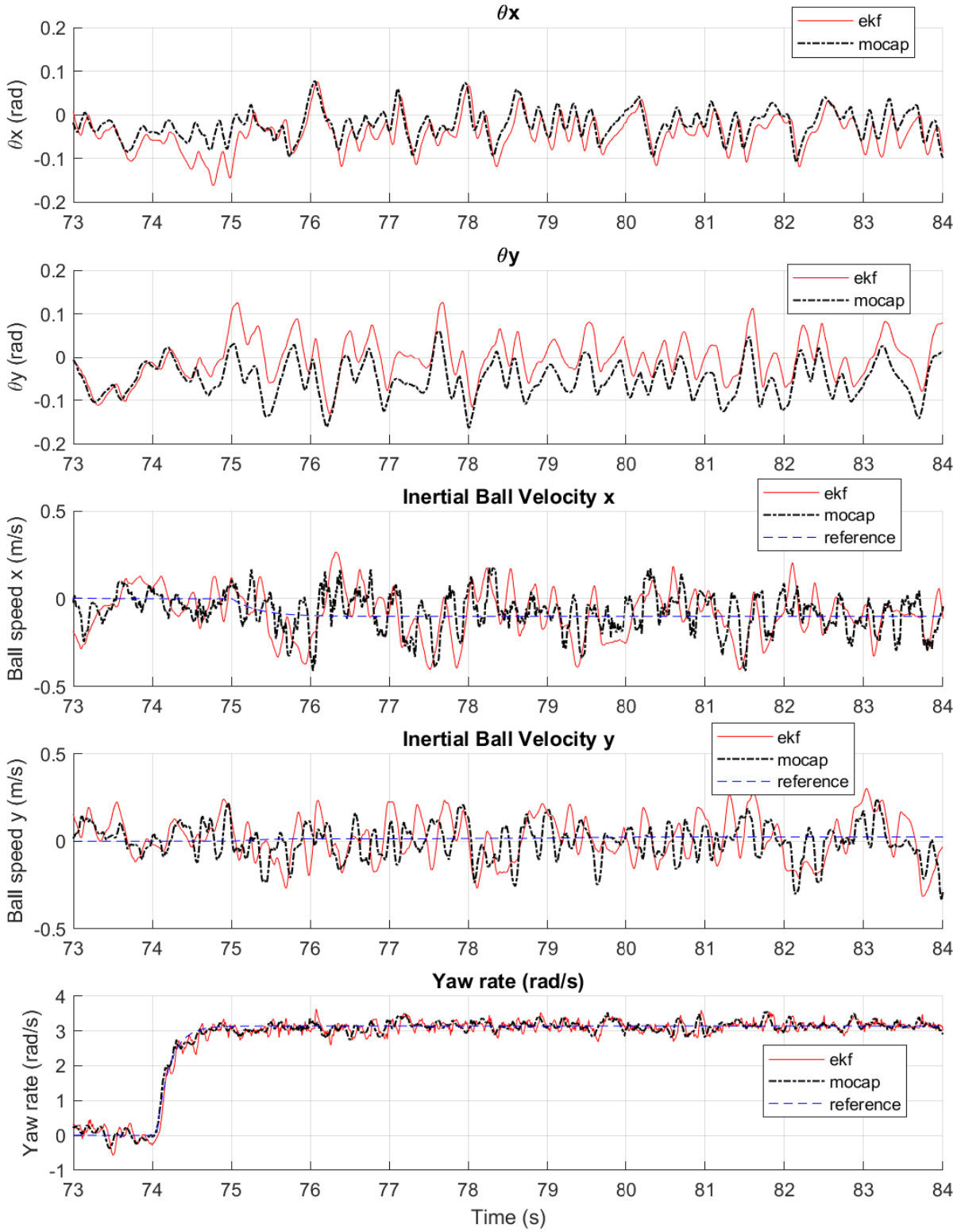


Figure 6.11: EKF estimates vs mocap plots under 0.5 Hz yaw rate and 10 cm/s drive speed.

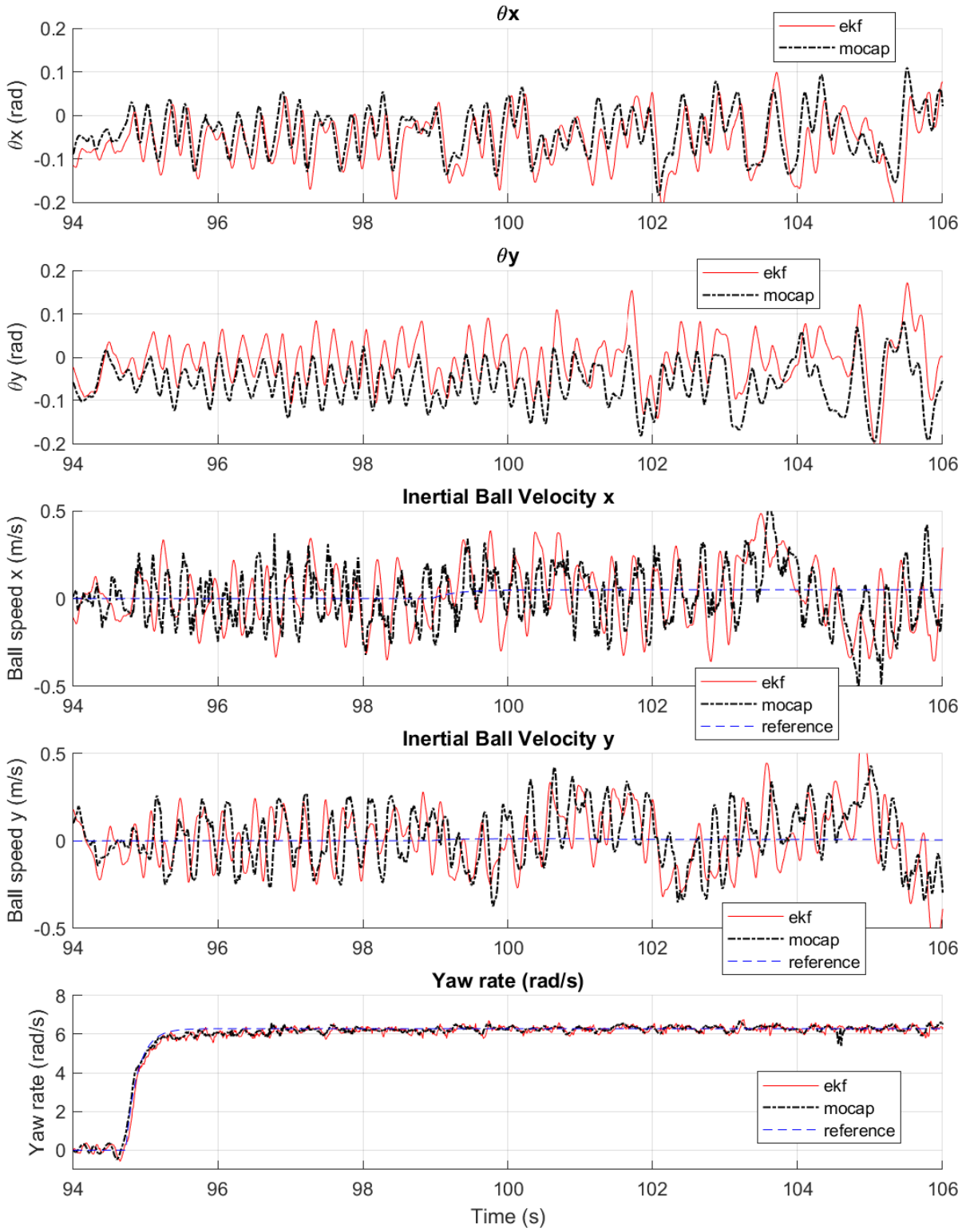
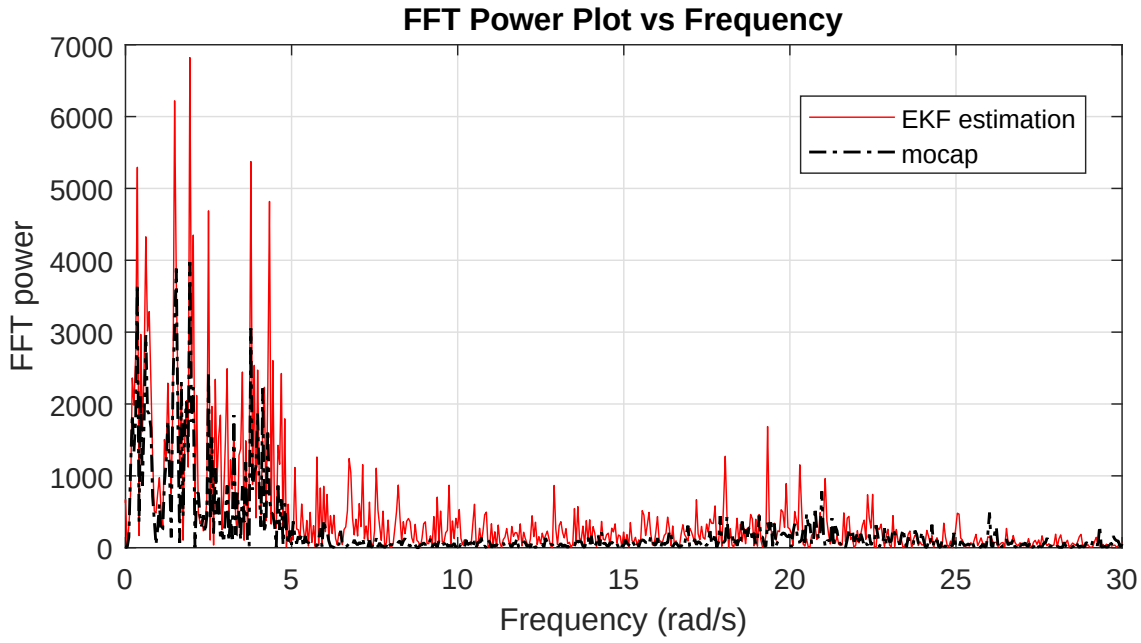
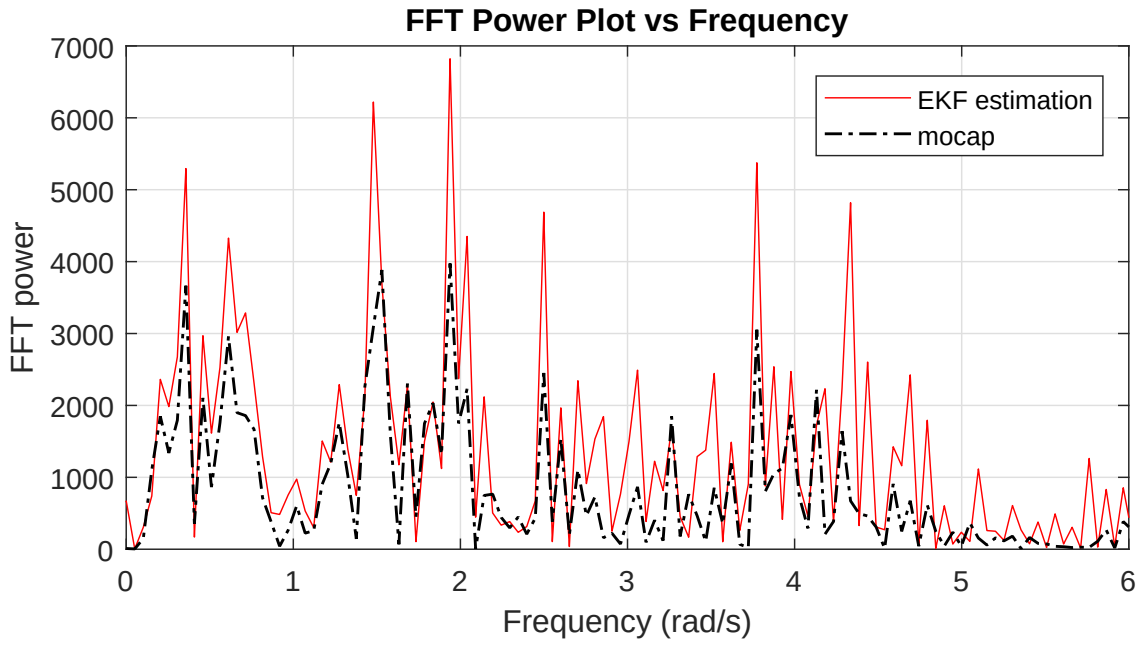


Figure 6.12: EKF estimates vs mocap plots under 1 Hz yaw rate and 5 cm/s drive speed.



(a) FFT plot.



(b) Zoomed in FFT plot.

Figure 6.13: FFT plot of the $\dot{\phi}_x$ EKF estimation and mocap measurements.

Table 6.1: List of RMS error of the θ and $\dot{\phi}$ estimations vs mocap in the x and y directions.

Driving conditions	RMSE θ (rad)	RMSE $\dot{\phi}$ (cm/s)
Idle	0.017	8.3
Translate 20 cm/s	0.031	9.6
Spin 0.5 Hz	0.056	13
Spin 1 Hz	0.065	19
Translate 10 cm/s & spin 0.25 Hz	0.036	9.9
Translate 10 cm/s & spin 0.5 Hz	0.057	14
Translate 5 cm/s & spin 1 Hz	0.077	21

error that slowly disappeared which we believe was caused by initialization error. Otherwise, there is no problem in the θ_x and θ_y estimates as the estimation errors are mostly offset errors. The problem is in the ball velocity estimations, where both the mocap measurements and the EKF estimates have large amount of noise. However, the signals seem to have approximately the same averaged values which incentivized the use of low-pass filter in the following mocap experiments.

The Fourier Transform plots of the ball velocity from the estimation and mocap can be seen in Fig. 6.13. As shown in the FFT plot, the EKF estimations are very noisy when compared to the mocap measurements. However, there is a significant high frequency response in the system measured by the mocap at around 17 to 22 rad/s range. This might be attributed to the DROW rollers transition which causes microslips and vibrations into the system. The mocap experiment shown in §6.4.5 attempts to analyze if the driving speed also affects the frequency response around this frequency range. The frequency separation between the important system dynamics (< 5 rad/s) and the high frequency vibrations (18-25 rad/s) are relatively small. This means that we need to use an aggressive low-pass filter with a cutoff frequency at no less than 10 rad/s in order to eliminate this noise. We choose to use a 1st order low-pass filter with cutoff frequency of 10 rad/s which increases up to 20 rad/s as the desired yaw rate increases, as mentioned in the §6.2.1.

6.4.4 Estimation Accuracy Experiment

This experiment is designed to verify the estimation accuracy of the linear model KF, high yaw-rate model EKF, and the DMP. The low-pass filtered ball velocity estimates were used starting from this experiment, which helped balancing the robot more smoothly and stably. The experiment tested the robot to spin in place under the following yaw rates: 0, $\pm$ 0.5, $\pm$ 1, and $\pm$ 1.5 Hz. Maintaining position hold while spinning in place is the prerequisite condition for position tracking under nontrivial yaw rates. Since the KF can't maintain position hold while spinning, the controller used in this experiment only used the estimated states from the EKF. By setting up the experiment this way, we can use the stable controller and compare the estimation performance separately from the control problem. The estimated states of θ_x, θ_y, $\dot{\phi}_x$, and $\dot{\phi}_y$ are compared with the motion capture measurements.

The following process noise covariance matrices were used in this experiment:

$$\begin{aligned} q_1^N &= q_2^N = 10^{-5}, \qquad & q_3^N &= q_4^N = q_5^N = 10^{-4}, \qquad & q_6^N &= q_7^N = 0.1 \\ q_8^N &= 2, & q_9^N &= q_{10}^N = 0.1, & q_{11}^N &= q_{12}^N = q_{13}^N = 0.001 \end{aligned} \tag{6.30}$$

Table 6.2 lists the controller gains and SLC outer loop poles used in this experiment. It is very difficult to control the MBBR with just a single control gains for all tested yaw rates, so the gains were tuned manually for each target yaw rates.

Figure 6.14 shows the RMS error of the estimations vs. the mocap measurements. The θ_x and θ_y compares the estimations from EKF, KF and DMP, while the $\dot{\phi}_x$ and $\dot{\phi}_y$ compares the estimations from EKF and KF only. The EKF θ estimations is significantly more accurate than the KF as we expected, but the DMP has even more accurate θ estimation than the EKF. We have shown that the DMP has great estimations for θ in Chapter 3, and this experiment also shown the same result. Surprisingly, the KF and EKF have almost identical $\dot{\phi}_x$ and $\dot{\phi}_y$ accuracy across all tested yaw rates. This result implies that the MBBR can be controlled by using the θ from the

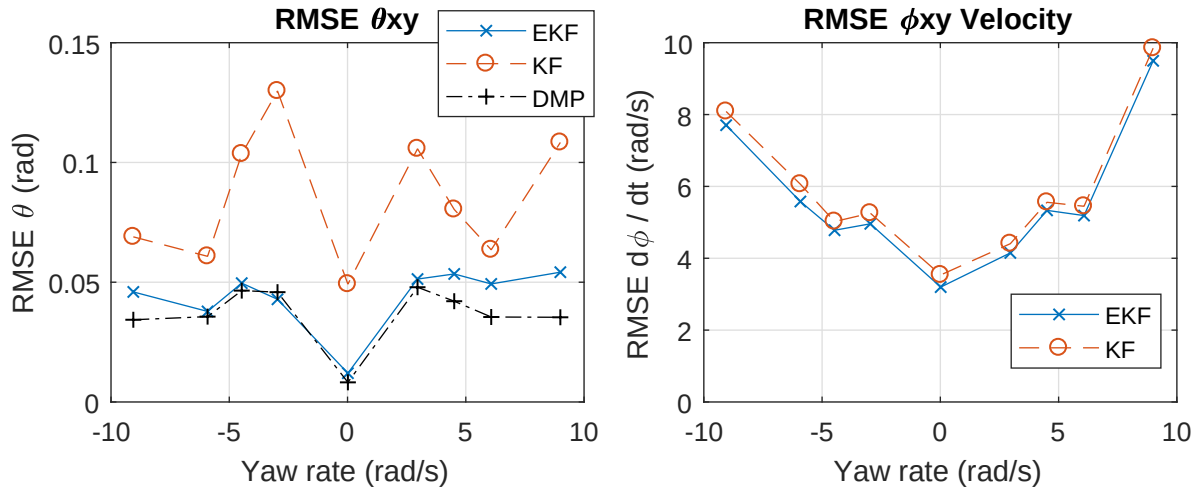


Figure 6.14: The RMS of the estimation error of KF, EKF, and DMP estimations vs mocap.

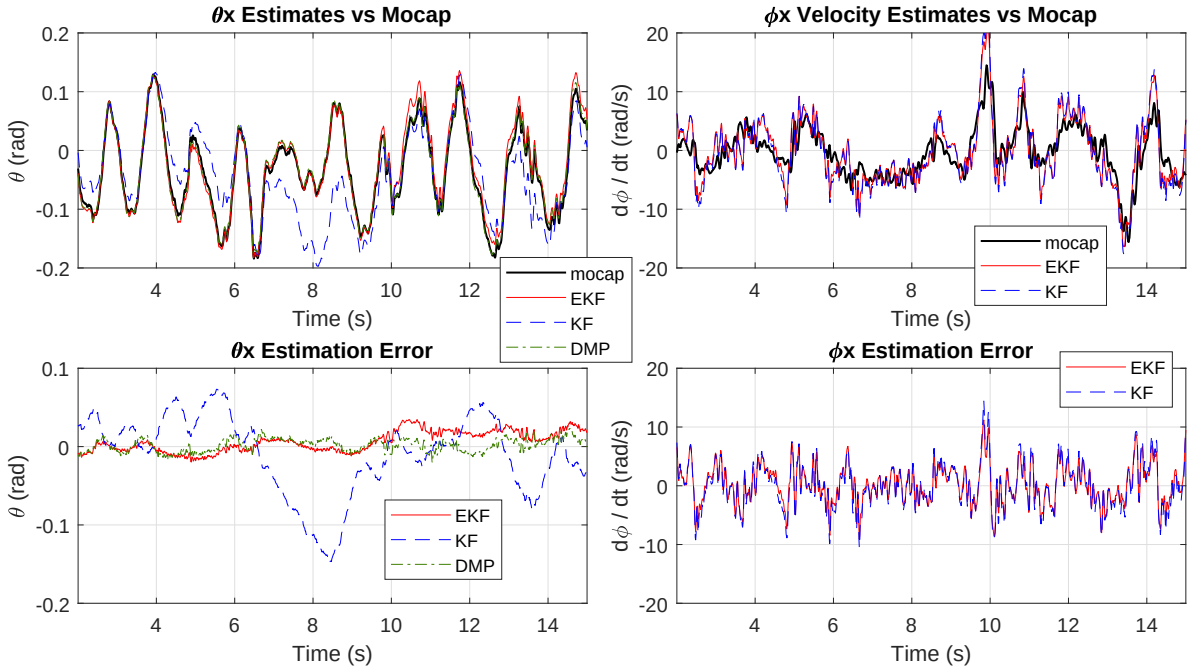


Figure 6.15: The estimations vs mocap and the estimation error plot at 0 Hz yaw rate.

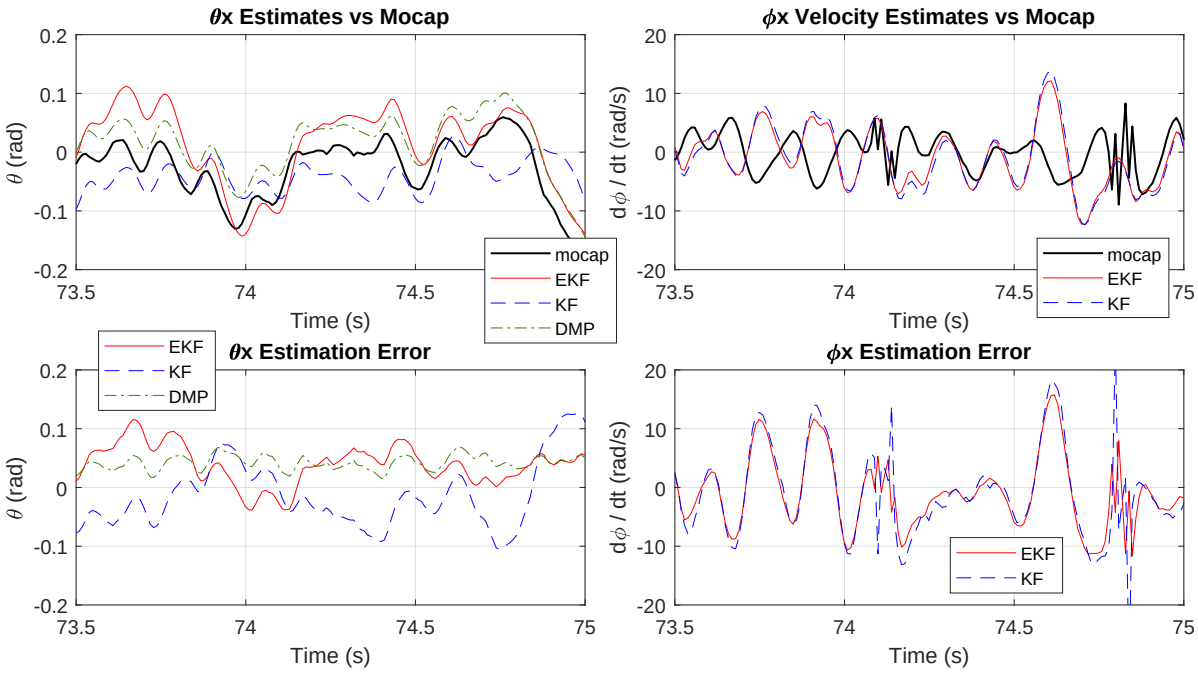


Figure 6.16: The estimations vs mocap and the estimation error plot at -1.5 Hz yaw rate.

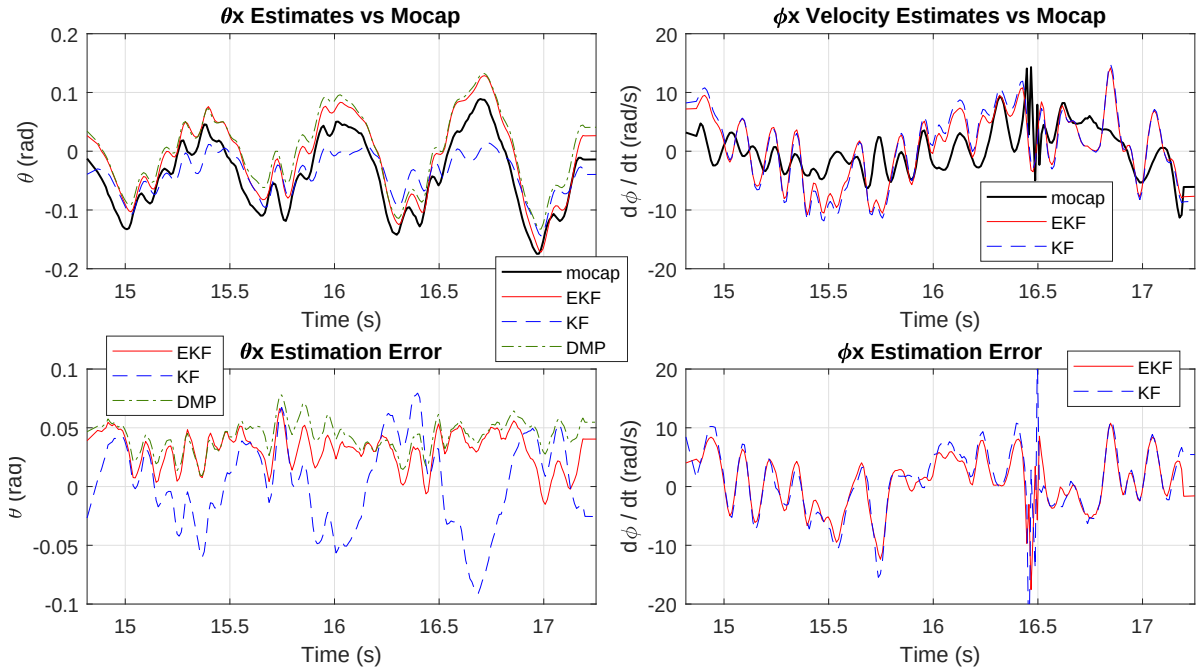


Figure 6.17: The estimations vs mocap and the estimation error plot at -1.0 Hz yaw rate.

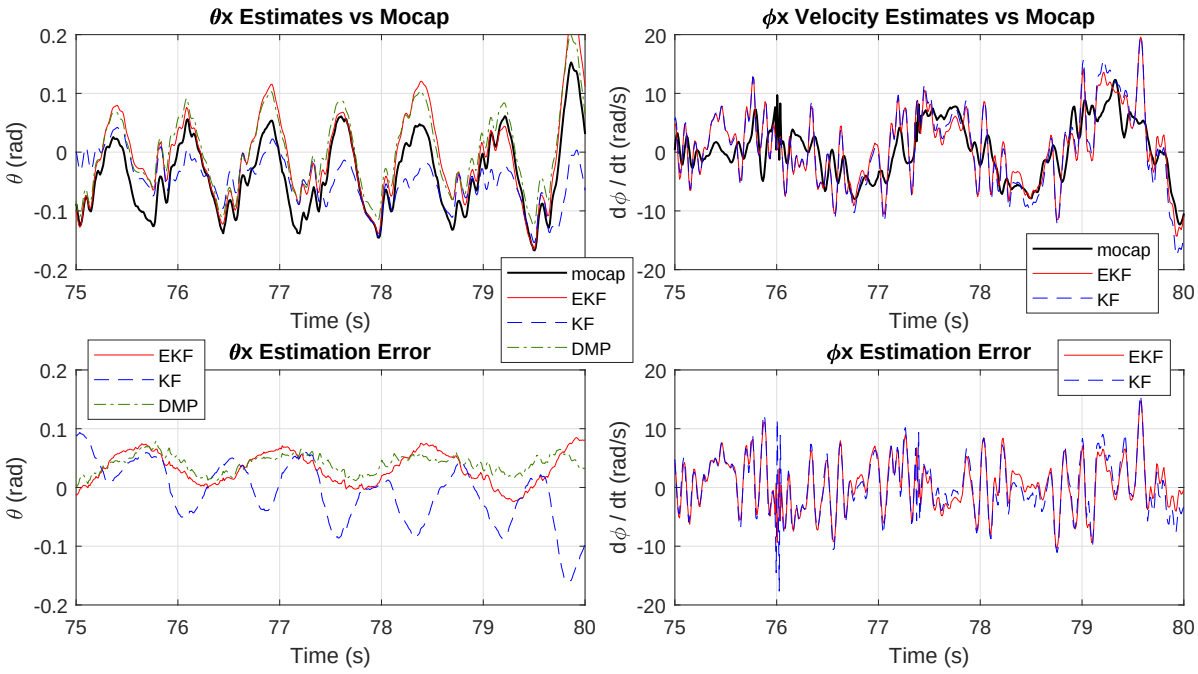


Figure 6.18: The estimations vs mocap and the estimation error plot at -0.75 Hz yaw rate.

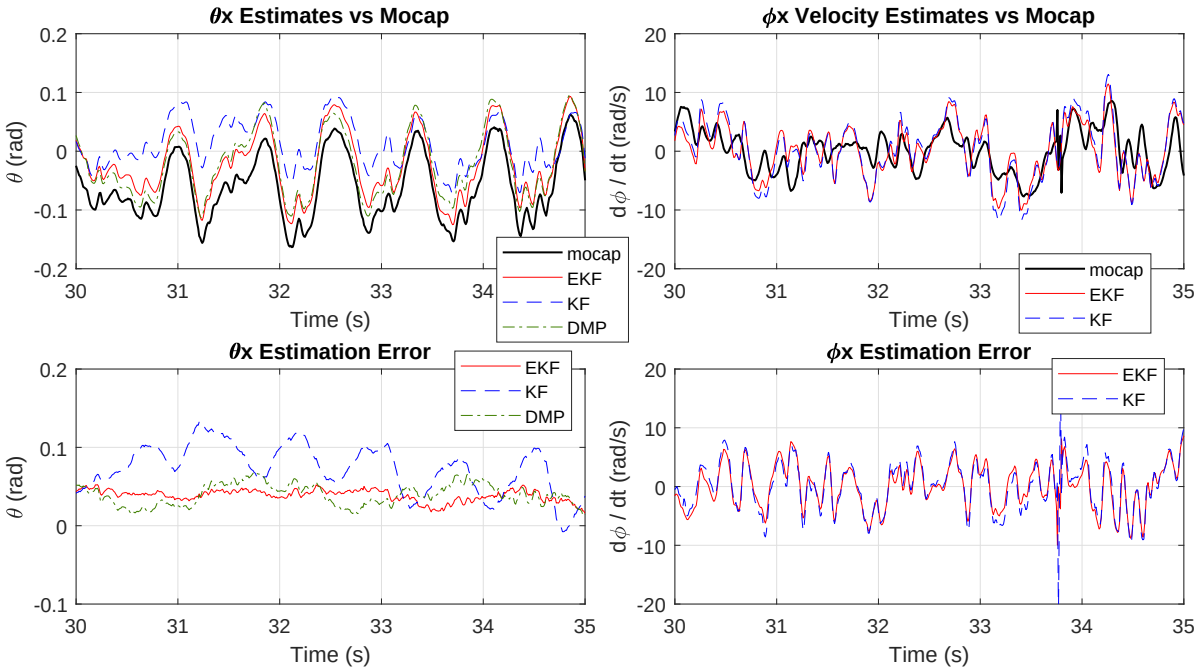


Figure 6.19: The estimations vs mocap and the estimation error plot at -0.5 Hz yaw rate.

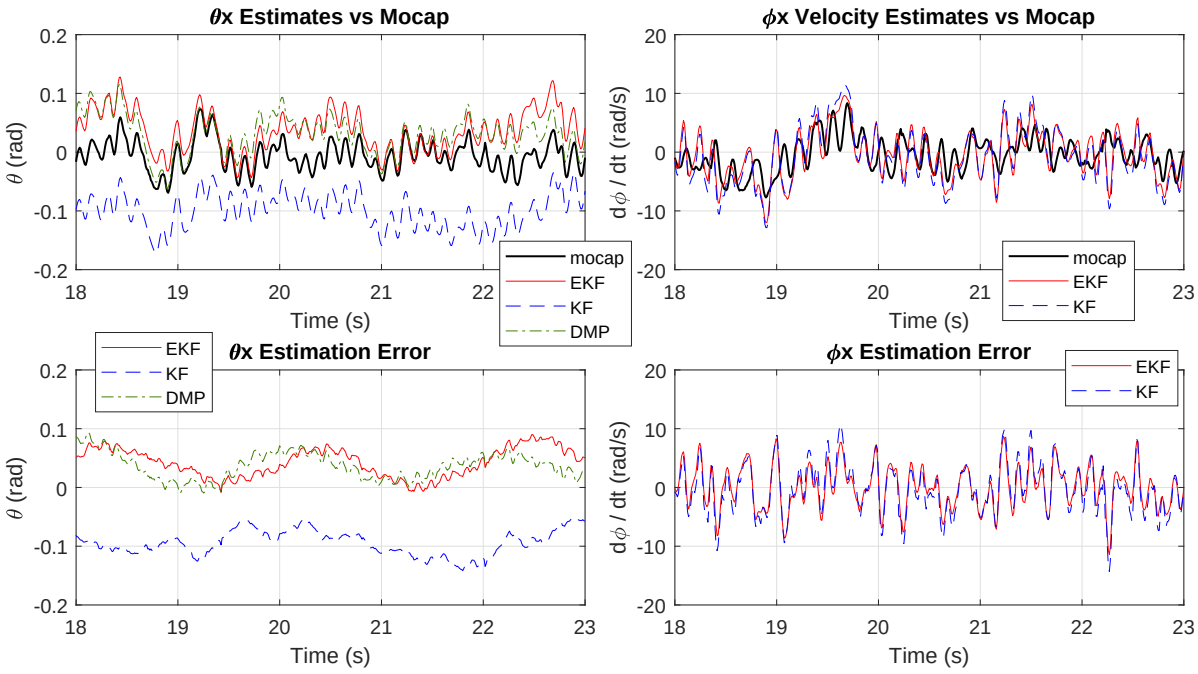


Figure 6.20: The estimations vs mocap and the estimation error plot at 0.5 Hz yaw rate.

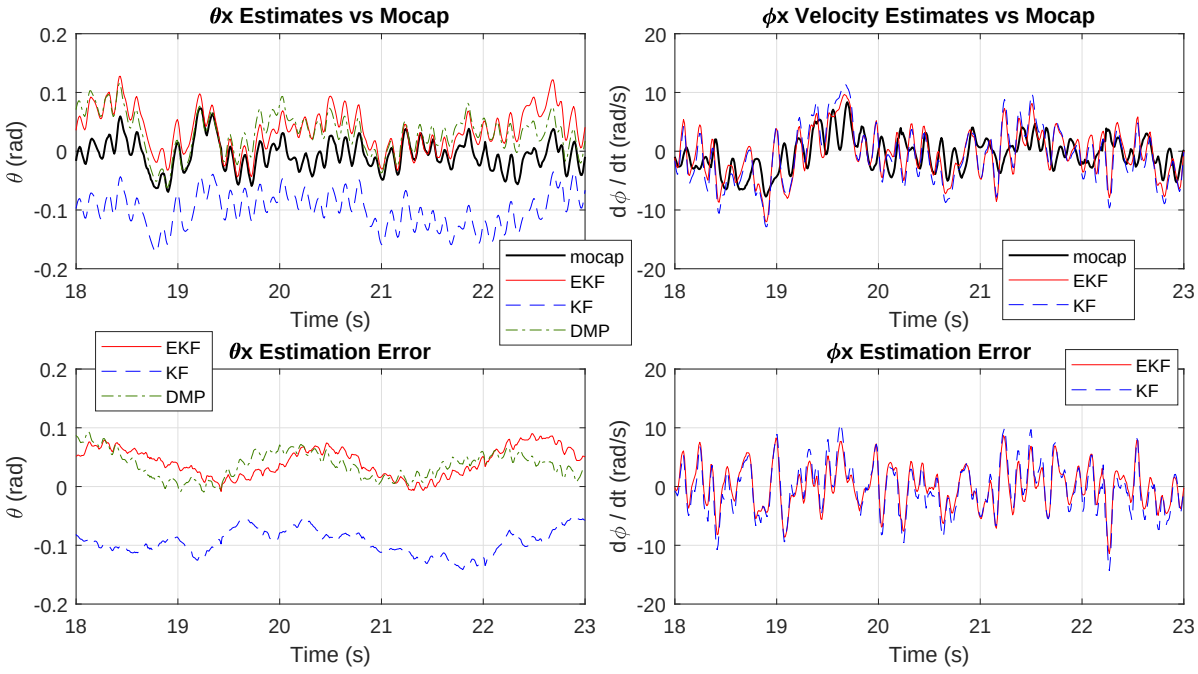


Figure 6.21: The estimations vs mocap and the estimation error plot at 0.75 Hz yaw rate.

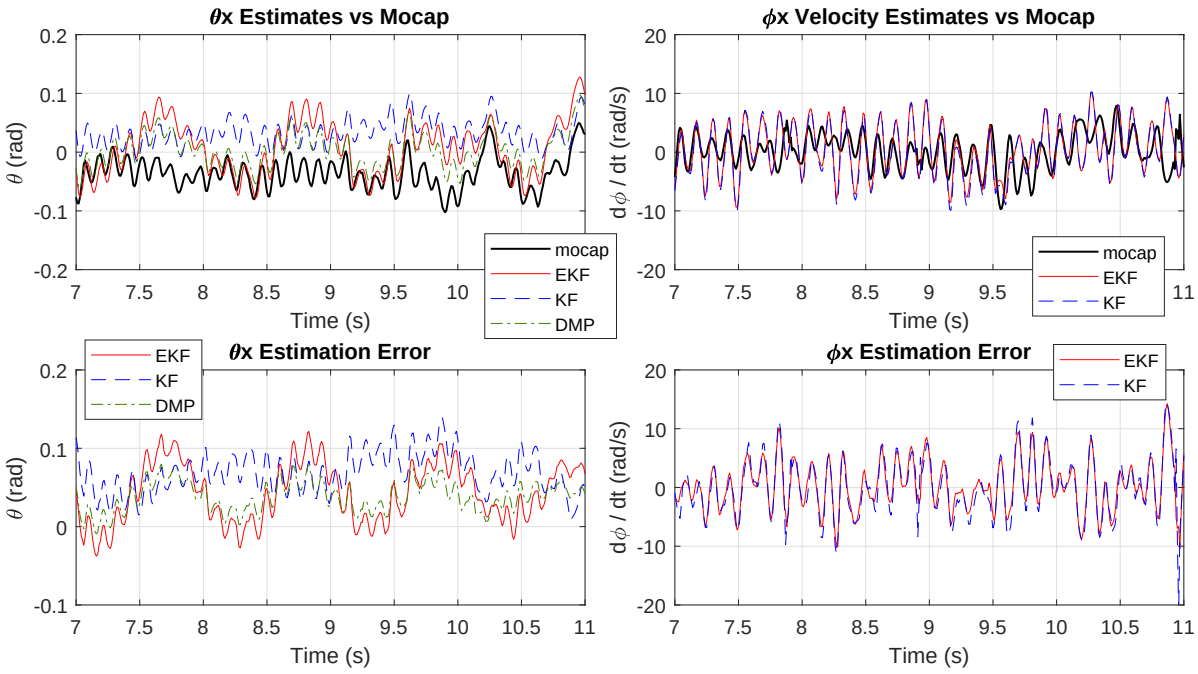


Figure 6.22: The estimations vs mocap and the estimation error plot at 1.0 Hz yaw rate.

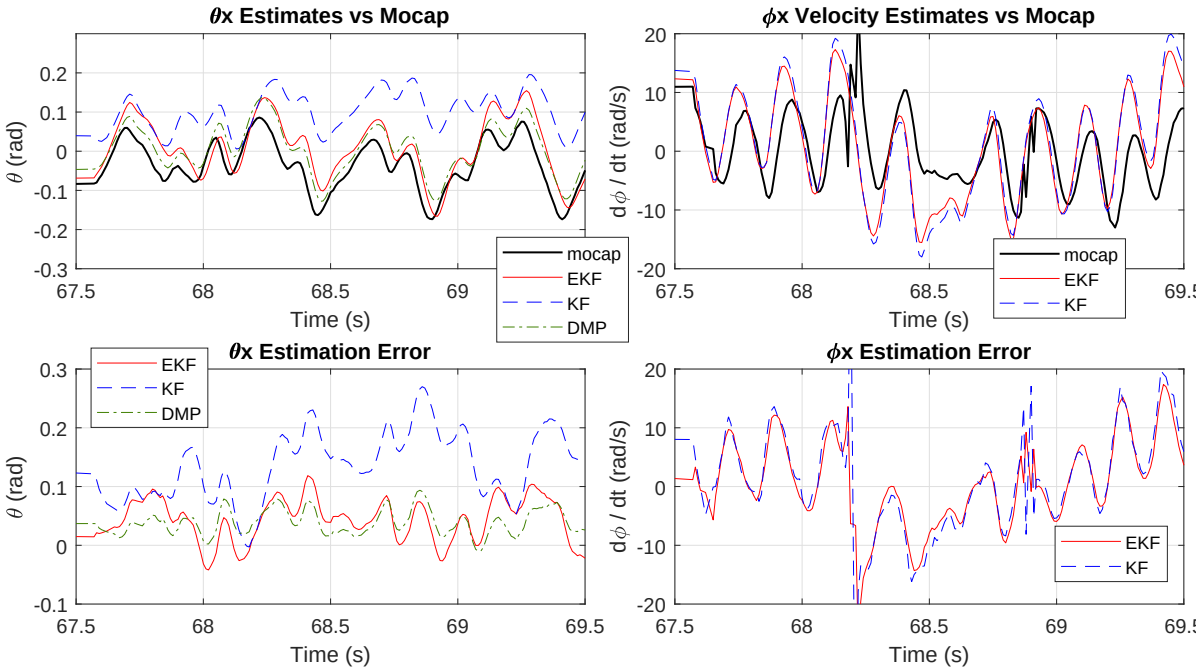


Figure 6.23: The estimations vs mocap and the estimation error plot at 1.5 Hz yaw rate.

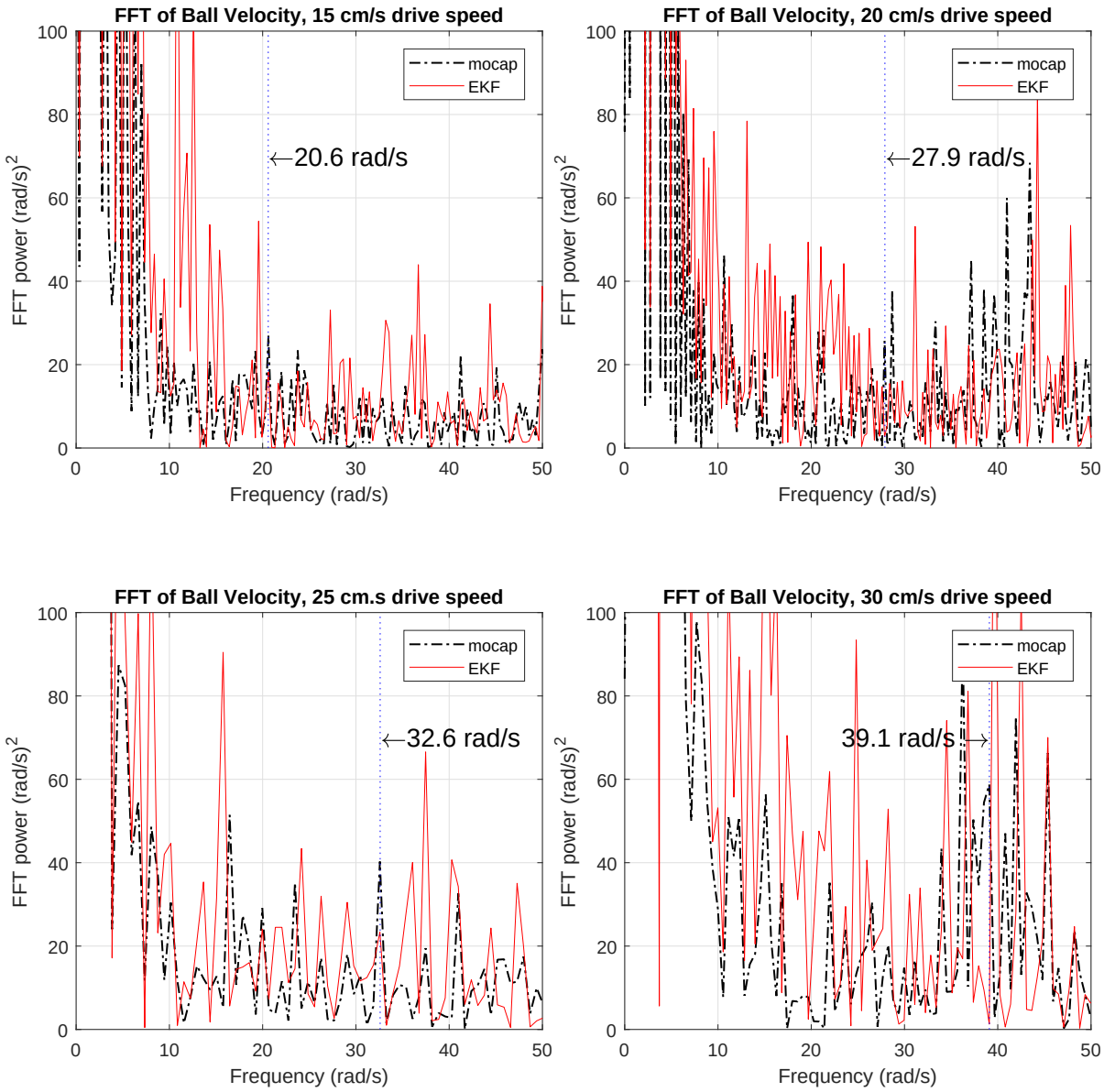


Figure 6.24: The FFT of the ball speed EKF estimation and Mocap measurement across different driving speed.

Table 6.2: List of the control gains and SLC poles for each target yaw rates for the spinning in place experiment.

Gains	**Target yaw rate magnitude (rad/s)**				
	0	0.5	0.75	1.0	1.5
c_1	10	12	14	13	12
c_2	0.4	0.4	0.4	0.4	0.6
c_3	0.5	0.5	0.6	0.6	0.5
c_4	0.075	0.10	0.10	0.075	0.060
c_5	0.4	0.4	0.4	0.4	0.4
c_6	1.2	1.2	1.2	1.2	1.2
c_{SLC}	0.20	0.20	0.15	0.10	0.10
ω_{SLC}	0.970	0.968	0.963	0.960	0.951

DMP estimation and the linear model KF's estimations for the remaining states, which happened to be true and will be demonstrated in §6.4.6.

Figures 6.15 to 6.23 show the estimated states vs the mocap measurement and their estimation errors. The DMP and EKF θ estimations are relatively close to each other, but the DMP's estimation error almost always have smaller amplitude than the EKF. Both $\dot{\phi}$ estimations and mocap measurements are very noisy in all driving conditions. Some of these vibrations might be caused by the DROW roller transition, which means that some of the vibration frequency increases linearly with respect to the drive rate. Figure 6.24 shows the FFT plot of the $\dot{\phi}$ from EKF estimations and the mocap measurements. As shown in this figure, some of the mocap FFT frequency responses seem to increase linearly in frequency as the drive rate increases. This vibration pattern might be caused by the roller transition in the DROWs where the vibration frequency increases as the drive rate increases. This implies that some of the high frequency noise in the $\dot{\phi}$ estimates is inherent in the system and can't be avoided. Therefore, a low-pass filter is used to prevent these vibrations to negatively affect the controller. On average, the $\dot{\phi}$ estimation errors have zero mean, which means that the inertial position drift due to the integration error of the $\dot{\phi}$ states is relatively small. This result can be seen more in the position tracking experiments in §6.4.5 and §6.4.6.

6.4.5 EKF Control Experiments

In this experiment, the states estimations from the EKF is used to control the MBBR and under several simultaneous spinning and translation rates. The DMP θ estimations are also shown to compare them with the EKF's estiamtions. The following estimated states are compared with the mocap measurements: θ_x, θ_y, θ_z, $\dot{\theta}_z$, ball inertial position and velocity. The MBBR were tested under the following driving conditions:

- 0.5 Hz yaw rate and 20 cm/s drive rate.
- 0.75 Hz yaw rate and 15 cm/s drive rate.
- 1.0 Hz yaw rate and 10 cm/s drive rate.
- 1.5 Hz yaw rate and 5 cm/s drive rate.

The drive rates for each yaw rates are the fastest drive rate where the controller can reliably stabilize the MBBR. It is possible to drive faster, but there will be more overshoot and less stability. Table 6.3 lists the controller gains used in this experiment and we used the following process noise covariance matrix values:

$$\begin{aligned} q_1^N &= q_2^N = 10^{-5}, \qquad q_3^N = q_4^N = q_5^N = 10^{-4}, \qquad q_6^N = q_7^N = 0.1 \\ q_8^N &= 2, \qquad\qquad q_9^N = q_{10}^N = 0.001, \qquad q_{11}^N = q_{12}^N = q_{13}^N = 0.001. \end{aligned} \tag{6.31}$$

Figures 6.25 to 6.28 show the EKF and DMP estimated states, mocap measurements, and the state reference values during the experiment. Note that the ball velocity shown are smoothed due to excessive noise in both the mocap and estimation data. The actual estimated states and measurements look more like the data in §6.4.3 and §6.4.4, so the data is smoothed out for clarity. The smoothing function is a simple zero-phase 1st order low-pass filter with a cutoff frequency of 10 rad/s. The low-pass filter is simply applied forward and backwards in time which then are averaged to get the smoothed signal.

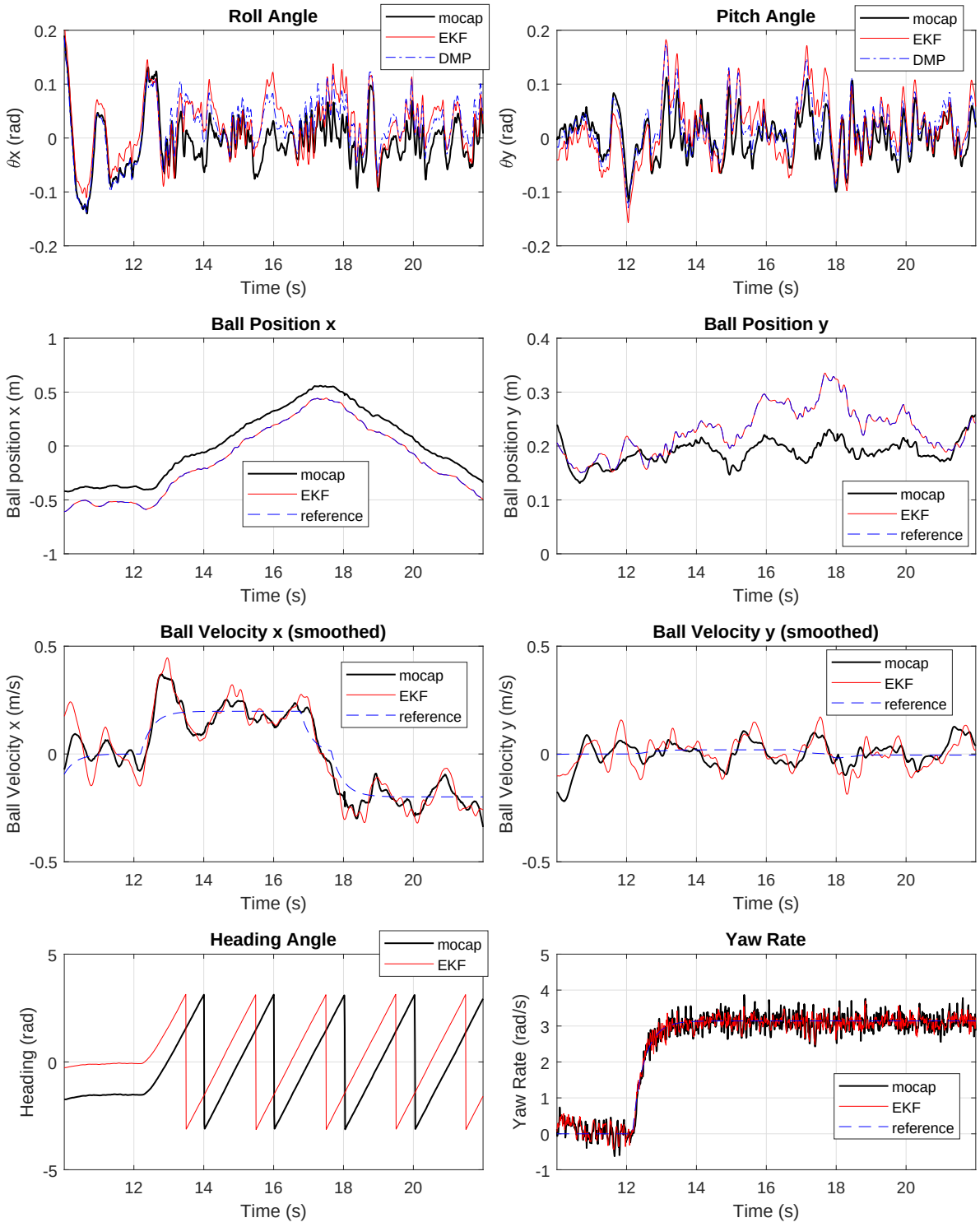


Figure 6.25: The EKF estimation and control performance under 0.5 Hz yaw rate and 20 cm/s drive rate.

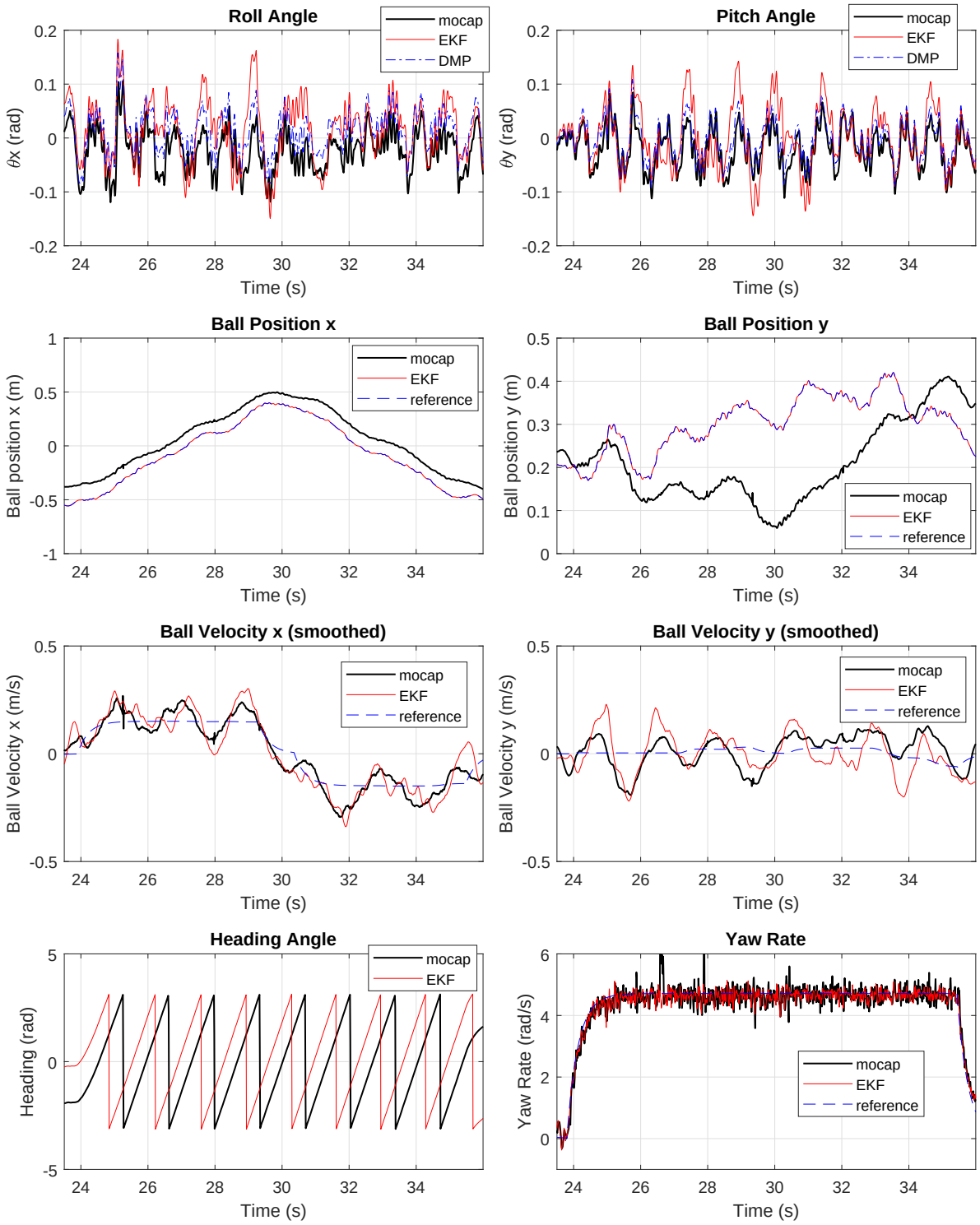


Figure 6.26: The EKF estimation and control performance under 0.75 Hz yaw rate and 15 cm/s drive rate.

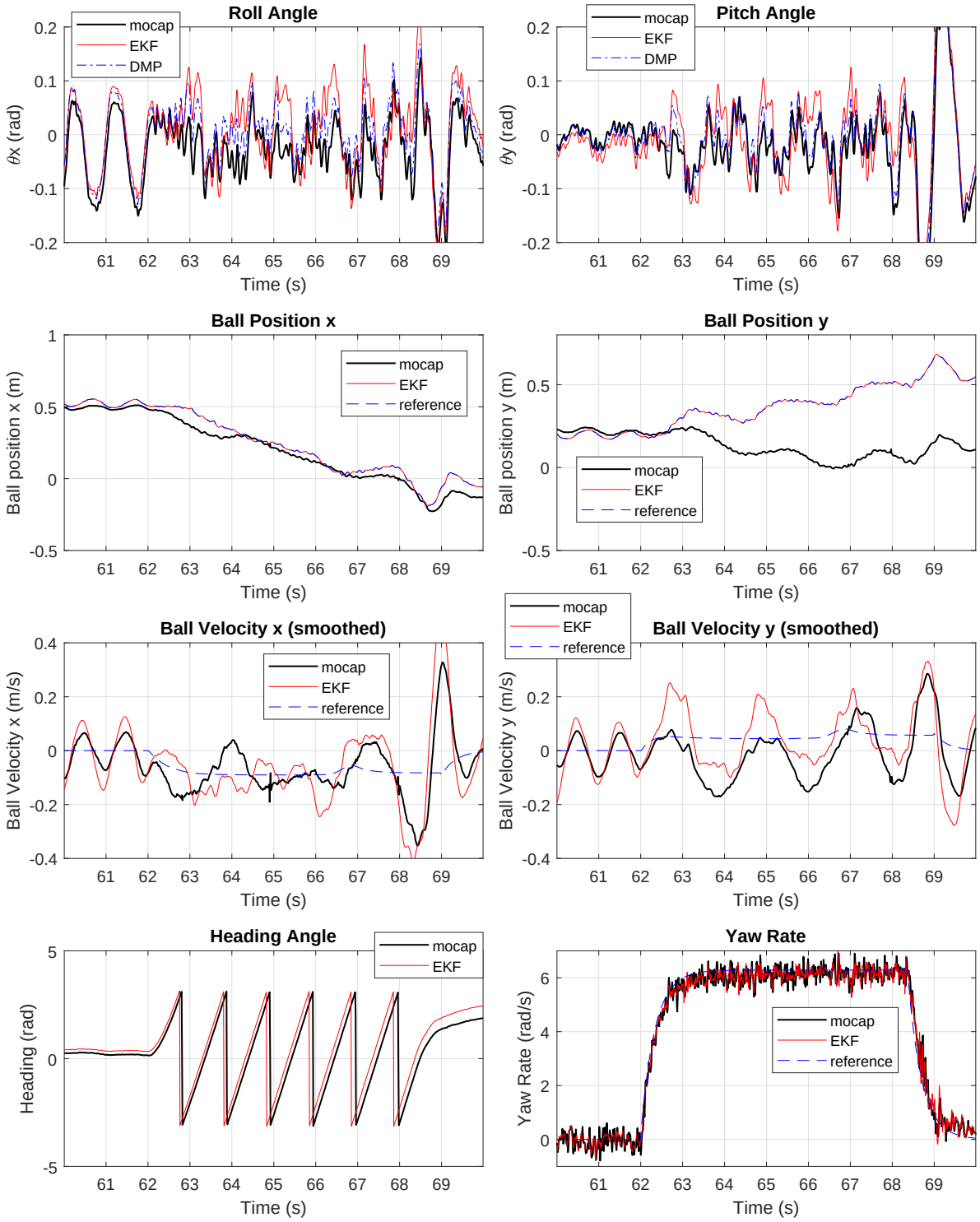


Figure 6.27: The EKF estimation and control performance under 1.0 Hz yaw rate and 10 cm/s drive rate.

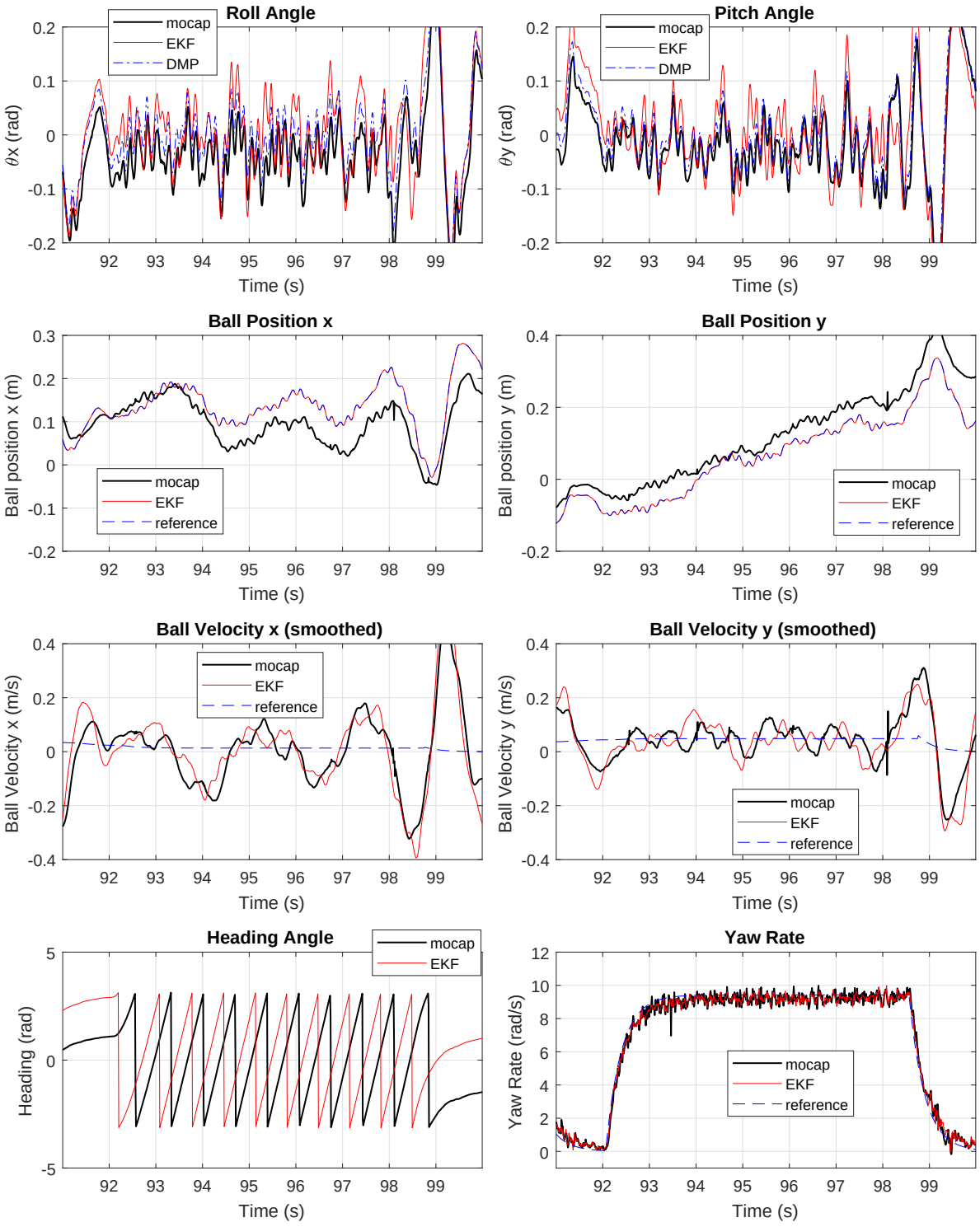


Figure 6.28: The EKF estimation and control performance under 1.5 Hz yaw rate and 5 cm/s drive rate.

Table 6.3: List of the control gains and SLC poles for each target yaw rates for the spinning EKF control experiment.

Gains	**Target yaw rate magnitude (rad/s)**				
	0	0.5	0.75	1.0	1.5
c_1	10	12	14	14	12
c_2	0.4	0.4	0.4	0.5	0.6
c_3	0.5	0.5	0.6	0.6	0.5
c_4	0.075	0.10	0.10	0.085	0.060
c_5	0.4	0.4	0.4	0.4	0.4
c_6	1.2	1.2	1.2	1.2	1.2
c_{SLC}	0.20	0.20	0.15	0.10	0.10
ω_{SLC}	0.970	0.968	0.963	0.960	0.951

The MBBR have achieved a stable driving and good position tracking performance in all tested yaw rates, except for the 1.5 Hz yaw rate case where it slowly lost balance after 8 seconds. Figure 6.28 shows that both the θ and ball speed estimations are relatively inaccurate compared to the lower yaw rates case. There are some integration error build up in the ball position estimates, especially in the direction where the robot is not driving into. This behavior also appeared in the simulation, which is shown in Fig. 6.6. However, this drift happened very slowly and generally not a huge issue for a remote controlled robot. The robot only have the encoders measurements for odometry and these measurements are not reliable under high yaw rates. The controller seems to track the ball position reference almost exactly during the experiment, which showed that the controller works well.

As shown in the data, the EKF θ estimation is less accurate when compared to the DMP during high yaw rates driving. The DMP accuracy and the mocap experiment in §6.4.4 have shown that the KF also have similar ball speed estimation accuracy as the EKF. All of the results indicates that we might be able to control the robot by using the KF estimations and use the DMP θ estimates. The controller did work very well and it showed that the driving performance is more stable than what the EKF can achieve by itself, which will be shown in the following subsection.

6.4.6 KF with DMP Control Experiment

The control experiment in §6.4.5 is repeated, but by using the estimated states from the DMP and linear model KF. The KF estimates all the states except for θ_x, θ_y, θ_z, $\dot{\theta}_z$, which will be covered by the DMP and raw gyro z measurement. The same controller gains as the EKF case were used (see Table 6.3) and the following process noise covariance matrix were used:

$$q_1^L = q_2^L = 10^{-5}, \qquad q_3^L = q_4^L = 10^{-4}, \qquad q_5^N = q_6^N = 0.1, \qquad q_7^N = q_8^N = 0.001 \tag{6.32}$$

Figures 6.29 to 6.32 show the estimation and control performance of the DMP+KF controller. The EKF estimation were also shown in the θ_x and θ_y plots as a comparison to the DMP's performance. The performance of this controller is better than the controller which uses the EKF estimations. This controller achieved a much stabler position tracking under 1.5 Hz yaw rate, which is the yaw rate where the EKF was struggling to balance. The ball velocity estimations at 1.5 Hz yaw rate has an offset error, but the estimated ball velocity states tracked the reference states stably unlike the EKF's case. The DMP+KF controller has similar ball position drifts that the EKF has, but this is not a huge issue as mentioned in the EKF's case.

The data showed in the θ estimations might show what the EKF is lacking compared to the DMP. Even though the EKF estimated the θ value at the correct phase, the magnitude is sometimes too large compared to the actual θ. The EKF's magnitude error fluctuation seems to be periodic in nature and might be the primary cause of instability.

6.5 Experimental Results Discussion

Several motion capture experiments have been done to show the state estimation and controller performance of the EKF, KF and DMP to control the MBBR under various yaw and drive rates. We have achieved our target performance for position tracking under high yaw rates

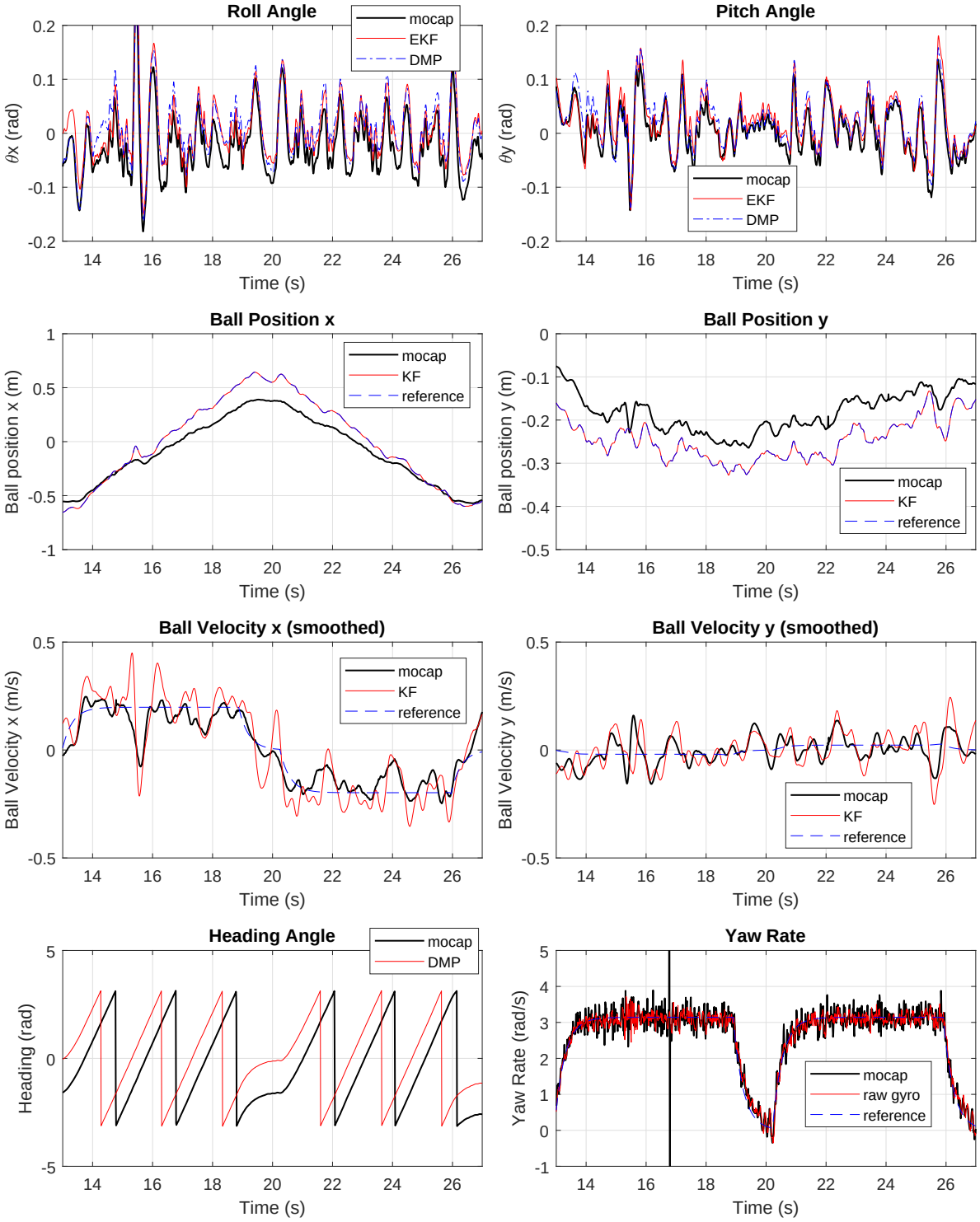


Figure 6.29: The DMP+KF estimation and control performance under 0.5 Hz yaw rate and 20 cm/s drive rate.

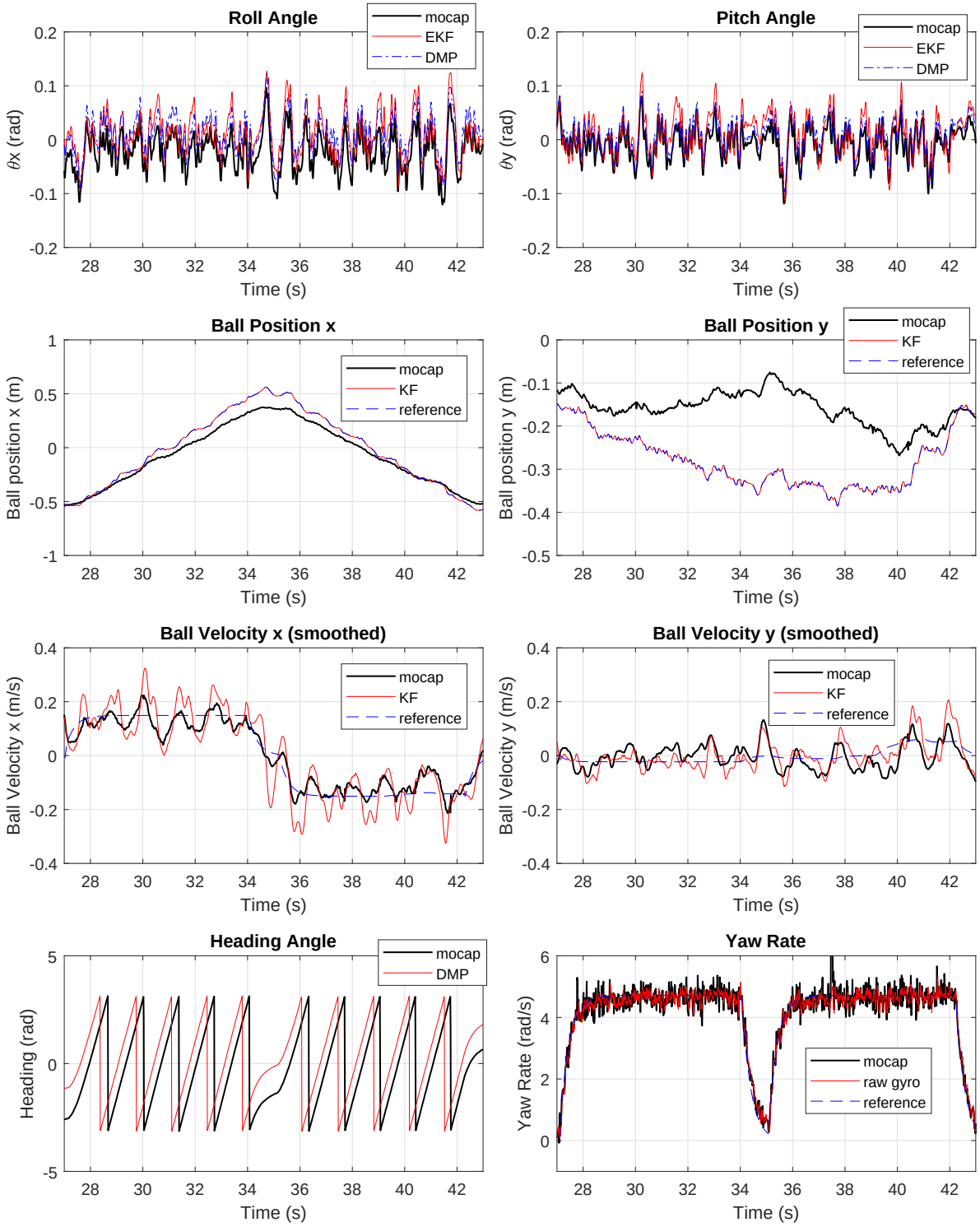


Figure 6.30: The DMP+KF estimation and control performance under 0.75 Hz yaw rate and 15 cm/s drive rate.

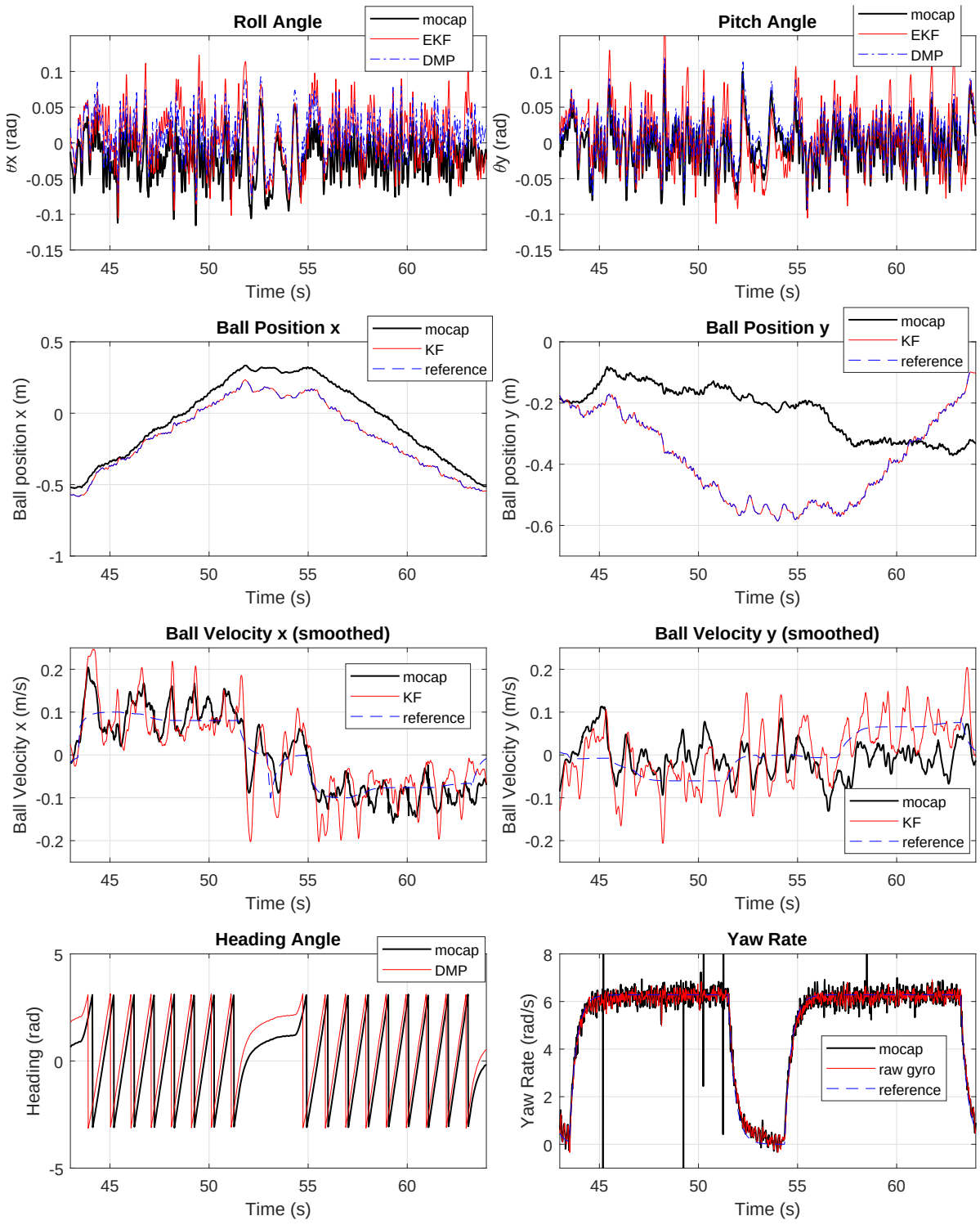


Figure 6.31: The DMP+KF estimation and control performance under 1.0 Hz yaw rate and 10 cm/s drive rate.

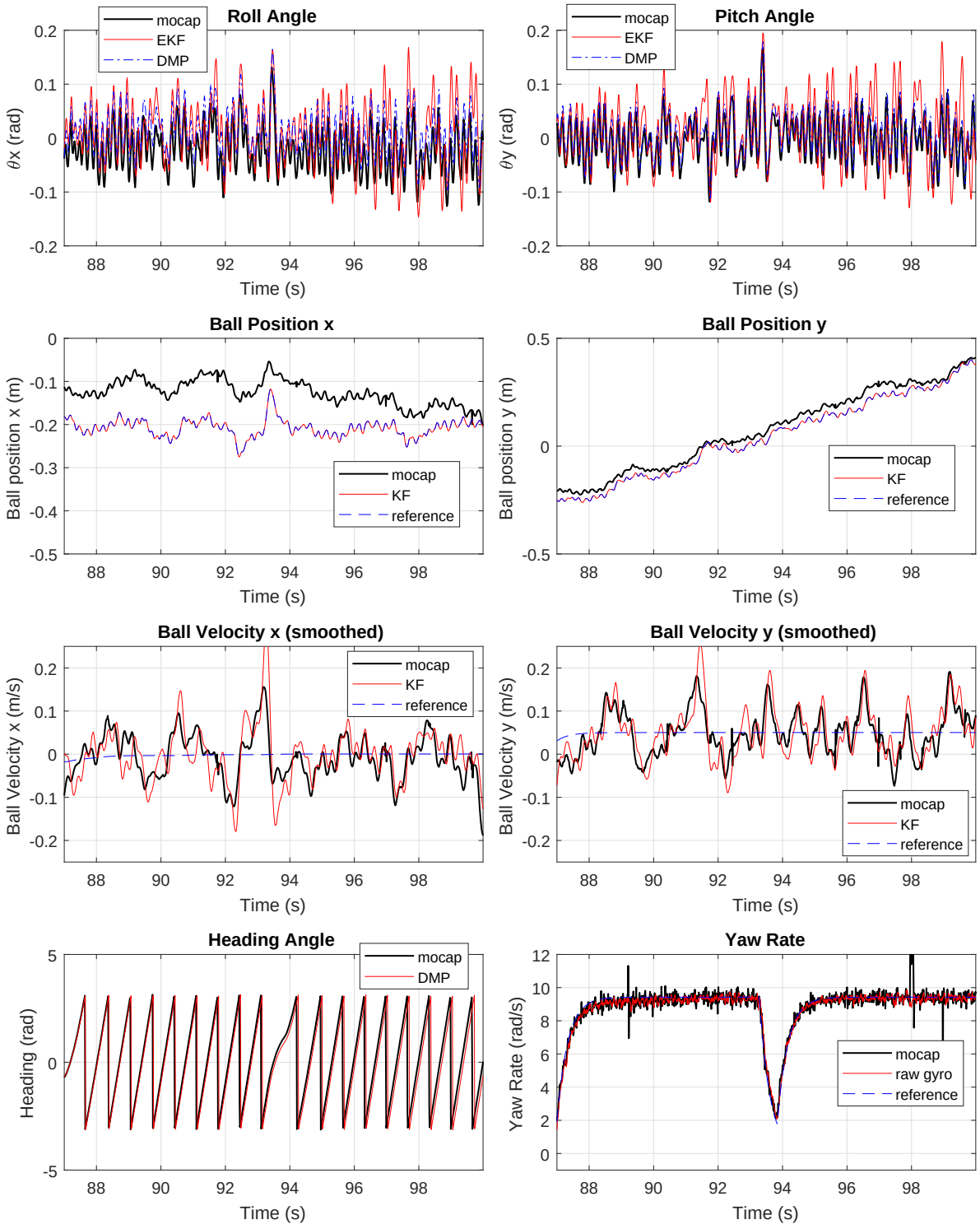


Figure 6.32: The DMP+KF estimation and control performance under 1.5 Hz yaw rate and 5 cm/s drive rate.

which achieve simultaneous spinning and driving objectives that we have set during the design phase.

It is unfortunate that the EKF is unable to outperform the DMP+KF controller, but it can achieve a very close performance to the DMP+KF. The EKF can work as a standalone estimator while the DMP+KF must use the proprietary DMP algorithm native to the InvenSense MPU-9250 or other similar devices, which might not be available to some robots. The problem in the EKF's estimation might be caused by model inaccuracies and implementation errors. Some of the unmodeled dynamics are the static friction, backlash, support balls interaction between the ball and the casing, and DROW rollers transition. None of these are simple to model, so it is very unlikely to include any of these dynamics into the model. The implementation of the EKF might be able to be improved by using a non-diagonal noise covariance matrices. The states in the high yaw-rate model are highly coupled to each other, which means that a non-diagonal process noise covariance matrix might be more appropriate for this system. However, there are 13 states in the EKF which means that there are 91 variables to tune and this can be extremely difficult to do. Most of the coupling terms are with respect to the yaw rate, so it might be possible to tune just the yaw rate components of the noise matrix.

Considering that the tracking performance is excellent with the DMP+KF combination, there is no need to tune the EKF further in the MBBR. However, there might be a better implementation of the EKF or the high yaw-rate BBR model in a different BBRs, such as when the robot is not circular symmetric (different inertia in x and y directions). There might also be other choice of model based nonlinear estimator other than EKF, such as the Unscented Kalman Filter. Considering that the DMP estimation accuracy is extremely good, it is very likely that a BBR can be balanced well by using the DMP with a good estimator for the ball inertial position and velocity. There might be an estimator only for the ball inertial position and velocity which works well and robust to the vibrations and model uncertainties in the BBR. Finding a better implementation of the high yaw-rate model or finding a good odometry algorithm for other miniature BBRs can be part of the future work.

In conclusion, the high yaw-rate model and the EKF works well, but not as well as the DMP+KF combination in the MBBR. In the case where an accurate orientation estimation such as the DMP is not available, the EKF can be used for a full state estimation and it has good estimation accuracy up to certain yaw rates. In addition, a linear controller is sufficient to achieve position tracking under high yaw rates and the major difficulty in this problem lies in the state estimations and minimizing the effect of the noise and vibration.

Acknowledgements

Clark Briggs from ATA Engineering lent us the Optitrack motion capture system which was greatly utilized in the experiments done in Chapter 3 and 6.

Chapter 5 and 6 partially uses the material as it appears in the published paper in IEEE International Conference on Robotics and Automation (ICRA) 2019, E. Sihite, D. Yang, and T. Bewley, "Modeling and state estimation of a Micro Ball-balancing Robot using a high yaw-rate dynamic model and an Extended Kalman Filter". The dissertation author was the primary investigator and author of this paper, contributed to the estimator and controller design, experimentation, and data analysis.

References

[1] T. B. Lauwers, G. A. Kantor, and R. Hollis, “A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive,” *IEEE Int’l Conf. on Robotics and Automation*, pp. 2884–2889, 2006.

[2] L. Hertig, D. Schindler, M. Bloesch, C. D. Remy, and R. Siegwart, “Unified state estimation for a ballbot,” *IEEE Int’l Conf. on Robotics and Automation (ICRA)*, pp. 2471–2476, 2013.

[3] T. Hoshino, S. Yokota, and T.Chino, “Omniride: A personal vehicle with 3 dof mobility,” *Int’l Conf. on Control, Automation, Robotics and Embedded Systems*, 2013.

[4] M. Neunert, F. Farshidian, and J. Buchli, "Adaptive real-time nonlinear model predictive motion control," *IROS 2014 Workshop on Machine Learning in Planning and Control of Robot Motion*, 2014.

[5] T. Bewley, *Numerical Renaissance: simulation, optimization, and control*. Renaissance Press, San Diego, 2015.

Chapter 7

Results, Future Work, and Conclusions

The MBBR has proven to successfully balance and achieve inertial position tracking for yaw rates up to 1.5 Hz. The high yaw-rate model of the MBBR works well when used in the EKF and simulations which indicates an accurate model. A filter which takes advantage of the non-minimum phase behavior of an inverted pendulum robot has also been successfully implemented in the linear feedback controller. The mechanical build of the latest MBBR design has proven to be very robust as it has been driven to its limits for more than a year and is still working fine. A novel orthogonal and midlatitude omniwheel placements was also utilized in this design worked very well. Additionally, the MBBR utilized a novel drive/coast model and compensation algorithm that we developed and has proven to be usable in real-time controls application. The EKF might not work as well as the DMP+KF on the MBBR, but the EKF by itself have achieved some of the high yaw-rates performance that we desired, but only up to 1 Hz yaw rates.

The future work can consist on improving the mechanical smoothness of the robot which is the most serious mechanical flaw in the current prototype. The high frequency vibrations and the friction problem in the robot must be addressed to deliver a smoother driving performance. The vibrations can be reduced by using single-row omniwheels which has shown to deliver less vibrations during the balancing. However, the fragility of the component must be addressed, either by using stronger material or thicker cross-sections. On the other hand, the high friction in

the current design is extremely hard to mitigate because the MBBR is very reliant on the support balls to prevent the ball from losing contact with the omniwheels. These support balls add more contact points on the ball which, in turn, also add more friction. We have not yet find a good design solution to remove the need of using the support balls and that might be a good starting topic when attempting to redesign the MBBR.

In order to make this robot into a commercial product, such as a toy robot which can dance/spin around gracefully, one must address the component which takes the most resource and time to produce: the omniwheels. In the current design for both SROW and DROW, each individual rollers have their own metal axle pins. Assembling the omniwheel by inserting the pins for each individual rollers takes a significant amount of time, which means high production time and cost. A mass production plan to reduce assembly time and cost must be researched in order to reduce the cost of production, which in turn, also reduce the cost of the final product.

There are several different design variation of the BBR based on the shape of the robot's body. One can design a saucer type of BBR which has a low center of gravity and more massive inertia in the yaw direction. This type of robot can "glide" on the surface, mimicking a hovercraft and possibly harder to control due to its lower center of gravity. The body can be wide enough to mount some infra-red LEDs range finder on the edge of the body, which can be used to detect the edge of a table when driven on top of it. Our MBBR is designed to be circularly symmetric, so a design which has different inertia in the x and y directions can also be considered. For each of these robots, the accuracy of the high yaw-rate model EKF can be tested to show that the same estimator and controller design methodology in MBBR is extensible to other types of BBRs.

The friction in the z components has not yet been identified, so a future work to build a system identification platform which can identify friction model in the yaw component can be investigated. The omniwheels force the ball differently between spinning clockwise and counter-clockwise, where they push the ball into and away from the omniwheels respectively. It is likely that each directions has different friction parameters and building such test platform can confirm this hypothesis.

Both the high yaw-rate model EKF and linear model KF has high noise in the ball velocity estimations. A more thorough investigation on whether this noise can be reduced by selecting a different noise covariance matrices or state augmentations can be done in the future. The noise covariance matrices used in this dissertation are all diagonal matrices. Considering that the dynamics during high yaw rates are highly coupled, non-diagonal noise covariance matrices might be more appropriate for this robot. This can be very difficult to investigate because the EKF has 13 states and 8 measurements while the KF has 8 states and 6 measurements. Either estimators have large states and figuring out the correct values for the covariance matrices will be a challenging problem. We can also investigate the implementation of the high yaw-rate model in a different nonlinear estimator, such as the unscented Kalman filter, and evaluate if there is an improvement in the estimation accuracy.

The current linear feedback controller with SLC outer loop θ reference works well as long as the robot has relatively accurate θ and ball velocity estimations. However, we can still aim to improve the robustness of the controller to make the robot more stable even under less accurate state estimations. The high yaw-rate model can be used to derive a nonlinear control algorithm, which will be an extremely challenging problem. However, the current controller works well enough that it might not be necessary to design a nonlinear controller.

Drive/brake motor drivers can also be used instead of drive/coast drivers which can eliminate an additional complexity in the controller design. Some drive/brake drivers must be driven with 2 PWM signals per motor, which might increase the product's overhead if they need to upgrade the microcontroller used. The energy efficiency of the drive/coast motor drivers has not yet been investigated. There might be some energy savings due to the regenerative current flowing back into the battery during the off cycle of the PWM signal. A more thorough research will be necessary to back this claim.

In conclusion, a small ball-balancing robot using a novel orthogonal mid-latitude omni-wheel placement, novel drive/coast compensation algorithm, and high yaw-rate dynamic model Extended Kalman Filter has successfully achieved position tracking under very high yaw-rates.

The target performance of inertial position tracking under very high yaw-rates, up to 1.5 Hz, has been fulfilled and we have successfully build a robot that can show the fluid and graceful movements that we desired in a ball-balancing robot.

Appendix A

Complete Symbolic Dynamic Models

This appendix contains the symbolic derivations for the MIP and MBBR high yaw-rate model in Chapter 3 and 5 respectively. These dynamic models were derived symbolically in Wolfram Mathematica 11.0.

A.1 MIP Nonlinear Model

This section contains the full nonlinear dynamic model for the high yaw-rate MIP which are derived in the Chapter 3. The full nonlinear MIP high yaw-rate model, which can be used in a simulation, is shown below:

$$M(\boldsymbol{q})\,\ddot{\boldsymbol{q}} = \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$$

$$\boldsymbol{q}(t) = \begin{bmatrix} \theta(t) \\ \phi(t) \\ \psi(t) \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u_x \\ u_z \end{bmatrix}, \quad M(\boldsymbol{q}) = \begin{bmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}, \quad \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}. \tag{A.1}$$

The equation above can be used to solve the system's acceleration $\ddot{\boldsymbol{q}} = M(\boldsymbol{q})^{-1}\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$, which must be calculated during the simulation. The elements of $M(\boldsymbol{q})$ and $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$ are shown

below:

$$
\begin{aligned}
m_{11} &= I_{b1} + 2I_{w1} + 2\,m_w\,r^2 + m_b\,(l^2 + r^2 + 2\,l\,r\,\cos(\theta)) \\
m_{12} &= m_{21} = 2I_{w1} + (m_b + 2m_w)r^2 + m_b\,l\,r\cos(\theta) \\
m_{22} &= 2I_{w1} + (m_b + 2m_w)r^2 \\
m_{33} &= (r/d)(I_{t2} + I_{t3} + 2I_{w2} + 2I_{w3} + l^2 m_b) + (d/r)(I_{w1} + m_w r^2) \\
&\quad - (r/d)(I_{t2} - I_{t3} + 2I_{w2} - 2I_{w3} + l^2 m_b)r^2\cos(2\theta)
\end{aligned}
\tag{A.2}
$$

$$
\begin{aligned}
b_1 &= \sin(\theta)((I_{t2} - I_{t3} + 2I_{w2} - 2I_{w3} + l^2 m_b)\cos(\theta)\dot{\psi}^2 + l\,m_b\,(g + r\,\dot{\theta}^2)) \\
b_2 &= 2\,(k_1\,u_x - 2\,k_2\dot{\phi}) + l\,m_b\,r\,\sin(\theta)\dot{\theta}^2 \\
b_3 &= 2\,k_1\,u_z - k_2\,(d/r)\,\dot{\psi} - (2/d)(I_{t2} - I_{t3} + 2I_{w2} - 2I_{w3} + l^2 m_b)\,r\,\sin(2\theta)\,\dot{\psi}\,\dot{\theta}
\end{aligned}
\tag{A.3}
$$

The sensor measurements can be simulated by using the sensor dynamics equations for the encoder, gyrometer and accelerometer measurements. The encoder and gyrometer measurements are linear while the accelerometer measurements have some nonlinear dynamics. After using the change of variables in (3.18), the encoders can directly measure the averaged wheel rotation φ and the heading angle ψ. The gyrometer simply measures the attitude angle rate of change ($\dot{\theta}$). Finally, the accelerometer measurement can be derived from the linear body acceleration:

$$
\ddot{\boldsymbol{p}}_b = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{\psi}((r + 2l\cos(\theta))\dot{\theta} + r\dot{\phi}) + l\sin(\theta)\,\ddot{\psi} \\ l\cos(\theta)sin(\theta)\dot{\psi}^2 - (l + r\cos(\theta))\ddot{\theta} - r\cos(\theta)\ddot{\phi} \\ -l\sin(\theta)^2\,\dot{\psi}^2 - l\dot{\theta}^2 + r\sin(\theta)(\ddot{\theta} + \ddot{\phi}) \end{bmatrix}, \tag{A.4}
$$

where the acceleration terms $\ddot{\theta}$, $\ddot{\phi}$, and $\ddot{\psi}$ are derived from (A.1).

A.2 MBBR Nonlinear Model

This section contains the nonlinear dynamic model for the high yaw-rate MBBR which hich are derived in the Chapter 5. The full nonlinear MBBR high yaw-rate model, which can be used in a simulation, is shown below:

$$M(\boldsymbol{q})\,\ddot{\boldsymbol{q}} = \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$$

$$\boldsymbol{q}(t) = \begin{bmatrix} \theta_x(t) \\ \theta_y(t) \\ \theta_z(t) \\ \phi_x^*(t) \\ \phi_y^*(t) \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}, \quad M(\boldsymbol{q}) \in \mathcal{R}^{5\times 5}, \quad \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u}) = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}, \tag{A.5}$$

where the matrix $M(\boldsymbol{q})$ is a 5 by 5 matrix with elements m_{ij} in row i and column j. The elements of the matrix $M(\boldsymbol{q})$ and $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$ are shown below:

$$\begin{aligned}
m_{11} &= I_{tx} + I_w + l^2\, m_t \\
m_{12} &= 0 \\
m_{13} &= -(I_{tx} + I_w + l^2\, m_t)\, \sin(\theta_y) \\
m_{14} &= l\, m_t\, r\, \cos(\theta_x) - I_w\, \cos(\theta_y) \\
m_{15} &= -l\, m_t\, r\, \sin(\theta_x)\, \sin(\theta_y) \\
m_{21} &= 0 \\
m_{22} &= 1/2\, (I_{ty} + I_{tz} + 2\, I_w + l^2\, m_t + (I_{ty} - I_{tz} + l^2\, m_t)\, \cos(2\theta_x)) \\
m_{23} &= (I_{ty} - I_{tz} + l^2\, m_t)\, \cos(\theta_x)\, \cos(\theta_y)\, \sin(\theta_x) \\
m_{24} &= 0 \\
m_{25} &= -I_w + l\, m_t\, r\, \cos(\theta_x)\, \cos(\theta_y)
\end{aligned} \tag{A.6}$$

$$
\begin{aligned}
m_{31} =& -\left(I_{tx} + I_w + l^2\, m_t\right)\, \sin(\theta_y) \\
m_{32} =& (I_{ty} - I_{tz} + l^2\, m_t)\, \cos(\theta_x)\, \cos(\theta_y)\, \sin(\theta_x) \\
m_{33} =& 1/2\,(2\, I_w + 2\, I_{tz}\, \cos(\theta_x)^2\, \cos(\theta_y)^2 + 2\, I_{ty}\, \cos(\theta_y)^2\, \sin(\theta_x)^2 + 2\, I_{tx}\, \sin(\theta_y)^2 \\
& - 1/4\, l^2\, m_t\,(-6 + 2\, \cos(2\theta_x) + \cos(2\,(\theta_x - \theta_y)) + 2\, \cos(2\theta_y) + \cos(2\,(\theta_x + \theta_y)))) \\
m_{34} =& -\, l\, m_t\, r\, \cos(\theta_x)\, \sin(\theta_y) \\
m_{35} =& l\, m_t\, r\, \sin(\theta_x) \\
m_{41} =& l\, m_t\, r\, \cos(\theta_x) - I_w\, \cos(\theta_y) \\
m_{42} =& 0 \\
m_{43} =& -\, l\, m_t\, r\, \cos(\theta_x)\, \sin(\theta_y) \\
m_{44} =& I_b + I_w + (m_b + m_t)\, r^2 \\
m_{45} =& 0 \\
m_{51} =& -\, l\, m_t\, r\, \sin(\theta_x)\, \sin(\theta_y) \\
m_{52} =& -\, I_w + l\, m_t\, r\, \cos(\theta_x)\, \cos(\theta_y) \\
m_{53} =& l\, m_t\, r\, \sin(\theta_x) \\
m_{54} =& 0 \\
m_{55} =& I_b + I_w + (m_b + m_t)\, r^2
\end{aligned}
\tag{A.7}
$$

$$
\begin{aligned}
b_1 =& 1/2\,(-2\,k_1\,u_x + 2\,g\,l\,m_t\,\cos(\theta_y)\,\sin(\theta_x) - 2\,k_2\,\dot{\theta}_x - (I_{ty} - I_{tz} + l^2\,m_t)\,\sin(2\theta_x)\,\dot{\theta}_y^2 \\
&+ 2\,k_2\,\sin(\theta_y)\,\dot{\theta}_z + I_{ty}\,\cos(\theta_x)\,\sin(\theta_x)\,\dot{\theta}_z^2 - I_{tz}\,\cos(\theta_x)\,\sin(\theta_x)\,\dot{\theta}_z^2 \\
&+ l^2\,m_t\,\cos(\theta_x)\,\sin(\theta_x)\,\dot{\theta}_z^2 + I_{ty}\,\cos(\theta_x)\,\cos(\theta_y)^2\,\sin(\theta_x)\,\dot{\theta}_z^2 \\
&- I_{tz}\,\cos(\theta_x)\,\cos(\theta_y)^2\,\sin(\theta_x)\,\dot{\theta}_z^2 + l^2\,m_t\,\cos(\theta_x)\,\cos(\theta_y)^2\,\sin(\theta_x)\,\dot{\theta}_z^2 \\
&- I_{ty}\,\cos(\theta_x)\,\sin(\theta_x)\,\sin(\theta_y)^2\,\dot{\theta}_z^2 + I_{tz}\,\cos(\theta_x)\,\sin(\theta_x)\,\sin(\theta_y)^2\,\dot{\theta}_z^2 \\
&- l^2\,m_t\,\cos(\theta_x)\,\sin(\theta_x)\,\sin(\theta_y)^2\,\dot{\theta}_z^2 + 2\,k_2\,\cos(\theta_y)\,\dot{\phi}_x^* + 2\,l\,m_t\,r\,\sin(\theta_x)\,\sin(\theta_y)\,\dot{\theta}_z\,\dot{\phi}_x^* \\
&+ 2\,\dot{\theta}_y\,((I_{tx} + I_w + l^2\,m_t + (I_{ty} - I_{tz} + l^2\,m_t)\,\cos(2\theta_x))\,\cos(\theta_y)\,\dot{\theta}_z - I_w\,\sin(\theta_y)\,\dot{\phi}_x^*) \\
&+ 2\,l\,m_t\,r\,\cos(\theta_x)\,\dot{\theta}_z\,\dot{\phi}_y^*)
\end{aligned}
\tag{A.8}
$$

$$
\begin{aligned}
b_2 =& 1/2\,(-2\,k_1\,u_y\,\cos(\theta_x) + 2\,k_1\,u_z\,\sin(\theta_x) + 2\,g\,l\,m_t\,\cos(\theta_x)\,\sin(\theta_y) \\
&- 2\,(k_2 - (I_{ty} - I_{tz} + l^2\,m_t)\,\sin(2\theta_x)\,\dot{\theta}_x)\,\dot{\theta}_y - I_{ty}\,\cos(\theta_y)\,\sin(\theta_y)\,\dot{\theta}_z^2 \\
&- I_{tz}\,\cos(\theta_y)\,\sin(\theta_y)\,\dot{\theta}_z^2 + l^2\,m_t\,\cos(\theta_y)\,\sin(\theta_y)\,\dot{\theta}_z^2 + I_{ty}\,\cos(\theta_x)^2\,\cos(\theta_y)\,\sin(\theta_y)\,\dot{\theta}_z^2 \\
&- I_{tz}\,\cos(\theta_x)^2\,\cos(\theta_y)\,\sin(\theta_y)\,\dot{\theta}_z^2 + l^2\,m_t\,\cos(\theta_x)^2\,\cos(\theta_y)\,\sin(\theta_y)\,\dot{\theta}_z^2 \\
&- I_{ty}\,\cos(\theta_y)\,\sin(\theta_x)^2\,\sin(\theta_y)\,\dot{\theta}_z^2 + I_{tz}\,\cos(\theta_y)\,\sin(\theta_x)^2\,\sin(\theta_y)\,\dot{\theta}_z^2 \\
&- l^2\,m_t\,\cos(\theta_y)\,\sin(\theta_x)^2\,\sin(\theta_y)\,\dot{\theta}_z^2 + I_{tx}\,\sin(2\theta_y)\,\dot{\theta}_z^2 - 2\,l\,m_t\,r\,\cos(\theta_x)\,\cos(\theta_y)\,\dot{\theta}_z\,\dot{\phi}_x^* \\
&- 2\,\dot{\theta}_x\,((I_{tx} + I_w + l^2\,m_t + (I_{ty} - I_{tz} + l^2\,m_t)\,\cos(2\theta_x))\,\cos(\theta_y)\,\dot{\theta}_z \\
&- I_w\,\sin(\theta_y)\,\dot{\phi}_x^*) + 2\,k_2\,\dot{\phi}_y^*)
\end{aligned}
\tag{A.9}
$$

$$
\begin{aligned}
b_3 =& -k_1\,u_z\,\cos(\theta_x)\,\cos(\theta_y) - k_1\,u_y\,\cos(\theta_y)\,\sin(\theta_x) + k_1\,u_x\,\sin(\theta_y) \\
&+ (I_{ty} - I_{tz} + l^2\,m_t)\,\cos(\theta_x)\,\sin(\theta_x)\,\sin(\theta_y)\,\dot{\theta}_y^2 - k_2\,\dot{\theta}_z \\
&+ \cos(\theta_y)\,\dot{\theta}_y\,(-(2\,I_{tx} - I_{ty} - I_{tz} + l^2\,m_t + (I_{ty} - I_{tz} + l^2\,m_t)\,\cos(2\theta_x))\,\sin(\theta_y)\,\dot{\theta}_z \\
&+ l\,m_t\,r\,\cos(\theta_x)\,\dot{\phi}_x^*) + \dot{\theta}_x\,(k_2\,\sin(\theta_y) + (I_{tx} + I_w + l^2\,m_t \\
&- (I_{ty} - I_{tz} + l^2\,m_t)\,\cos(2\theta_x))\,\cos(\theta_y)\,\dot{\theta}_y - (I_{ty} - I_{tz} + l^2\,m_t)\,\cos(\theta_y)^2\,\sin(2\theta_x)\,\dot{\theta}_z \\
&- l\,m_t\,r\,\sin(\theta_x)\,\sin(\theta_y)\,\dot{\phi}_x^* - l\,m_t\,r\,\cos(\theta_x)\,\dot{\phi}_y^*)
\end{aligned}
\tag{A.10}
$$

$$
\begin{aligned}
b_4 =& k_1\, u_x\, \cos(\theta_y) + k_1\, u_z\, \cos(\theta_x)\, \sin(\theta_y) + k_1\, u_y\, \sin(\theta_x)\, \sin(\theta_y) \\
&+ l\, m_t\, r\, \sin(\theta_x)\, \dot{\theta}_x^2 + l\, m_t\, r\, \cos(\theta_x)\, \cos(\theta_y)\, \dot{\theta}_y\, \dot{\theta}_z \\
&+ \dot{\theta}_x\, (k_2\, \cos(\theta_y) - I_w\, \sin(\theta_y)\, \dot{\theta}_y - l\, m_t\, r\, \sin(\theta_x)\, \sin(\theta_y)\, \dot{\theta}_z) - k_2\, \dot{\phi}_x^*
\end{aligned} \tag{A.11}
$$

$$
\begin{aligned}
b_5 =& k_1\, u_y\, \cos(\theta_x) - k_1\, u_z\, \sin(\theta_x) + l\, m_t\, r\, \cos(\theta_x)\, \sin(\theta_y)\, \dot{\theta}_x^2 + k_2\, \dot{\theta}_y - k_2\, \dot{\phi}_y^* \\
&+ l\, m_t\, r\, \cos(\theta_x)\, \sin(\theta_y)\, \dot{\theta}_y^2 + l\, m_t\, r\, \dot{\theta}_x\, (2\, \cos(\theta_y)\, \sin(\theta_x)\, \dot{\theta}_y - \cos(\theta_x)\, \dot{\theta}_z)
\end{aligned} \tag{A.12}
$$

The sensor measurements are all nonlinear. Let the $\boldsymbol{y}_{gy}^B$ and $\boldsymbol{y}_{en}^B$ be the gyrometer and encoder measurements respectively in body frame, with the following sensor dynamics:

$$
\boldsymbol{y}_{gy}^B = \begin{bmatrix} \Omega_x^B \\ \Omega_y^B \\ \Omega_z^B \end{bmatrix} = \begin{bmatrix} \dot{\theta}_x - \sin(\theta_y)\, \dot{\theta}_z \\ \cos(\theta_x)\, \dot{\theta}_y + \cos(\theta_y)\, \sin(\theta_x)\, \dot{\theta}_z \\ -\sin(\theta_x)\, \dot{\theta}_y + \cos(\theta_x)\, \cos(\theta_y)\, \dot{\theta}_z \end{bmatrix} \tag{A.13}
$$

$$
\boldsymbol{y}_{en}^B = \begin{bmatrix} \varphi_x^B \\ \varphi_y^B \\ \varphi_z^B \end{bmatrix} = \begin{bmatrix} -\dot{\theta}_x + \sin(\theta_y)\, \dot{\theta}_z + \cos(\theta_y)\, \dot{\phi}_x^* \\ -\cos(\theta_x)\, \dot{\theta}_y - \cos(\theta_y)\, \sin(\theta_x)\, \dot{\theta}_z + \sin(\theta_x)\, \sin(\theta_y)\, \dot{\phi}_x^* + \cos(\theta_x)\, \dot{\phi}_y^* \\ \sin(\theta_x)\, \dot{\theta}_y - \cos(\theta_x)\, \cos(\theta_y)\, \dot{\theta}_z + \cos(\theta_x)\, \sin(\theta_y)\, \dot{\phi}_x^* - \sin(\theta_x)\, \dot{\phi}_y^* \end{bmatrix} \tag{A.14}
$$

Let $\boldsymbol{y}_{ac}^B$ be the accelerometer measurement which has the following sensor dynamics:

$$
\boldsymbol{y}_{ac}^B = \begin{bmatrix} y_{ac1}^B & y_{ac2}^B & y_{ac3}^B \end{bmatrix}^T \tag{A.15}
$$

$$
\begin{aligned}
y_{ac1}^B =& 1/2\, (2\, g\, \sin(\theta_y) + 2\, l_{ax}\, \dot{\theta}_y^2 + l_{ax}\, \dot{\theta}_z^2 + l_{ax}\, \cos(\theta_y)^2\, \dot{\theta}_z^2 - l_{ax}\, \sin(\theta_y)^2\, \dot{\theta}_z^2 \\
&+ l_{az}\, \cos(\theta_x)\, \sin(2\theta_y)\, \dot{\theta}_z^2 + l_{ay}\, \sin(\theta_x)\, \sin(2\theta_y)\, \dot{\theta}_z^2 - 4\, \dot{\theta}_x\, ((l_{ay}\, \cos(\theta_x) \\
&- l_{az}\, \sin(\theta_x))\, \dot{\theta}_y + \cos(\theta_y)\, (l_{az}\, \cos(\theta_x) + l_{ay}\, \sin(\theta_x))\, \dot{\theta}_z) - 2\, R\, \cos(\theta_y)\, \dot{\theta}_z\, \dot{\phi}_x^* \\
&- 2\, l_{az}\, \cos(\theta_x)\, \ddot{\theta}_y - 2\, l_{ay}\, \sin(\theta_x)\, \ddot{\theta}_y + 2\, l_{ay}\, \cos(\theta_x)\, \cos(\theta_y)\, \ddot{\theta}_z \\
&- 2\, l_{az}\, \cos(\theta_y)\, \sin(\theta_x)\, \ddot{\theta}_z - 2\, R\, \cos(\theta_y)\, \ddot{\phi}_y^*)
\end{aligned} \tag{A.16}
$$

$$
\begin{aligned}
y^B_{ac2} =& 1/4\,(-4\,g\,\cos(\theta_y)\,\sin(\theta_x) + 4\,l_{ay}\,\dot{\theta}_x^2 + 4\,\sin(\theta_x)\,(l_{az}\,\cos(\theta_x) + l_{ay}\,\sin(\theta_x))\,\dot{\theta}_y^2 \\
& - 8\,l_{ay}\,\sin(\theta_y)\,\dot{\theta}_x\,\dot{\theta}_z - 8\,\cos(\theta_x)\,(l_{az}\,\cos(\theta_x)\,\cos(\theta_y) + l_{ay}\,\cos(\theta_y)\,\sin(\theta_x) \\
& - l_{ax}\,\sin(\theta_y))\,\dot{\theta}_y\,\dot{\theta}_z + 3\,l_{ay}\,\dot{\theta}_z^2 + l_{ay}\,\cos(\theta_x)^2\,\dot{\theta}_z^2 - l_{ay}\,\cos(\theta_y)^2\,\dot{\theta}_z^2 \\
& + l_{ay}\,\cos(\theta_x)^2\,\cos(\theta_y)^2\,\dot{\theta}_z^2 - l_{ay}\,\sin(\theta_x)^2\,\dot{\theta}_z^2 - l_{ay}\,\cos(\theta_y)^2\,\sin(\theta_x)^2\,\dot{\theta}_z^2 \\
& - l_{az}\,\sin(2\theta_x)\,\dot{\theta}_z^2 - l_{az}\,\cos(\theta_y)^2\,\sin(2\theta_x)\,\dot{\theta}_z^2 + l_{ay}\,\sin(\theta_y)^2\,\dot{\theta}_z^2 \\
& - l_{ay}\,\cos(\theta_x)^2\,\sin(\theta_y)^2\,\dot{\theta}_z^2 + l_{ay}\,\sin(\theta_x)^2\,\sin(\theta_y)^2\,\dot{\theta}_z^2 + l_{az}\,\sin(2\theta_x)\,\sin(\theta_y)^2\,\dot{\theta}_z^2 \\
& + 2\,l_{ax}\,\sin(\theta_x)\,\sin(2\theta_y)\,\dot{\theta}_z^2 - 4\,R\,\sin(\theta_x)\,\sin(\theta_y)\,\dot{\theta}_z\,\dot{\phi}_x^* - 4\,R\,\cos(\theta_x)\,\dot{\theta}_z\,\dot{\phi}_y^* + 4\,l_{az}\,\ddot{\theta}_x \\
& + 4\,l_{ax}\,\sin(\theta_x)\,\ddot{\theta}_y - 4\,l_{ax}\,\cos(\theta_x)\,\cos(\theta_y)\,\ddot{\theta}_z - 4\,l_{az}\,\sin(\theta_y)\,\ddot{\theta}_z + 4\,R\,\cos(\theta_x)\,\ddot{\phi}_x^* \\
& - 4\,R\,\sin(\theta_x)\,\sin(\theta_y)\,\ddot{\phi}_y^*)
\end{aligned}
\tag{A.17}
$$

$$
\begin{aligned}
y^B_{ac3} =& 1/4\,(-4\,g\,\cos(\theta_x)\,\cos(\theta_y) + 4\,l_{az}\,\dot{\theta}_x^2 + 4\,\cos(\theta_x)\,(l_{az}\,\cos(\theta_x) + l_{ay}\,\sin(\theta_x))\,\dot{\theta}_y^2 \\
& - 8\,l_{az}\,\sin(\theta_y)\,\dot{\theta}_x\,\dot{\theta}_z + 8\,\sin(\theta_x)\,(\cos(\theta_y)\,(l_{az}\,\cos(\theta_x) + l_{ay}\,\sin(\theta_x)) \\
& - l_{ax}\,\sin(\theta_y))\,\dot{\theta}_y\,\dot{\theta}_z + 3\,l_{az}\,\dot{\theta}_z^2 - l_{az}\,\cos(\theta_x)^2\,\dot{\theta}_z^2 - l_{az}\,\cos(\theta_y)^2\,\dot{\theta}_z^2 \\
& - l_{az}\,\cos(\theta_x)^2\,\cos(\theta_y)^2\,\dot{\theta}_z^2 + l_{az}\,\sin(\theta_x)^2\,\dot{\theta}_z^2 + l_{az}\,\cos(\theta_y)^2\,\sin(\theta_x)^2\,\dot{\theta}_z^2 \\
& - l_{ay}\,\sin(2\theta_x)\,\dot{\theta}_z^2 - l_{ay}\,\cos(\theta_y)^2\,\sin(2\theta_x)\,\dot{\theta}_z^2 + l_{az}\,\sin(\theta_y)^2\,\dot{\theta}_z^2 \\
& + l_{az}\,\cos(\theta_x)^2\,\sin(\theta_y)^2\,\dot{\theta}_z^2 - l_{az}\,\sin(\theta_x)^2\,\sin(\theta_y)^2\,\dot{\theta}_z^2 + l_{ay}\,\sin(2\theta_x)\,\sin(\theta_y)^2\,\dot{\theta}_z^2 \\
& + 2\,l_{ax}\,\cos(\theta_x)\,\sin(2\theta_y)\,\dot{\theta}_z^2 - 4\,R\,\cos(\theta_x)\,\sin(\theta_y)\,\dot{\theta}_z\,\dot{\phi}_x^* + 4\,R\,\sin(\theta_x)\,\dot{\theta}_z\,\dot{\phi}_y^* - 4\,l_{ay}\,\ddot{\theta}_x \\
& + 4\,l_{ax}\,\cos(\theta_x)\,\ddot{\theta}_y + 4\,l_{ax}\,\cos(\theta_y)\,\sin(\theta_x)\,\ddot{\theta}_z + 4\,l_{ay}\,\sin(\theta_y)\,\ddot{\theta}_z - 4\,R\,\sin(\theta_x)\,\ddot{\phi}_x^* \\
& - 4\,R\,\cos(\theta_x)\,\sin(\theta_y)\,\ddot{\phi}_y^*).
\end{aligned}
\tag{A.18}
$$

The accelerations terms $\ddot{\boldsymbol{q}}$ can be solved from (A.5).