Towards KriCatch, A Slip Catching Practice System for the game of Cricket

MSc Report

Committe :

Prof.dr.ir. G.J.M. Krijnen
Dr.ir. D. Dresscher
Dr. A.H. Mader
Dr. S. Vanapalli

September 2018

038RAM2018
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Towards

A Slip Catching Practice System for the game of Cricket

By

Ajinkya Arun Bhole

in partial fulfillment of the requirements for the degree of

Master of Science in **Systems and Control**
Faculty of Electircal Engineering, Mathematics and Computer Science
University of Twente, The Netherlands

Assessment Committee:

prof.dr.ir. Stefano Stramigioli
prof.dr.ir. Gijs Krijnen
dr.ir. Douwe Dresscher
dr. Angelika Mader
dr. Srinivas Vanapalli

September 6, 2018

Abstract

This work presents KriCatch 1.0, output of the first design iteration for KriCatch, a slip catching practice system for the game of Cricket.

Present slip catching practice systems for the game of Cricket comprise of a thrower throwing a ball towards a coach or a passive equipment, which then swerve the ball for a user's catching practice. These methods cannot provide a catching practice with good efficacy, versatility and a degree of realism, pointing towards the need for one. In this respect, KriCatch is being designed.

KriCatch 1.0 comprises of a thrower who throws a ball towards a robotic manipulator which can swerve the ball to a desired point. To achieve this, four subsystems are designed.

The first subsystem senses the position of the thrown ball, estimates and predicts the trajectory of the ball and finally provides an interception point for the robotic manipulator.

The second subsystem decides the spot where the thrown ball should be sent to.

The third subsystem decides the strategy for how the ball needs to be sent to, to the spot decided by the second subsystem, by the robotic manipulator.

The fourth subsystem finally plans and executes the trajectory for the robotic manipulator, to intercept the ball.

To evaluate the designed system, a prototype of the designed system is setup and experimented upon. It is observed that the ball is successfully intercepted by the robotic manipulator when the interception point lies in the manipulators achievable workspace. However, the ball cannot be sent accurately to the desired spots. This is mainly because the robotic manipulator is not able to perfectly track the commanded trajectory to intercept the ball, on account of its joint velocity, acceleration and jerk limits.

To Vasant Shankar Bhole

Dear Ajoba,

I can't thank you enough for introducing me to the wonderful game of Cricket,
For showing me and Aaji those interesting Cricket matches,
And, for mostly saying yes to play Cricket with me. It was always fun for me to make you run around in the front yard to collect the ball.
Someday soon, KriCatch will be realized. I hope this will bring some more Rahul Dravids to the game!

Tumcha bhaari naatu,
Ajju

Acknowledgements

This report marks my final step as a Systems and Controls student at the University of Twente and the first step in my journey towards the development of KriCatch. It cannot express the long days spent in the lab, the tiredness debugging the codes, the joy for each successfully built and working code, the hope for good results and the sadness with each failed attempt.

The environment created at the Robotics and Mechatronics Lab at the University of Twente is probably the best a student interested in robotics could ever hope for.

I would like to take this opportunity to thank Stefano Stramigioli, head of the RAM lab, for believing in me, my idea and letting me work on it with the RAM group. You are one of the most energetic personalities I have ever met. I can never understand where you get all your energy from (maybe you have some magical energy tank hidden somewhere).

Thanks Douwe Dresscher. Inspite of your busy schedule, you agreed to mentor me on this project, which does not come in your research domain. When asked for an advise or presented with a problem, you can process it, think critically and suggest great ideas at a brisk. You gave me the total control of the steering wheel for my thesis, which was a great learning experience for me.

Marcel Schwirtz, thanks a tonne for taking my constant bugging, for always having five minutes for me and being my 'instant problem solver'. Someday I will beat your cycling records. Alfred de Vries, thanks for all your efforts in finding time-slots for me to work in the Smart-XP Lab. Someday, I will build a robot which will perform all your crazy ideas.

Sahib Dhanjal (aka Chopa), it is because of you, that I never had to use `roswtf`. Ishan Kanade, thanks for being my Ubuntu repair guy. Keep solving my computer related problems, you guys.

Kaul (aka Judgeet Singh), I cannot thank you enough for grooming my English writing skills during the four wonderful years we spent in Pilani.

Thesis work can become really stressful sometimes. Florentine, I had a great time sharing and listening to thesis miseries over lunch in Waaier. Also, thanks for bringing in adventures in my dull mainstream life during thesis. Peshwe Amit, bhau, thanks for all your endless hilarious yapping in our Haveli. If KriCatch becomes a success someday, all business plans of yours are on tap.

Maushi, Aaji, I would have gotten nowhere without your blessings. Rashu, thanks for your regular 'cool.. good luck dude' wishes.

Mummi, Pappa and Appu, I can dream big and follow my heart only because of your unconditional support. It is so easy for me to find my motivation to work hard and give in my best at anything and everything that comes my way, just by looking at you three.

Picture abhi baaki hai,

Ajinkya Bhole

Preface

This tour de force in experimental robotics marks the embarkation towards the journey of design and development of '*KriCatch*', a slip catching practice system for the game of Cricket.

It is identified that the current catching practice equipment for Cricket, available in the market, cannot provide a catching practice with good efficacy, versatility and a degree of realism, pointing towards the need for one.

Iterative design is a process of designing a product in which the product is tested and evaluated repeatedly at different stages of design to eliminate usability flaws before the product is designed and launched. In other words, iterative design is a process of improving and polishing the design over time.

The iterative design methodology is used to develop KriCatch and this work describes the first iteration of this design process leading to KriCatch 1.0.

Contents

1 Introduction

1.1 Context

"..Sports is actually a chance for us to have other human beings push us to excel."

-John Keating, (Robin William's character in Dead Poet's Society)

And to keep up the gradient of this spirit, the expertise of a sports player and thus a sport overall must keep advancing. Application of technology to sports equipment has a great impact on the performance of players and has a potential to revolutionize the entire sporting culture through variegated approaches, such as improvement in training methods, development of new equipment, improvement of existing equipment, strategy planning and tactical analysis.

Practice makes perfect, and practicing using training equipment is a tremendously effective method to achieve desired skill sets in a sport. Now-a-days many are moving to automated training equipment, which offer increased efficiency of practice and great versatility (Sato et al., 2017; Miyazaki et al., 2006; Li et al., 2012; ProBatter Sports, 2011). Moreover, in many sports, automation eliminates the needs of any accompanying personage to provide practice.

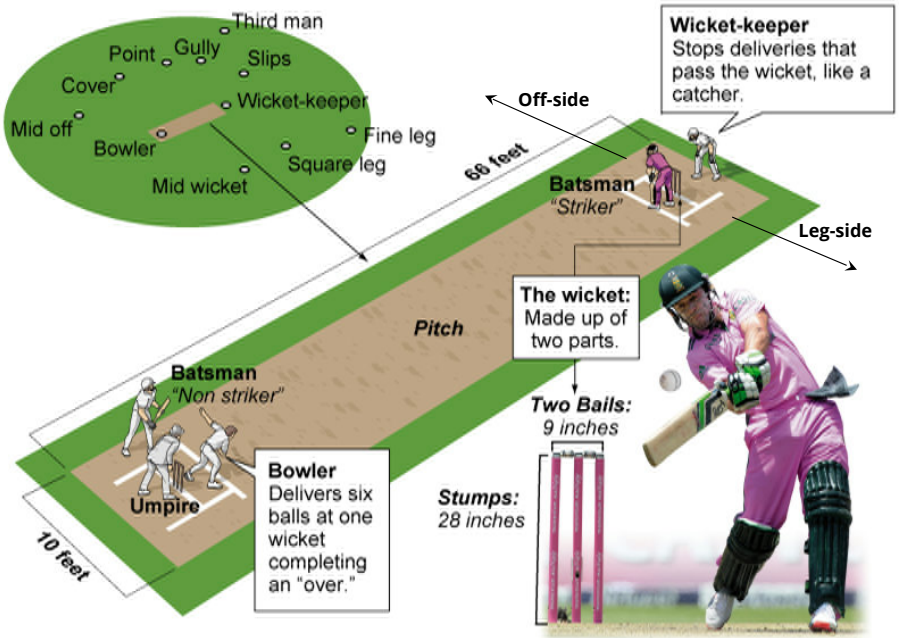

Figure 1.1: The basics of the game of Cricket. Image Source: http://www.chicagotribune.com/

1.2 Identifying the Problem

Cricket is a widely popular bat-and-ball sport (Wikipedia, 2018b). 'Catches win matches' is probably the oldest adage in the game of Cricket. Catching in Cricket requires great concentration, agility, hand-eye coordination and sure-handedness. Knowledge of good catching techniques alone does not suffice, ample practice is required to master the art of catching and building up confidence.

Before discerning the problem, some basic facts about Cricket and the terminologies used in the game are introduced ahead. Cricket is played with two teams of 11 players each. Each

team takes turns batting and playing the field, as is done in the game of Baseball. In Cricket, the batter is called the batsman and the pitcher is called the bowler. The bowler tries to knock down the bail of the wicket (refer Figure 1.1). A batsman tries to prevent the bowler from hitting the wicket, by hitting the ball. Two batsmen are on the pitch at the same time. The batsmen can run, after the ball is hit. A run is scored each time the two batsmen change their places on the pitch.

The Cricket field can be divided in two ways:

1. When the batsman is facing the bowler, the side to the right hand side of the batsman is called the *off-side* and the one on the left hand side is called the *leg-side* (Figure 1.1).

2. As is shown in Figure 1.2 the ground can be divided into four sections, *outfield*, *infield*, *close-infield* and the *pitch*.

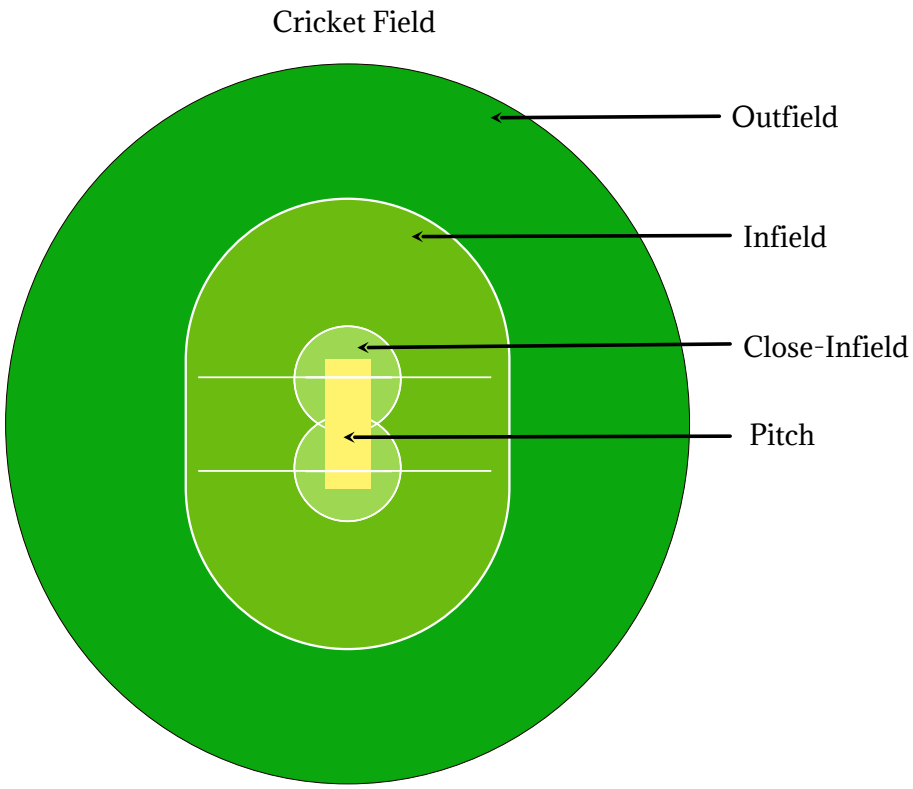

Figure 1.2: A standard Cricket ground showing the Pitch, Close-Infield, Infield and the Outfield.

Based on the second way of division, the catches taken in Cricket are differentiated as outfield, infield and close-infield catches.

A slip fielder (collectively, a slip cordon or the slips) is placed in the *close-infield*, behind the batsman on the *off side* field (Figure 1.3). The catches coming in this region are called slip catches. These catches are the most difficult to make, as the players have a very less reflex time and need to be extremely attentive. Naturally, these are the most practiced catches in game.

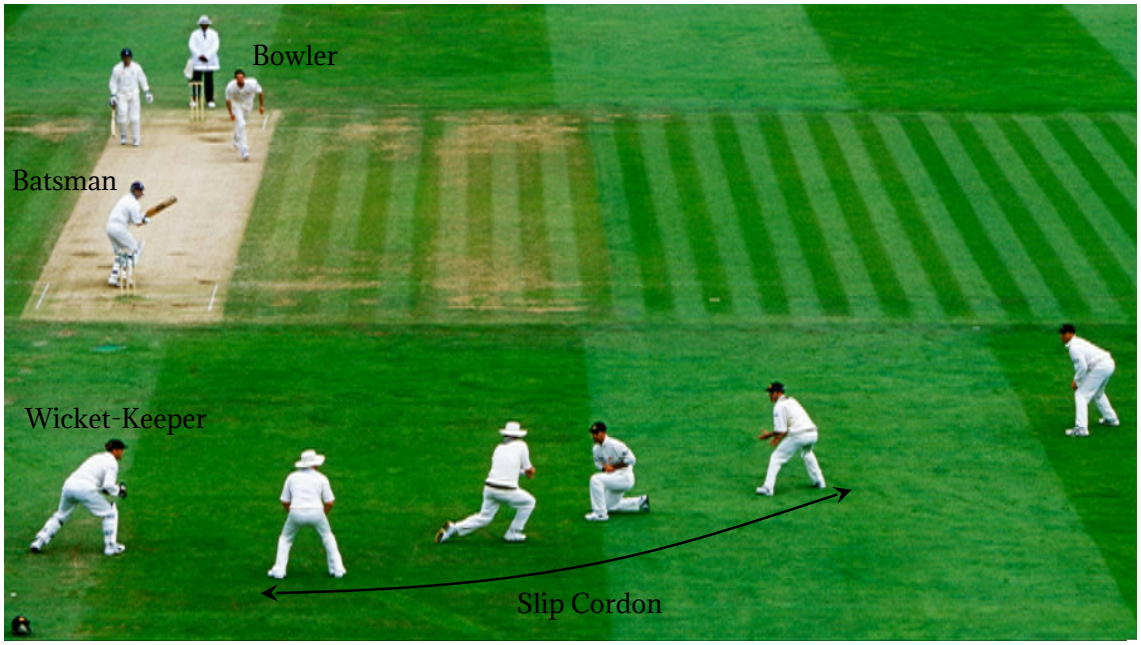

Figure 1.3: A Slip Catch scenario in the game of Cricket

There are numerous ways and equipment to practice slip catching which go as follows:

1. A traditional way of practicing slip catches is by shooting a ball on a pitch roller. The ball hits the curved surface of the roller and gets swerved towards the fielders. (Figure 1.4 (a))

2. The slip catching cradle (Figure 1.4 (b)) is similar to using a pitch roller and consists of long, thin ash lathes over a bowed metal frame.

3. Katchet (Figure 1.4 (c)) is a portable equipment with an uneven surface which can provide unpredictable catches.

4. Using Reflex ball (Figure 1.4 (d)) is another technique to practice for slip catches. This ball is made up of uneven surface and bounces in unpredictable direction when it hits the ground.

5. Finally, the most realistic way to practice slip catching requires a well-practiced coach to make it worthwhile. As shown in Figure 1.4 (e), the feeder (F) throws the ball such that it reaches the coach (C) at chest height, wide to the off side and the coach deflects the ball with a bat into the slip cordon (S) for practicing catches.

Figure 1.4: Slip catching ways and equipment used for in Cricket.

A good catching practice equipment is the one which can provide catches in a specific desired direction and location (for providing practice on a player's catching weak spots) as well as in directions which are unforeseeable by the player (to provide training for unorthodox shots and feinting movements done by the batsmen), Moreover, the practicing method must hold a degree of realism to simulate an actual game scenario.

Using the pitch roller and the slip catching cradle cannot provide unpredictable catches as the fielder is able to predict beforehand where the ball will deflect according to the surface of the equipment. While on the other hand, Katchet and Reflex ball can fulfill the criterion of being unpredictable but cannot provide catches in a desired spot. The above-mentioned methods and equipment also do not provide the catching practice with a degree of realism. The method of having a well-practiced coach holds a good degree of realism but makes the practice coach dependent. Moreover, in case the catches are required in specific spots, a coach usually cannot provide good repeatability in providing catches. This method is also unsafe as

there are chances of the ball hitting the coach in case he/she accidentally misses it or does not duck at the appropriate moment in case required.

1.3 Defining the Problem:

Based on the identified problem in the previous section, following problem is evident:

There is a requirement for an equipment / system for slip catching practice which can provide unpredictable catches as well as catches in desired spots with a good degree of realism.

The main goal of this work is to design KriCatch, a slip catching practice system which solves the above-mentioned problem. An iterative design methodology (Wikipedia, 2018c) is used to develop KriCatch and this work focuses on the first iteration of this design process leading to KriCatch 1.0.

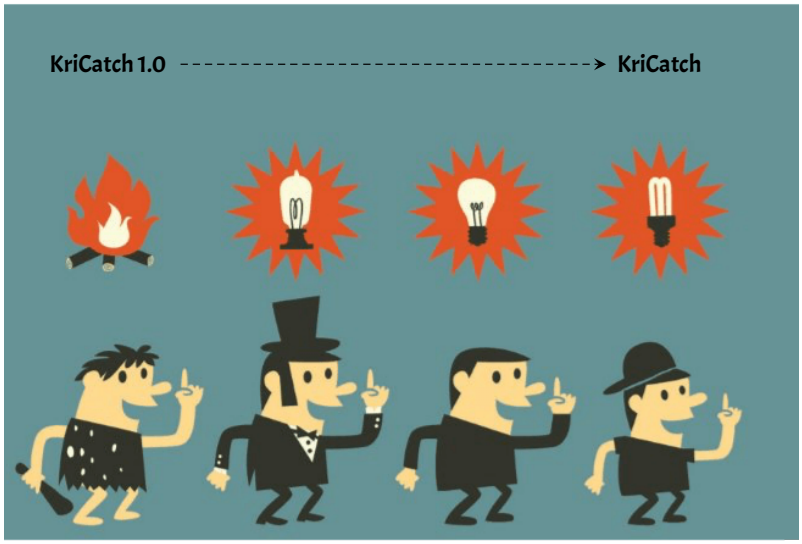

Figure 1.5: Iterative design process for the design of KriCatch.

1.4 Content

Chapter 2 discerns in detail, the functional requirements for KriCatch. Chapter 3 describes the system architecture of KriCatch 1.0. The experimental results performed on KriCatch 1.0 are presented in Chapter 4. Finally, conclusions and recommendations are described in Chapter 5.

2 Analysis

Before starting the prototyping phase of KriCatch 1.0, it is important to understand its functional requirements. This chapter describes some easily discernable functional requirements for KriCatch. Thereafter, a functional analysis is performed. Necessary subsystems are identified and their function allocation is done to meet the overall system's functional requirements.

The simplest way to provide slip catching practice is by having a machine which can shoot a ball in desired spots for the player. But, this kind of equipment does not provide a catching practice with a good degree of realism.

To understand how an apt catching practice system should look like, it is important to firstly analyze how an event of slip catch occurs, how can a slip catch be taken successfully and what are some of the challenges the players in Cricket face while using the present catching practice equipment in the market.

2.1 How does an event of slip catch occur?

The goal of a batsman in the game of Cricket is to hit the ball by finding gaps amongst the fielders so as to make runs. It is therefore never the intent of a batsman to hit the ball straight towards a player. An event of a catch usually occurs when, the ball hits a spot on the bat which was not desired by the batsman (possibly due to a swing in the ball trajectory or an error made by the batsman).

2.2 How can a slip catch be taken successfully?

Based on the discussion in the previous section on how an event of slip catch occurs in the game of Cricket, the coaches always advice the fielders *to keep an eye on the swing and the face of the bat.* The fielders are also advised *to observe and predict the trajectory and swing of the thrown ball.* This is because if the ball is swinging, there are high chances that the batsman makes an error and the ball hits an undesired spot on the bat.

2.3 Feinting Motions by the Batsmen

Now-a-days batsmen play many unorthodox shots and also *feint the motion of their body and bat to trick the fielders.* As can be seen from the Figure 2.1, that the batsman initially feinted his motion by moving to his leg-side and the wicket-keeper followed his motion. The batsman then swerved the ball in the direction opposite to that of the wicket-keeper's direction of travel. It is therefore also necessary for the fielders to get used to such tactics used by the batsmen.

Figure 2.1: An example of unorthodox batting method: The Late Cut Shot played by Eoin Morgan.

2.4 Functional Requirements

Based on the above discussion, it is clear that an apt catching practice system should:

- Imitate the bowler, so that the fielder needs to keep an eye on the direction and swing of the ball

- Imitate the batsman, so that he/she can observe the swing and face of the bat
- Imitate feinting motions done by the batsmen

2.5 Functional Analysis

Two of the functional requirements identified in the previous section are imitating the bowler and the batsman.

2.5.1 Imitation of Bowler:

Advanced and successful ball shooting machines like TrueMan (BOLA, 2015) (Figure 2.2) which can shoot a ball at desired speed, spin and spots, and can also provide bowler animations, have already been developed. To imitate a bowler, either such kind of a machine can be used or one can simply have a player throw the ball for the practice.

Figure 2.2: The TrueMan bowling machine by BOLA

2.5.2 Imitation of Batsman:

In case, it is desired not to have a coach dependent practice, it is necessary to have a machine which can sufficiently imitate a batsman. It is not a good idea to have a player to act as a batsman to provide catches because there is a possibility that the player might assimilate the actions of giving catches and might accidentally do the same during a real match scenario.

The batsman should therefore be imitated by an actively controllable machine, which can swerve the ball to desired spots. This machine is referred to as the Batsman Imitating Machine (BIM) in the rest of this work.

To imitate only the batsman's arm, the BIM can be a robotic manipulator, with main requirements including sufficient degrees of freedom, work-space and speed to swerve the ball on desired spots and also to create feinting motions. In case it is also required to imitate the body motions of the batsman, a robotic manipulator can be set up on a moving base, making the BIM a mobile robotic manipulator. The feinting motions can also be simulated by planning them on the robotic manipulator.

2.5.3 Subsystems and their Functions:

To swerve the ball to a desired spot, the bat placed on the BIM needs to be placed in a specific location and orientation, depending on the incoming trajectory of the ball. For this, it is necessary to know the incoming trajectory of the ball.

When a ball shooting machine is used to imitate the bowler and the parameters which define the trajectory of the thrown ball are known with sufficient accuracy, it is easy to predict the trajectory and state of the ball at which the BIM must intercept the ball for swerving.

But, in case a person is throwing the ball, to imitate the bowler, it is necessary to have a sensing subsystem for estimating and predicting the trajectory of the thrown ball.

In this work, it is assumed that we have a person throwing the ball for the practice. Based on this, following subsystems and their functions are proposed:

- Subsystem 1: Senses the position of the thrown ball, estimates and predicts the trajectory of the ball and finally provides an interception point for the BIM.
- Subsystem 2: Decides where the thrown ball should be deviated to.
- Subsystem 3: Decides the strategy for the BIM, for how to send the ball to the spot decided by Subsystem 2, by providing the orientation of the normal vector to the surface of the bat used on the BIM.
- Subsystem 4: Plans and executes the trajectory of the bat used on the BIM.

Figure 2.3 shows a rough overview of the connections of the subsystems used for KriCatch 1.0.

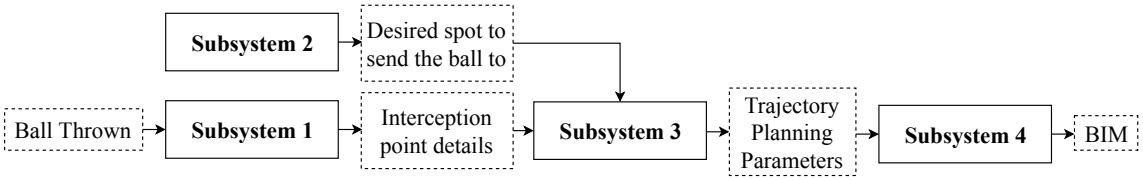

Figure 2.3: A rough idea of the connections of the subsystems used for KriCatch 1.0

3 System Architecture

This chapter firstly discusses the general workflow of KriCatch 1.0 and finally describes every subsystem involved in detail.

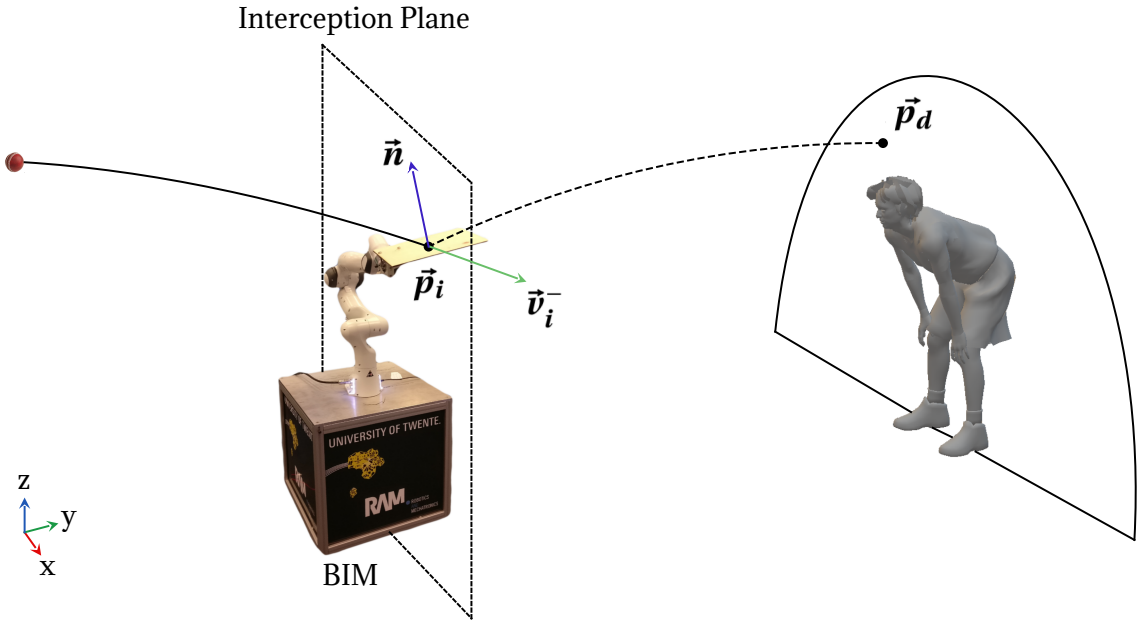

Figure 3.1: A general workflow of KriCatch 1.0

The general workflow of KriCatch 1.0 goes as follows (refer Figure 3.1):

- A ball thrower throws the ball, which is sensed by a motion-capture system. Subsystem 1 takes the position measurements of the ball from the motion-capture system and starts estimating and predicting the trajectory of the thrown ball. The interception point for the BIM is constrained in a specific fixed plane (Interception Plane) in the workspace of the BIM. The predicted trajectory of the ball and the Interception Plane can be used to obtain the interception point details for the BIM, namely, the interception time (t_i), interception point ($\vec{p}_i$) and the velocity of the ball ($\vec{v}_i^-$) at the interception point.
- Subsystem 2 decides the spot ($\vec{p}_d$) to which the thrown ball need to be sent to.
- Subsystem 3 decides the strategy for how to send the ball to the desired spot ($\vec{p}_d$) from the interception point ($\vec{p}_i$). This is done by calculating the orientation of the bat of the BIM at the interception point. The orientation of the bat is specified in the form of a vector ($\vec{n}$) normal to the surface of the bat.
- The BIM is a robotic manipulator with a bat attached to its end-effector. Subsystem 4 finally plans and executes the trajectory of the bat such that it reaches from its initial position and orientation to the interception point specified by Subsystem 1 and and an orientation specified by Subsystem 3, at the interception time.

The subsystems mentioned in the above workflow are discussed in detail the following sections. These subsystem are integrated together in a ROS environment (Quigley et al., 2009).

3.1 Subsystem1: Sensing the thrown ball

It is necessary to predict the motion of the ball in advance as this provides the system with some time to plan and execute the the trajectory for the BIM. For this it is necessary to firstly sense the motion of the ball. Based on the measurements from a sensing system, estimation and prediction of the trajectory of the thrown ball can be done. Using the predicted trajectory, the interception point for the BIM can be calculated. This subsystem outputs the interception point details namely, the interception time (t_i), interception point ($\vec{p}_i$) and the velocity of the ball ($\vec{v}_i^-$) at the interception point. The methodology used to obtain these outputs are explained in the following subsections.

3.1.1 Sensing Ball Position:

This subsection briefly discusses OptiTrack (Point, 2011), a marker-based motion capture system used to procure the position of the thrown ball.

OptiTrack has been used in several works wherein robotic manipulators are made to catch thrown objects ((Brescianini and D'Andrea, 2018; Cigliano et al., 2015; Kim et al., 2014)). The OptiTrack system used in this work provides position measurements with a position accuracy of about 0.01 mm and measurements at the rate of 250 Hz.

The data from OptiTrack can be streamed over a network using the OptiTrack Streaming Engine. The `mocap_optitrack` package (Kathrin Gräve, 2018) developed for ROS can be used to provide 6D poses of rigid bodies defined by a set of markers placed onto it. This package was modified to provide the position data of individual markers present in the OptiTrack environment. Any ball covered with retro-reflective tape can then be tracked using the modified package. Figure 3.2 shows the ball covered with a retro-reflective tape used in this work.

Figure 3.2: Ball covered with retro-reflective tape used in OptiTrack environment

It is necessary to predict the motion of the ball in advance as this provides the system with some time to plan and execute the the trajectory for the BIM. For this it is necessary to estimate and predict the trajectory of the thrown ball and calculate the interception point for the BIM. The methodology used to perform these tasks are explained in following subsections.

3.1.2 Estimation of the state of ball:

To predict the trajectory of a thrown ball, it is required to estimate its current state of motion.

The Bayesian state estimation approach is widely used for state estimation, as it provides a systematic and general approach to handle the effect of various random uncertainties on the process states and measurements. Bayesian state algorithms use a first-principles based dynamic model and the statistical properties of random disturbances and measurements to obtain the posterior distribution of the state estimates.

The Kalman Filter (KF) introduced by Rudolf E. Kalman (Kalman, 1960) is a widely used Bayesian estimation technique. This technique works best in-case the system under consideration is linear. In case of a non-linear system, which is generally the case, there is another approach with almost the same computational complexity, but with better performance. That approach is the Extended Kalman filter (EKF) (Anderson and Moore, 1979). In EKF the state distribution is approximated by a Gaussian Random Variable (GRV), which is then propogated through the first order linearization of the non-linear system.

EKF has been successfully implemented in many previous works involving prediction of ball trajectories under the effect of gravity and drag (Koç et al., 2018; Brescianini and D'Andrea, 2018; Zhang et al., 2014). An EKF algorithm implemented in C++ language, is used in this work. The general framework for KF and EKF is described in the Appendix A.1-A.2.

The EKF algorithm requires the system's dynamic model. In this work, it is assumed that only gravity and drag forces act on the thrown ball. The following ballistic model with a quadratic drag, which is generally used to take into consideration drag forces (Müller et al., 2011) is used in this work:

$$\ddot{\vec{p}} = -k_d ||\dot{\vec{p}}|| \dot{\vec{p}} + \vec{g} \tag{3.1}$$

where, $\vec{p}$ is the position vector of the ball, k_d is the drag coefficient and $\vec{g} = [0, 0, -9.8]^T$ is the gravity vector.

It is necessary to estimate the position and velocity of the ball as these parameters are required for the BIM to intercept the ball and send it to a desired spot. *The values for the drag coefficient of the ball must be estimated online, rather than measured and stored, as individual balls can show large differences in behaviour.* The estimated state (x) of the ball thus has seven elements. $x_{1:3}$ for the velocity, $x_{4:6}$ for the position and x_7 for the drag coefficient of the ball.

The discrete form of the dynamics of the state vector can be given as follows:

$$\begin{bmatrix} x_{1:3}^{t+1} \\ x_{4:6}^{t+1} \\ x_7^{t+1} \end{bmatrix} = \begin{bmatrix} dt(-k_d ||x_{1:3}^t|| x_{1:3}^t + \vec{g}) + x_{1:3}^t \\ dt(x_{1:3}^t) + x_{4:6}^t \\ x_7^t \end{bmatrix} \tag{3.2}$$

where, $x_{1:3}^t$, $x_{4:6}^t$ and x_7^t represent the velocity, position and drag coefficient states at a certain time t and dt is the sample time which is set according to the frequency at which OptiTrack system sends the position data.

The position data coming from the OptiTrack system is considered as the measurement vector for the EKF.

To test the algorithm, experiments are performed in the OptiTrack environment where a ball is thrown at random velocities from random positions. To validate the estimation of the drag coefficient of the ball, it is theoretical calculated beforehand as follows:

$$k_d = \frac{\rho A_{ball} C_d}{2 m_{ball}} \tag{3.3}$$

where, $\rho = 1.2 kg/m^3$, $C_d = 0.4$ (McCormick, 1995), the circular area $A_{ball} = 1.257 \times 10^{-3} m^2$ and the mass of the ball $m_{ball} = 3.6655 \times 10^{-3} kg$. Substituting these, we obtain $k_d = 0.05 m^{-1}$

The estimation results using EKF algorithm are shown in Figures 3.3-3.5.

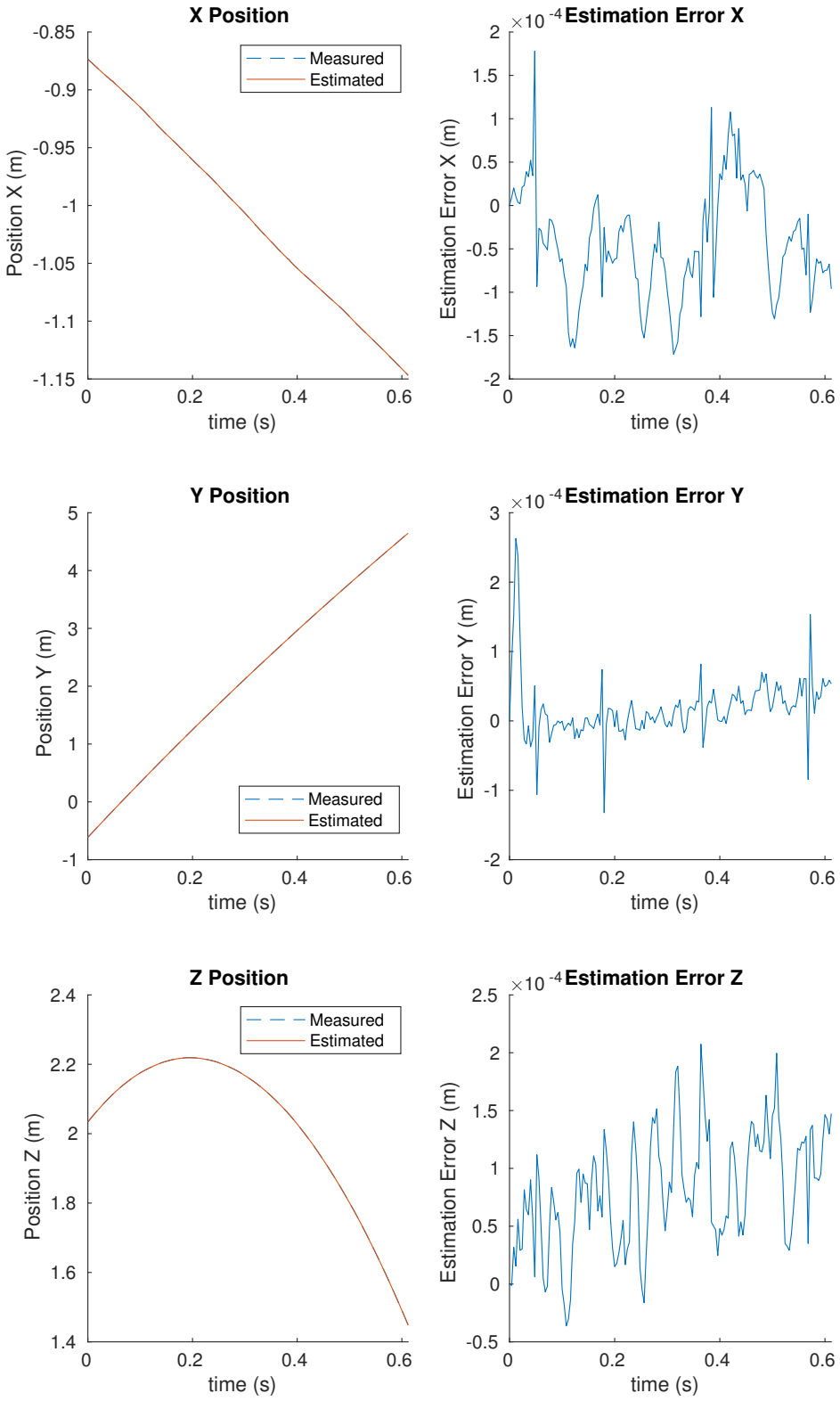

Figure 3.3: Position Estimates and their Estimation errors after application of the EKF algorithm

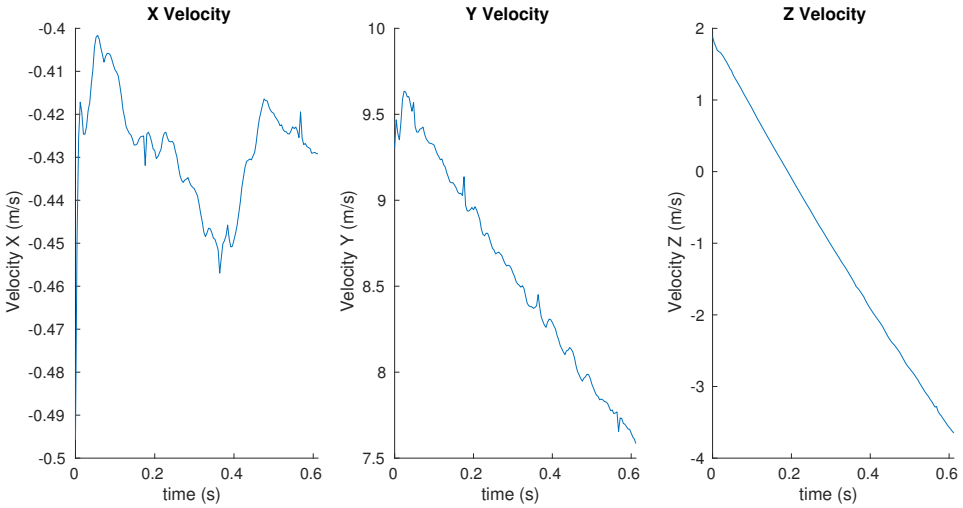

Figure 3.4: Velocity estimates of the ball after application of the EKF algorithm

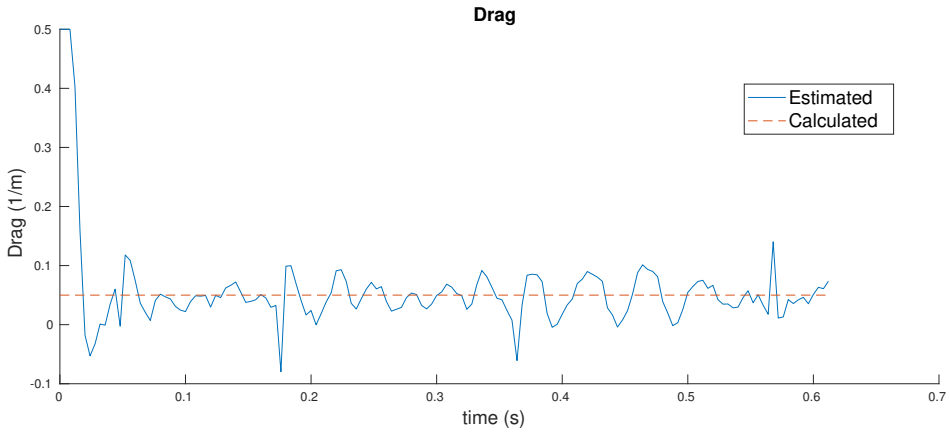

Figure 3.5: Estimate of the drag coefficient of the ball after application of the EKF algorithm

From the estimation results shown in Figure 3.3-3.5 it can be seen that the estimate of the position of the ball is quite accurate and the estimation errors lie in the range of -0.3 mm to 0.3 mm. It can be also seen that the estimate of the drag coefficient of the ball converges to the theoretically calculated value in about 0.05 secs.

Although it can be seen that the estimation of drag coefficient has spikes and goes to negative values at some points, which is unrealistic. To avoid negative values of the drag coefficient estimates, the EKF algorithm is modified to add a constraint on its estimated state, leading to a Constrained-EKF (CAEKF) algorithm. The framework for adding constraints on the estimated state is explained in Appendix A.3.

A constraint which keeps the estimated drag coefficient above the value of zero is used in the CEKF algorithm. The estimation results using CEKF algorithm are shown in Figures 3.6-3.8.

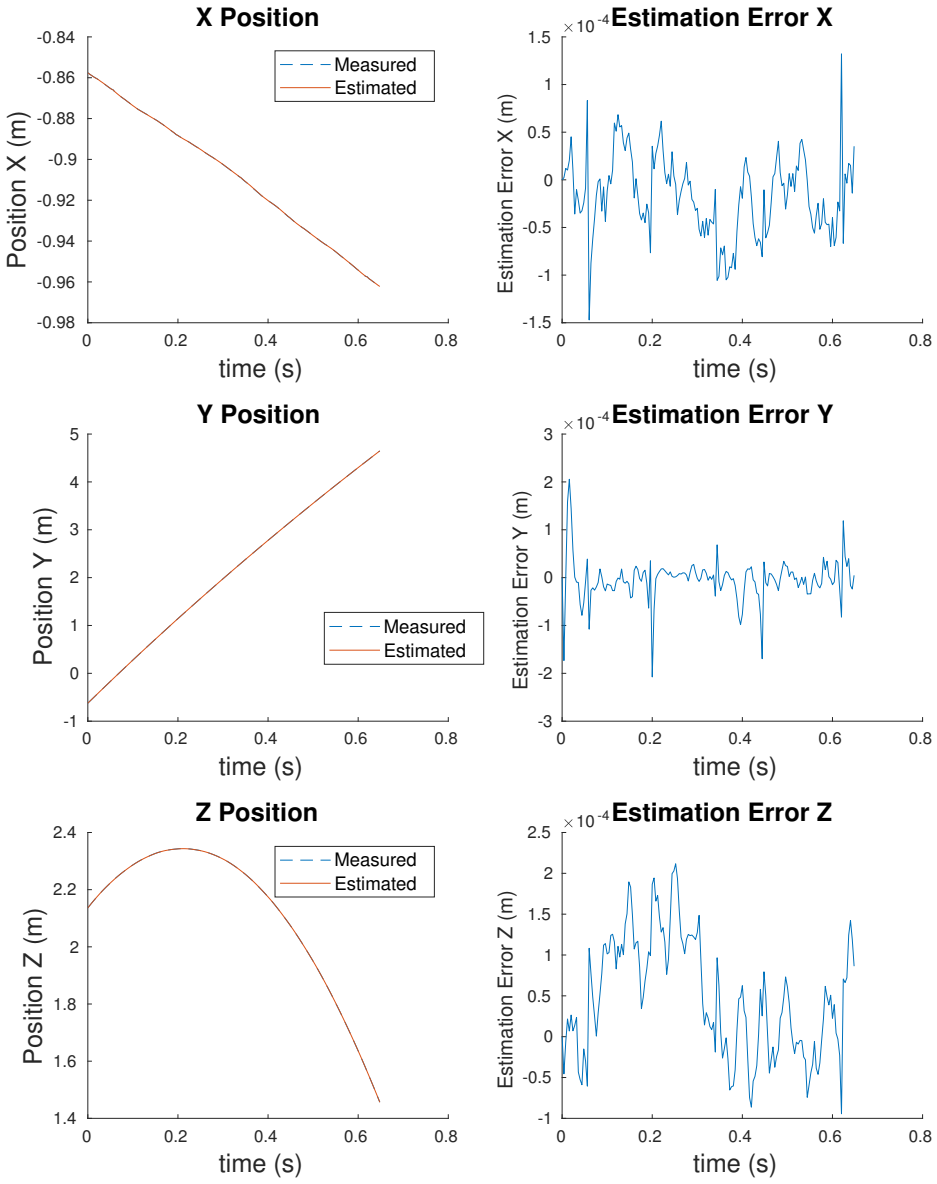

Figure 3.6: Position Estimates and their Estimation errors after application of the CEKF algorithm

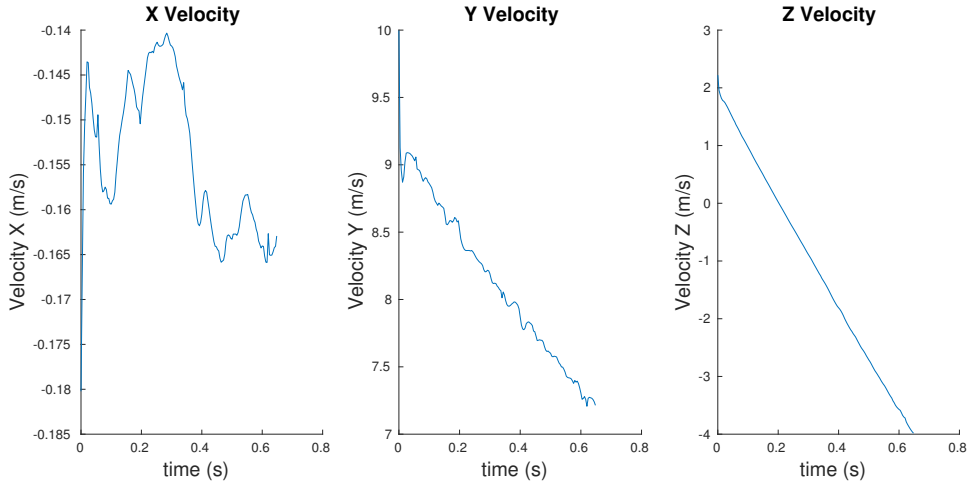

Figure 3.7: Velocity estimates of the ball after application of the CEKF algorithm

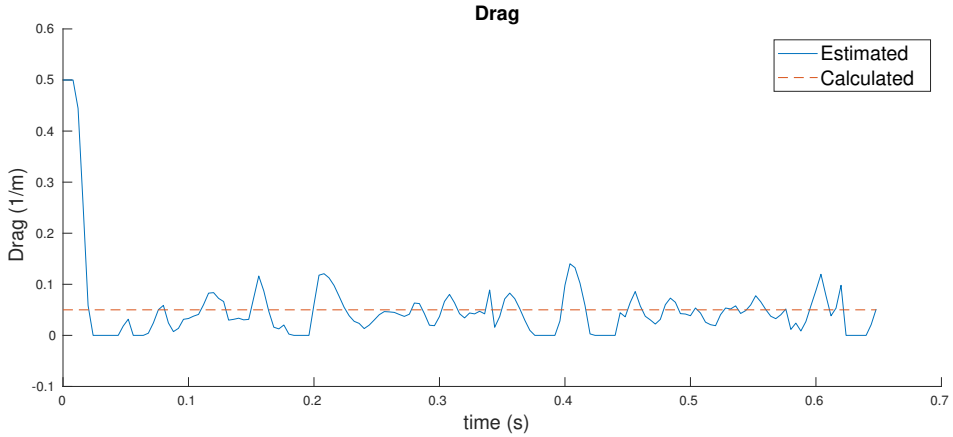

Figure 3.8: Estimate of the drag coefficient of the ball after application of the CEKF algorithm

It can be seen that the drag coefficient is now constrained to non-negative values and converges to the theoretically calculated value in about 0.05 secs. Also, the estimation error in the position of the ball are in a similar range to that of the EKF algorithm. The estimation of the drag estimate still remains spiky.

It is well known that the co-variance matrices of process noise ($\boldsymbol{C_w}$) and measurement noise ($\boldsymbol{C_v}$) have a significant impact on the Kalman filter's (KF) gain in estimating dynamic states (Almagbile et al., 2010). A high Kalman Gain can spike the estimates even for a very small innovation (difference between current measurement and its corresponding predicted measurement). The spiking can be mitigated by modifying the EKF's Kalman gain to be adaptive, leading to an Adaptive Extended Kalman Filter (AEKF). The adaptive part of the algorithm can make sure that the Kalman filter becomes less aggressive once convergence is achieved reducing the spikes. This can also mitigate the problems in tuning the filter by trial and error methods, which can be a tedious task. Moreover, the noise levels may change for changing environment making the chosen tuning parameters of the KF ineffective. The AEKF algorithm can adapt the Kalman gain making it more robust against noise. The AEKF algorithm used in this work is described in Appendix A.4.

The estimation results using the AEKF algorithm are shown in Figures 3.9-3.11.

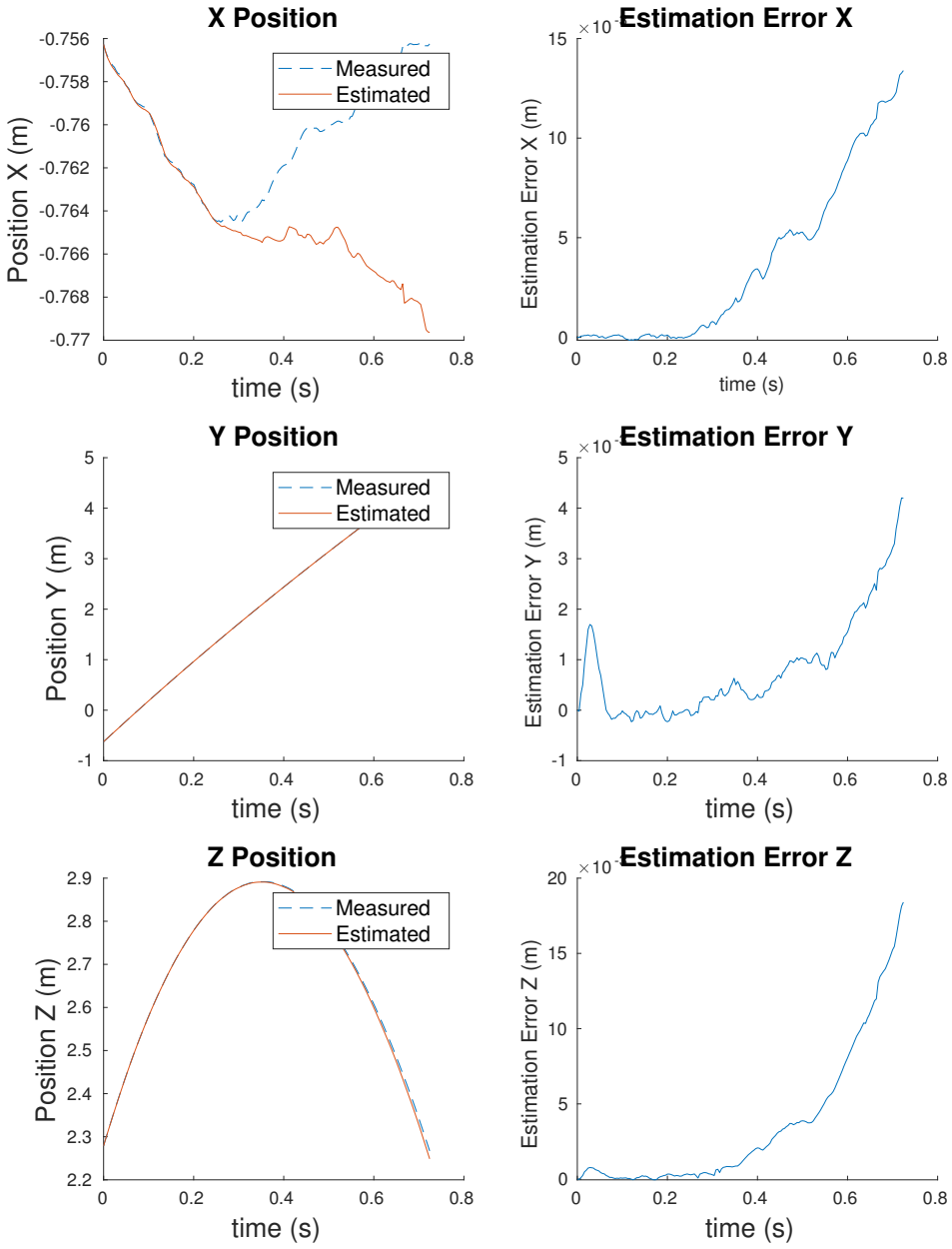

Figure 3.9: Position Estimates and their Estimation errors after application of the CEKF algorithm

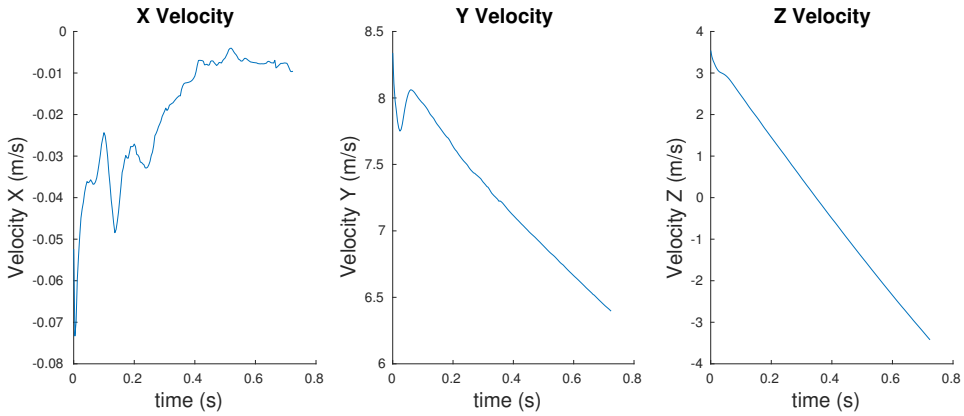

Figure 3.10: Velocity estimates of the ball after application of the CEKF algorithm

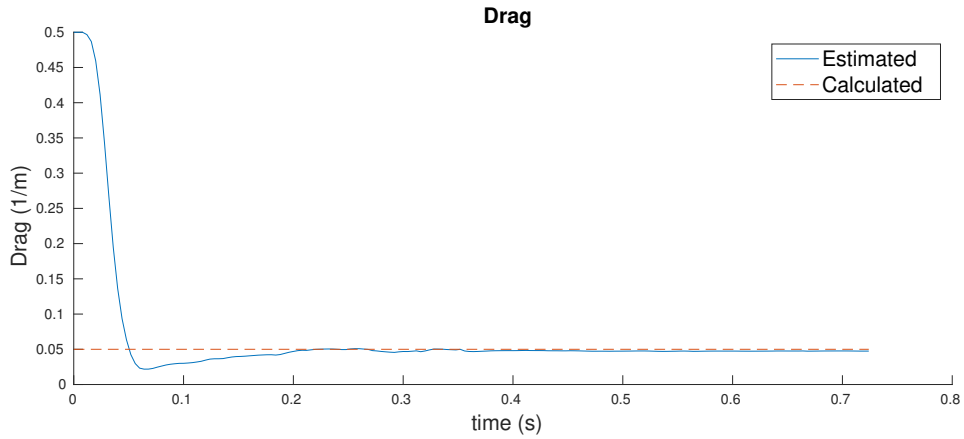

Figure 3.11: Estimate of the drag coefficient of the ball after application of the CEKF algorithm

It can be seen that the drag coefficient now has no spikes. But, its value converges to the theoretically calculated value in about 0.2 secs, which is much more than that of EKF and CEKF. Also, the estimation error in the position of the ball are quite large as compared to the results from EKF and CEKF. The AEKF algorithm is thus chucked out.

The EKF and CEKF algorithms provide almost similar results for the estimation of position and velocity of the ball. The drag estimate obtained from CEKF is constrained to be non-zero, making it more realistic. Although the CEKF algorithm is computationally more expensive than EKF, this is of no concern because, each iteration of the EKF and CEKF is finished well before the minimum possible sampling time $\boldsymbol{dt}$ of the algorithms, which is set according to the frequency at which the OptiTrack provides position measurements. *It is therefore decided to make use of the CEKF algorithm for estimating the ball trajectory.*

3.1.3 Prediction of the ball trajectory:

As seen from the CEKF estimation results (Figures 3.6-3.8) in the previous section, the estimation algorithm converges in about 0.05 secs. Once the estimation has converged, the prediction of the ball trajectory can be started. The trajectory of the ball can be predicted by numerically integrating the discrete dynamic model (Equation 3.2), by setting its initial value to the currently estimated state of the ball. The estimation and prediction of the ball trajectory take place continuously till the ball is intercepted.

3.1.4 Calculation of the interception point details:

The trajectory (i.e. the state) of the ball is predicted till it intersects an Interception Plane (Figure 3.1). This provides the interception point details, namely, the interception time (t_i), interception point ($\vec{p}_i$) and the velocity of the ball ($\vec{v}_i^-$) at the interception point. As the state of the ball is continuously estimated, the predicted interception point details keep bettering. Figure 3.12 shows the development in the x and z co-ordinates of the predicted interception point. The y co-ordinate of the interception point remains constant as the interception plane is parallel to the XZ plane ((Figure 3.1)).

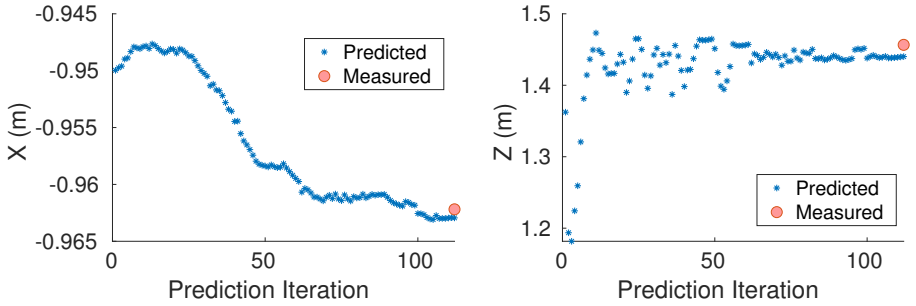

Figure 3.12: Development of x and z co-ordinates of the predicted interception point with each prediction iteration.

The result for the interception position of the ball is acceptable, as the size of the bat used in this work is 20 cm× 40 cm ,which is quite large compared to the distance error in the measured and predicted interception point, which is usually around 0.5 cm.

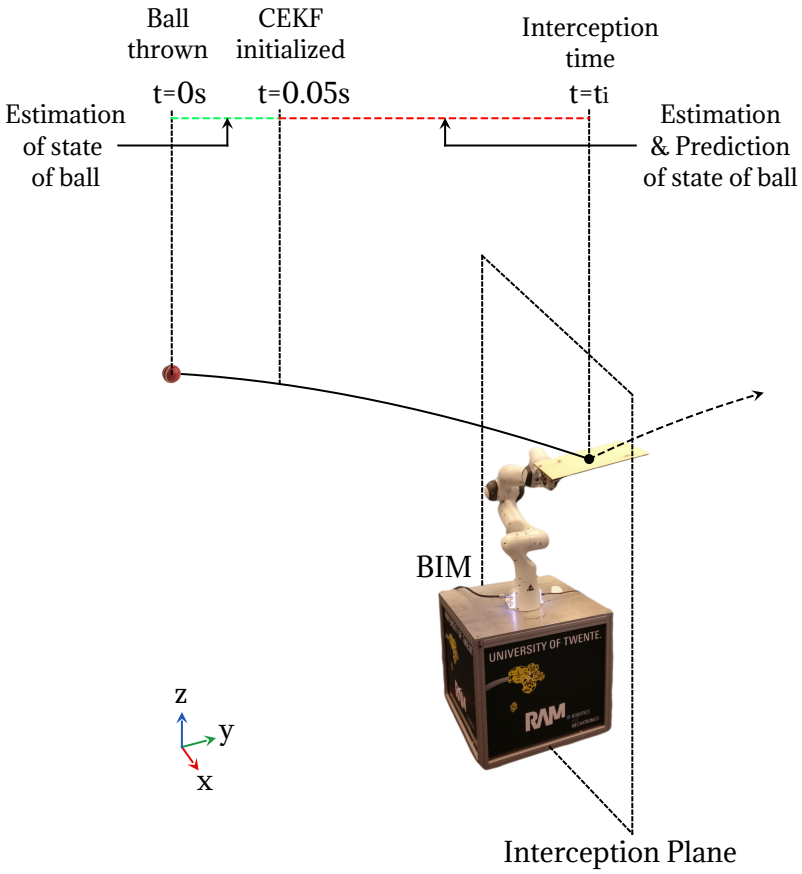

Figure 3.13: Work-flow of Subsystem 1

3.1.5 Subsystem Overview

Figure 3.13 shows an overall work-flow of Subsystem 1 and Figure 3.14 shows an overview of the inputs and outputs Subsystem 1.

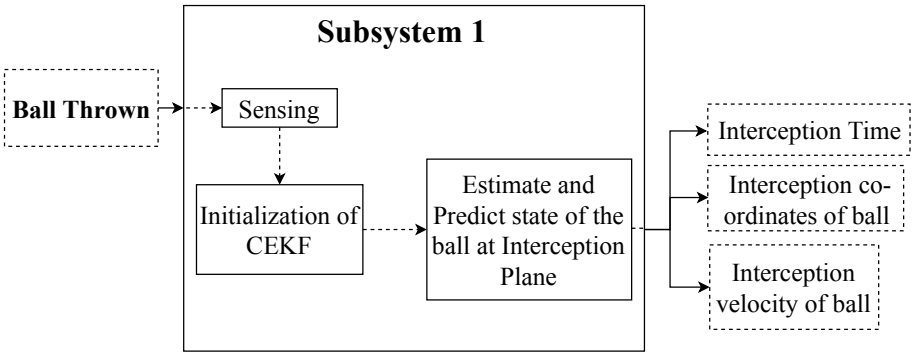

Figure 3.14: Subsystem 1 Overview

3.2 Subsystem 2: Decide where to send the ball to

This susbsystem decides the spot for sending the thrown ball to, by the BIM. This spot is called 'catching point' in the discussion ahead.

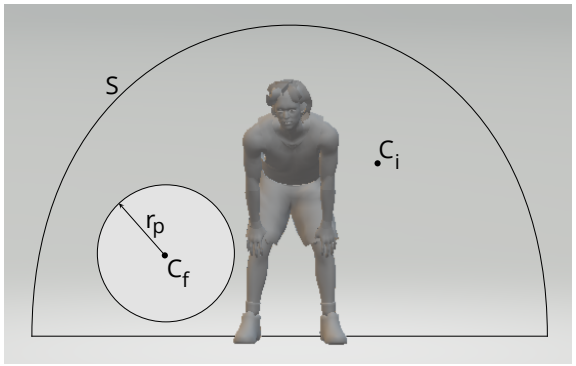

Figure 3.15: Selection of catching points

To decide the catching point, firstly a semi-circular catching region 'S' is defined around the player as is shown in Figure 3.15. For every throw of the ball, a random point ($\boldsymbol{c_f}$) is then selected in this circular region for providing catches.

It is assumed that the thrown ball has enough speed so that it can reach any point in the catching region after it is swerved from the BIM. This is a fitting assumption as players usually have a good idea of what speed should the ball be thrown, so that it can reach the fielder taking catches.

To provide a feinting motion, the catch is set up as follows: A random point ($\boldsymbol{c_f}$) is selected in the catching region S and another random point ($\boldsymbol{c_i}$) is selected such that it lies in the region S, but outside a circle of a pre-defined radius $\boldsymbol{r_p}$ having its center at $\boldsymbol{c_f}$ (See Figure 3.15). During the interception of the thrown ball the bat of the BIM is firstly made to orient itself to show as if it is trying to provide a catch at the point $\boldsymbol{c_i}$, but the bat finally orients itself to provide a catch at the point $\boldsymbol{c_f}$ at the interception point, thus in all, feinting its motion. The decision to feint the motion of the ball is made randomly by the subsystem.

3.2.1 Subsystem Overview

Figure 3.16 shows an overview of Subsystem 2.

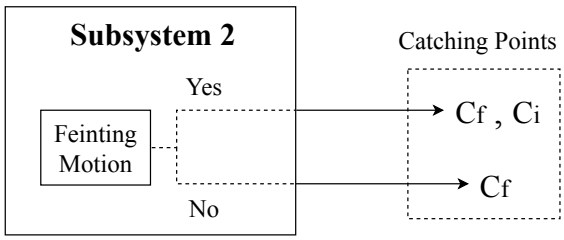

Figure 3.16: Subsystem 2 Overview

3.3 Subsystem 3: Strategy for how to send the ball to a desired spot

This subsystem discusses the strategy for how to orient the bat of BIM to send the ball to the output point specified by Subsystem 2. Precisely, the task of Subsystem 3 can be stated as follows:

Given the interception point for the BIM $\vec{p}_i$, interception velocity of the ball $\vec{v}_i^-$ (from Subsystem 1) and the catching point(s) $\vec{p}_d$ (from Subsystem 2), find the orientation of the bat in the form of a normal vector $\vec{n}$ to the surface of the bat.

The ball thrown by the thrower is intercepted by the BIM in a pre-defined fixed plane, called the Interception Plane (IP), in the workspace of the BIM (Figure 3.17). This plane is defined parallel to the face of the user taking catches. The user is asked to stand at a pre-defined distance from the interception plane.

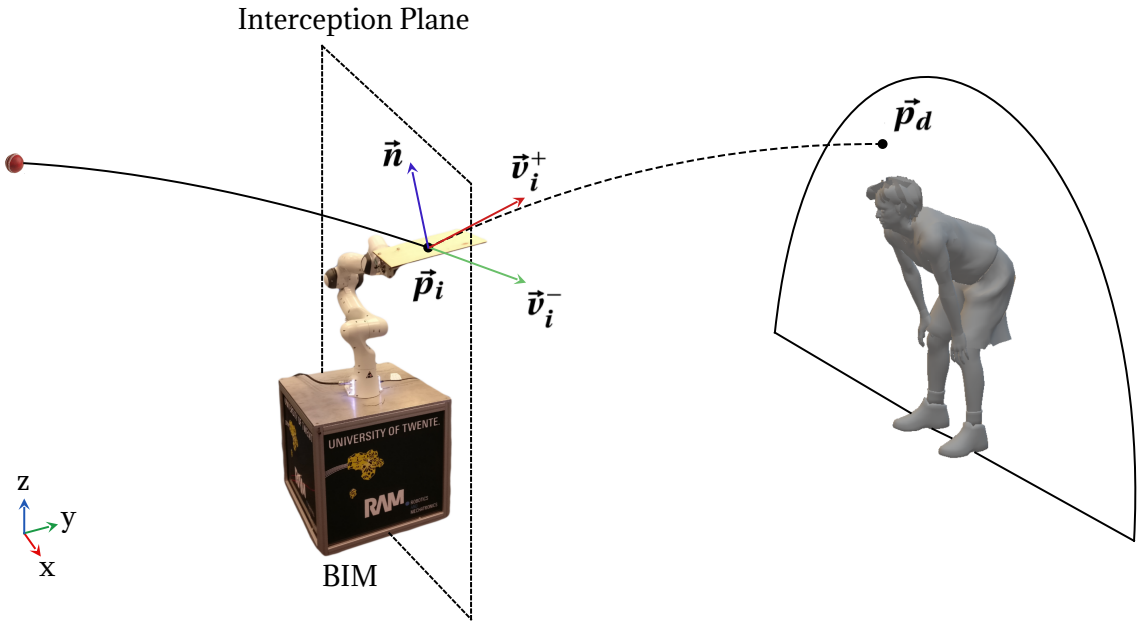

Figure 3.17: The various points of interest and variables involved in deciding how to send the ball to a desired point

As shown in the Figure 3.17, $\vec{p}_i$ is the interception point, $\vec{v}_i^-$ is the velocity before impact, $\vec{v}_i^+$ is the velocity after the impact and $\vec{p}_d$ is the desired point, the ball has to be sent to by the BIM.

The surface of the bat placed on the BIM in this work is flat and it is assumed that this surface only applies a force in the direction normal to it, when the ball hits it. Thus, we have:

$$\vec{n} = \hat{v}_i^+ - \hat{v}_i^- \tag{3.4}$$

where, $\vec{n}$ is the normal vector to the surface of the bat, $\hat{v}_i^+$ and $\hat{v}_i^-$ are unit vectors in the direction of $\vec{v}_i^+$ and $\vec{v}_i^-$ respectively.

Also, the bat of the BIM is made to have a zero velocity at the time of impact, giving the folowing equation:

$$||\vec{v}_i^+|| = \beta||\vec{v}_i^-|| \tag{3.5}$$

where, $\beta = \mathbf{0.6}$ is the coefficient of restitution, which is calculated experimentally. The procedure to calculate the co-efficient of restitution is explained in Appendix B.

As can be seen in Equation 3.4, to calculate $\vec{n}$ is it required to calculate $\vec{v}_i^+$. The problem of Subsystem 3 can thus be recast as follows:

Given $\vec{p}_i$, $\vec{v}_i^-$, β and $\vec{p}_d$, it is required to calculate $\vec{v}_i^+$.

This problem comes under the class of *Two-point boundary value problem (TPBVP)* (Wikipedia, 2018a). A detailed analysis of the problem and its solution methodology is described in the following subsections.

3.3.1 Analysis of the Two Point Boundary Value Problem

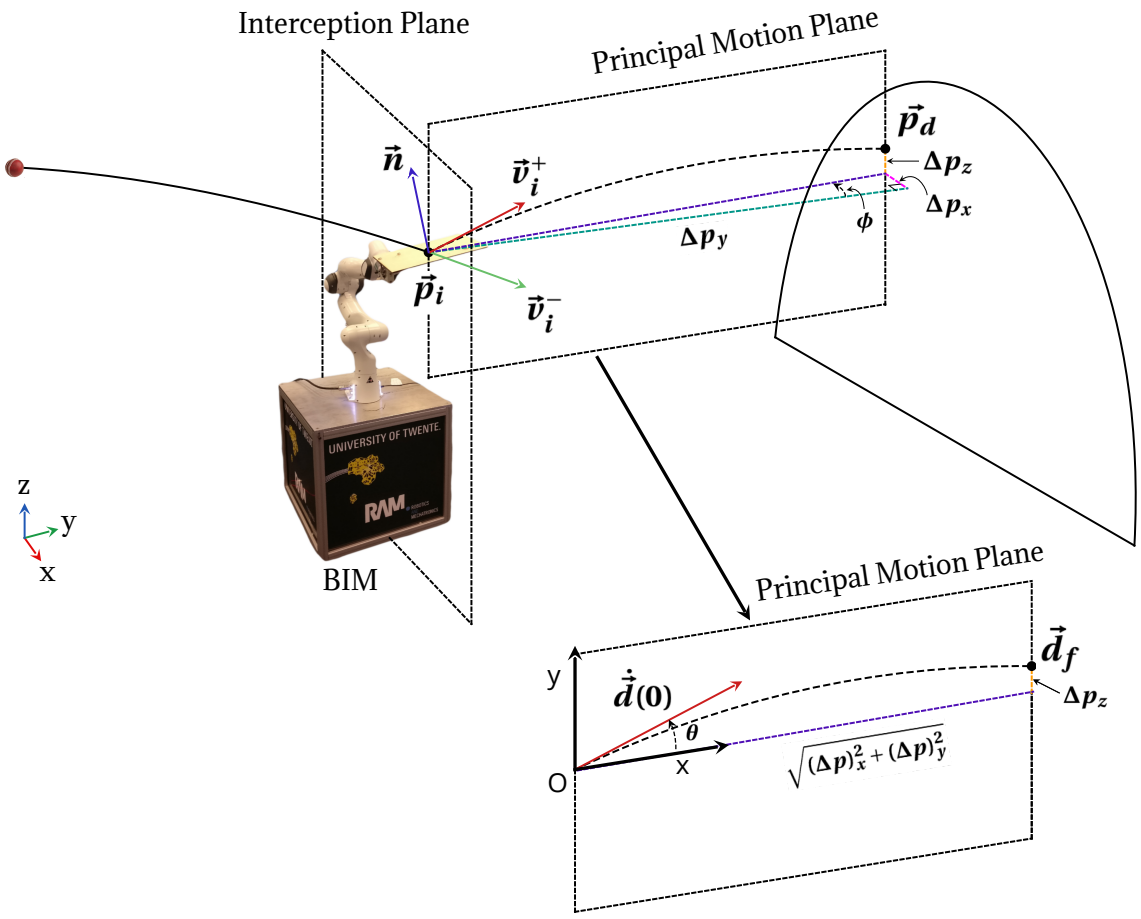

Figure 3.18: Motion of ball in Principal Motion Plane

To make the analysis simple, the problem of sending the ball to the desired position is formulated in the principal motion plane (PMP) i.e. plane perpendicular to the ground passing through the points $\vec{p}_i$ and $\vec{p}_d$ (See Figure 3.18). The origin (O) of the PMP is set to the intercep-

tion point $\vec{p_i}$ and the desired point $\vec{p_d}$ in the PMP can now be given as follows :

$$\vec{d_f} = \begin{bmatrix} \sqrt{(\Delta p)_x^2 + (\Delta p)_y^2} \\ \Delta p_z \end{bmatrix} \tag{3.6}$$

where, $\Delta p = \vec{p_d} - \vec{p_i}$ and Δp_x, Δp_y and Δp_z are the x,y and z components of Δp.

Let $\vec{d(t)}$ represent a position of the ball in the PMP at a specific time instant t. We have the following differential equation and initial conditions involved in the problem:

$$\ddot{\vec{d}}(t) = -k_d ||\dot{\vec{d}}(t)|| \dot{\vec{d}}(t) + \vec{g} \tag{3.7}$$

$$\vec{d}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.8}$$

$$\dot{\vec{d}}(0) = \begin{bmatrix} ||\vec{v}_i^+|| cos(\theta) \\ ||\vec{v}_i^+|| sin(\theta) \end{bmatrix} \tag{3.9}$$

$$\vec{d}(t_f) = \vec{d_f} \tag{3.10}$$

where, k_d is the drag coefficient of the ball, $\vec{g} = [0, -9.8]^T$ is the gravity vector, θ is the launch angle and t_f is the time at which the ball reaches the point $\vec{p_d}$.

Equation 3.7 is the dynamic model of the thrown ball. Equations 3.8-3.10 are the boundary constraints of the problem.

As can be seen in Equation 3.7 there are four integration constants involved. From equations 3.8-3.10 we have six boundary conditions and two free parameters i.e. θ and t_f. A TPBVP is well defined if the number of integration constants and free parameters add to the number of boundary conditions, which is the case here. *The formulated TPBVP is thus well-defined.*

3.3.2 Existence and Uniqueness of the TPBVP solution

For a given $||\vec{v}_i^+||$, there exists an envelope in which all the possible trajectories of the ball lie (Chudinov, 2005). In case the catching point $\vec{p_d}$ lies:

- inside this envelope, the TPBVP has two set of solutions for θ and t_f.
- on this envelope, the TPBVP has a unique solution for θ and t_f.
- outside this envelope, the TPBVP has no solution.

As described in Subsystem 2, it is assumed that the ball is thrown with a velocity such that it is able to reach the desired point when swerved by the BIM. The TPBVP thus always has a solution. In case the TPBVP has two set of solutions for θ and t_f, the solution with a lower value of θ is chosen. This is because the trajectory of the ball in a slip catch is usually almost flat.

3.3.3 Solution Methodology for the TPBVP

The TPBVP formulated in the previous section does not have an analytic solution. Hence, it needs to be solved using appropriate numerical methods (Ascher et al., 1994).

The `scikits.bvp_solver`, a python package (John Salvatier, 2012) which can solve a general TPBVP is used to solve the formuated problem in the subsection 3.3.1. The package takes

around 10 ms to solve the formulated problem and is computationally fast enough to guarantee that it can suitably be implemented in the task at hand wherein time is a cardinal resource.

The parameter $\boldsymbol{\theta}$ obtained from solving the TPBVP is then used to obtain the velocity of the ball post impact i.e. $\vec{v}_i^+$. This is done as follows (refer Figure 3.18):

$$\vec{v}_i^+ = \begin{bmatrix} ||\vec{v}_i^+||cos(\theta)sin(\phi) \\ ||\vec{v}_i^+||cos(\theta)cos(\phi) \\ ||\vec{v}_i^+||sin(\theta) \end{bmatrix} \tag{3.11}$$

where, $||\vec{v}_i^+||$ is calculated from Equation 3.5 and $\phi = tan^{-1}(\frac{\Delta p_x}{\Delta p_y})$.

The unit vectors $\hat{v}_i^+$ and $\hat{v}_i^-$ can then be calculated and substituted in equation 3.4 to obtain the required normal vector ($\vec{n}$) for the surface of the bat.

3.3.4 Choice of BIM

Based on the above analysis, it can be seen that the BIM needs to have atleast 4 degrees of freedom (DOFs). Two for translating in the interception plane and two to set the normal vector of the surface of the bat. Based on the availability of resources, it was decided to make use of a 7 DOF robotic manipulator, 'Panda' (Franka Emika GmbH, 2018b) as the BIM.

3.3.5 Subsystem Overview

Figure 3.19 shows an overview of Subsystem 3 .

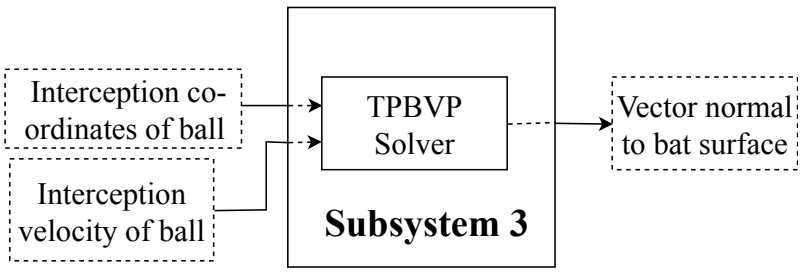

Figure 3.19: Subsystem 3 Overview

3.4 Subsystem 4: Trajectory Planning and Execution on BIM

This subsystem discusses the trajectory planning and its execution on the BIM.

3.4.1 Trajectory Planning

To plan the trajectory for the bat of the BIM, the interception time, interception point and the required orientation of the bat at the interception point are required. The interception time and point are provided by Subsystem 1 while the orientation of the bat is provided by the Subsystem 3.

Assume that, at a certain time t_o, the interception time t_i, the interception point $\vec{p}(t_i)$, the vector normal to the surface of the bat $\vec{n}(t_i)$ required to send the ball to desired point are available. A trajectory planner is then evoked to plan a trajectory for the bat of the BIM starting from its current configuration to the interception point configuration.

The trajectory for $\vec{p}(t)$ (hitting point of the bat) and $\vec{n}(t)$ (normal vector to the surface of the bat) is planned separately and is explained in the following subsections.

Trajectory Planning for $\vec{p}(t)$:

To achieve the desired translation of the hitting point of the bat $\vec{p}(t)$, a third order polynomial trajectory is used:

$$\vec{p}(t) = \vec{a_0} + \vec{a_1}(t_i - t_o) + \vec{a_2}(t_i - t_o)^2 + \vec{a_3}(t_i - t_o)^3 \tag{3.12}$$

We have the following boundary conditions on this trajectory:

- $\vec{p}(t_o)$ is the position of the hitting point of the bat when the trajectory planner is evoked.
- $\dot{\vec{p}}(t_o)$ is the velocity of the hitting point of the bat when the trajectory planner is evoked.
- $\vec{p}(t_i)$ is the interception point received from Subsystem 1.
- $\dot{\vec{p}}(t_i)$ is zero.

Based on these boundary conditions we have,

$$\vec{a_0} = \vec{p}(t_o) \tag{3.13}$$

$$\vec{a_1} = \dot{\vec{p}}(t_o) \tag{3.14}$$

$$\vec{a_2} = \frac{-3(\vec{p}(t_o) - \vec{p}(t_i)) - 2\dot{\vec{p}}(t_o)(t_i - t_o))}{(t_i - t_o)^2} \tag{3.15}$$

$$\vec{a_3} = \frac{2(\vec{p}(t_o) - \vec{p}(t_i)) + \dot{\vec{p}}(t_o)(t_i - t_o))}{(t_i - t_o)^3} \tag{3.16}$$

Trajectory Planning for $\vec{n}(t)$:

The trajectory for $\vec{n}(t)$ i.e. the *unit* normal vector to the surface of the bat is planned by planning the trajectory of a rotation vector $\vec{r}(t)$, such that following equation holds:

$$\vec{n}(t) = e^{[\vec{r}(t)]}\vec{n}(t_o) \tag{3.17}$$

where, $[\vec{r}(t)]$ is the skew-symmetric cross-product matrix representation of $\vec{r}(t)$.

Following holds from Equation 3.17:

$$\dot{\vec{n}}(t) = \dot{\vec{r}}(t) \times \vec{n}(t) \tag{3.18}$$

It is clear from Equation 3.17 that $\vec{r}(t_o) = \vec{0}$. Thus, the following third order polynomial trajectory is used for $\vec{r}(t)$:

$$\vec{r}(t) = \vec{b_1}(t_i - t_o) + \vec{b_2}(t_i - t_o)^2 + \vec{b_3}(t_i - t_o)^3 \tag{3.19}$$

We have the following boundary conditions on this trajectory:

- $\vec{n}(t_o)$ is the unit vector normal to the surface of the bat when the trajectory planner is evoked.
- $\dot{\vec{n}}(t_o)$ is the rate of change of the unit vector normal to the surface of the bat when the trajectory planner is evoked.
- $\vec{n}(t_i)$ is the unit vector normal to the surface of the bat at interception point obtained from Subsystem 3.

- $\dot{\vec{n}}(t_i)$ is zero.

Based on the above boundary conditions we have,

$$\vec{b_1} = \dot{\vec{r}}(t_o) \tag{3.20}$$

$$\vec{b_2} = \frac{3(\vec{r}(t_i) - 2\dot{\vec{r}}(t_o)(t_i - t_o)}{(t_i - t_o)^2} \tag{3.21}$$

$$\vec{b_3} = \frac{(-2\vec{r}(t_i) + \dot{\vec{r}}(t_o)(t_i - t_o))}{(t_i - t_o)^3} \tag{3.22}$$

As discussed in Subsystem 2, in case the subsystem decides to feint the motion of the bat, the Subsystem 2 provides two normal vectors for the surface of bat $\vec{n}_i$ (for the catching point c_i) and $\vec{n}_f$ (for the catching point c_f). The trajectory to set these normal vectors is done in two phases. Let t_s be the time at which subsystem 4 is evoked to start planning the trajectory for BIM. In the first phase, a fraction $k = 0.5$ of the time $t_i - t_s$ is allocated to set the normal vector to the surface of the bat from $\vec{n}(t_s)$ to $\vec{n}_i$ and in the second phase, the remaining time $(1-k)(t_i - t_s)$ is allocated to set the normal from $\vec{n}_i$ to $\vec{n}_f$. Both the phases are applied a third order trajectory profile as mentioned in Equation 3.19.

3.4.2 Motion Execution

The planned trajectory from the previous section is executed on the BIM i.e. the Panda arm by providing it with the twist command $T(t)$ given as follows:

$$T(t) = \begin{bmatrix} \dot{\vec{p}}(t) \\ \dot{\vec{r}}(t) \end{bmatrix} \tag{3.23}$$

where, $\dot{\vec{p}}(t)$ and $\dot{\vec{r}}(t)$ are the velocity profiles defined in the previous section and $\hat{r}$ is the axis of rotation for the normal vector to the surface of the bat.

The Panda arm is controlled in the ROS environment using the `franka_ros` package (Franka Emika GmbH, 2018a). The twist commands are sent at a frequency of 1000 Hz to the manipulator. A new trajectory is planned at a frequency of 250 Hz, based on the updated interception point details obtained from Subsystem 1 and Subsystem 3.

3.4.3 Subsystem Overview

Figure 3.20 shows an overview of Subsystem 4.

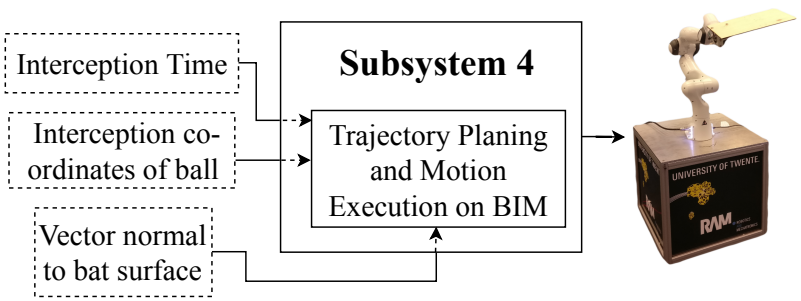

Figure 3.20: Subsystem 4 Overview

3.5 KriCatch 1.0: System Overview

A complete system overview of KriCatch 1.0 is shown in Figure 3.21. Once the ball is thrown, Subsystem 1 starts the sensing, estimation and prediction of the ball's trajectory and outputs the interception point details. Based on the catching points obtained from Subsystem 2 and the interception point details from Subsystem 1, Subsystem 3 decides the strategy for how the send the ball to the desired catching points by providing the normal vector to the surface of the bat. Subsystem 4 now has all its required inputs to plan the trajectory of the bat towards the interception point and finally execute it on the BIM.

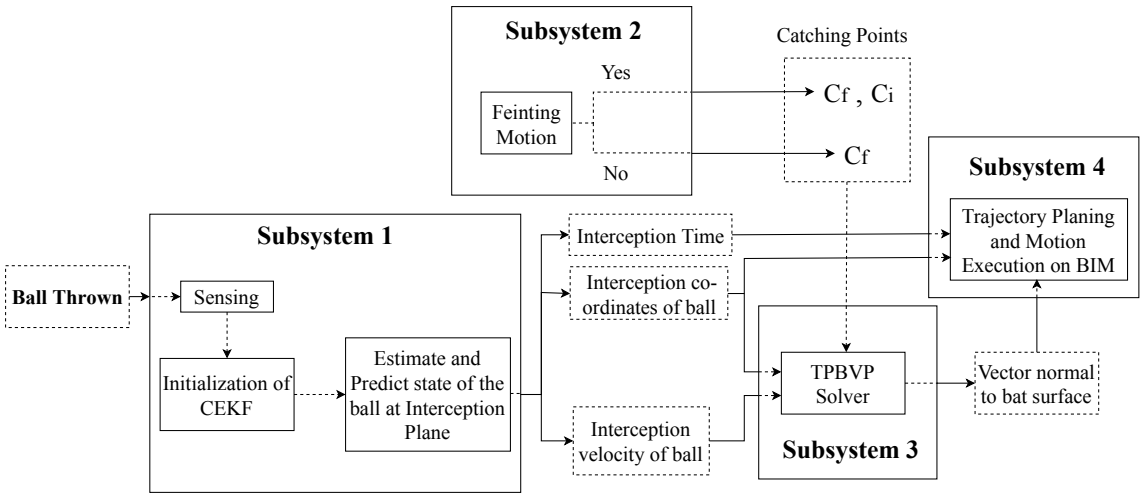

Figure 3.21: A complete system overview of KriCatch 1.0.

4 Experiments on KriCatch 1.0

The experimental setup of KriCatch 1.0 is shown in Figure 4.1. The distance between the thrower and the BIM, ***D***, shown in the figure, available in the OptiTrack environment is around 4.5 m. This is slightly less than the usual distance (5-6 m) set between a thrower and a coach providing catching practice as shown in Figure 1.4 (e). Also, the ball is usually thrown towards the coach at speeds which require the coach to intercept the ball in about 0.3-0.4 secs.

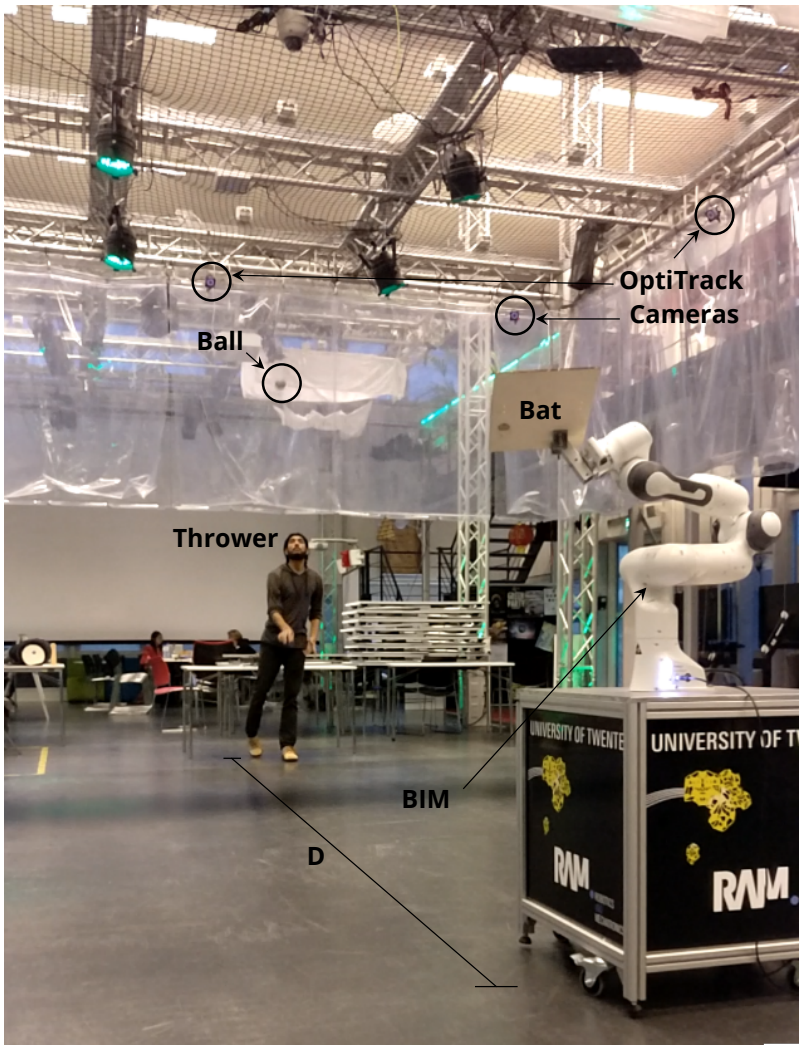

Figure 4.1: Experimental setup of KriCatch 1.0.

The robotic manipulator used in this work, i.e. the Panda Arm has limits on its joint velocity, acceleration and jerk, which are violated in case it is asked to traverse outside a radius of 5 cm from its initial position, within a time period of 0.3-0.4 secs.

To avoid facing the joint limits, following steps need to be taken:

- the ball needs to be thrown at lower speeds so that the interception time reaches around 0.6-0.7 secs
- the ball needs to be thrown in a radius of 10 cm from the starting position of the hitting point on the bat and the tip-tilt rotation of the bat is restricted within 20 degrees of the initial orientation. This is the achievable work-space of Panda arm for the task at hand.

- the feinting motion is disabled
- a low pass filter has to be applied to the sent twist commands to the manipulator to avoid joint velocity and acceleration command discontinuity errors given by the manipulator.

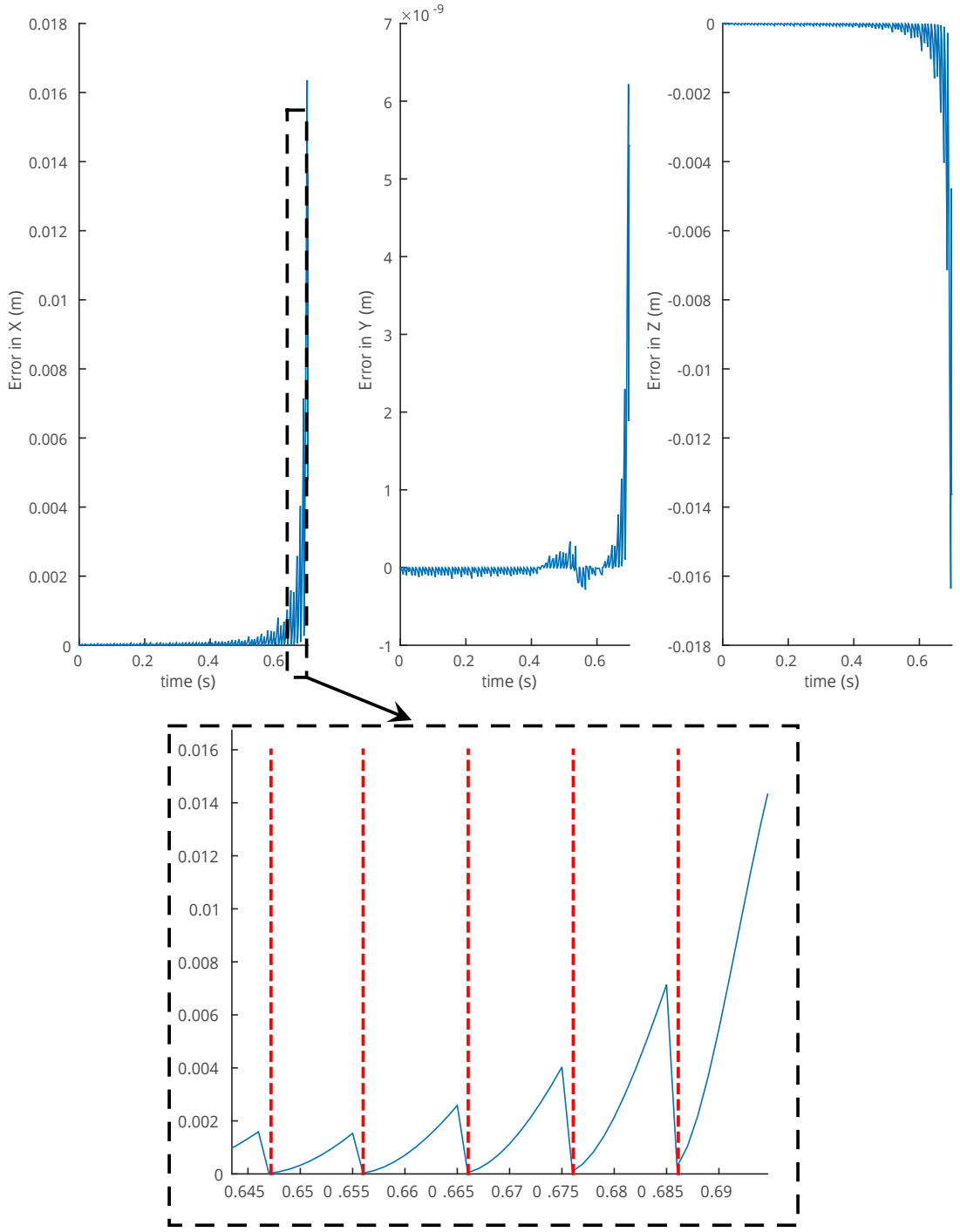

Figure 4.2: Error in position tracking of the bat of BIM for an example trajectory.

The effect of applying the low pass filter, on the position tracking of the hitting point of the bat for an example trajectory is shown in Figure 4.2. The plot show errors between the commanded position and the actual tracked position of the hitting point on the bat. At the red dotted lines, the trajectory is re-planned on the basis of the new interception point details. It can be seen

that the errors in the x and z positions at the end of the trajectory went to 1.6 cms and -1.6 cms respectively. Through experiments, it has been observed that the magnitude of this error can go up to 3 cms for each co-ordinate in case the trajectory needs to be completed in a minimum of 0.5 seconds.

4.1 Ball Interception Results

Figure 4.3 shows different episodes wherein the BIM successfully intercepts and swerves the ball.

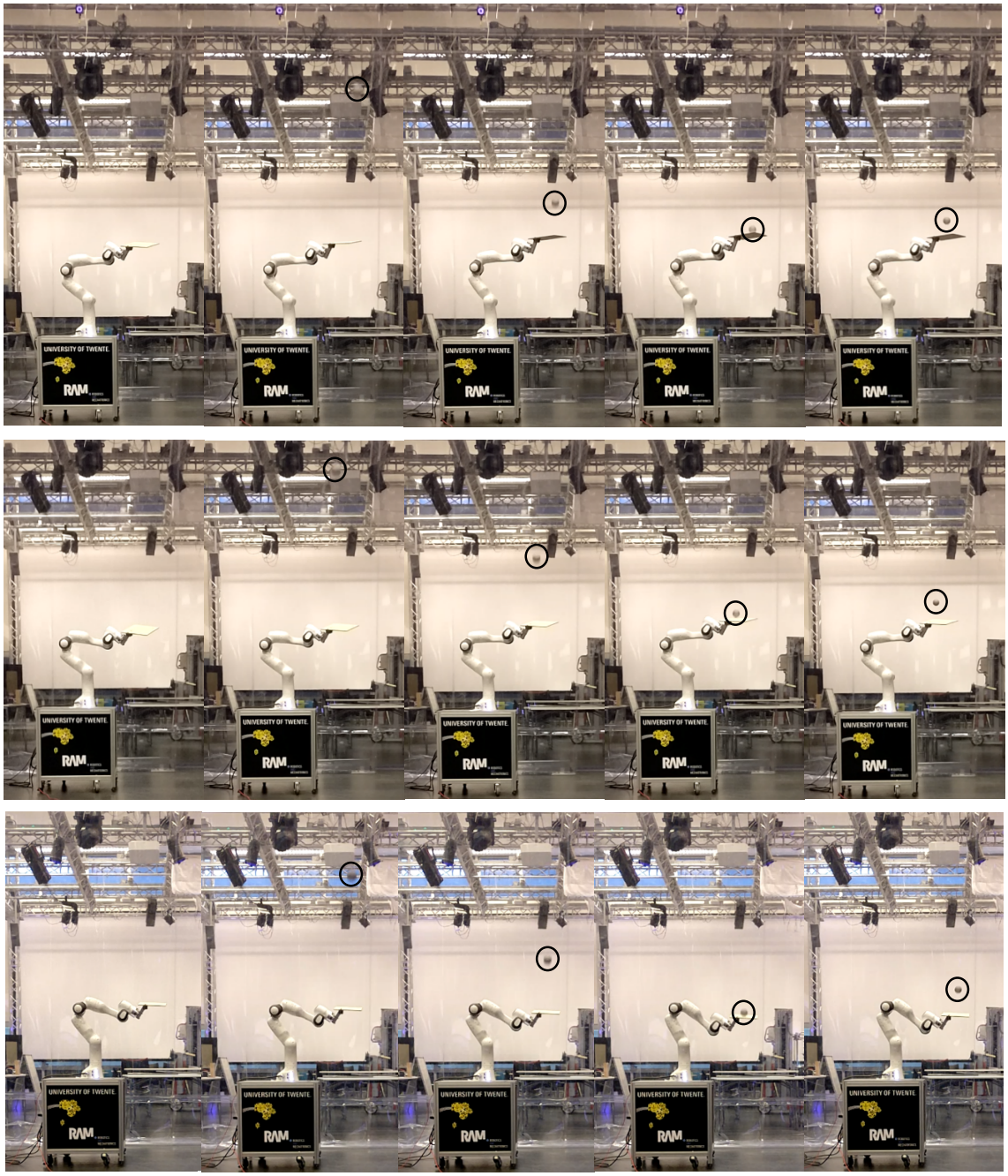

Figure 4.3: Different scenarios wherein the BIM intercepts and swerves the ball.

4.2 Catching Point Results

To examine the working of KriCatch 1.0, it is also important to check if the ball indeed gets swerved to the desired catching points. Two of the episodes of providing a catch are shown in Figure 4.4 and 4.5. It can be seen that the ball indeed gets swerved towards the catching points, although not with a good accuracy (within a radius of 10 cms from the desired catching point).

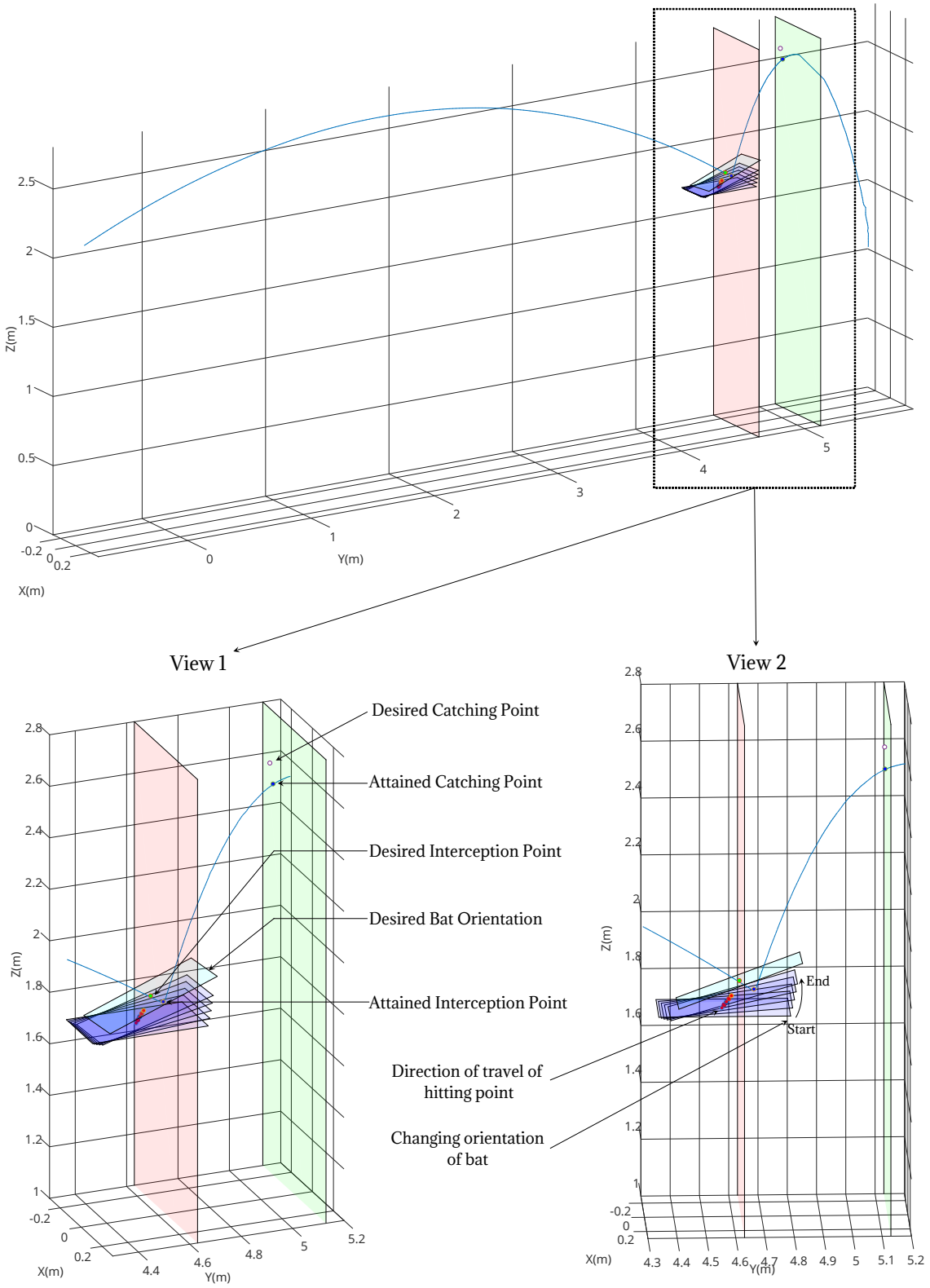

Figure 4.4: Scenario 1 showing the details of a catch given by the BIM.

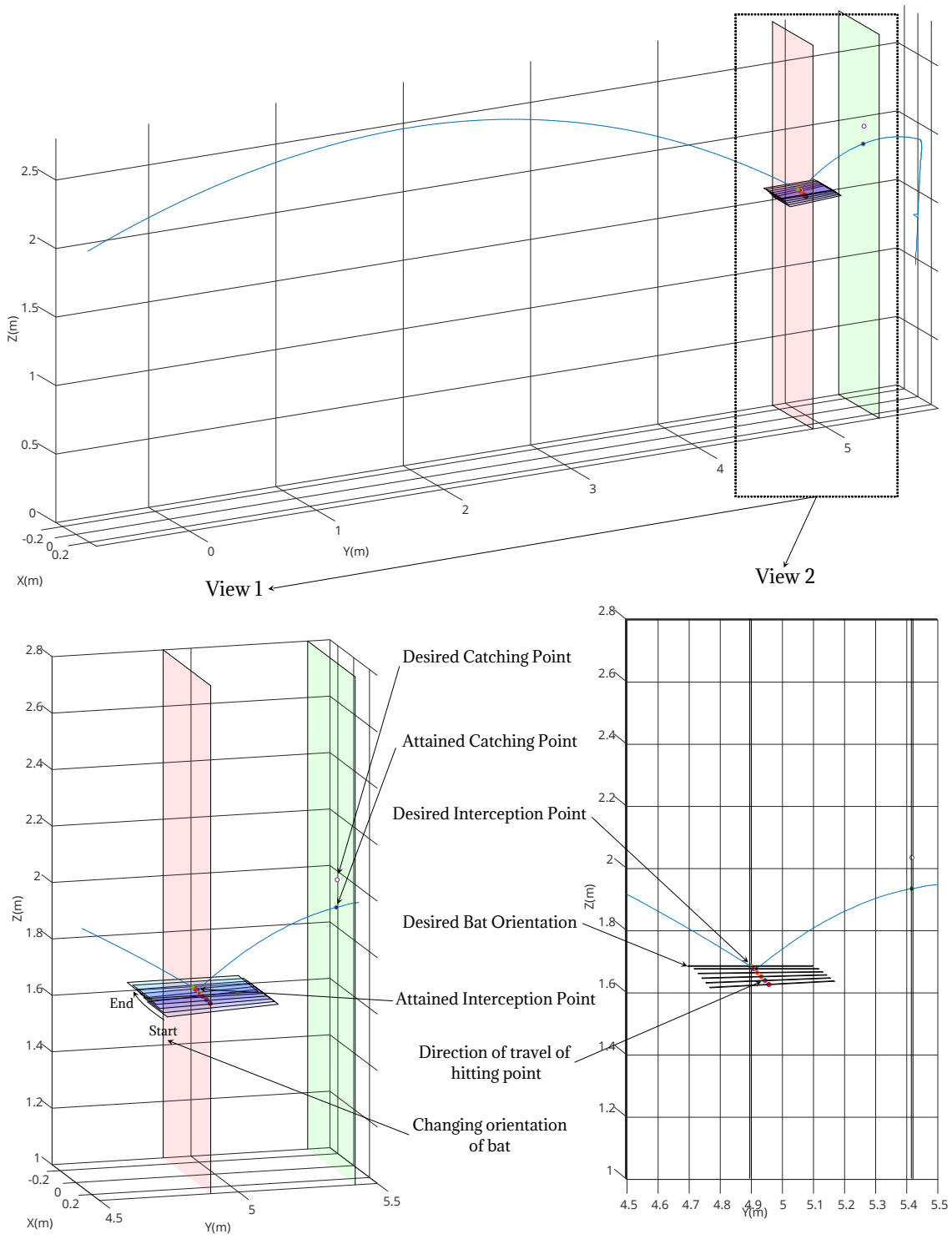

Figure 4.5: Scenario 2 showing the details of a catch given by the BIM.

4.3 Discussion on the results

The main reason for the ball not getting accurately sent to desired catching point can be accounted to the the low-pass filter applied for tracking the planned trajectory of the bat. From Figures 4.4 and 4.5 it can be seen that, the position and orientation of the hitting point of the bat try to follow the desired position and orientation at the interception point. Although the desired position and orientation at the interception point is not accurately reached due to the application of the low pass filter. This results in, the ball hitting the bat at points different than

that of the hitting point. Moreover, due to the application of the low pass filter, the velocity of the hitting point is not exactly zero when the ball hits the bat, which is an assumption taken into consideration for Subsystem 3 to calculate the orientation of the bat. This non-zero velocity of the bat at the interception point provides and extra velocity to the ball resulting in an error for the attained catching point.

Another reason can be accounted to the assumption in the Subsystem 3 that the surface of the bat applies only a force normal to the surface of the bat, when the ball hits it. In reality, there are forces which act tangential to the surface of the bat, which result in more loss of energy from the ball. Moreover, this tangential impulse at the surface can make the ball spin resulting to forces acting on the ball due to magnus effect. This conclusion can be verified by looking at the Figure 4.5. It can be seen that although the hitting point almost reaches the interception point, the ball gets sent to a lesser vertical height than that of the expected point i.e. the desired catching point.

5 Conclusions and Recommendations

Following conclusions are made based on the experiments described in the previous chapter:

- The EKF algorithm converges in about 0.05 seconds. This is around 10% of the available interception time for the BIM. A better estimation algorithm or system model can help getting additional convergence speed.
- Based on the observation and opinions from experienced Cricket players, setting the velocity of the bat to zero at the interception point does not provide a good degree of realism for the system. In reality, during an event of slip catch, the bat usually possesses a non-zero velocity at the time it intercepts the ball.
- The interception point is constrained in a fixed plane. Although this makes the problem of 'how to send the ball to the desired spot' easier and thus computationally inexpensive, it suffers from the fact that it generates restrictive strokes and can result in unnatural strategies when compared with human playing
- The contact model of the bat hitting the needs improvement by taking into consideration the tangential forces which act on the ball during contact.
- The TPBVP formulates for Subsystem 3 is solved using a package written in Python. A C++ implementation of package can be faster in terms of time.
- The Panda manipulator used as the BIM, in terms of speed is too slow to intercept the ball thrown at the usual speeds during a slip catching practice session in Cricket.
- It is difficult for the thrower to throw the ball accurately in the limited working space of the Panda manipulator.
- The ball thrown in the achievable work space of the Panda arm is not intercepted accurately at the calculated interception point as a low pass filter is applied on the twist command sent to the manipulator. This leads to the ball hitting the bat around the hitting point making it difficult for the ball to be sent to the desired catching point.

Based on the above conclusions and from far-sight, following ideas have been gathered for the next iteration of KriCatch:

- The trajectory estimation of the ball can be improved by considering other non-linear estimation algorithms (example Unscented Kalman Filter/ Particle Filter) along with a better ballistic model of the ball, taking into consideration the magnus effect. This needs to be done by taking care of the criterion of computation time and resources.
- Implementation of an Artifical-Intelligence algorithm that tracks the progress and level of the user and provides catching points and feinting motions accordingly.
- The BIM must be chosen/custom made with appropriate specifications such that it has a larger working space (as compared to the one in this work), in which a thrower can easily throw the ball.
- The strategy used in Subsystem 3 for how to send the ball to desired points involves the constraint of setting the velocity of the bat to zero at the interception point. This constraint must be freed, to make the swerving action of the BIM more realistic. At the same time, it must be taken care that the new strategy used is not computationally expensive and is real-time implementable.

- The constraint of intercepting the ball in a specific fixed plane (Interception Plane) can be freed and be chosen on a criterion which makes the BIM to intercept the ball without breaking its hardware limits. Again, the solution to this problem must be real-time implementable.
- Presently, the ball is thrown directly to the BIM without bouncing it on the floor. Usually in the game of Cricket, the ball reaches the batsman after bouncing once on the floor. It is then important to integrate this bouncing action in the trajectory estimation and prediction subsystem.

A Appendix 1

A.1 Optimal Online Estimation of Linear Gaussian Systems: The Kalman Filter

Online estimation is the estimation of the present state using all the measurements that are available, i.e. all measurements up to the present time. This subsection explains the framework for the online estimation of the states of time-discrete processes. Most physical processes evolve in the continuous time. Nevertheless, it can be assumed that these systems can be described adequately by a model where the continuous time is reduced to a sequence of specific times.

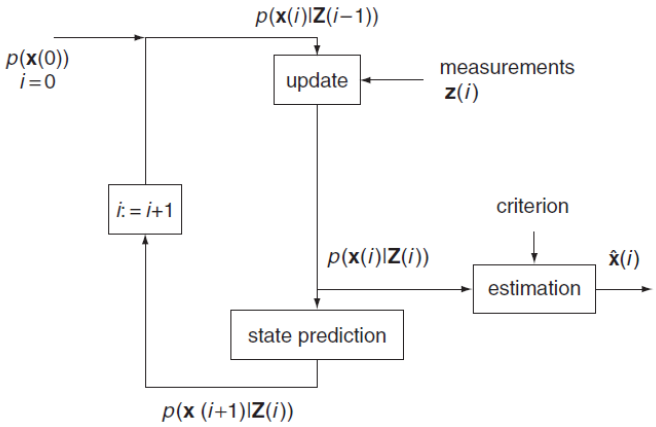

Figure A.1: An overview of online estimation of Linear Gaussian System Systems (Lei et al., 2017)

Figure A.1 presents an overview of the scheme for the online estimation of the state. The connotation of the phrase online is that for each time index i an estimate $\hat{\boldsymbol{x}}(\boldsymbol{i})$ of $\boldsymbol{x}(\boldsymbol{i})$ is produced based on $\boldsymbol{Z}(\boldsymbol{i})$, i.e. based on all measurements that are available at that time. The crux of optimal online estimation is to maintain the posterior density $\boldsymbol{p}(\boldsymbol{x}(\boldsymbol{i}) \mid \boldsymbol{Z}(\boldsymbol{i}))$ for running values of $\boldsymbol{i}$. This density captures all the available information of the current state $\boldsymbol{x}(\boldsymbol{i})$ after having observed the current measurement and all previous ones.

The state model is said to be linear if the transition from one state to the next can be expressed by a so-called linear system equation:

$$\boldsymbol{x}(\boldsymbol{i}+\mathbf{1}) = \boldsymbol{F}(\boldsymbol{i})\boldsymbol{x}(\boldsymbol{i}) + \boldsymbol{L}(\boldsymbol{i})\boldsymbol{u}(\boldsymbol{i}) + \boldsymbol{w}(\boldsymbol{i}) \tag{A.1}$$

$$\boldsymbol{y}(\boldsymbol{i}) = \boldsymbol{H}(\boldsymbol{i})\boldsymbol{x}(\boldsymbol{i}) + \boldsymbol{v}(\boldsymbol{i}) \tag{A.2}$$

$\boldsymbol{F}(\boldsymbol{i})$ is the system matrix. The vector $\boldsymbol{u}(\boldsymbol{i})$ is the control vector (input vector). $\boldsymbol{L}(\boldsymbol{i})$ is the gain matrix. $\boldsymbol{w}(\boldsymbol{i})$ is the process noise, $\boldsymbol{y}(\boldsymbol{i})$ is the measurement vector and $\boldsymbol{H}(\boldsymbol{i})$ is the measurement matrix which maps $\boldsymbol{x}(\boldsymbol{i})$ to $\boldsymbol{y}(\boldsymbol{i})$. The process noise represents the unknown influences on the system, for instance, formed by disturbances from the environment. The process noise can also represent an unknown input/control signal. Sometimes process noise is also used to take care of modelling errors. The general assumption is that the process noise is a white random sequence with normal distribution and has the following properties:

$$\boldsymbol{E}[\boldsymbol{w}(\boldsymbol{i})] = \mathbf{0} \tag{A.3}$$

$$\boldsymbol{E}[\boldsymbol{w}(\boldsymbol{i})\boldsymbol{w}^{\boldsymbol{T}}(\boldsymbol{j})] = \boldsymbol{C}_{\boldsymbol{w}}(\boldsymbol{i})\boldsymbol{\delta}(\boldsymbol{i},\boldsymbol{j}) \tag{A.4}$$

$\boldsymbol{C_w(i)}$ is the covariance matrix of $\boldsymbol{w(i)}$. Since w(i) is supposed to have a normal distribution with zero mean, $\boldsymbol{C_w(i)}$ defines the density of $\boldsymbol{w(i)}$ in full.

The update steps, i.e. the determination of $\boldsymbol{p((x(i) \mid Z(i))}$ shown in Figure A.1 are given as follows:

$$\hat{\boldsymbol{z}}(\boldsymbol{i}) = \boldsymbol{H(i)}\bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i-1}) \tag{A.5}$$

$$\boldsymbol{S(i)} = \boldsymbol{H(i)C(i \mid i-1)H^T(i)} + \boldsymbol{C_v(i)} \tag{A.6}$$

$$\boldsymbol{K(i)} = \boldsymbol{C(i \mid i-1)H^T(i)S^{-1}(i)} \tag{A.7}$$

$$\bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i}) = \bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i-1}) + \boldsymbol{K(i)}(\boldsymbol{z(i)} - \hat{\boldsymbol{z}}(\boldsymbol{i})) \tag{A.8}$$

$$\boldsymbol{C(i \mid i)} = \boldsymbol{C(i \mid i-1)} - \boldsymbol{K(i)S(i)K^T(i)} \tag{A.9}$$

The interpretation is as follows: $\hat{\boldsymbol{z}}(\boldsymbol{i})$ is the predicted measurement. It is an unbiased estimate of $\boldsymbol{z(i)}$ using all information from the past. The so-called innovation matrix $\boldsymbol{S(i)}$ represents the uncertainty of the predicted measurement. The uncertainty is due to two factors: the uncertainty of $\boldsymbol{x(i)}$ as expressed by $\boldsymbol{C(i \mid i-1)}$, and the uncertainty due to the measurement noise $\boldsymbol{v(i)}$ as expressed by $\boldsymbol{C_v(i)}$. The matrix $\boldsymbol{K(i)}$ is the Kalman gain matrix. This matrix has large, when $\boldsymbol{S(i)}$ is small and $\boldsymbol{C(i \mid i-1)H^T(i)}$ is large, that is, when the measurements are relatively accurate. When this is the case, the values in the error covariance matrix $\boldsymbol{C(i \mid i)}$ will be much smaller than $\boldsymbol{C(i \mid i-1)}$.

The prediction, i.e. the determination of $\boldsymbol{p(x(i+1) \mid Z(i))}$ given $\boldsymbol{p(x(i) \mid Z(i))}$, boils down to finding out how the expectation $\boldsymbol{x(i \mid i)}$ and the covariance matrix $\boldsymbol{C(i \mid i)}$ propagate to the next state. We thus have:

$$\bar{\boldsymbol{x}}(\boldsymbol{i+1} \mid \boldsymbol{i}) = \boldsymbol{F(i)}\bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i}) + \boldsymbol{L(i)u(i)} \tag{A.10}$$

$$\boldsymbol{C(i+1 \mid i)} = \boldsymbol{F(i)C(i \mid i)F^T(i)} + \boldsymbol{C_w(i)} \tag{A.11}$$

The above presented recursive equations are generally referred to as the discrete Kalman filter (DKF).

A.2 Sub-Optimal Solution for Online Estimation of Non-Linear Systems: The Extended Kalman Filter

The general case of nonlinear systems and nonlinear measurement functions can be given as follows:

$$\boldsymbol{x(i+1)} = \boldsymbol{f(x(i), u(i), i)} + \boldsymbol{w(i)} \tag{A.12}$$

$$\boldsymbol{y(i)} = \boldsymbol{h(x(i), i)} + \boldsymbol{v(i)} \tag{A.13}$$

The vector $\boldsymbol{f(.,.,.)}$ is a nonlinear, time variant function of the state x(i) and the control vector u(i).

Any Gaussian random vector that undergoes a linear operation retains its Gaussian distribution. A linear operator only affects the expectation and the covariance matrix of that vector. This property is the basis of the Kalman filter. It is applicable to linear-Gaussian systems, and it permits a solution that is entirely expressed in terms of expectations and covariance matrices. However, the property does not hold for nonlinear operations. In nonlinear systems, the state

vectors and the measurement vectors are not Gaussian distributed, even though the process noise and the measurement noise might be. Consequently, the expectation and the covariance matrix do not fully specify the probability density of the state vector. The question is then how to determine this non-Gaussian density, and how to represent it in an economical way. Unfortunately, no general answer exists to this question.

assuming that the nonlinearities of the system are smooth enough to allow linear or quadratic approximations, Kalman-like filters become within reach. These solutions are suboptimal since there is no guarantee that the approximations are close. An obvious way to get the approximations is by application of a Taylor series expansion of the functions.

$$\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{\epsilon}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\epsilon} + \boldsymbol{H.O.T.} \tag{A.14}$$

$$\boldsymbol{h}(\boldsymbol{x}+\boldsymbol{\epsilon}) = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{H}(\boldsymbol{x})\boldsymbol{\epsilon} + \boldsymbol{H.O.T.} \tag{A.15}$$

where, H.O.T. represent the higher order terms.

The update steps then become:

$$\hat{\boldsymbol{z}}(\boldsymbol{i}) = \boldsymbol{H}(\boldsymbol{x}(\boldsymbol{i}))\bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i}-\mathbf{1}) \tag{A.16}$$

$$\boldsymbol{S}(\boldsymbol{i}) = \boldsymbol{H}(\boldsymbol{x}(\boldsymbol{i}))\boldsymbol{C}(\boldsymbol{i} \mid \boldsymbol{i}-\mathbf{1})\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{x}(\boldsymbol{i})) + \boldsymbol{C}_{\boldsymbol{v}}(\boldsymbol{i}) \tag{A.17}$$

$$\boldsymbol{K}(\boldsymbol{i}) = \boldsymbol{C}(\boldsymbol{i} \mid \boldsymbol{i}-\mathbf{1})\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{x}(\boldsymbol{i}))\boldsymbol{S}^{-\mathbf{1}}(\boldsymbol{i}) \tag{A.18}$$

$$\bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i}) = \bar{\boldsymbol{x}}(\boldsymbol{i} \mid \boldsymbol{i}-\mathbf{1}) + \boldsymbol{K}(\boldsymbol{i})(\boldsymbol{z}(\boldsymbol{i}) - \hat{\boldsymbol{z}}(\boldsymbol{i})) \tag{A.19}$$

$$\boldsymbol{C}(\boldsymbol{i} \mid \boldsymbol{i}) = \boldsymbol{C}(\boldsymbol{i} \mid \boldsymbol{i}-\mathbf{1}) - \boldsymbol{K}(\boldsymbol{i})\boldsymbol{S}(\boldsymbol{i})\boldsymbol{K}^{\boldsymbol{T}}(\boldsymbol{i}) \tag{A.20}$$

And the prediction step can be given as follows:

$$\bar{\boldsymbol{x}}(\boldsymbol{i}+\mathbf{1} \mid \boldsymbol{i}) = \boldsymbol{f}(\boldsymbol{x}(\boldsymbol{i}), \boldsymbol{u}(\boldsymbol{i}), \boldsymbol{i}) \tag{A.21}$$

$$\boldsymbol{C}(\boldsymbol{i}+\mathbf{1} \mid \boldsymbol{i}) = \boldsymbol{F}(\boldsymbol{i})\boldsymbol{C}(\boldsymbol{i} \mid \boldsymbol{i})\boldsymbol{F}^{\boldsymbol{T}}(\boldsymbol{i}) + \boldsymbol{C}_{\boldsymbol{w}}(\boldsymbol{i}) \tag{A.22}$$

A.3 Constrained Extended Kalman Filter

In case it is known that the state estimates are bounded in certain range, it can be useful to integrate these constrains in the EKF algorithm to obtain better convergence.

The structure of the EKF does not include constraints on the states. Dan Simon et. al. (Simon and Chia, 2002) have proposed a methodology to integrate inequality constraints for state estimation which uses the Projection Approach. This methodology is briefly described ahead:

Consider the following linearized model of the plant:

$$\boldsymbol{x}(\boldsymbol{i}+\mathbf{1}) = \boldsymbol{F}(\boldsymbol{i})\boldsymbol{x}(\boldsymbol{i}) + \boldsymbol{L}(\boldsymbol{i})\boldsymbol{u}(\boldsymbol{i}) + \boldsymbol{w}(\boldsymbol{i}) \tag{A.23}$$

$$\boldsymbol{y}(\boldsymbol{i}) = \boldsymbol{H}(\boldsymbol{i})\boldsymbol{x}(\boldsymbol{i}) + \boldsymbol{v}(\boldsymbol{i}) \tag{A.24}$$

Suppose at each time instant $\boldsymbol{i}$ the states are subject to following constraints:

$$\boldsymbol{D_i x(i) = d_i} \tag{A.25}$$

Let $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{u}}$ be the state estimated at time $\boldsymbol{i}$ by an unconstrained estimator. $\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{c}}$ be the constrained estimate taking into account equation A.25. The associated covariances of estimation error are denoted $\boldsymbol{P}_{\boldsymbol{i}}^{\boldsymbol{u}}$ and $\boldsymbol{P}_{\boldsymbol{i}}^{\boldsymbol{c}}$. The Principle of Projection (Simon and Chia, 2002) is then applied, which is basically solving the constrained optimization problem:

$$\min_{\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{c}}}((\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{c}} - \hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{u}})^{\boldsymbol{T}} \boldsymbol{W_i^{-1}} (\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{c}} - \hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{u}})) \qquad \boldsymbol{s.t.} \quad \boldsymbol{D_i \hat{x}_i^c = d_i} \tag{A.26}$$

$\boldsymbol{W_i}$ is a symmetric positive definite weighting matrix. The solution is obtained through the use of the Lagrange multiplier, and summarized by the following set of equations:

$$\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{c}} = \hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{u}} + \boldsymbol{L_i}(\boldsymbol{d_i} - \boldsymbol{D_i}\hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{u}}) \tag{A.27}$$

$$\boldsymbol{P_i^c} = (\boldsymbol{I_n} - \boldsymbol{L_i D_i}) \boldsymbol{P_i^u} (\boldsymbol{I_n} - \boldsymbol{L_i D_i})^{\boldsymbol{T}} \tag{A.28}$$

$$\boldsymbol{L_i} = \boldsymbol{W_i^{-1}} \boldsymbol{D_i^T} (\boldsymbol{D_i W_i^{-1} D_i^T})^{\boldsymbol{-1}} \tag{A.29}$$

The constrained estimated state has the following properties:

Choosing $\boldsymbol{W_i} = \boldsymbol{I_n}$ results in a constrained estimate that is closer to the true state than the unconstrained estimate.

Now, suppose that at each time step $\boldsymbol{i}$, the state is subject to the following linear soft inequality constraint:

$$\boldsymbol{D_i x(i) \leq d_i} \tag{A.30}$$

A way of dealing with such a problem is using the active set method (Sircoulomb et al., 2008). It consists in testing at each time step $\boldsymbol{i}$ the scalar inequalities of A.30. For the $\boldsymbol{k^{th}}$ inequality, two scenarii can occur:

- The inequality is satisfied, and so do not have to be taken into account.
- The inequality is not satisfied. Then, an equality constraint is applied to the boundary: $\boldsymbol{D_i x(i) = d_i}$, where for any matrix $\boldsymbol{M_i}$ at time $\boldsymbol{i}$, $\boldsymbol{M_k, i}$ denotes the $\boldsymbol{k^{th}}$ row of $\boldsymbol{M_i}$

Consequently, dealing with soft inequality constraints reduces to the application at each time step of the active equality constraints.

A.4 Adaptive adjustment of Noise Covariance (C_v and C_w) in Kalman Filter

The performance of the KF is highly affected by $\boldsymbol{C_v}$ and $\boldsymbol{C_w}$. Improper choice of $\boldsymbol{C_v}$ and $\boldsymbol{C_w}$ may significantly degrade the KF's performance and even make the filter diverge. Shahrokh Akhlaghi et.al. (Akhlaghi et al., 2017) have proposed an adaptive filtering approach to adaptively estimate $\boldsymbol{C_w}$ and $\boldsymbol{C_v}$ based on *innovation and residual* to improve the dynamic state estimation accuracy of the extended Kalman filter. At the EKF's predication step, the innovation is the difference between the actual measurement and its predicted value. On the other hand, the residual is the difference between actual measurement and its estimated value using the information available.

Residual based adaptive estimation of $\boldsymbol{C_v}$ is given as follows:

$$\boldsymbol{C_v(i) = \alpha C_v(i-1) + (1-\alpha)(\epsilon(i)\epsilon(i)^T + H(i)P^-(i)H(i)^T)} \tag{A.31}$$

where, $\boldsymbol{i}$ is the current iteration number, $\boldsymbol{\alpha}$ is a forgetting factor, $\boldsymbol{\epsilon(i)}$ is the residual, $\boldsymbol{H(i)}$ is the measurement matrix and $\boldsymbol{P^-(i)}$ is the state covariance matrix predicted in $\boldsymbol{i-1}$ step.

Innovation based adaptive estimation of $\boldsymbol{C_w}$ is given as follows:

$$\boldsymbol{C_w(i) = \alpha C_w(i-1) + (1-\alpha)(K(i)d(i)d(i)^T K(i)^T)} \tag{A.32}$$

where, $\boldsymbol{d(i)}$ is the innovation, $\boldsymbol{K(i)}$ is the Kalman gain matrix.

A flow of the adaptive algorithm is shown in Figure A.2:

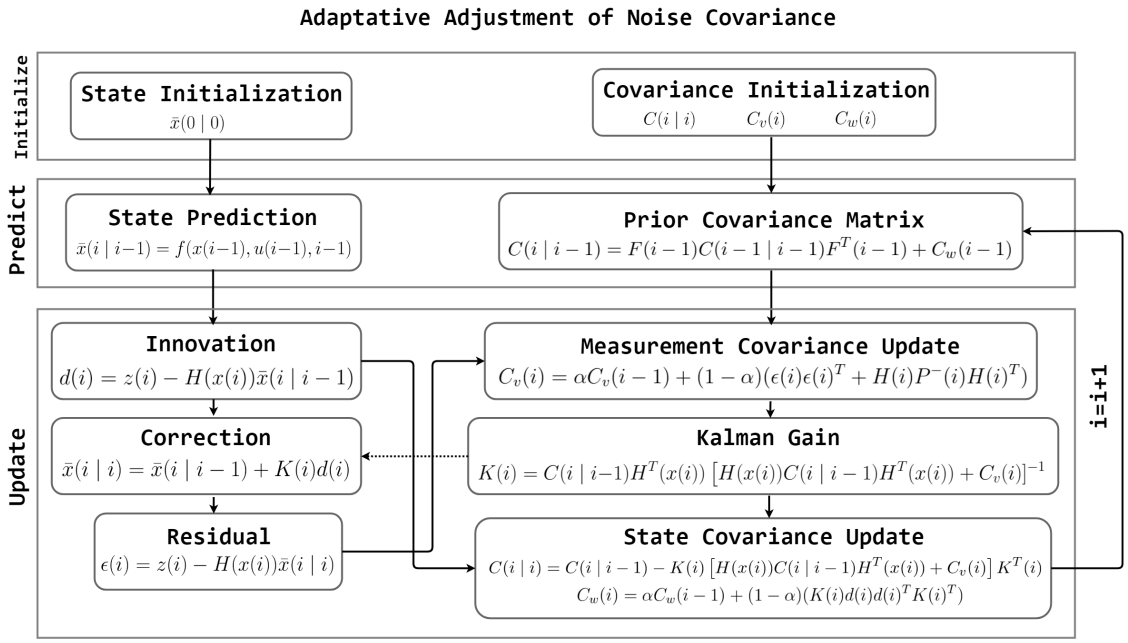

Figure A.2: Flowchart of the adaptive EKF algorithm

B Appendix 2

B.1 Calculation of co-efficient of restitution of ball

An experimental method to approximately calculate the co-efficient of restitution as is explained in (Briggs, 1945) is used in this work. During the experiment, the ball is thrown from a specific height (***H***) onto the bat of the BIM, and the height of rebound (***h***) was measured (See Figure B.1). The position of the ball during the experimentation is measured using the OptiTrack system.

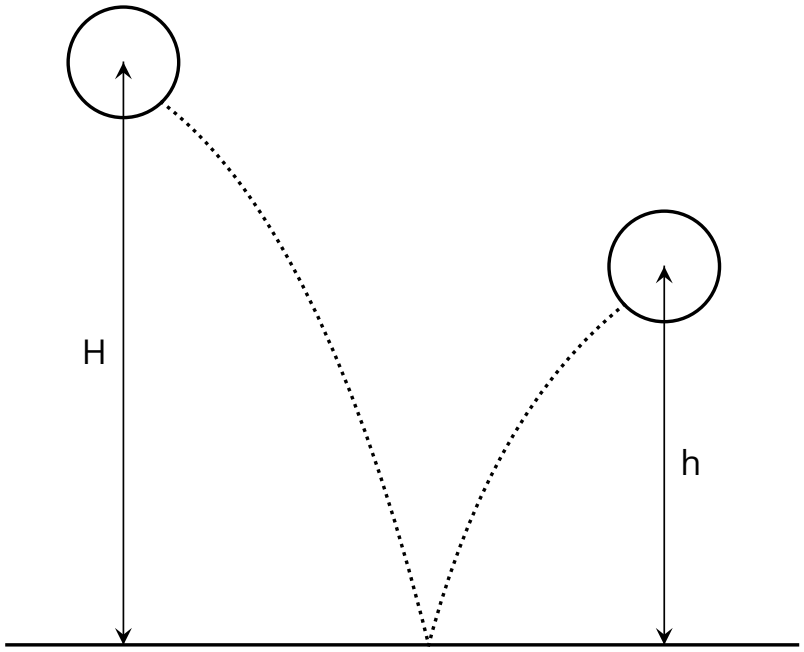

Figure B.1: Bouncing experiment to calculate the co-efficient of restitution of the ball.

The co-efficient of restitution $\boldsymbol{\beta}$ is calculated as follows:

$$\boldsymbol{\beta} = \sqrt{\frac{e^{\frac{-2k_d h}{m}} - 1}{1 - e^{\frac{-2k_d H}{m}}}} \tag{B.1}$$

where, $\boldsymbol{k_d}$ is the drag co-efficient of the ball calculated through estimation experiments from Subsystem 1, ***H*** is the height from which ball is thrown, ***h*** is the rebound height and ***m*** is the mass of the ball.

Bibliography

Akhlaghi, S., N. Zhou and Z. Huang (2017), Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation, *arXiv preprint arXiv:1702.00884*.

Almagbile, A., J. Wang and W. Ding (2010), Evaluating the performances of adaptive Kalman filter methods in GPS/INS integration, **vol. 9**, no.1, pp. 33–40.

Anderson, B. D. and J. B. Moore (1979), Optimal filtering, *Englewood Cliffs*, **vol. 21**, pp. 22–95.

Ascher, U. M., R. M. Mattheij and R. D. Russell (1994), *Numerical solution of boundary value problems for ordinary differential equations*, volume 13, Siam.

BOLA (2015), TrueMan by BOLA, http://www.bola.co.uk/TrueMan.html, [Online; accessed 12-August-2018].

Brescianini, D. and R. D'Andrea (2018), Computationally Efficient Trajectory Generation for Fully Actuated Multirotor Vehicles, *IEEE Transactions on Robotics*.

Briggs, L. J. (1945), *Methods for measuring the coefficient of restitution and the spin of a ball*, US Department of Commerce, National Bureau of Standards.

Chudinov, P. (2005), The envelope of projectile trajectories in midair, *arXiv preprint physics/0511059*.

Cigliano, P., V. Lippiello, F. Ruggiero and B. Siciliano (2015), Robotic ball catching with an eye-in-hand single-camera system, **vol. 23**, no.5, pp. 1657–1671.

Franka Emika GmbH (2018a), franka*ros*, , [Online; accessed 12-August-2018].

Franka Emika GmbH (2018b), Panda, https://www.franka.de/, [Online; accessed 12-August-2018].

John Salvatier (2012), Python package for solving two-point boundary value problems. https://pypi.org/project/scikits.bvp_solver/

Kalman, R. E. (1960), A new approach to linear filtering and prediction problems, **vol. 82**, no.1, pp. 35–45.

Kathrin Gräve, A. B. (2018), mocap*optitrack*, , [Online; accessed 30-August-2018].

Kim, S., A. Shukla and A. Billard (2014), Catching objects in flight, **vol. 30**, no.EPFL-ARTICLE-198748.

Koç, O., G. Maeda and J. Peters (2018), Online optimal trajectory generation for robot table tennis, *Robotics and Autonomous Systems*, **vol. 105**, pp. 121–137.

Lei, B., G. Xu, M. Feng, F. van der Heijden, Y. Zou, D. de Ridder and D. M. Tax (2017), *Classification, parameter estimation and state estimation: an engineering approach using MATLAB*, John Wiley & Sons.

Li, H., H. Wu, L. Lou, K. Kühnlenz and O. Ravn (2012), Ping-pong robotics with high-speed vision system, in *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*, IEEE, pp. 106–111.

McCormick, B. W. (1995), *Aerodynamics, aeronautics, and flight mechanics*, volume 2, Wiley New York.

Miyazaki, F., M. Matsushima and M. Takeuchi (2006), Learning to dynamically manipulate: A table tennis robot controls a ball and rallies with a human being, in *Advances in Robot Control*, Springer, pp. 317–341.

Müller, M., S. Lupashin and R. D'Andrea (2011), Quadrocopter ball juggling, in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE, pp. 5113–5120.

Point, N. (2011), Optitrack, *Natural Point, Inc.,[Online]. Available: http://www. naturalpoint. com/optitrack/.[Accessed 22 2 2014]*.

ProBatter Sports (2011), ProBatter PX2 Cricket Video Simulator, http://probatter.com/cricket/, [Online; accessed 08-June-2018].

Quigley, M., K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng (2009), ROS: an open-source Robot Operating System, in *ICRA workshop on open source software*, volume 3, Kobe, Japan, p. 5.

Sato, K., K. Watanabe, S. Mizuno, M. Manabe, H. Yano and H. Iwata (2017), Development of a block machine for volleyball attack training, in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, pp. 1036–1041.

Simon, D. and T. L. Chia (2002), Kalman filtering with state equality constraints, **vol. 38**, no.1, pp. 128–136.

Sircoulomb, V., G. Hoblos, H. Chafouk and J. Ragot (2008), State estimation under nonlinear state inequality constraints. A tracking application, in *Control and Automation, 2008 16th Mediterranean Conference on*, IEEE, pp. 1669–1674.

Wikipedia (2018a), Boundary value problem — Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Boundary_value_problem, [Online; accessed 08-June-2018].

Wikipedia (2018b), Cricket — Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Cricket&oldid=844653304, [Online; accessed 08-June-2018].

Wikipedia (2018c), Iterative design — Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/wiki/Iterative_design, [Online; accessed 29-August-2018].

Zhang, Y., Y. Zhao, R. Xiong, Y. Wang, J. Wang and J. Chu (2014), Spin observation and trajectory prediction of a ping-pong ball, in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, pp. 4108–4114.