Fast Interceptor of a Dynamic Object

by

Sergio A. Cafarelli

B.S.E. Electrical Engineering & Computer Science
Princeton University (1987)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author .....
Department of Aeronautics and Astronautics
August 20, 2009

Certified by .....
Jonathan P. How
Professor
Thesis Supervisor

Accepted by .....
David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Fast Interceptor of a Dynamic Object

by

Sergio A. Cafarelli

Submitted to the Department of Aeronautics and Astronautics
on August 20, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis presents a path planning and control strategy that enables an unmanned non-holonomic vehicle to intercept a fast moving object. The path planning is performed under model uncertainty, with respect to the vehicle's maneuverability, as well as uncertainty in the estimation of the object's future trajectory and position. This problem involves the tracking of the dynamic object in a cluttered environment and the accurate estimation of its future position in the presence of noisy measurements. The ground vehicle (interceptor) is required to intercept the dynamic object at a predicted (catch) location in a finite amount of time. This time restriction presents quite a challenge given the inherent limitation in the vehicle's steering and maneuverability. The solution strategy is divided into three sub-problems: 1) prediction, 2) path planning and 3) control. The prediction of the parameters that describe the dynamic's object in space is accomplished via Kalman Filtering which, in conjunction with an impact predictor, provide the waypoints needed to construct a reference path that will place the interceptor on a collision course with the dynamic object (target.) A pure pursuit algorithm was used to steer the interceptor along a reference trajectory, which was designed to make the vehicle engage the dynamic object on a near tail-on aspect. In the endgame, the pure pursuit algorithm was modified to ensure arrival to the catch point while a position controller was added to ensure timely arrival to the predicted catch location. The problem statement was then augmented to include obstacle avoidance. The dynamic object was required to navigate around fixed obstacles in order to catch the dynamic object. Results will show that the proposed strategy performed very well in the absence of obstacles and was well suited to handle the maneuverability constraints of the non-holonomic vehicle. Results also will show that, with minor modifications to the path planner, the interceptor successfully managed to avoid obstacles and catch the dynamic object although at a slightly lower success rate. The proposed solution was first demonstrated in simulation and then tested using MIT's RAVEN testbed.

Thesis Supervisor: Jonathan P. How
Title: Professor

Acknowledgments

I would like to thank Prof. Jonathan How for extending me the opportunity to be part of the Aerospace Controls Laboratory and for his support, patience and interest in completing this work. I also would like to thank Prof. John Hansman for his help, support and guidance throughout my short time at MIT. I would like to thank Mr. Vishnu Desaraju for his assistance, log hours at the lab, and all of his innumerable contributions to this project. I also would like to thank Buddy Michini for his help in developing the infrastructure needed for this project, as well as Josh Redding for his help in understanding all aspects of the VICON and Motion Analysis Systems. I would like to extend my gratitude to Kenneth Lee, Karl Kulling, Cameron Fraser, Frant Sobolic, Brandon Ludders, and Justin Teo whose help and advice were instrumental in developing a working vehicle model. In addition, I would like to thank my other colleagues in the Aerospace Controls Laboratory who were always willing to lend a hand, as well as Mrs. Kathryn Fischer, Mrs. Barbara Lechner and Mrs. Marie Stuppard for all their support and encouragement throughout this endeavor.

To my wife, Deborah Cafarelli, I want to express my deepest gratitude for her constant moral support and my deepest respect for being gracious and patient in the face of my never ending stress. I would like to thank my daughter Sophia, whose smiles, hugs and kisses gave me strength in times of weakness; my in-laws, Mr. Bob and Bobbie Dean, for holding up the home front during my absence; and my mother, Blanca, for her never ceasing prayers. Finally, all glory be given to my Lord and saviour, Jesus Christ, from whom all blessings flow. To Him, I dedicate any and all success I have had throughout my life.

"We are not to think lowly of ourselves when it is empirically inappropriate to do so, nor are we to think more highly of ourselves than is warranted." Table Talk on Rom. 12:3-8

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Guidance of moving objects is a highly researched topic in aerospace and defense industries. In very general terms, the goal of guidance is to control the path of an object in order for it to reach a particular destination. Of interest is the missile intercept problem, in which the purpose of guidance is to reach a desired target by ensuring that the position of the intercepting object coincides with that of the target. Note that there is no explicit time requirement for the missile to intercept the target. Of course, the only implicit time requirement is for the missile to intercept the target before it loses its ability to remain airborne. In the endgame, target interception is achieved either through a direct hit (kinematic kill) or by secondary methods such as explosive devices detonated by proximity fuses. The former method implies a zero miss distance while the latter translates to a small miss distance. Though both are considered successful engagements, this work will focus on the methods by which to achieve a direct hit.

Missile guidance laws can be divided into homing and external guidance. Homing guidance is associated with missiles that are capable of sensing the target through active, semi-active or passive means. During active homing, the missile radiates the target and uses the target's returned signal for tracking, guidance and control. In semi-active homing, the target is illuminated by an external source and the missile uses the returned signal from the target for intercept. Finally, in passive homing, the missile senses radiation that is naturally emitted by the target (e.g. infrared

radiation, sounds waves) in order to track and intercept the target.

External Guidance can be divided into three categories: commanded, line-of-sight and pursuit. Command guided missiles receive guidance instructions from external sources such as ground stations or integrated air defense systems. Line-of-sight guidance laws (e.g. beam rider and commanded-line-of-sight) use an external source to steer and guide the missile to its intended target. During pursuit guidance, the missile flies directly to the target at all times. This can be achieved by having the missile's longitudinal axis pointed at the target or by having the missile's velocity vector pointed at the target at all times. The former is called attitude pursuit while the latter is called velocity pursuit. In either case, most pure pursuit engagements end in a rear intercept. There are many implementations of pursuit guidance, some of which include lead pursuit, proportional navigation and pure collision to name a few.

As mentioned before, the missile intercept problem is concerned with the eventual kill of the target. This thesis is more interested in the retrieval of the target which imposes the explicit time constraint of arrival to the intercept point at the right time and with enough accuracy to allow the target to fall within the footprint of a retrieving device.

1.1 Objectives

This thesis examines the intercept problem as it applies to the guidance and control of a non-holonomic vehicle whose objective is to intercept and retrieve a dynamic object (i.e. target). In this application, the dynamic object is a golf ball whose motion is considered quasi deterministic in the sense that, in between bounces, it is possible to accurately estimate the parameters that define its motion in space and time. The element of uncertainty in the ball's trajectory is introduced every time the ball impacts the ground, as it is very likely that such impacts produce substantial changes in the ball's direction of motion. The intercept problem is further complicated by the explicit time constraint imposed by the need to retrieve the ball. In other words, it is not

sufficient that the vehicle is able to plot a collision course with the target (which implies that the ball may impact anywhere in footprint of the vehicle). The problem requires that the vehicle's catching device is placed accurately underneath the ball in order to catch it.

The non-holonomic constraints of the ground vehicle, the estimation uncertainty of the ball's trajectory, and the time constraint associated with the retrieval of the target make this problem very challenging. The primary goal of this thesis is to find a repeatable solution to the problem of intercept and retrieval of a dynamic object in an obstacle free environment. The secondary goal of this thesis is to successfully intercept and retrieve the dynamic object in the presence of obstacles.

The work presented in this thesis follows the work of [14] to develop an implementation of a pure pursuit controller and the work of [9] to develop a path planing strategy for obstacle avoidance.

1.2 Thesis Overview

Chapter 2 provides a brief discussion on the infrastructure required for target tracking and autonomous vehicle control. It then discusses the hardware requirements, hardware selection and software architecture.

Chapter 3 provides a brief explanation on the dynamic object's selection process followed by a discussion on the strategy for detection and tracking of the object in the presence of clutter. It then discusses the use of Kalman filtering to estimate the parameters that describe the dynamic object's motion in 3-D space and the algorithm used to predict the future positions of the dynamic object, to include the intercept location.

Chapter 4 explains the overall path planning and endgame strategy required to initially position the interceptor on a collision path with the target and, in the endgame, maximize the interceptor's ability to catch the dynamic object. It then discussed the control strategy required to track the prescribed path and place the interceptor at the required intercept point at the precise time to enable it to catch the dynamic object.

The steering controller is based on a pure pursuit path tracking algorithm and the position controller makes use of both feedforward and feedback to ensure the timely arrival of the vehicle to the desired goal location.

Chapter 5 discusses the path planning modifications needed to avoid a stationary obstacle whose location is known a-priori and readily available to the path planner. The strategy builds on some of the key ideas presented by L.E. Dubins [9] in his work on path length minimization under constrained motion with some modifications that align with the pure pursuit path planing strategy.

Chapter 6 first presents system level results in an obstacle free environment. Under this condition, the engagements are categorized as Type I or Type II engagements, where the difference is purely based on the vehicle's initial position and heading with respect to the ball's initial trajectory. Lastly, this chapter presents the results for engagements in which an obstacle was present during the intercept engagement.

Chapter 2

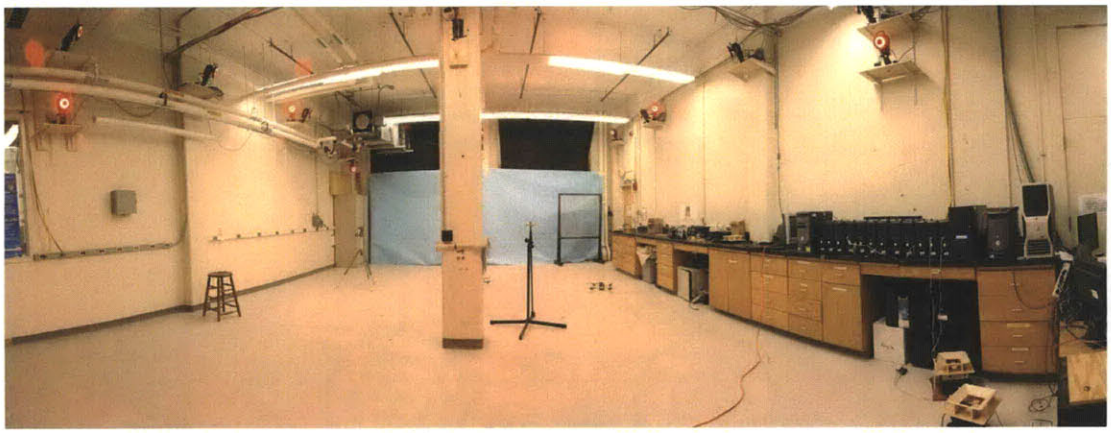
Hardware Description

2.1 Background

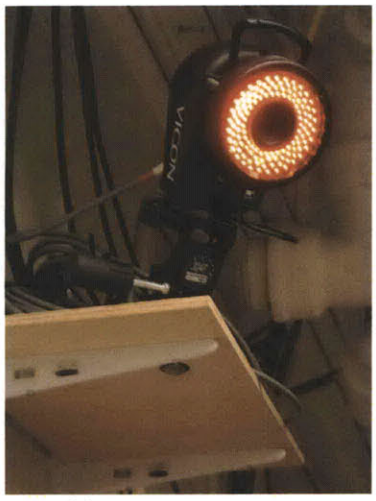
2.1.1 RAVEN

The path planning, navigation and control strategies that will be described in subsequent chapters were implemented in the Aerospace Control Laboratory (ACL) Real-time indoor Autonomous Vehicle test ENvironment (RAVEN) [11], a multi-vehicle testbed allowing for rapid-prototyping of high-level mission management and path planning algorithms (see Figure 2-1(a)). This capability is achieved by using a very accurate Vicon MX motion capture system [27] to produce high bandwidth state estimates of numerous aerial and ground vehicles, as well as in-house vehicle controllers to provide low-level control and stabilization of the vehicle hardware. RAVEN also hosts another tracking system called Raptor-4 Digital Real Time System built by Motion Analysis® [26]. This particular tracking system consists of 12 Raptor-4 Digital Cameras and *Cortex* software, which captures vehicle motion with very good accuracy. The system also includes a suite of tools called *Calcium* that creates a skeletal structure of the markers that define a particular object. The creation of unique skeletal structures allows for the tracking of multiple objects.

VICON and Motion Analysis are optical motion capture systems (see Figs. 2-1(b) and 2-1(c)) that track the position of reflective (fluorescent) markers in space. An ob-



(a) RAVEN Flight Facility



(b) Vicon MX Camera

(c) MA Raptor-4 Camera

Figure 2-1: RAVEN Elements

ject is identified by a unique clustering of three or more markers, while unassociated dots (e.g. dots that do not belong to any particular object) are either transmitted or rejected by the object processing software. Apart from the obvious tracking capabilities of these optical systems, they can also be used to emulate on-board sensors suites such as speed indicators or more complicated INS systems. The data measured by the motion capture system is transmitted as UDP packets via ethernet connections to the vehicle's ground based control computer, which in turn commands the vehicle through an XBeePro RC transmitter. Both motion capture systems operate at a nominal sampling rate of 100Hz with millimeter accuracy.

2.1.2 Interceptor

The interceptor chosen for this particular application is the Duratrax® Mini Quake remote control vehicle shown in Fig. 2-2. The vehicle is 9.5" in length, 7.4" in width and 4.7" in height with a ground clearance of 1.25". The speed controller is an ESC1000 Micro Electronic speed controller with forward and reverse (but no braking capability). It includes a Futaba® 2-channel RC radio with an RX-100 receiver as well as an SX-100 servo. The RX-100 receiver is replaced with a Robostix that transmits steering and throttle commands to the vehicle. The car is powered by a 600mAh NiCd battery, whose battery life is not very extensive, and will drain quickly if the vehicle is subjected to constant aggressive maneuvering.

Figure 2-2: Interceptor with Catching Device

Remote control operations are done via Digi's XBee-Pro® 802.5.4 OEM RF module, which is connected to a robostix that decodes the steering and throttle serial message sent by the host computer. The (crude) catching device is an unassuming plastic receptacle attached to the top of the vehicle. The opening of the receptacle is 9cm in diameter (just smaller than twice the size of a golf ball.)

Robostix

The Robostix is an expansion board for the Gumstix motherboard that hosts an AVR ATmega128 microcontroller unit. Though created as an expansion board, it can also be used in a stand alone configuration. The Robostix board has a large number of IO

pins which can be used to control external devices such as sensors and actuators. Fig. 2-3 shows the Robostix connected to the XBee transmitter, the ESC speed controller, and the steering servo mechanism. As mentioned before, the Robostix decodes the serial message sent by the host computer into throttle and steering commands. The serial message was limited to 3 bytes of information that contains a start message string followed by the desired throttle and steering command. By reducing the size of the message sent to the Robostix, the response time of the unit was increased from 20 to 100Hz.

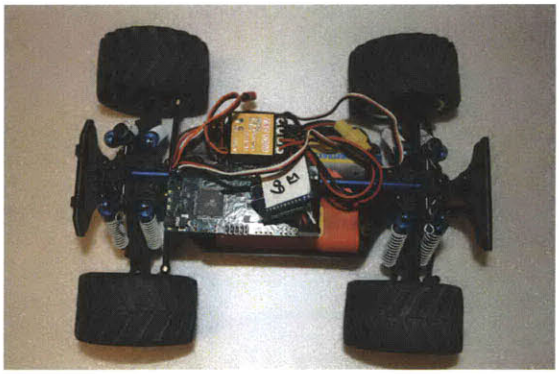


Figure 2-3: Interceptor Hardware

2.2 Software Implementation

The software for the simulation and testbed implementations was written in the C, C++, and MATLAB programming languages. The software consisted of a master program that controlled four asynchronous threads. All the threads shared information stored in a Master Workspace that contained all relevant variables common to each thread. In order to prevent conflict when several threads access the same variable in the workspace (a common problem when using threads), local variables were created to handle all required computations and updated at the end of the thread. The four threads implemented are:

1. Path Planner: this thread contains the Kalman filter estimator, the impact

predictor, the global waypoint assignments and the target tracking modules.

2. Controller: this thread contains the pure pursuit steering controller, the position controller and obstacle avoidance modules.

3. Communications: this thread managed the communications with the tracking system, the vehicle computer and external data acquisition systems.

4. Diagnostics Port: this thread provided all the data required for data reduction and analysis.

The software architecture is shown in Fig. 2-4 below. All threads run at 100Hz, except for the Diagnostic Port thread that runs at 200Hz (in order to capture all tracking data which can provide updates at rates up to 120Hz.)

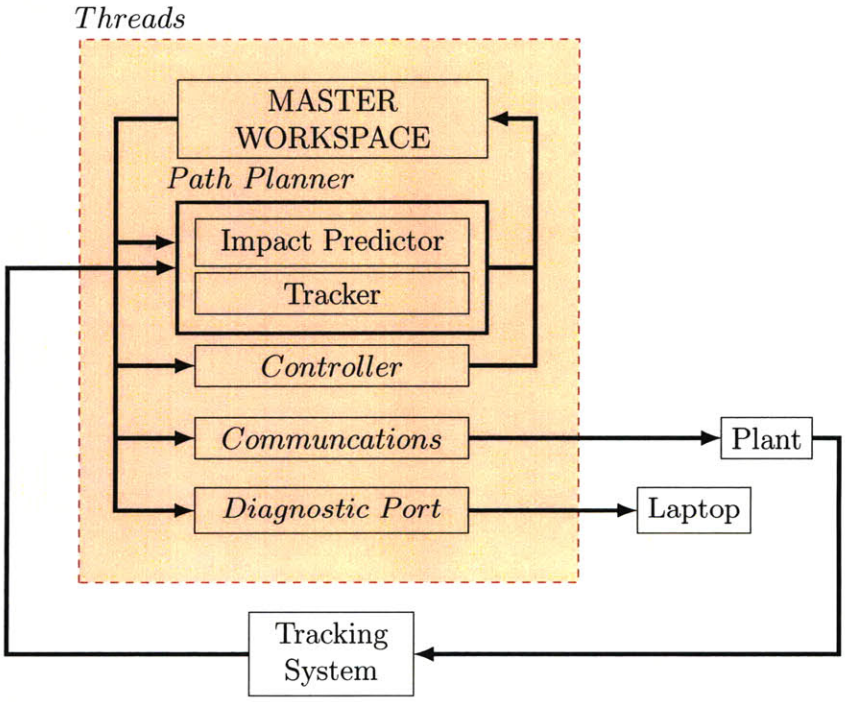


Figure 2-4: Software Architecture

.

Chapter 3

Impact Prediction

3.1 Introduction

The task of intercepting a dynamic object hinges on the ability to detect, track and accurately predict an intercept location. As mentioned in Chapter 2 the optical tracking system provides all critical information to track the dynamic object (and interceptor.) This tracking must be performed in the presence of clutter defined as extraneous markers or objects that are present during the engagement. Furthermore, most optical systems do suffer from various optical anomalies (e.g. ghosting) that contribute to errors in tracking and estimation. The goal of this chapter is to discuss the selection of the dynamic object, the tracking strategy and the rejection of clutter. The last section will cover the impact prediction algorithm and show some results regarding the accuracy of the predictor.

3.2 Dynamic Object Selection

The selection of the dynamic object is crucial if the vehicle is to have a reasonable chance to get under the object and catch it. The ideal object should be small to prevent potential ghosting, light so that it does not damage the vehicle upon impact (catch) and should have a high coefficient of restitution in order to maximize the time of flight in between bounces. Various objects were selected as potential candidates

Figure 3-1: Dynamic Object Candidates

(see Fig. 3-1) but only two were chosen for experimentation, namely: a ping pong ball and a golf ball. The other objects were rejected because they were either too heavy or, after attaching reflective tape to the surface, their bounce characteristics were dramatically affected (as was the case of the small rubber ball shown in the leftmost location of the figure) The two selected objects were used to test the robustness of the impact predictor. The ping pong ball is light (2.7 grams) and small (40 mm diameter) with relatively high coefficient of restitution (0.88). The golf ball is heavier (45.93 grams) and slightly larger (42.67 mm) with high coefficient of restitution (0.83). Unfortunately, after attaching reflective tape to the ping-pong ball, the coefficient of restitution was reduced to ≈ 0.7. Table 3.1 includes the weight and size of all candidate objects.

Table 3.1: Candidate Dynamic Object Measurements

Metric	Rubber Ball	Ping Pong Ball	Golf Ball	Large Rubber Ball	Racket Ball	Synthetic Ball
Weight (g)	<2	2.7	45.93	60	40	90
Diameter (mm)	28.45	39.60	45.9	48.39	55.88	56.52

Test results showed (see Table 3.2) that the golf ball was able to maintain its coefficient of restitution and provided the longest times of flight between bounces. Therefore, the dynamic object of choice for this application was a golf ball.

Table 3.2: Object Time of Flight

Bounce	Ping Pong Ball TOF	Golf Ball TOF
2	0.78	1.23
3	0.56	1.04
4	0.41	0.88

3.3 Tracking

The dynamic object must be tracked in the presence of clutter, which is composed of extraneous individual markers or cluster of markers (e.g. other objects) and must be robust with respect to optical ghosting or possible data dropouts. The selected tracking strategy is one used most commonly in radar and passive systems called track-before-detect [10, 16]. For this application, the basic concept is to track all potential objects for a period of time until enough information has been collected to determine which tracks are associated with a dynamic object. Since the dynamic object is a bouncing (golf) ball, persistent positive motion in the +z direction is a reasonable metric to determine persistence of motion. During the tracking process, a pre-history buffer is maintained for all potential tracks, which contains past information on the objects' trajectory. This information is used to perform an initial estimate of parameters that describe the ball's motion in each axis. This section will explain in detail the tacking strategy.

3.3.1 Data Association

The data association algorithm selected for tracking was based on nearest neighbor[6]. At the beginning of a run all detected markers within a data frame will form individual tracks, $\mathcal{T}$. After every subsequent frame, all markers that lie in the vicinity of a particular track will be associated with that track. The rest of the markers are either associated with other pre-existing tracks or may initiate new tracks. This process is illustrated in Fig. 3-2 where the i^{th} marker represents that last known location

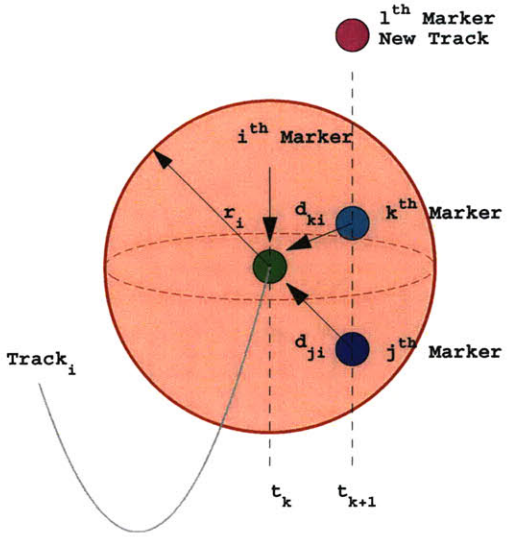


Figure 3-2: Data Association

associated with track $\mathcal{T}_i$ at time t_k, and r_i represents the region of "proximity" of any new markers. A new data frame is collected at time t_{k+1} containing 3 new markers. The j^{th} & k^{th} markers are associated with track $\mathcal{T}_i$, since they are located in the proximity region (e.g. d_{ki} & $d_{ji} < r_i$), while the l^{th} marker forms a new track. One crucial point that must be mentioned is the possibility that a particular track may not be updated for a number of frames. When this happens, the proximity region is increased in direct proportion to the time elapsed since the last update. This adjustment allows for additional marker motion in case a marker is obscured, dropped or not reported by the tracking system.

Fig. 3-2 also illustrates an interesting but very frequent situation where the optical tracking systems provides multiple measurements associated with a single marker (a phenomenon known as ghosting). Traditional nearest neighbor algorithms select the single data point that is physically closest[1] to the previous point. This particular selection process may lead to tracking errors that will ultimately affect the performance of the impact predictor. Alternatively, the proposed data association algorithm computes track location (e.g. coordinates) by averaging the position of all candidate

[1]Or applies some heuristic to determine proximity

Algorithm 1 Nearest Neighbor Data Association

$\mathcal{T}_i \leftarrow x_i$
for $k \in New\_Markers$ **do**
  $\delta r_k \leftarrow ||[x_k\ y_k\ z_k] - [x_i\ y_i\ z_i]||_2$
  if $\delta r_k < r_i$ **then**
    $\mathcal{T}_i \leftarrow x_k$
  else
    $\mathcal{T}_k \leftarrow x_k$ *(assign a new track or associate with pre-existing track)*
  end if
end for

markers associated with a particular track. Mathematically,

$$\mathcal{T}_i(t_{k+1}) = \frac{1}{N}\sum_{j=1}^{N} \mathbf{P}_j(t_{k+1}) \tag{3.1}$$

where $\mathcal{T}_i(t_{k+1})$ contains the updated coordinates of the i^{th} track, $\mathbf{P}_j(t_{k+1}) = [x_j, y_j, z_j]$ represent the coordinates of the j^{th} marker at time t_{k+1}, and N is the total number of markers associated with track $\mathcal{T}_i$. The averaging process in Eq. 3.1 will clearly introduce errors in track position, but the error will be less than if the wrong ghost point is selected randomly during track evolution. Fortunately, this situation (the averaging of data points) fits well within the context of Kalman Filtering in that one could assign more uncertainty to instances where track information contains multiple markers. To this end, the algorithm maintains a metric, ϵ, that measures the largest deviation between the reported track position and its associated (candidate) markers.

$$\epsilon = \max ||Track_i(t_{k+1}) - P_j||_2 \quad j = \{1, N\} \tag{3.2}$$

This error metric is later used by the Kalman Filter to adjust the uncertainty in the observation model.

3.3.2 Motion Detection

The track management process explained in the previous section provides a list of potential tracks that need to be processed to determine which track is associated

with a dynamic object. The motion detection algorithm assumes that the dynamic object will initially move upwards (e.g. $+z$ direction), and this vertical displacement is used in a cost function to determine the track's confidence level as a function of time [7]. The cost function, J, has an initial value $J_0 = 5$ and penalizes downward or pure horizontal motion as follows:

$$J_k = \max\left(0, J_{k-1} + \frac{z_k - z_{k-1} - \alpha\Delta t}{|z_k - z_{k-1} - \alpha\Delta t|}\right) \tag{3.3}$$

where α is a tuning parameter that reflects the minimum amount of vertical displacement that the dynamic object is required to travel in an interval Δt. Notice that Eq. (3.3) is lower-bounded at 0 to prevent accumulating very large negative values prior to the toss of a ball (the dynamic object)[2]. In our application the value $\alpha = 2.0$ provided very consistent results when tracking various dynamic objects. The confidence level is updated after every measurement, and when it reaches a value of $J = 15$ then the track is associated with the dynamic object that must be intercepted and the confidence level is no longer updated. The number of samples needed to successfully establish a track will very depending on the evolution of the cost function (Eq. 3.3). However, results showed that a minimum of 8 consecutive samples with continuous vertical displacement are needed for track initiation. This minimum number of samples are the basis for determining the size of a buffer that will be referred to as prehistory. This buffer contains the dynamic objects' Time-Space-Position-Information (TSPI)[3]. The need for this data will be explained in Section 3.3.3.

Before leaving this section it is important to mention that the motivation for developing a motion detection algorithm was to improve tracking of the dynamic object in the presence of clutter; to robustly handle ghosting and data dropouts, and to track multiple dynamic objects. The rejection of clutter is necessary because the RAVEN room hosts many vehicles and objects most of which are fitted with reflective markers. Furthermore, it is not uncommon for the tracking system to

[2]This may happen if the ball is lowered before been tossed - a very natural movement for an individual wanting to toss a ball with an upward arc

[3]TSPI refers to the (x, y, z) coordinates of an object as a function of time

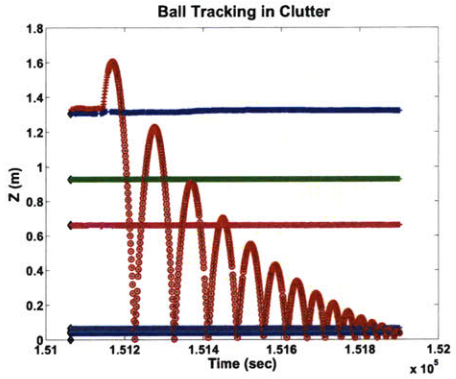
Algorithm 2 Motion Detection

$J_0 \leftarrow 5$
$z_k \leftarrow \mathcal{T}_i(z_k)$
$J_k = \max\left(0, J_{k-1} + \frac{z_k - z_{k-1} - \alpha\Delta t}{|z_k - z_{k-1} - \alpha\Delta t|}\right)$
if $J_k > 15$ **then**
  Dynamic Object $\leftarrow \mathcal{T}_i$
end if

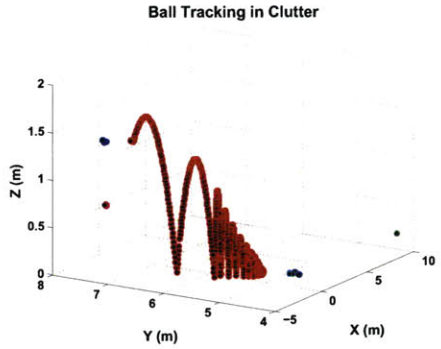
observe unwanted reflections that may persist for the duration of a run. These random reflections contain inherent motion in all axes which, if not properly filtered, may cause track association problems. This situation is depicted in Fig. 3-3(a) where a ball (shown in red) is embedded in clutter that contains: 2 stationary markers (shown in green and magenta), a vehicle (shown in purple and cyan) and unwanted reflections (shown in dark blue). When the tracking data stream is processed by the motion detection algorithm six (6) distinct tracks are formed, as shown in Fig. 3-3(b), but the information associated with the ball is not corrupted. Two additional experiments were conducted to determine if the motion detection algorithm is capable of tracking multiple dynamic objects. In the first experiment, four balls were released one after the other in various directions. The tracking algorithm was able to properly identify and track each one of the balls even through bounces off the wall and other objects. The results are shown in Fig. 3-3(c). For the final experiment, three balls were held together and then released all at once (e.g. grapeshot). At the beginning of the engagement, a single track was formed (green line in Fig. 3-3(d)) due to the fact that all the markers were in close proximity of each other. As the markers begin to separate, new tracks are formed (red, blue and cyan tracks) and each distinct track was maintained for the duration of each marker's motion.

3.3.3 Track Prehistory

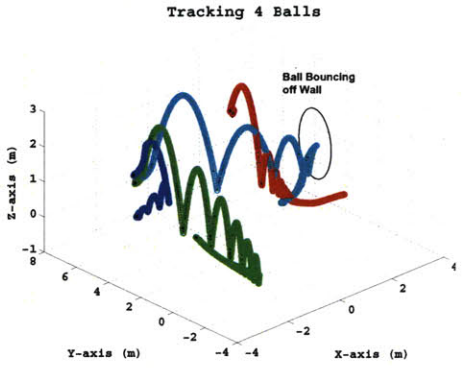
During the tracking process, each track contains both a confidence level computed according to the metric described in the previous section, and a buffer called track pre-history. Track pre-history is a circular First-In-First-Out (FIFO) buffer that contains



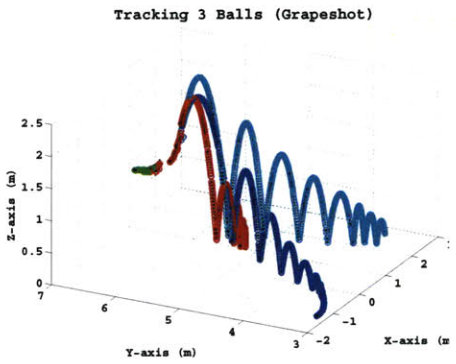
(a) Single Ball Embedded in Clutter



(b) Tracking of a Single Ball in Clutter: x, y, z vs. time



(c) Tracking of Several Balls Released at Different Times



(d) Tracking of Several Balls Released at Once

Figure 3-3: Robustness of Tracking Algorithm

time stamped position information (e.g. t, x, y, z) of the last eight track samples. Prehistory data is used to obtain preliminary estimates of the parameters that describe the ball's trajectory on each axis. Though this will be discussed in more detail in Section 3.4, it is worth mentioning that the trajectory of the dynamic object as a function of time is assumed to be linear in the x & y axis and quadratic in the z axis. Define the state vector $\mathbf{x} = \begin{bmatrix} a_1 & a_2 & a_3 & b_1 & b_2 & c_1 & c_2 \end{bmatrix}^T$, as the coefficients of the quadratic and linear time trajectories in z, x and y, respectively. Furthermore define

the matrices $\mathbf{M_1}$, $\mathbf{M_2}$, & $\mathbf{M}$ as:

$$\mathbf{M_1} = \begin{bmatrix} t_0^2 & t_0 & 1 \\ t_1^2 & t_1 & 1 \\ \vdots & \vdots & \vdots \\ t_7^2 & t_7 & 1 \end{bmatrix}, \mathbf{M_2} = \begin{bmatrix} t_0 & 1 \\ t_1 & 1 \\ \vdots & \vdots \\ t_7 & 1 \end{bmatrix}, \mathbf{M} = \begin{bmatrix} \mathbf{M_1} & \mathbf{0}_{8x2} & \mathbf{0}_{8x2} \\ \mathbf{0}_{8x3} & \mathbf{M_2} & \mathbf{0}_{8x2} \\ \mathbf{0}_{8x3} & \mathbf{0}_{8x2} & \mathbf{M_2} \end{bmatrix}$$

and the measurement vectors $\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}$ and the matrix $\mathbf{B}$ as:

$$\mathbf{b}_1 = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_7 \end{bmatrix} \mathbf{b}_2 = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix} \mathbf{b}_3 = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_7 \end{bmatrix} \mathbf{B} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ \mathbf{b_3} \end{bmatrix}$$

The desired parameters are found by solving the over-determined system matrix equation $\mathbf{Mx} = \mathbf{B}$. whose solution is simply

$$\mathbf{x} = (\mathbf{M^T M})^{-1} \mathbf{M}^T \mathbf{b}$$

The vector $\mathbf{x}$ is the best estimate (in the least squared sense) of the parameters that describe the motion of the ball in all three axes. These estimates are now used as the initial values, or a-priori estimates ($\mathbf{x}_0^- = \mathbf{x}$), to the Kalman Filter used for Impact Prediction.

3.4 Impact Predictor

The objective of the impact predictor is to estimate the potential locations where the interceptor may be able to catch the ball. The algorithm selected for this estimation is a Kalman Filter, which will be used to estimate the (constant) coefficients that describe the linear and parabolic motion of a bouncing ball. It is assumed that within the finite (indoor) environment of the RAVEN room, both drag and Magnus effects play a negligible role in the flight of the ball, at least when compared to the effects of

gravity. Even for throws that impart spin on the ball, it is assumed that the motion in the x-y plane will remain rectilinear but not continuously straight throughout all the bounces. In other words, the ball's motion in the x-y plane is piecewise linear from bounce to bounce.

3.4.1 Formulation

The objective of the Kalman Filter is to estimate a vector of constant coefficients given a series of noisy measurements. Defining the state vector as before, $\mathbf{x} = [\ a_1 \quad a_2 \quad a_3 \quad b_1 \quad b_2 \quad c_1 \quad c_2\ ]^T$, it is clear that the dynamics and observation equations are given by:

$$\dot{\mathbf{x}} = 0 \tag{3.4}$$

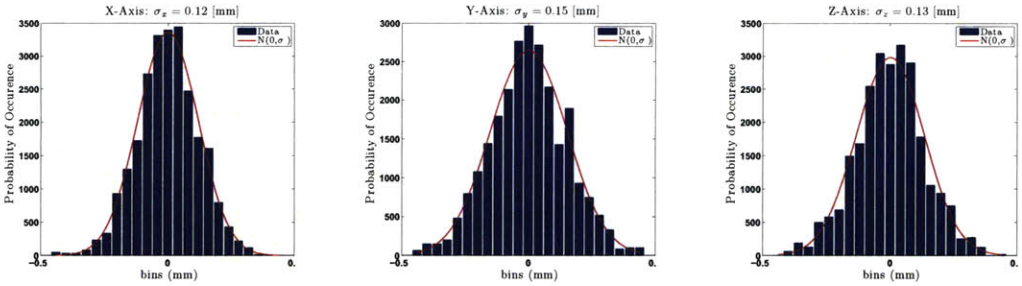
$$\mathbf{z_k} = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{3.5}$$

where:

$$\mathbf{H}_k = \begin{bmatrix} t_k^2 & t_k & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_k & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t_k & 1 \end{bmatrix}$$

$$\mathbf{v}_k = \begin{bmatrix} v_z & v_x & v_y \end{bmatrix}^T$$

and $\{v_z, v_x, v_y\}$ are assumed to be uncorrelated zero mean white processes with covariance $R_k = E[\mathbf{v}_k\mathbf{v}_k^T] = diag(\sigma_z^2, \sigma_x^2, \sigma_y^2)$. The error variance of each individual axis is computed experimentally by measuring the position of a single dot at different locations inside the RAVEN room. For each location and each axis, the mean value is subtracted from the data to form a single error vector. The error vectors are then plotted in a histogram and compared to the equivalent Gaussian density function with the appropriate variance $N(0, \sigma)$. The results, included in Figs 3-4(a), 3-4(b) & 3-4(c), show sub-millimeter tracking accuracy and provide the values that were used for R_k.



(a) X-axis Measurement Error (b) Y-axis Measurement Error (c) Z-axis Measurement Error

Figure 3-4: Axial Measurement Error

The Discrete Kalman Filter can now be written as:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}_k \mathbf{x}_k + \mathbf{w}_k \tag{3.6}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3.7}$$

where

$$\mathbf{\Phi}_k = I_{7x7} \tag{3.8}$$

$$\mathbf{w}_k = [w_z, w_x, w_y]^T \tag{3.9}$$

$$\mathbf{Q}_k = E[\mathbf{w}_k \mathbf{w}_k^T] = \begin{bmatrix} w_z & 0 & 0 \\ 0 & w_x & 0 \\ 0 & 0 & w_z \end{bmatrix} \tag{3.10}$$

Note the addition of process noise in Eq. 3.6. This is done primarily to stabilize the covariance update computation during the first few iterations. In the previous section, it was mentioned that processing of track pre-history information would provide estimates that can be used as initial conditions to the Kalman Filter. Though these initial estimates are better than assuming no prior knowledge of the parameter values (which would yield a very large initial covariance estimate, $P_0^- \rightarrow \infty$), 8 samples may not contain sufficient information to significantly lower the uncertainty in the estimated quantities and therefore, P_0^- will remain large (at least initially.) When P_0^- is large, it is common practice to implement the alternate form of Kalman Filter

[8] for at least a few iterations before reverting back to the nominal Kalman Filter form.

It is important to mention that the measurement noise covariance ($\mathbf{R}_k$), can be updated in the event that the tracking algorithm associates more than one marker to a particular track. When this occurs, the covariance matrix is increased by a quantity proportional to the maximum error between the reported location and the markers' locations. This process allows the algorithm to assign less confidence to measurements that contain the average of multiple markers.

As the ball travels towards its impact point, the Kalman Filter continuously computes and updates the state vector, $\mathbf{x}$ (trajectory parameters), and uses these values to compute catch and impact locations for the current and future bounces. The predictive computations of subsequent bounces assume the following:

- The impact between the ball and the ground is modeled as an inelastic collision with a constant coefficient of restitution ρ [12]. Therefore the z component of the velocity vector before (v_z^-) and after impact (v_z^+) is given by $v_z^+ = -\rho v_z^-$.

- The tracking system identifies the ball as a single marker and provides no spin information. Therefore, the algorithm assumes that at the time of impact, the angle of entry and the angle of departure are the same (see Fig. 3-5) and that the current rectilinear motion in the x-y plane is maintained throughout the ball's entire trajectory (e.g. for all subsequent bounces). Though this is the assumption made by the impact predictor, the ball is clearly allowed to change direction at the time of impact with the ground. However, this work will assume that the ball will continually move forward after each bounce where forward motion is defined in terms of the angle between the velocity vectors before and after impact. Mathematically, define the deflection angle, β, as the angle between the current velocity vector (in the plane) and the velocity vector after the next bounce, namely:

$$\beta = \cos^{-1}\left(\frac{\mathbf{v}_k \cdot \mathbf{v}_{k+1}}{|\mathbf{v}_k||\mathbf{v}_{k+1}|}\right)$$

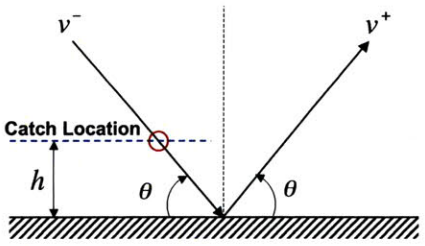


Figure 3-5: Geometry at Impact

Then, it is assumed that $|\beta| \leq 90^o \ \forall t$.

The computation of the catch and impact location for all subsequent bounces is straight forward. Start by defining T_0^G as the global track initiation time reported by the tracking system. Further assume that at a time, $t_k > T_0^G$, the Kalman Filter state estimate is $\mathbf{x}_k = \begin{bmatrix} a_1 & a_2 & a_3 & b_1 & b_2 & c_1 & c_2 \end{bmatrix}^T$, then, the relative and global impact times of the next bounce are given by:

$$T_1^R = -\frac{a_2 + \sqrt{a_2^2 - 4a_1a_3}}{2a_1} = -\frac{a_2 + \Delta}{2a_1} \tag{3.11}$$

$$T_1^G = T_0^G + T_1^R \tag{3.12}$$

For all subsequent bounces, the relative and global impact times are given by:

$$T_n^R = -\frac{\rho^{n-1}}{a_1}\Delta \tag{3.13}$$

$$T_n^G = T_0^G + \sum_{i=1}^{n} T_i^R \tag{3.14}$$

The impact locations on the $x - y$ plane are simply:

$$x_n = b_1 \sum_{i=1}^{n} T_i^R + b_2 \tag{3.15}$$

$$y_n = c_1 \sum_{i=1}^{n} T_i^R + c_2 \tag{3.16}$$

The catching (intercept) times are easily computed by determining the time the ball will cross a plane located a distance h above the ground, which corresponds to the

height of the catching mechanism. The relative and global catching times for the first and subsequent bounces are given by:

$$T_{c_1}^R = -\frac{a_2 + \sqrt{a_2^2 - 4a_1(a_3 - h)}}{2a_1} \tag{3.17}$$

$$T_{c_n}^R = -\frac{\rho^{n-1}\Delta}{2a_1}\left[1 + \sqrt{1 + \frac{4a_1 h}{\rho^{2(n-1)}\Delta^2}}\right] \tag{3.18}$$

$$T_{c_n}^G = T_0 + \sum_{i=1}^{n-1} T_{c_i}^R + T_{c_n}^R \tag{3.19}$$

and the catch locations on the $x - y$ plane are simply:

$$x_{c_n} = b_1\left(\sum_{i=1}^{n-1} T_{c_i}^R + T_{c_n}\right) + b_2 \tag{3.20}$$

$$y_{c_n} = c_1\left(\sum_{i=1}^{n-1} T_{c_i}^R + T_{c_n}\right) + c_2 \tag{3.21}$$

Note that in Eq. 3.18, $\exists T_{c_n}^R \in \Re$ iff $h \leq -\frac{\rho^{2(n-1)}\Delta^2}{4a_1}$, which simply means that in order to catch the ball at a height, h, the ball bounce must achieve a maximum height $H > h$.

3.4.2 Results: Impact Prediction Error

The accuracy of the Impact Predictor was tested experimentally by repeatedly throwing a ball and collecting trajectory information with the optical tracking system. The raw data was reduced to determine the actual impact and catch locations[4]. The catch location was computed via interpolation by identifying the samples before and after the ball crossed the catching plane. The impact location was determined in one of two ways: 1) by projecting the last two known locations above ground (before and after ground impact) and computing the intercept point of the two lines or 2) by applying an LSE to the parabolic trajectory before and after a particular bounce and finding the intercept point. The catch plane was located six inches (15.24 cm) above

[4]The impact location is defined as the point where the ball contacts the ground, while the catch location is defined as the point where the ball crosses a catching plane.

ground level, which simulates the height of the capturing device. The ball was thrown both with and without spin to observe the stability of the filter and accuracy of the predictions. As stated before, the primary goal of the impact predictor is to accurately estimate both the catch location and the time at which the interceptor must rendezvous with the target (e.g. catch the ball). The secondary goal is to provide intermediate points (e.g. predicted impact locations) to a path planner that plots a pure-pursuit course that directs the vehicle towards the target. Results show that, if the ball were to be intercepted prior to the n^{th}, then the accuracy with which the catch point can be predicted depends primarily on the data collected after the $(n-1)$ bounce. This is a direct consequence of the ball changing its direction of motion every time it impacts the ground[5]. Clearly, if the ball were to remain in a straight path after every bounce, then all information from previous bounces would help in reducing the prediction error of the catch point. However, these deviations are expected and therefore, the information from all previous bounces will not necessarily improve accuracy in the prediction of the catch locations (see Fig. 3-11). Regardless of the ball's directional changes, the prediction of the intermediate points prior the actual catch (i.e. ground impact locations) proved to be extremely useful for path planning. Results show that these impact locations can be used as waypoints to the path planner in order to develop a reference path that will place the interceptor in a pure pursuit path towards the target

The results can be best understood by way of an example. In general, it is assumed that the ball will be caught prior to the third bounce and intermediate results are shown in order to gain more insight on the prediction process. Figures 3-6 to 3-8 show the post processing results of a sample simulation frozen at 3 distinct instants in times. Each figure includes four plots which show the ball's trajectory as a function of time in the z (height), x and y axis, and a top view of the engagement. Henceforth, these plots will be referred to as: Plot 1 (Height vs. Time), Plot 2 (x vs time), Plot 3 (y vs. time) and Plot 4 (Top View).

Plots 1 through 3 include the current ball's location (black star), the predicted

[5]Recall the impact predictor's assumption regarding ball trajectory for subsequent bounces.

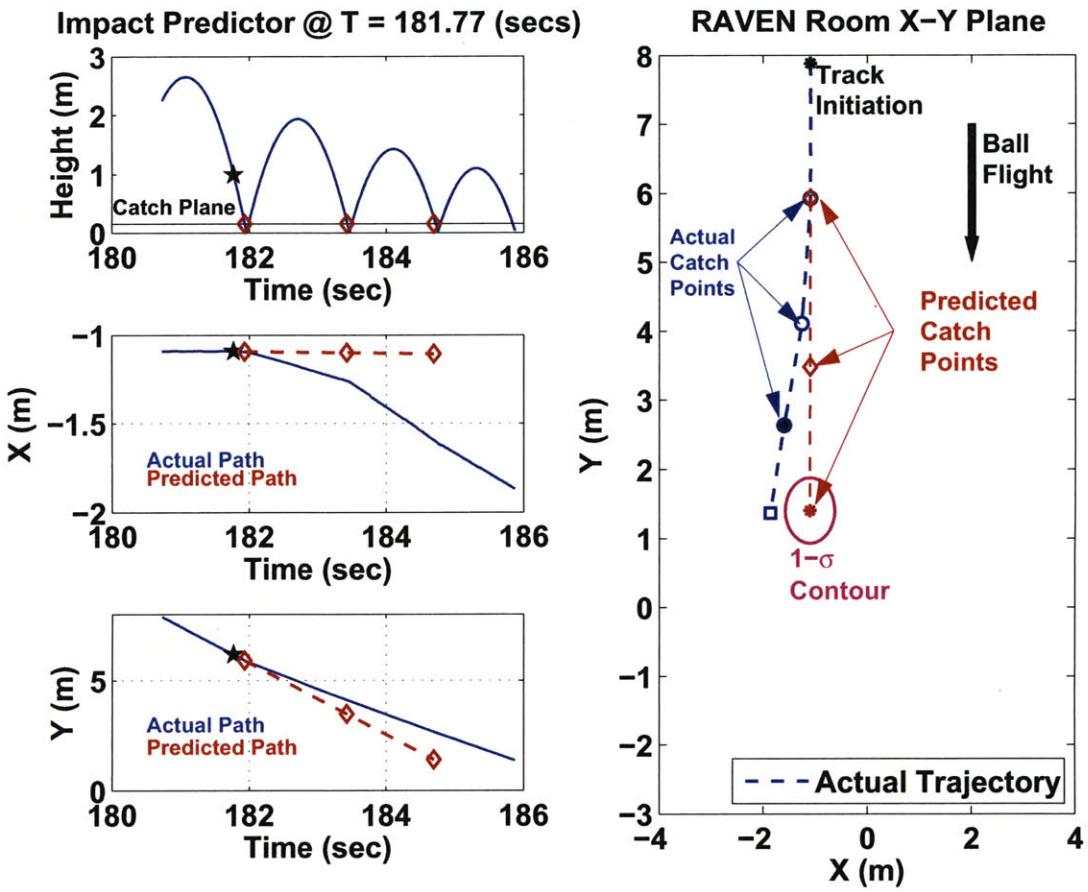


Figure 3-6: Post Processing Prediction Prior to First Bounce

catch time and locations (red diamonds), the actual ball trajectory (blue line), and the predicted ball trajectory in the x-y plane (red dotted line). The trajectory data shown in Plot 1 is used by the impact predictor to estimate *catching times*, while the trajectory data from Plots 2 and 3 are used to determine the predicted *catch locations*. Plot 4 shows the actual catch locations (blue dots), the predicted catch points (red diamonds) and the 1-σ state propagation error of the final catch point.

As mentioned in the previous section, the Kalman filter estimates the state vector $\mathbf{x}_k$ at each time sample t_k. These estimates are used to continuously update the predicted catch and impact points. As new measurements are available, the Kalman Filter error covariance starts to settle and the predicted locations start to converge.

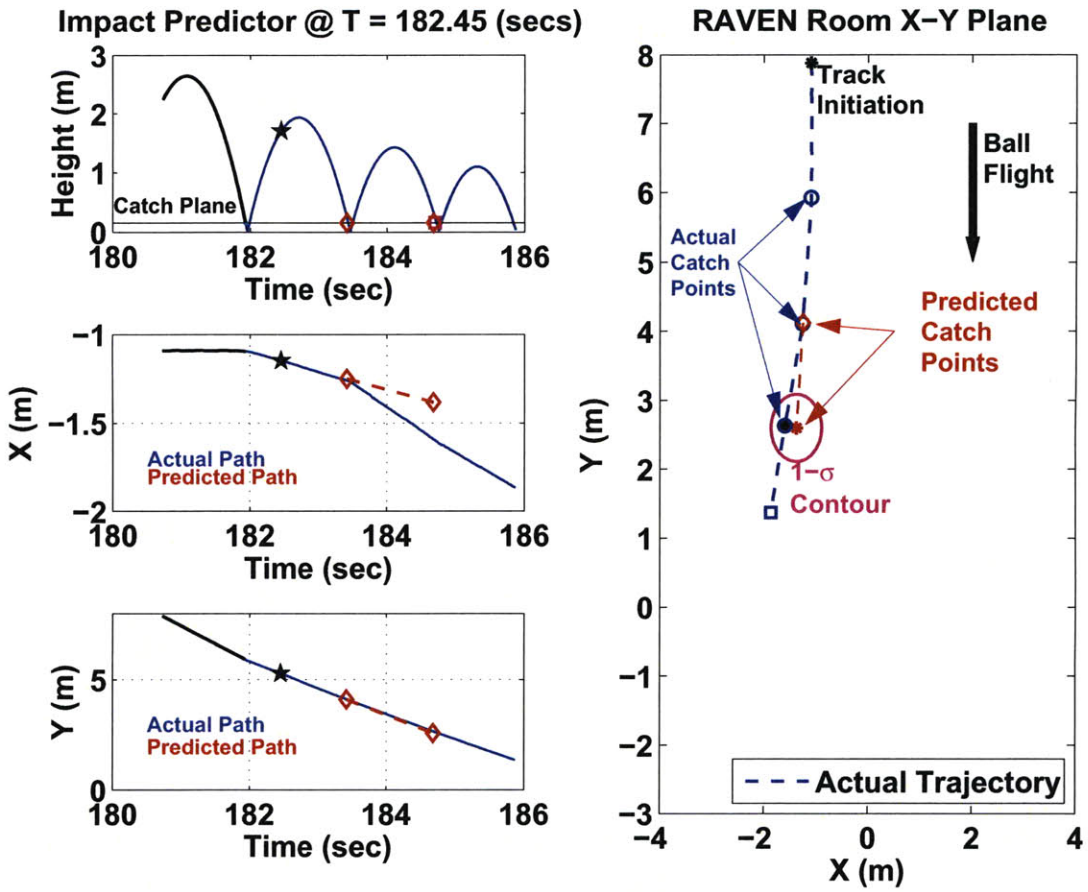


Figure 3-7: Post Processing Prediction Prior to Second Bounce

The behavior of the impact predictor can be better understood by presenting the results of one particular run at three different time instances. Fig. 3-6 shows a snapshot of the simulation prior to the first bounce. Plot 1 shows the parabolic trajectory of the ball (blue) along with the predicted catch times (red). These times are computed using Eqs.3.17 - 3.19. Plots 2 and 3 show the predicted catch locations (red) which are computed using Eq. 3.20 and Eq 3.21. Note that, except for the first bounce, the predicted locations for all future bounces are nowhere near the actual locations. This is a direct consequence of the assumption that the ball will maintain it's current heading (in the $x - y$ plane) for all subsequent bounces. Obviously, this assumption may lead to large positional errors, especially when the ball changes direction after

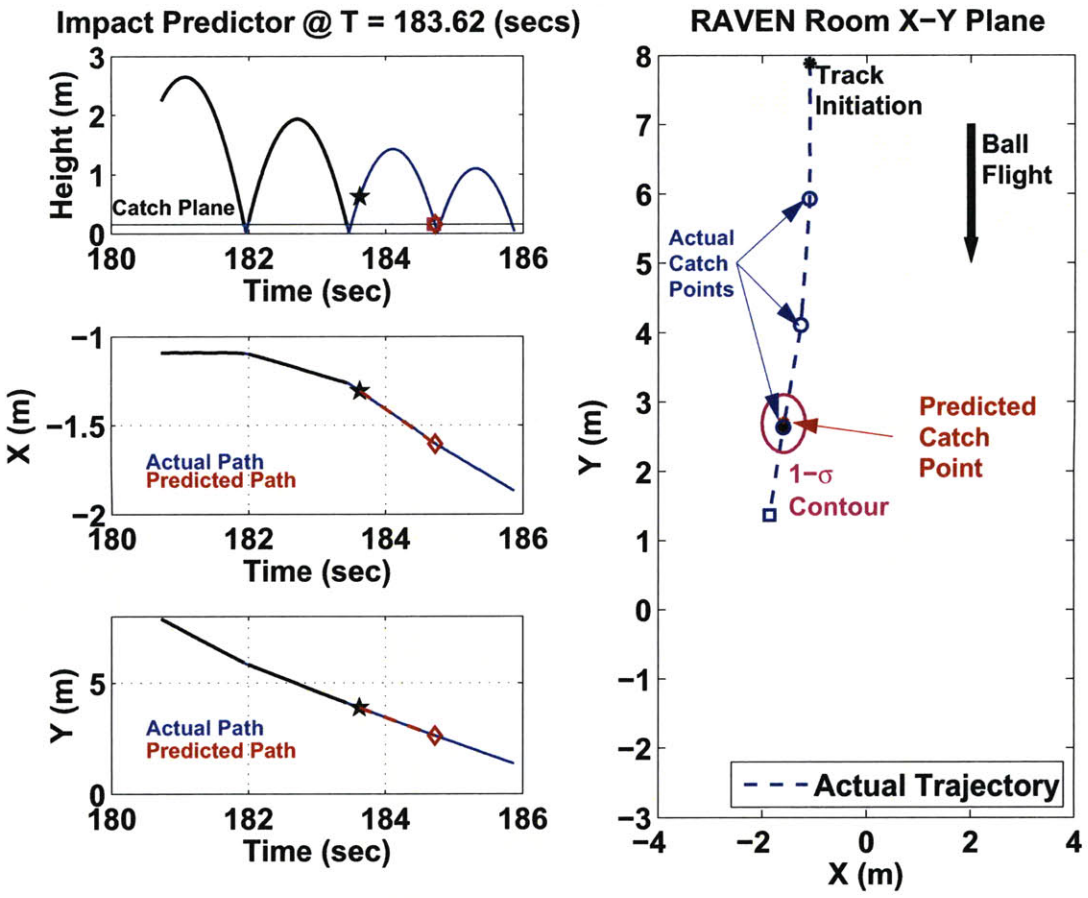


Figure 3-8: Post Processing Prediction Prior to Third Bounce

a bounce. It is clear from the figure that the information obtained prior to the first bounce does not provide much insight as to the final catch location. However, for path planning purposes, the predicted path is in the general direction of the intercept point and can be used to provide initial direction to the interceptor (more on path planning in chapter 4.) As the ball progresses towards the second bounce, the error in the catch location has decreased substantially as depicted in Fig. 3-7. The predicted path continues to converge towards the desired (final) catch location, which indicates that it continues to provide proper direction to the interceptor. Note that the estimate for the current catch point is very good and the vehicle may be able to catch the ball at the end of the current bounce provided that the controller can place

the vehicle at the catch location at the appropriate time. As the ball progresses to the final bounce, the estimator has successfully converged to the desired catch location (see Fig. 3-8) but with non-zero error. A post test analysis of the temporal and positional prediction error is required in order to quantify the various error terms as a function of time.

Prior to discussing prediction error analysis, it is necessary to define the following time variables: t_0 is the track initiation time, t_1 and t_2 are the catch times prior to the first and second bounce, respectively, and t_A is the final catch time. Now define 1) an estimation vector $\hat{M}(t)$ that contains the time varying estimates of the time and position associated with the final catching point, and 2) the constant vector $M(T_A)$, which contains the actual time and location of the final catching point. The vectors can be written as:

$$\hat{M}(t) = \begin{bmatrix} \hat{T}_3^c \\ \hat{x}_3^c \\ \hat{y}_3^c \\ \hat{z}_3^c \end{bmatrix}, \quad M_A = \begin{bmatrix} T_3^c \\ x_3^c \\ y_3^c \\ z_3^c \end{bmatrix}$$

where $\hat{M}(t)$ is defined for $t \in (t_0, t_A]$[6]. The error vector can now be defined as: $\delta M = \hat{M}(t) - M_A = \begin{bmatrix} \delta T & \delta x & \delta y & \delta z \end{bmatrix}^T$. Obviously, the impact predictor attempts to find a solution for which $\hat{z}_3^c = z_3^c$, which implies that (for the purpose of analyzing prediction errors) $\delta z = 0$. The non-zero components of the error vector are then:

$$\delta_{time} = \hat{T}_3^c - T_3^c \tag{3.22}$$

$$\delta x = \hat{x}_3^c - x_3^c \tag{3.23}$$

$$\delta y = \hat{y}_3^c - y_3^c \tag{3.24}$$

which are the errors that are presented in Fig. 3-9. The plots of time and location errors are divided into three segments corresponding to estimation performed prior

[6]The reason t_0 is not included in the error computation is because the Kalman Filter will not provide any state estimates to the impact predictor until the covariance matrix has "settled".

to the first, second and third bounces. The catch time error is measured in msec and intercept location error is measured in cm. The plots also include a red "star" which correspond to the errors associated with the snapshots presented in Figures 3-6 to 3-8. Fig. 3-9 shows that with each bounce, the error in prediction continually drops dramatically until the moment the vehicle catches the ball. Of particular interest are the end game errors shown in Fig. 3-10. In this particular engagement, the ball's time of flight during the last bounce is ≈ 1.15 seconds. During this time, the error in impact time starts at 50 msec (roughly 5 system samples) and within 0.24 seconds it quickly drops below 20 msec. The error in x is well within 1 cm throughout the entire endgame, while the error in Y drops below 2 cm after ≈ 0.17 seconds.

The time-varying RMS error $\epsilon_{RMS}(t)$ can be computed by using:

$$\epsilon_{RMS} = ||\delta P^T \delta P||_2 \tag{3.25}$$

where $\delta P = \begin{bmatrix} \delta x & \delta y \end{bmatrix}^T$ and δx and δy are defined in Eq. 3.23 and 3.24, respectively. The total RMS error for the entire engagement and for the endgame are shown in Fig. 3-11.

The plot also includes (in red) the radius of the catching device in order to provide a better idea as to the amount of time available to the interceptor to accurately catch the ball. It is of interest to note that in this example (and in general) the ball's trajectory information from the first bounce provides little insight as to the location of the final catching point. The most useful information is obtained in between the second and third bounce which is not a surprising results but it helps illustrate the time critical aspect of the problem.

Before leaving this section, it is important to understand the spread of the intercept location's error in both the x and y direction for a large number of runs. Fig. 3-12(a) was constructed by plotting $\delta P(t)$ for $t \in [t_2, T_A]$. The figure also includes the outline of both the catching device and the ball. The percentage of points that fall within the capturing device's area can be easily obtained by plotting the RMS error distribution and adding the total number of points that fall within a ra-

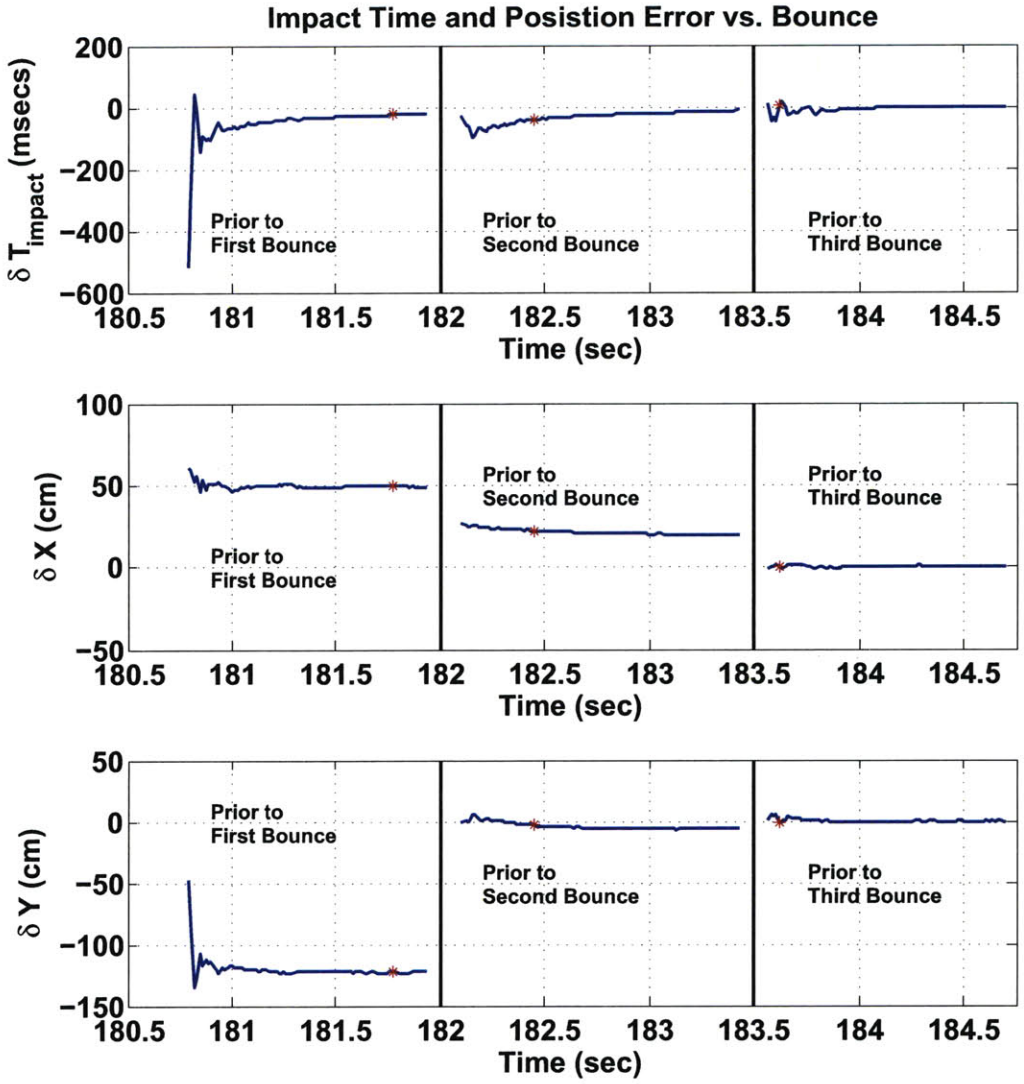


Figure 3-9: Catch Location Estimation Error in Time and Position

dius of 4.5 cm (the radius of the catching device). Results show a theoretical capture rate of approximately 95% (see Fig. 3-12(b)) assuming that the control and navigation strategies perfectly place the vehicle at the proper place and time (a highly optimistic goal.)

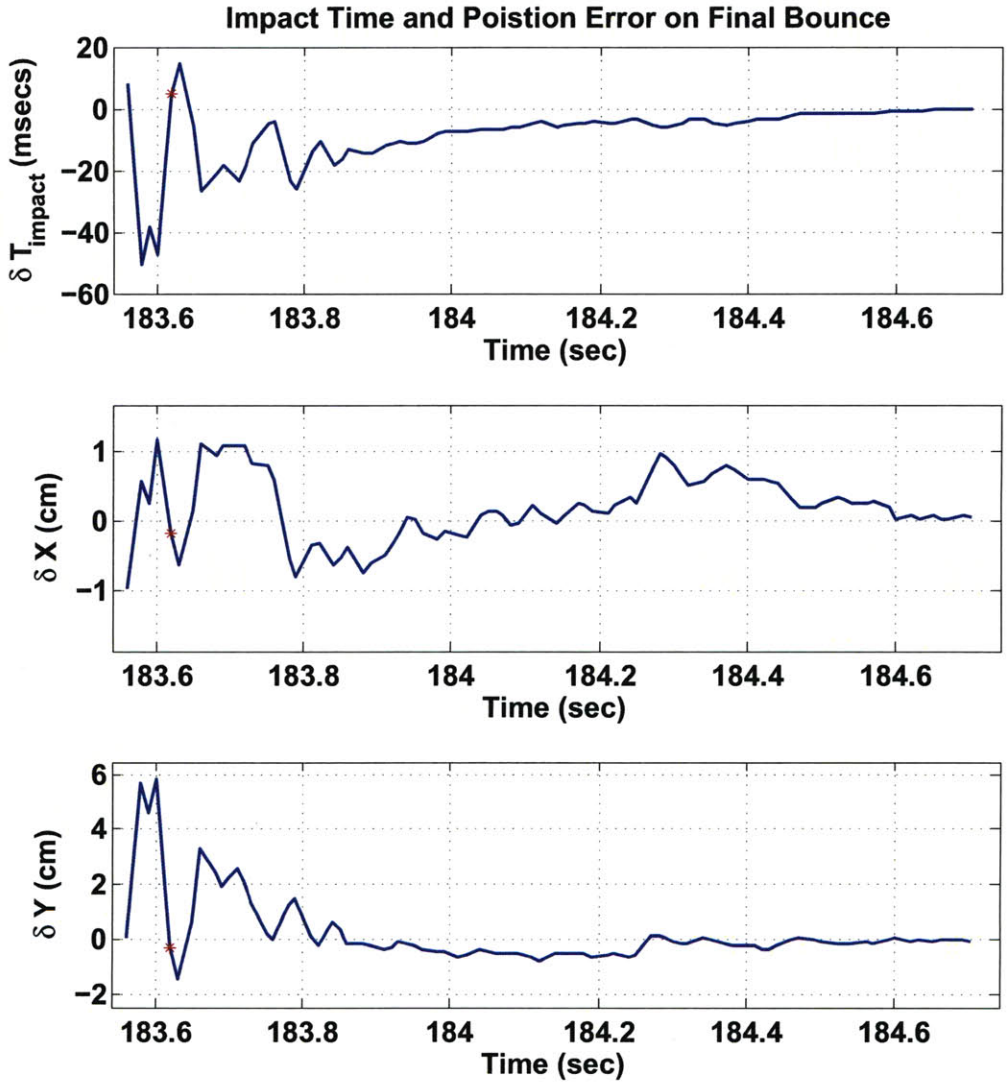
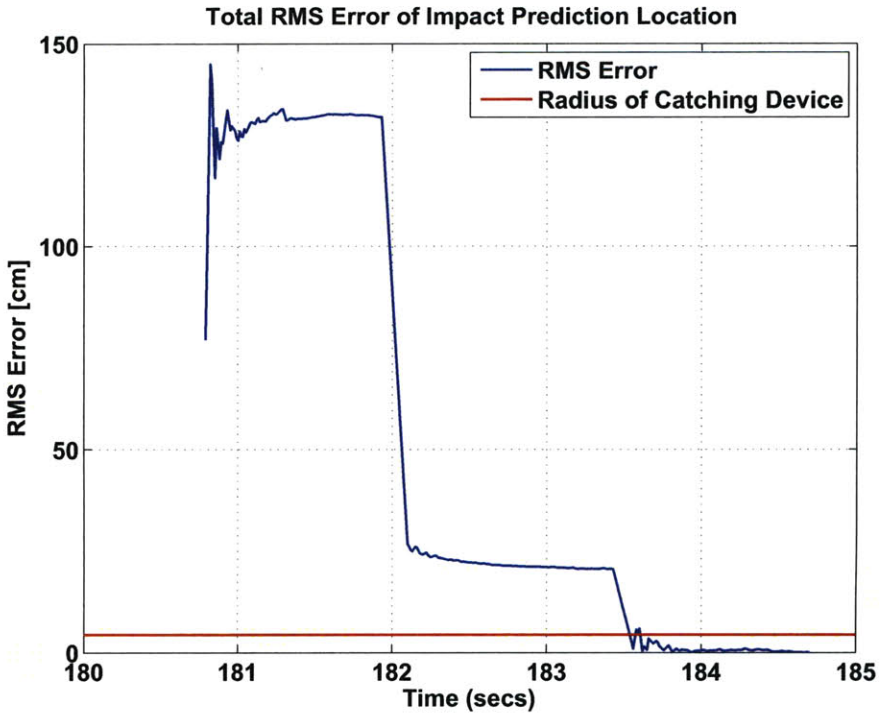
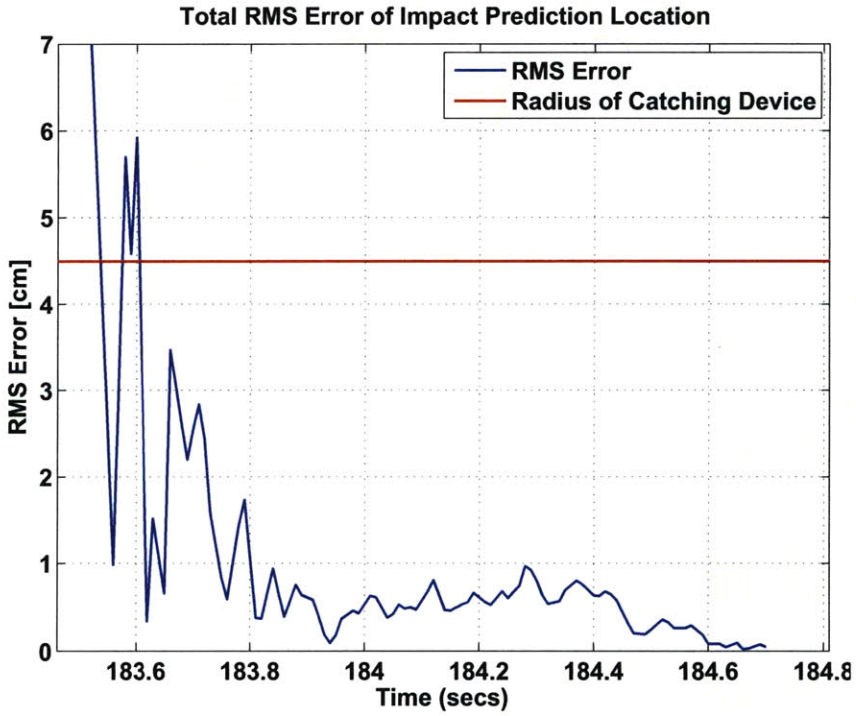


Figure 3-10: Post Analysis End Game Error

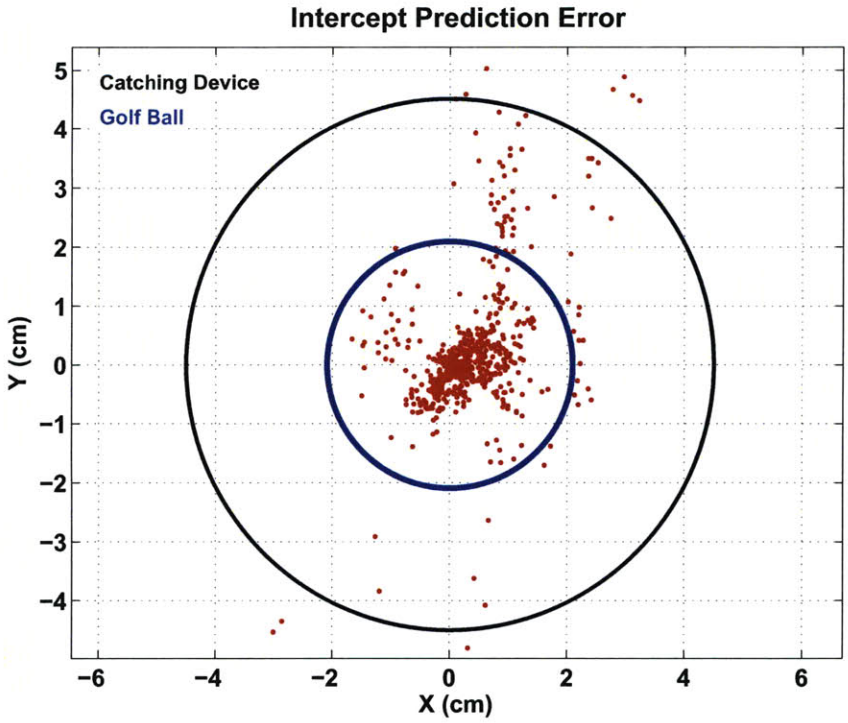


(a) RMS Error vs Bounce

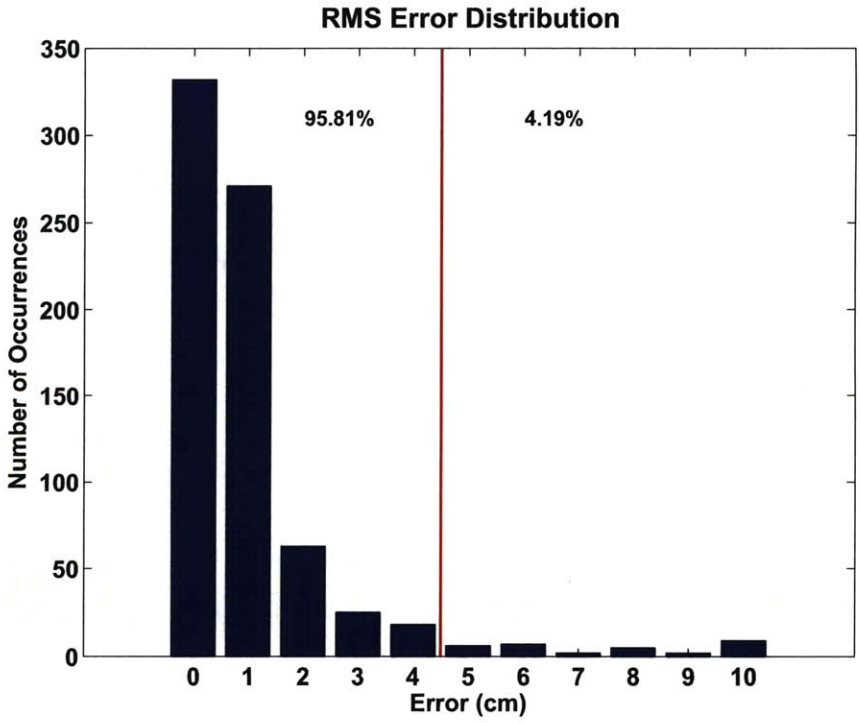


(b) End Game RMS Error

Figure 3-11: Root Mean Squared Error of Catch Location Estimation



(a) End Game Error Location



(b) End Game RMS Error Distribution

Figure 3-12: Distribution of Intercept Location

Chapter 4

Navigation and Control

4.1 Introduction

Once the task of properly tracking the dynamic object and accurately predicting the intercept location is complete, the next goal is to control the path of an object in order to intercept a target which is also in motion. In this application, the path is not merely a collision course but rather a course that would maximize the ability of the interceptor to catch or retrieve the moving object. In essence, it is a very precise intercept problem. The problem is divided into two parts (as depicted in Fig. 4-1):

1. The development of a reference path that would place the vehicle at the intercept location, and

2. The generation of control commands that would allow the vehicle to follow a pre-computed path in an efficient manner so that it arrives at the intercept location at the appropriate time[1].

Path planning and path following have been researched extensively. For this application, the pure pursuit path tracking strategy was selected, since it has been widely employed for planned navigation of non-holonomic vehicles.

[1]Early arrival to the intercept point is acceptable but one runs the risk that the vehicle may not be able to react quickly to any "last minute" prediction updates.

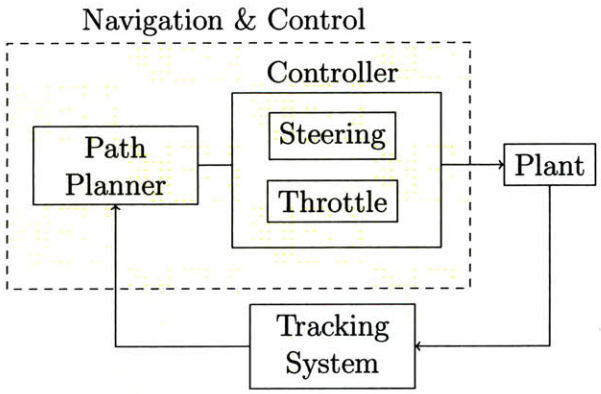


Figure 4-1: Navigation and Control System Architecture

This chapter covers in detail the path planning and control strategy, to include modifications to the reference path during the end game, where timely and accurate arrival to the intercept location is of utmost importance.

4.2 Path Planning

The strategy by which the interceptor approaches the dynamic object must be considered very carefully if one wishes to retrieve rather than merely intercept a dynamic object. The distinction between the two strategies is that the retrieval requires precise control of the interceptor in both space and time so that the dynamic object to be intercepted falls within the footprint of the catching/retrieving device (which is often smaller than the vehicle in which it is transported.) The objective of intercept is equally challenging but in this case the dynamic object is allowed to impact *any* part of the intercepting vehicle. In the context of this application, the vehicle must plot a path that would maximize its ability to catch a bouncing ball. Adding to the path planning complexity is the room's finite dimensions, the vehicle's maneuverability constraints, the presence of potential obstacles and the uncertainty in the estimate of the final catch location. A plausible strategy is to plot a course that directly connects the vehicle's initial position, P_0, to the predicted catching location, P_c. The potential problem with this strategy is that for some end game geometries, it may make it

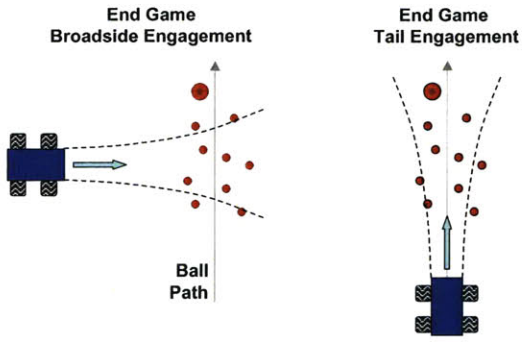


Figure 4-2: Broad Side and Tail On End Game Geometries

difficult, if not impossible, for the interceptor to successfully catch the ball. Consider a broadside engagement, in which the angle between the vehicle's velocity vector and the ball's trajectory is near (or greater than) 90 degrees as shown in the left panel of Fig. 4-2. In the figure, the small red dots indicate initial catch location estimates, while the large red ball indicates actual ball location. The dotted lines show the vehicle's maneuverable region that would prevent it from either drifting or spinning out of control. It is clear that any prediction error that results in a large lateral error may yield a miss because the vehicle will be unable to perform "last minute" maneuvers in a controlled fashion. In order to mitigate this particular constraint, it is reasonable to consider the use of pure pursuit in which the vehicle attempts to intercept/catch the dynamic object by approaching it in a near tail-on fashion as shown in the right panel of Fig. 4-2.

The classical pure pursuit problem dates back to 1732 when the French mathematician Pierre Bouguer posed the problem of a pirate ship intercepting a merchant ship[2][18]. The problem stated that a merchant ship was fleeing at a constant speed, V_m and a pirate ship was pursuing it at a constant speed, $V_p > V_m$,[3] along a path

[2]It is argued by Paul Nahin that the construction of the "pursuit curves" dates back to the 1700's when the French physician Claude Perrault posed a problem to Gottfried Leibniz whose solution gave rise to the what Christian Huygens referred to as tractrix curves. This particular book is highly recommended as a very informative (an quite enjoyable) presentation of classic chase and escape problems.

[3]Not a restriction posed in the original problem, but one that is implied in this work.

that is always moving in the direction of the merchant ship (e.g. the tangent to the path at any instant in time is always pointing in the direction of the merchant vessel.) The resulting curve, called *ligne de poursuite*, delineated a trajectory that lined up the pirate ship right behind the merchant ship until the inevitable tail intercept was accomplished.

The described strategy is very attractive for this application given the fact that the object to be intercepted (a bouncing ball) will not purposefully make evasive maneuvers in the endgame. However, as explained in the previous section, after every bounce there is an element of uncertainty in the ball's path due to ball spin or irregularities in the impact plane (ground). Some of this lateral error may be mitigated by modifying the size of the catching device, but in general this situation requires a more aggressive control strategy in the endgame.

Reference Path

The reference path is piece-wise linear whose waypoints are connected by straight lines. The waypoint assignment comes directly from the impact predictor and represent either the impact locations of each ball bounce or the catch or goal location (from this point forth, the goal location and the catch location will be used interchangeably and will represent the same point)[4]. As discussed in Chap. 3, the engagement commences at the moment the track is initialized and all pre-history information is processed; however, waypoint assignment is delayed until the diagonal elements of the estimator covariance have settled (e.g. stop oscillating), which nominally occurs 0.2 seconds after track initiation (see Fig. 3-9 and 3-10).

Waypoint locations are constantly updated as new information (ball pose) is available to the Kalman filter. Clearly, constant updates to the waypoint result in constant changes to the reference path, which could potentially induce some instability to the steering controller. However, the vehicle itself acts as a low pass filter which helps attenuate these variations. Furthermore, by the time the ball has reached the apex

[4]For this application, impact point predictions are limited to three ball bounces mainly due to the room's physical dimensions and to the addition of safety buffer zones around the lab's perimeter to protect the vehicle.

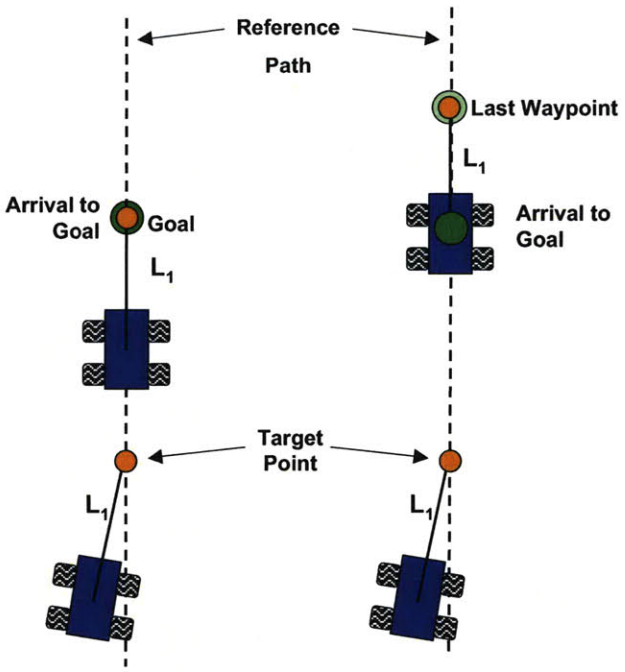


Figure 4-3: Waypoint Modification

of its current parabolic trajectory, variations to the impact and catch locations are small when compared to the distance that the vehicle must travel.

The only modification to the waypoints provided by the impact predictor is the assignment of the catch location. Though the details of the pure pursuit implementation are discussed in Section 4.3.1, it is useful to offer a brief explanation here for clarity. The car steers towards the goal by following a target point that lies on the reference path and is located a distance L_1 away from the vehicle as shown on the left panel of Fig. 4-3. Arrival to a goal is determined by the arrival of the target point to the goal rather than by the physical arrival of the vehicle to the goal. This situation can be easily remedied by moving the goal point along the reference path a distance L_1 away from it's original location. This new location is now the final waypoint of the reference path, which will ensure the arrival of the car to the desired goal location as shown in the right panel of Fig. 4-3. From this point forth, the term "waypoint" will refer exclusively to the points in the plane that define the reference path, and

the term "goal" will refer to the catch location.

4.2.1 End Game Strategy

End game refers to all the actions that take place during the last portion of the engagement[5] that enable the interceptor to catch the ball. This portion of the engagement is extremely important because all longitudinal and lateral errors must be minimized if the interceptor is to have a chance at catching the ball repeatedly. As discussed in Chap. 3, there is no reason to believe that after each bounce, the ball will maintain the same direction of motion; in fact, one should expect random directional changes after every bounce This means that last segment of the reference path can change abruptly and the path tracking algorithm must be able to quickly adjust to these changes. It is important to note that when using pure pursuit, the interceptor may arrive to the goal point with non-zero lateral error. The magnitude of the lateral error will depend on factors such as vehicle turning radius, vehicle speed, look ahead distance, and the angle between consecutive segments. Even in the case where two consecutive waypoints form a straight line, a seemingly small lateral error on the order of 2 to 4 cm could prove detrimental to the success of the interceptor whose catching device has a radius of 4.5 cm. To illustrate this point, a simulation was performed whose engagement geometry is illustrated in Fig. 4-4. The goal of the simulation was to have the vehicle follow a series of pre-defined reference paths in order to determine the closest point of approach (PCA) to the goal point (the green square in the figure). Each path is associated with a vehicle's initial position in the x-y plane, namely: (2,-2), (1,-2), (0,-2), (-1,-2), (-2,-2). The vehicle's initial heading was either 0^o (along the x-axis) or 90^o (along the y axis) and the goal was located at (0,3). The lateral error was computed as a function of time according to the following formula:

$$\epsilon_{lateral}(t) = ||\mathbf{P}_{vehicle} - \mathbf{P}_{goal}||_2 \tag{4.1}$$

$$\epsilon_{min} = \min(\epsilon_{lateral}(t)) \tag{4.2}$$

[5]This portion corresponds to the last segment of the reference path

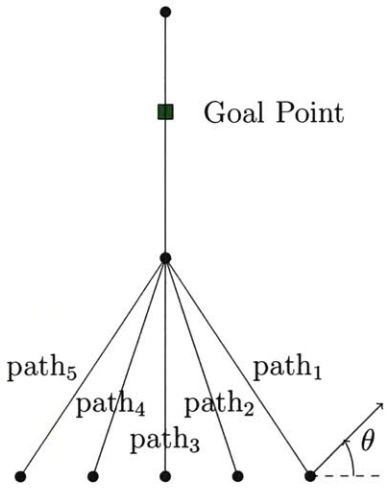


Figure 4-4: Pure Pursuit Simulation of the End Game

where $\mathbf{P}_{goal} = [0, 3]$. The simulation results shown in Fig. 4-5 focus on the latter part of the end game, namely 0.7 seconds prior to reaching the point of closest approach. The figure reveals that there were only two cases in which the lateral error was small enough to merit catching a moving object (repeatedly). The two cases were associated with the following initial conditions: $\theta_0 = 0^o$, $P_0 = [-1, 2]$ and $\theta_0 = 90^o$, $P_0 = [0, 2]$. (The results for the latter case are not surprising since all the vehicle is required to do is drive in a straight line to the goal point. Of course, in the presence of noise disturbances in either the vehicle's position or steering, this error would increase.)

This example clearly demonstrated that in the endgame, the reference path must be modified to force the interceptor to go through the catching point, thus minimizing lateral error.

The endgame path planning strategy is similar to the motion planning discussed in Kuwata et al. [13] in that the dynamics of the vehicle are propagated through various candidate reference paths. The goal is to determine which reference path yields the smallest lateral error, defined as the distance between the center of the vehicle's catching device and intercept point. This strategy is depicted in Fig. 4-6, where the green star represents the catching point, the straight lines represent the candidate reference paths and the dotted lines represent the actual vehicle trajectories. In this example, the black line represents the reference path provided by the planner. Note

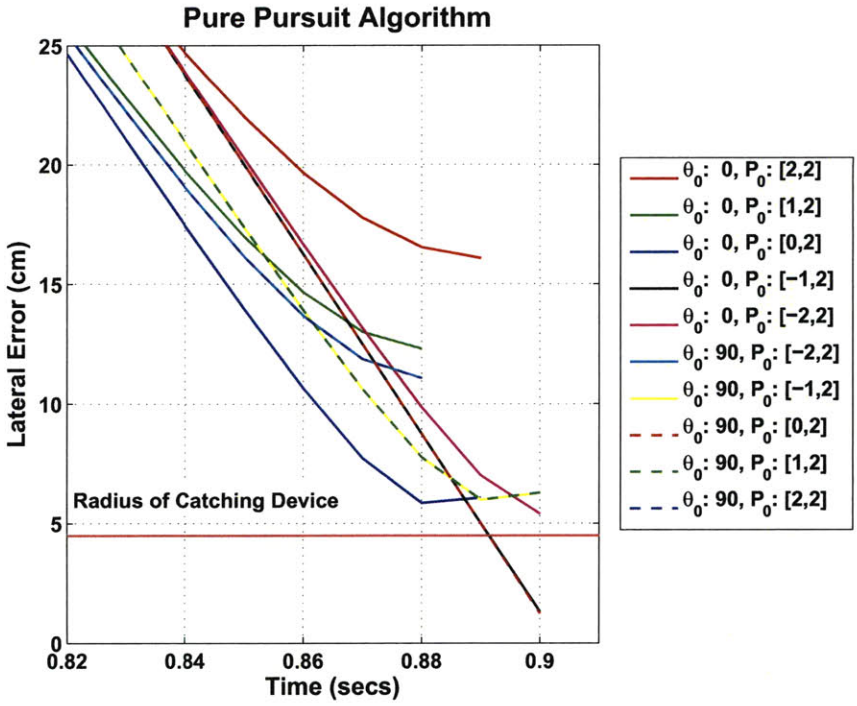


Figure 4-5: End Game Lateral Error Using Pure Pursuit

that if the vehicle were to follow this path, it would not cross the desired catch point as depicted by the black dotted line which, again, represents the expected vehicle trajectory. On the other hand, if the vehicle were to follow the orange reference path, the actual trajectory will place it right over the goal point; therefore, the strategy is to propagate the vehicle dynamics through various reference paths and select the path that would minimize the lateral error to the catch point. Assume that as the vehicle enters the engagement's end game, its position and heading are $\mathbf{P}_0$ and θ_0 respectively. The dynamic model used to describe the interceptor dynamics is the kinematic bicycle model[6] below:

$$\dot{x} = V\sin(\theta) \tag{4.3}$$

$$\dot{y} = V\cos(\theta) \tag{4.4}$$

$$\dot{\theta} = \frac{V}{L_w}\tan(\delta) \tag{4.5}$$

[6]Dynamic effect such as slipping are ignored.

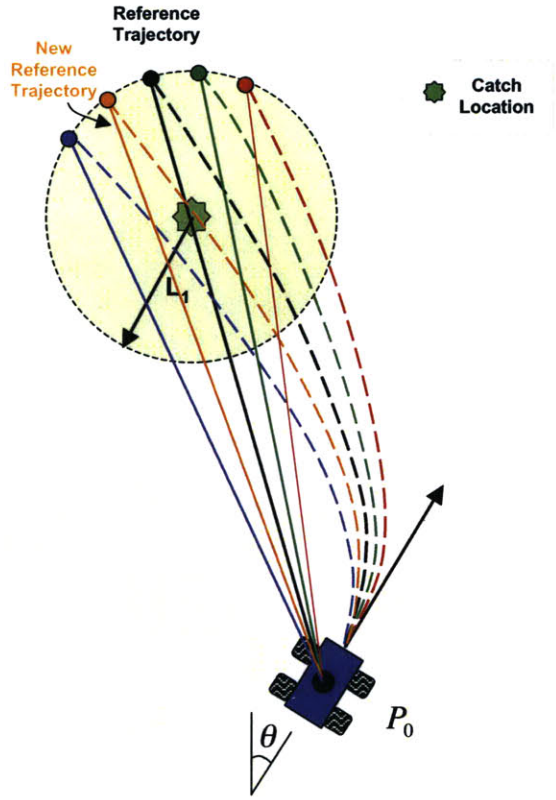


Figure 4-6: Waypoint Modification

$$\dot{V} = 0 \tag{4.6}$$

where (x, y) and V are the vehicle's position and speed, respectively, θ is the vehicle heading, L_w is the vehicle wheel base, and δ is the steering angle. Note that the model assumes constant speed when propagating the vehicle dynamics. The speed is derived from engagement information that is available to the path planner, namely, distance to target (D) and time-to-catch (t_c). The average speed is simply $\bar{V} = \frac{D}{t_c}$. The vehicle dynamics are first propagated using the reference path provided by the planner (black line in Fig. 4-6) to determined the estimated lateral error. The dynamics are then propagated through all potential reference paths and the lateral errors recorded. The path that yields the minimum lateral error is selected and the planner's reference path is updated. The process is repeated until the vehicle reaches the catch point.

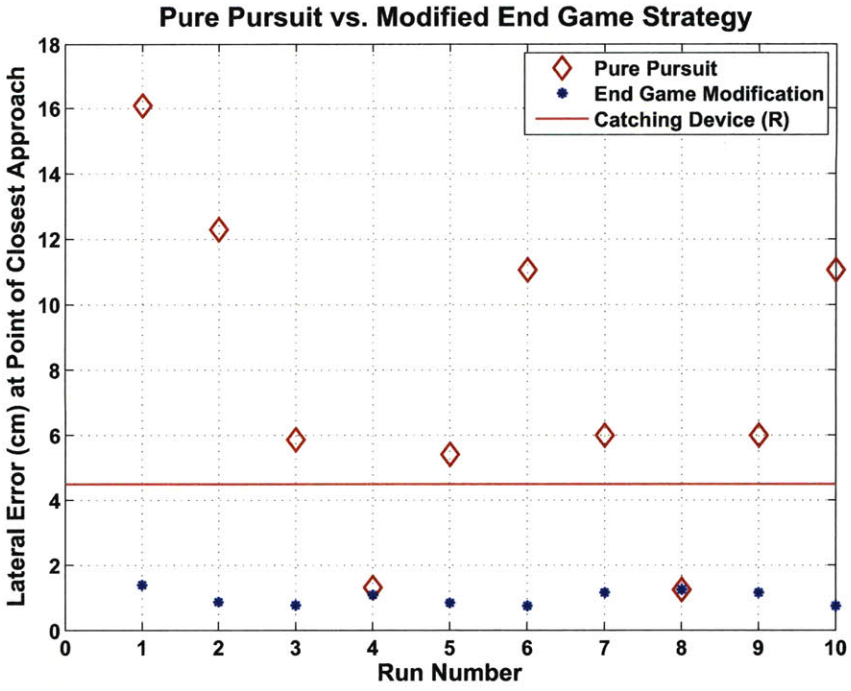


Figure 4-7: Lateral Error Comparison between Pure Pursuit and End Game Pure Pursuit Modification

The benefit of this strategy is best illustrated by running the simulations described in Fig. 4-4 with the new end game pure pursuit modification. (Recall that in the simulation the vehicle's initial position and heading were changed for variability.) The results, shown in Fig. 4-7, clearly illustrate that the new strategy will provide the necessary accuracy independent of initial conditions.

The method of propagating dynamics is accompanied by a computational cost that increases with the total number of reference paths to be explored. Furthermore, the accuracy of the solution will depend both in the number of reference paths to be searched and the sampling rate at which the dynamics are propagated. Early computations showed that searching through ten reference paths and propagating dynamics at a sampling rate of 10 Hz, will yield a solution in ≈ 0.054 seconds. This computational cost decreases as the vehicle approaches the goal point, since the total distance to go decreases, and the vehicle's heading is nearly collinear with the reference path. To illustrate the latter point, the simulations described in Fig. 4-4

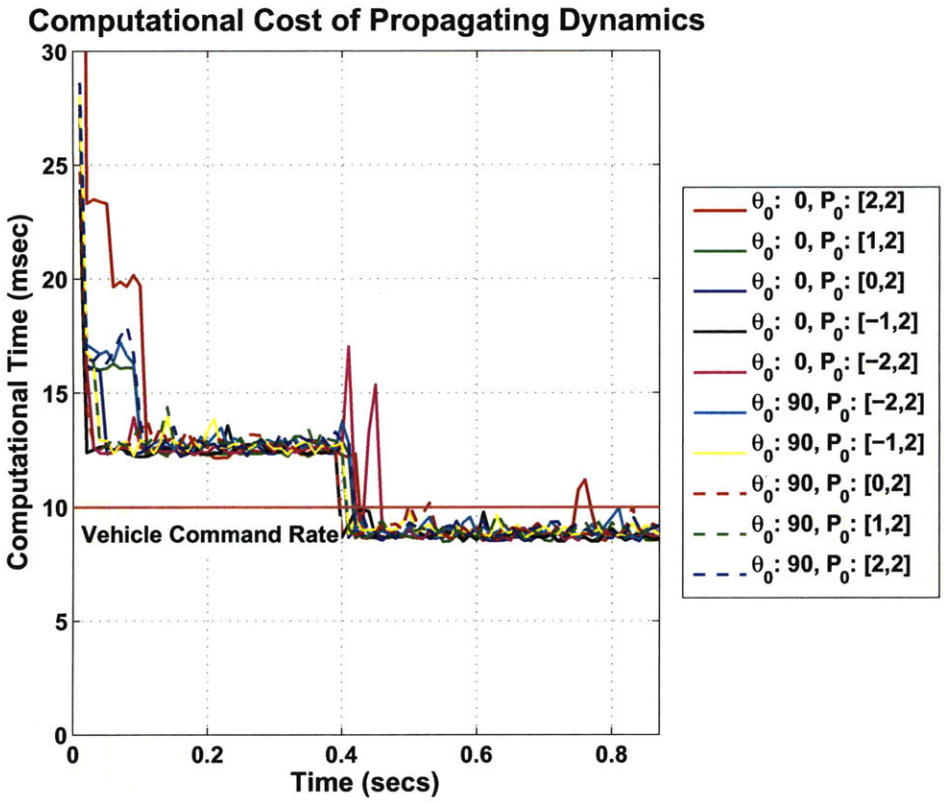


Figure 4-8: Computational Costs for Searching Best Path in the End Game

were used to compute the amount of (processing) time needed to arrive at a solution as a function of time. The dynamics were propagated at 2Hz and a total of 10 reference paths were continuously searched. The results are shown in Fig. 4-8, where the red line represents the rate at which the control commands are send to the vehicle. The figure clearly shows that for $t < 0.4$ msec, the computational costs exceed the vehicle command rate so the search parameters must be modified. For $t > 0.4$, the computational cost is manageable but still relatively high. The reason for the drop in computational cost can best be explained by examining at Fig. 4-9. The figure shows an endgame engagement where the vehicle is following a reference path that consists of two segments that represent the path associated with ball's trajectory prior to the second bounce and the the ball's trajectory prior to being caught. The magenta lines are circles of radius L_1, used to illustrate the arrival at the various waypoints. The

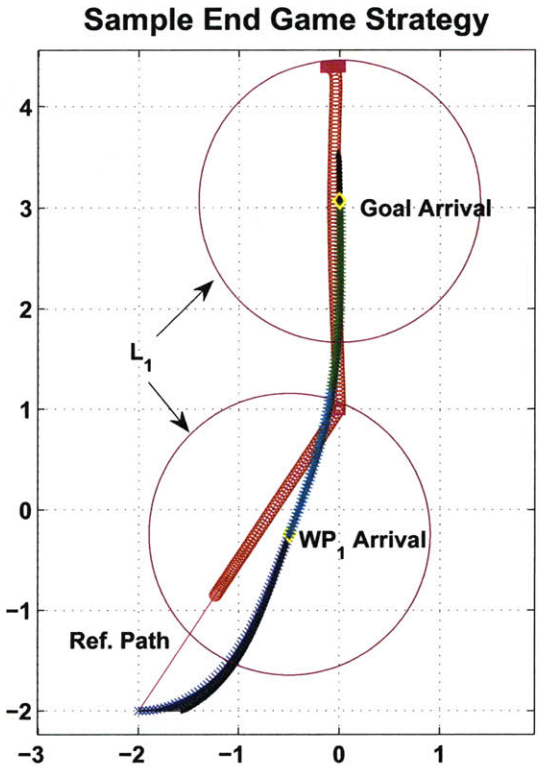


Figure 4-9: Sample Dynamic Reference Path Modification

vehicle's position at arrival to both waypoint #1 and the goal are marked with a yellow diamond. The red circles represent the target point[7] that the vehicle follows in order to properly track the reference path, and the blue markers and black arrows represent the vehicle's position and heading, respectively, as a function of time. The figure also includes cyan and green markers. The cyan markers correspond to points where the computational time exceeded the threshold of 10 msec, while the green marker correspond to points where the computational time were below threshold. It is clear that the computational time decreases as a function of distance to goal and as the vehicle lines up with the reference path; therefore, the parameters used in the end game must be carefully managed in order to minimize any computational costs while still ensuring the arrival of the vehicle at the desired goal point.

[7]The target point lies on the reference path

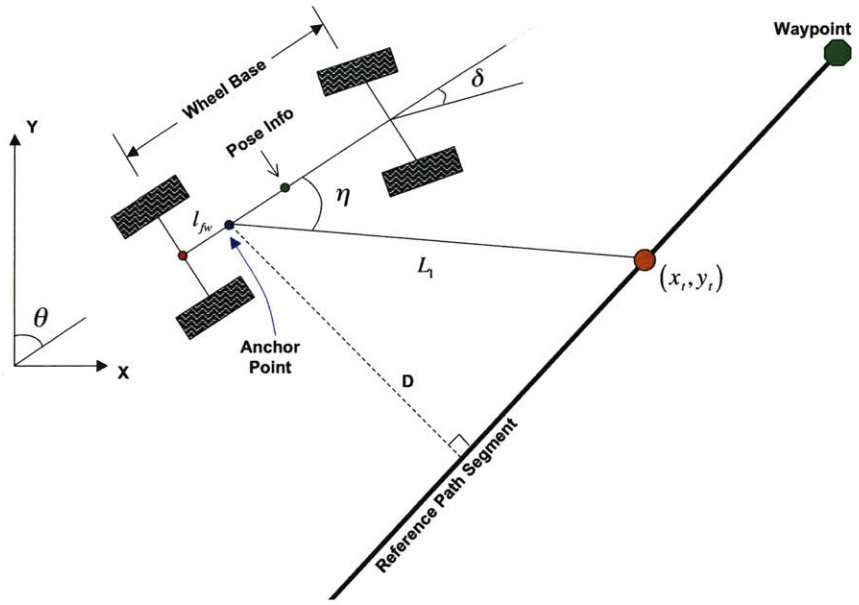


Figure 4-10: Geometry of Pure Pursuit Steering Controller

4.3 Control

4.3.1 Steering Controller

Pure pursuit controllers have been used extensively as steering controllers for autonomous ground vehicles [2, 3, 17, 20, 22]. The objective of this controller is to track a reference path by computing the error between the vehicle and a selected look-ahead reference point that lies on the path. The variables that define the pure-pursuit steering controller are shown in Fig. 4-10. In the figure, the vehicle position information (x, y, z) is referenced to the center of the vehicle, and the vehicle attitude, θ, is the angle between the y-axis and the line perpendicular to the rear axle's center point. The angles δ and η are the steering and the look ahead angle, respectively, where the lookahead angle is measured between the L_1 line segment and the vehicle center line. The look ahead distance, L_1 is defined as the line segment connecting the anchor point to the target point (x_t, y_t) in the reference path. The target point is obtained by selecting the forward[8] intercept point between a circle of radius L_1 centered at the

[8]The forward intercept point is defined as the point nearest to the goal location

anchor point and the reference path. If the vehicle were to be located at a distance $d > L_1$ away from the reference path, then the target point is placed at the point on the reference path that is closest to the vehicle's position, which implies that the vehicle will move perpendicular to the reference path. The wheel base, (L_w), is the distance between the front and rear axle. The vehicle control point, also referred to as anchor point, is located at a distance l_{fw} in front of the rear axle's center point. It has been shown by Kuwata et al. [14] that selecting a control point located in front of the rear axle center point (e.g. $l_{fw} > 0$) will result in better stability when compared to the conventional pure-pursuit controller ($l_{fw} = 0$). The same author has also shown that, in forward drive, the required steering angle that will place the anchor point in a collision course with the reference path is given by:

$$\delta = -\tan^{-1}\left(\frac{L_w \sin(\eta)}{\frac{L_1}{2} + l_{fw}\cos(\eta)}\right) \tag{4.7}$$

At the beginning of the engagement, the target point is co-aligned with the vehicle's initial position, which will cause the vehicle to travel directly to the first waypoint. Note that the initial target point could have been assigned to coincide with the position where the ball was initially tracked. This is a perfectly valid strategy but there is a subtle difference in performance, namely the total distance that the vehicle would travel prior to arriving at the first waypoint. By selecting the vehicle position as the first target point, the vehicle will take the most direct route to the first waypoint, while the latter strategy will direct the car first to the perpendicular line connecting the car with the reference path and then to the first waypoint. Given that timelines are of the essence, the first strategy was selected not only at the beginning of the engagement but also in the endgame.

The selection of the L_1 distance is very important in order to maintain steering control stability. A longer look ahead distance generates smoother steering control signals but at the expense of tracking errors or lateral steady state errors. On the other hand, shorter look ahead distances may reduce tracking and lateral errors but may induce oscillatory or unstable vehicle behavior. As discussed in [14], a good

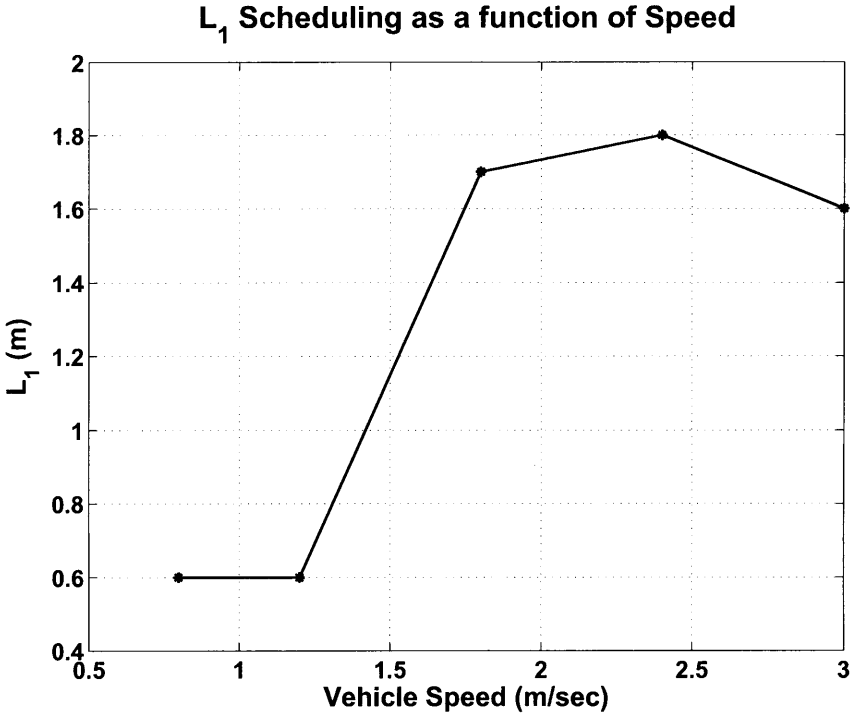


Figure 4-11: L_1 scheduling vs. Vehicle Speed

strategy is to schedule the look-ahead distance as a function of vehicle speed. This relationship was derived through experimentation yielding the results depicted in Fig. 4-11.

Another experiment was conducted in order to map the relationship between commanded steering and actual steering (this is done primarily to correct for trimming.) The vehicle was issued various steering commands while traveling at a nominal constant speed, and the vehicle's position was recorded using RAVEN's tracking systems. Position data was used to compute the curvature of the vehicle path and thus deduce a turning radius. Given the turning radius, r, the vehicle steering, δ, was computed using $r = D/2 + L_w/\sin(\delta)$, where L_w ad D are the vehicle wheel base and track[9], respectively. The results are shown in Fig. 4-12 along with a simple linear regression to extract the slope (m_s) and bias (b_s) of the vehicle's steering. The final steering

[9]Track is defined as the distance between the front wheels

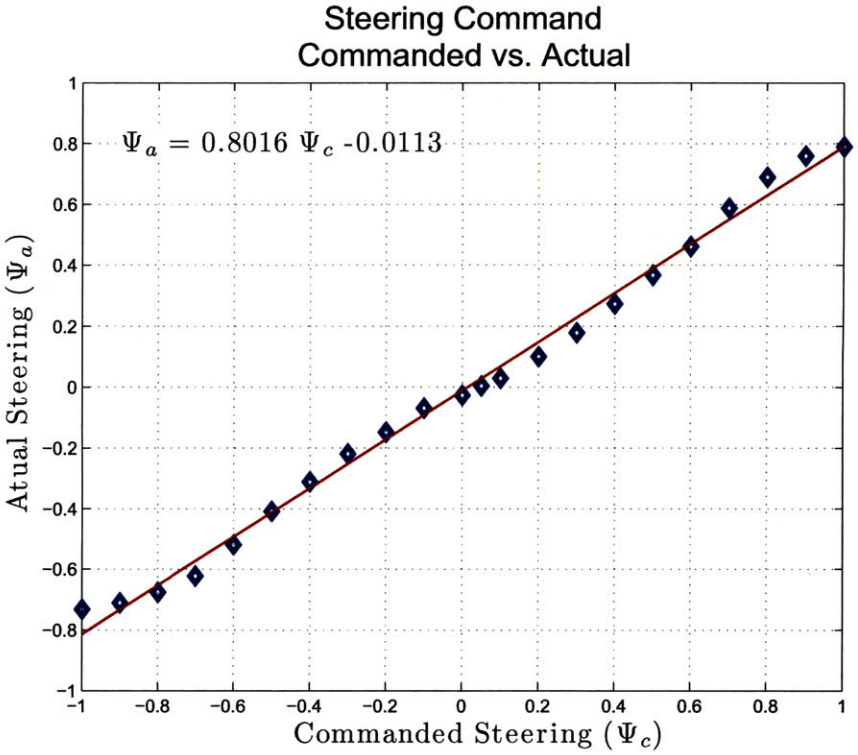


Figure 4-12: Steering Angle vs. Commanded Steering

command implemented in the code was:

$$\psi_a = \begin{cases} \alpha_1 m_s \psi_c + \alpha_2 b_s & |\psi_c| \leq 0.9 \\ \texttt{sat}(\psi_d/0.9) & \text{otherwise} \end{cases} \tag{4.8}$$

where: ψ_c is the commanded steering, ψ_a is the actual steering command sent to the truck, and α_1 and α_2 are tunable parameters.

4.3.2 Position Controller

The controller selected to manage the timely arrival of the vehicle to the catch point is depicted in Fig. 4-13. The controller is composed of a feedforward path and a proportional controller [15, 24]. The feedforward term computes the nominal throttle required to traverse the distance from the vehicle to the catch point in the allotted time. The feedforward model was developed from empirical data by collecting the

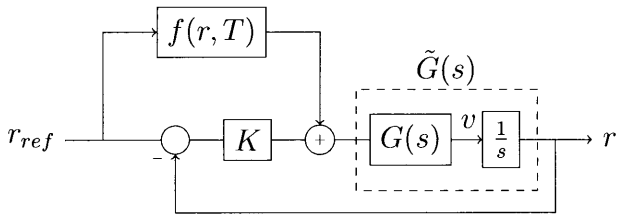


Figure 4-13: Position Controller

vehicle response to various throttle step commands. The vehicle was placed at the farthest point in the room (that would allow proper tracking) in order to provide the vehicle with ample distance to reach steady state speeds. The Motion Analysis Tracking system was used to collect vehicle position and velocity for various throttle step commands ranging from 0.2 to 1.0. In order to account for vehicle performance variability, the experiment was repeated several times for each step command. The data collected represents the vehicle's response to throttle step commands. The next step was to fit an exponential function of the form $V_{ss}(1 - e^{-t/\tau})$ to each data set. A sample set is shown in Figure 4-14, where the data collected is shown in blue and the exponential fit in green. The time constant, τ, as a function of throttle setting was fairly constant leading to a single expression that characterizes the vehicle's step response, namely:

$$V = V_{ss}\left(1 - e^{-\frac{t}{\tau}}\right) \tag{4.9}$$

The vehicle's total displacement can be found by integrating Eq. 4.9 from $[t, T]$:

$$\frac{ds}{dt} = V_{ss}\left(1 - e^{-\frac{t}{\tau}}\right)$$

$$\int_t^T ds = s_T - s = \int_t^T V_{ss}\left(1 - e^{-\frac{u}{\tau}}\right) du$$

$$\Delta s = V_{ss}\left((T - t) + \tau\left(e^{-\frac{T}{\tau}} - e^{-\frac{t}{\tau}}\right)\right)$$

In the event that the vehicle commences the engagement with a non-zero initial velocity, V_0, Eq (4-14) can be modified as follows: $V = V_{ss}\left(1 - e^{-\frac{t}{\tau}}\right) + V_0$, $t \geq 0$.

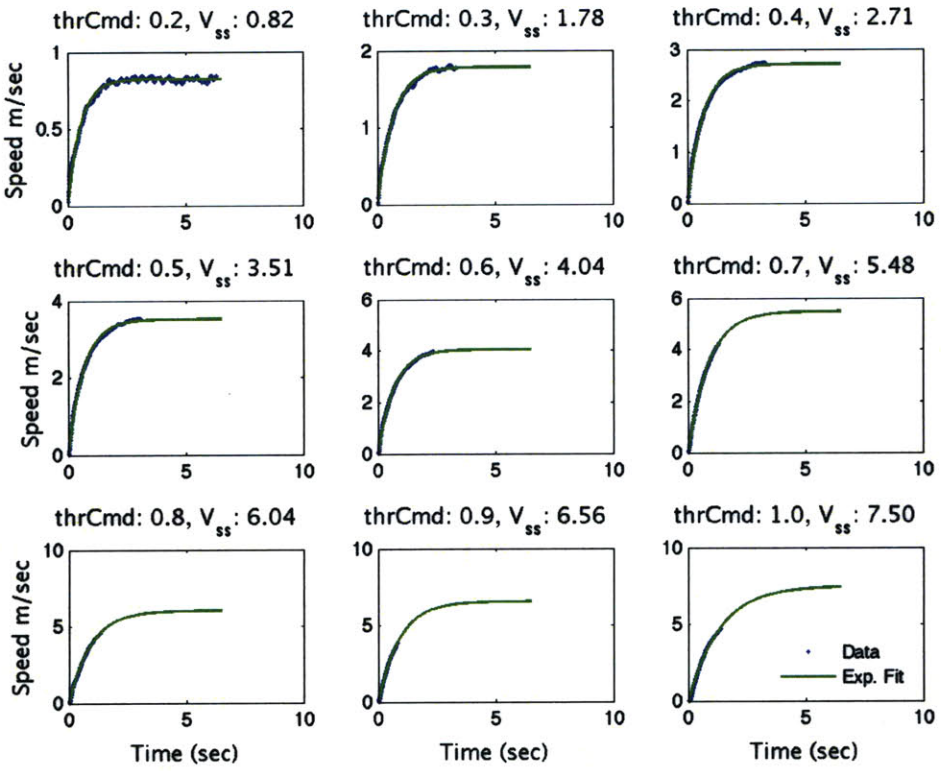


Figure 4-14: Throttle vs. Speed

The total displacement is now:

$$\Delta s(t) = V_{ss}\left((T-t) + \tau\left(e^{-\frac{T}{\tau}} - e^{-\frac{t}{\tau}}\right)\right) + V_0(T-t) \tag{4.10}$$

The profiles in Fig. 4-14 can also be used to derive a relationship between the vehicle's steady state vehicle speed, V_{ss}, and throttle setting, th. The results are shown in Fig. 4-15, which shows a linear relationship for throttle setting ranging from 0.2 to 0.7. At higher throttle settings, the relationship becomes non-linear and ultimately settles to a (derived) maximum steady state speed of 7 m/sec. Note that given the limited size of the room, the vehicle did not reach steady state for throttle settings higher than 0.7 and, therefore, the stated maximum steady state speed is

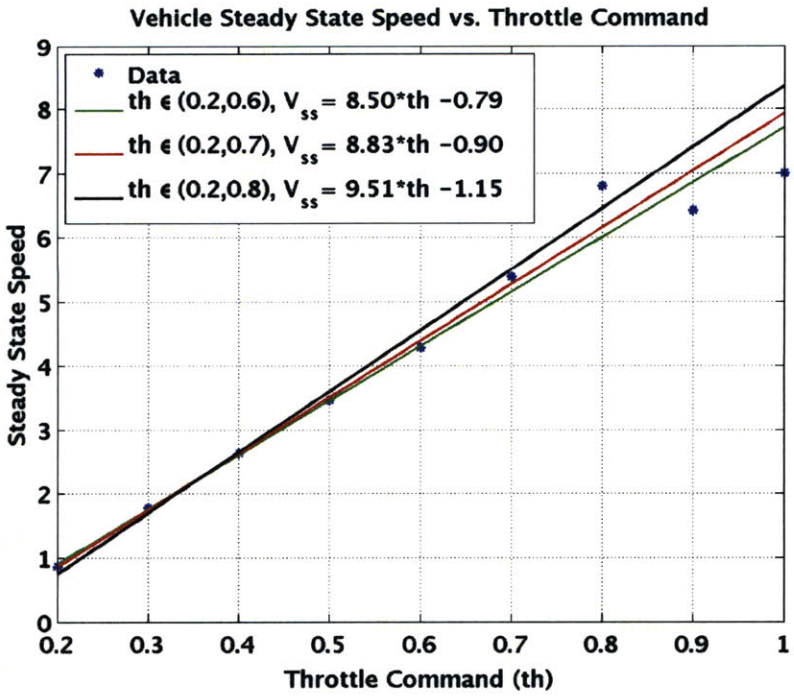


Figure 4-15: Throttle vs. Steady State Speed V_{ss}

inferred and not measured directly. The linear relationship can now be written as:

$$th = AV_{ss} + B \tag{4.11}$$

Combining Eq (4.11) and Eq. (4.10) yields

$$th = \frac{1}{A}\left(\frac{\Delta s - V_0(T-t)}{\left((T-t) + \tau\left(e^{-\frac{T}{\tau}} - e^{-\frac{t}{\tau}}\right)\right)} - B\right) \tag{4.12}$$

In essence, the feedforward term is attempting to model the inverse of the plant dynamics, so that (see Fig. 4-13):

$$f(r,T) \approx \tilde{G}^{-1}(s)$$

Clearly, it is naive to believe that it is possible to obtain or physically realize such an inverse; hence, one must assume that model uncertainty and potential disturbances

will induce errors that must be removed via feedback control. The controller selected is a proportional controller that augments the feedforward throttle signal in an attempts to null the positional error. As shown in Fig. 4-13, the control signal into the plant is:

$$u = K_p(r_{ref} - r) + f(r_{ref}, t_c) \tag{4.13}$$

Given the step response curves shown in Fig. 4-14 and the fact that these responses are approximated by an exponential of the form, $V = V_{ss}(1 - e^{-t/\tau})$ then it is reasonable to expect that, to first order, the plant is of the form:

$$G(s) = \frac{Vss}{\tau s + 1} \tag{4.14}$$

and adding an integrator to obtain position yields

$$\tilde{G}(s) = \frac{Vss}{s(\tau s + 1)} \tag{4.15}$$

The system error can be computed as follows (where the capital letters refer to the Lapace transform of each term):

$$E = R_{ref} - (kE + F)\tilde{(G)}$$
$$E(1 + k\tilde{G}) = R_{ref} - F\tilde{G}$$

and threfore

$$E = \frac{R_{ref}}{1 + k\tilde{G}} - \frac{F\tilde{G}}{1 + k\tilde{G}} \tag{4.16}$$

Where F is the Laplace transform of $f(r_{ref}, t_c)$. The steady state error to a unit step input is given by:

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) \tag{4.17}$$

$$= \lim_{s\to 0} s\left[\frac{1}{1 + k\tilde{G}}\frac{1}{s} - \frac{F\tilde{G}}{1 + k\tilde{G}}\right] \tag{4.18}$$

$$= \lim_{s \to 0} \left[\frac{1}{1 + k\tilde{G}} - s \frac{F\tilde{G}}{1 + k\tilde{G}} \right] \tag{4.19}$$

Since $\tilde{G}$ is a Type I system the first term in Eq. 4.19 goes to zero and all that is left is:

$$e_{ss} = -\lim_{s \to 0} s \frac{F\tilde{G}}{1 + k\tilde{G}} = -\lim_{s \to 0} s \frac{F}{k} \tag{4.20}$$

Since $f(r_{ref}, t_c)$ is a bounded integrable function, then $e_{ss} \to 0$.

Chapter 6 will show the results of the integration of the steering and position controller.

Chapter 5

Obstacle Avoidance

5.1 Introduction

Obstacle and collision avoidance is a capability that is present in all autonomous vehicles that operate in a clutter environment. Obstacles can be either stationary or dynamic. For the case of stationary obstacles, their physical location may or may not be known to the vehicle. For the latter case, the vehicle would require either on-board sensors to autonomously explore the environment [21] or support from outside agents to properly map the operating environment. For this particular application, the stationary objects are assumed to have known locations and this location is readily available to the path planner.

There are various obstacle avoidance and path planning strategies in the literature [4, 5, 19, 23, 25]. Of particular interest is the path planning strategy presented in [1, 9]. Dubins showed that for a vehicle with turning radius constraints, the optimal path that transports it from an initial state, specified in terms of position and heading, to a final state is composed of a series of arcs of circumference and line segments. This work seeks to apply a similar strategy with some modifications in order to fit the original architecture. It is clear that the modified strategy is sub-optimal (with respect to minimal distance), but it attempts to minimize vehicle skidding throughout the engagement.

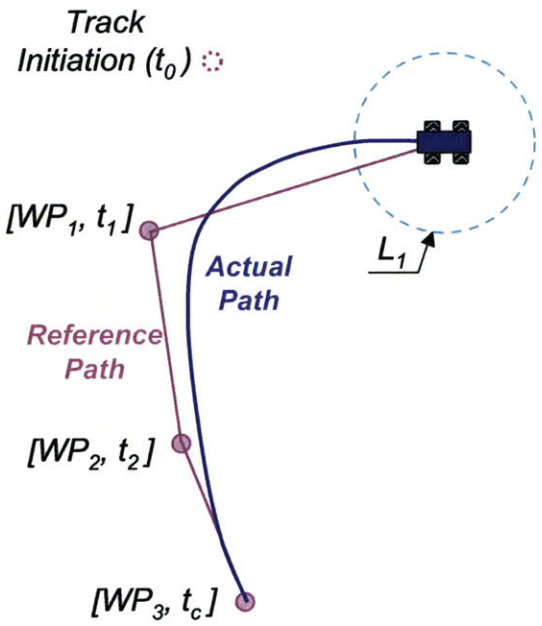
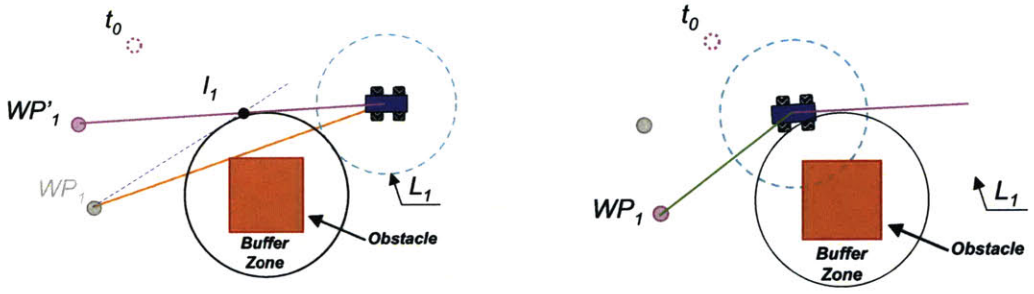


Figure 5-1: Nominal Reference Path Construction

5.2 Obstacle Avoidance Strategy

Chapter 4 covered in detail the path planning strategy for an obstacle free environment. The reference path is constructed by using line segments to connect the predicted impact locations (WP_i) as shown in Fig. 5-1. Note that the first line segment is always constructed by connecting the initial vehicle's position to the first waypoint. The figure also shows the target location at track initiation along with the times associated with each waypoint (i.e. the predicted time associated with each impact prediction location.) The figure also shows (in blue) a nominal vehicle trajectory that would result in tracking the reference path and a circle of radius L_1 centered on the vehicle, which is used as a guide to determine arrival to a particular waypoint as described in Section 4.2.

When an obstacle is present, the path planner needs to determine if the reference path places the vehicle on a collision course with the obstacle. In this application, the obstacle is surrounded by a circular buffer zone which the vehicle must avoid. This buffer zone provides some margin to protect the test asset, but more importantly



(a) Obstructed Path (b) Unobstructed Path

Figure 5-2: Reference Path Modification

it simplifies the mathematics of collision avoidance. Once the planner determines that the reference path penetrates the buffer zone, it modifies the path by selecting a different waypoint, thus constructing a temporary path that drives the vehicle away from the obstacle. The selection of the new waypoint is best explained by examining Fig. 5-2(a). As stated before, the initial reference path (in orange) connects the vehicle to the first waypoint, WP_1. Since the path penetrates the buffer zone, a new waypoint must be selected. The planner first constructs two tangent lines: one from the vehicle's location to the buffer zone and the other from WP_1 to the buffer zone. The planner then computes the intersection of these two tangent lines and draws a line, l_1, connecting the vehicle's current location to the intersect point. The new waypoint,WP'_1, is computed by extending the intersect point a distance $d > L_1$ along l_1. The reason for extending the waypoint location is similar to that explained in Section 4.2, where the physical arrival of the vehicle to the waypoint is required if the vehicle is to traverse the obstructing arc of the buffer zone and continue towards the *original* waypoint.

Before proceeding, it is important to differentiate between actual waypoints, WP_i, and temporary waypoints, WP'_i. The former are maintained in a master list used to construct the desired pure pursuit reference path, while the latter are just temporary waypoints uses by the planner to traverse along the arc of the buffer zone. Therefore, the process of computing temporary waypoints is repeated until the vehicle has clear

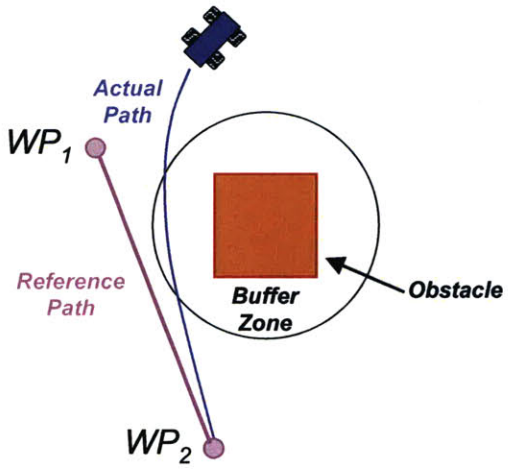


Figure 5-3: Possible Breach

line of sight to the *original* waypoint. When this happens, the planner immediately modifies the path to send the vehicle to WP_1 as shown in Fig. 5-2(b).

Having clear line of sight from the vehicle to the waypoint is very important because there may be a situation where the reference path lies completely outside the buffer zone, and yet the vehicle may penetrate the buffer zone when tracking the path (see Fig. 5-3). To prevent this situation from happening during the second segment of the reference path[1] the process of finding a tangential path to the obstacle must be repeated except that this time the initial waypoint of the reference path is not the vehicle location, but WP_1. The reason for maintaining WP_1 as the initial waypoint for this path segment was to attempt to line up the vehicle with the ball's trajectory as soon as possible. This seemed reasonable since the ball's deflection after each bounce is *generally* in the same direction as the motion prior to the bounce. If the ball were to continuously wrap around the obstacle, then it would be best to maintain the initial point of the reference path with the vehicle. The implication of maintaining WP_1 as the initial point of the reference path is that now the vehicle will be required to follow a path that connects WP_1 with the temporary (extended) waypoint, WP_2', as shown in Fig. 5-4. As before, the process is repeated until the

[1]In our application the second segment is the path from the first to the second bounce

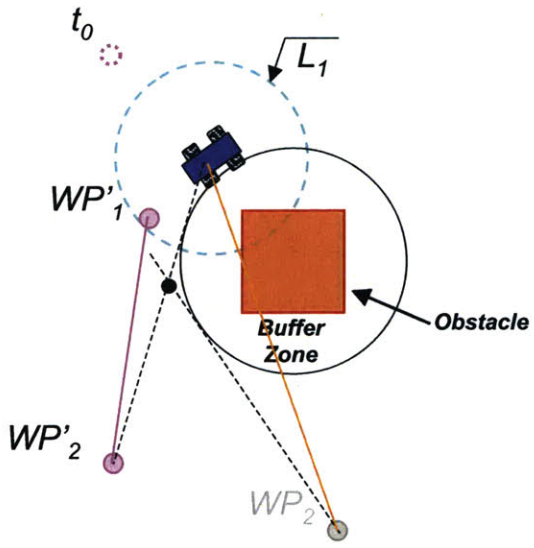


Figure 5-4: Reference Path Modification for Segment #2

vehicle has clear line of sight to the next *original* waypoint, WP_2. Clearly, the vehicle will trace an arc as it attempts to avoid the buffer zone, but this arc is not optimal in that it has a larger radius than if the vehicle were to continue on the perimeter of the buffer zone (which is the optimal path found by Dubins.).

The best way to test and visualize how the algorithm works is through modeling. A MATLAB simulation was developed to test the obstacle avoidance algorithm for various vehicle initial conditions and waypoint selections. The results of one such run are shown in Figures 5-5 to 5-7, where the selected geometry is somewhat arbitrary and not representative of an actual ball trajectory; however, it illustrates the functionality of the algorithm. The black dotted line in each figure represents the engagement area, which is a smaller footprint of the RAVEN room, mainly to provide a buffer to protect the car. As before, the obstacle (a column in the middle of the room) is shown in red surrounded by the buffer zone (in gray). The orange and green lines represent obstructed and unobstructed lines of sight, respectively, from the vehicle to the current waypoint, while the magenta circles represent waypoints (both original and temporary.) The vehicle position is marked with a blue "x" and the black line (arrow) indicates the vehicle heading. The red circles are the target points (or what

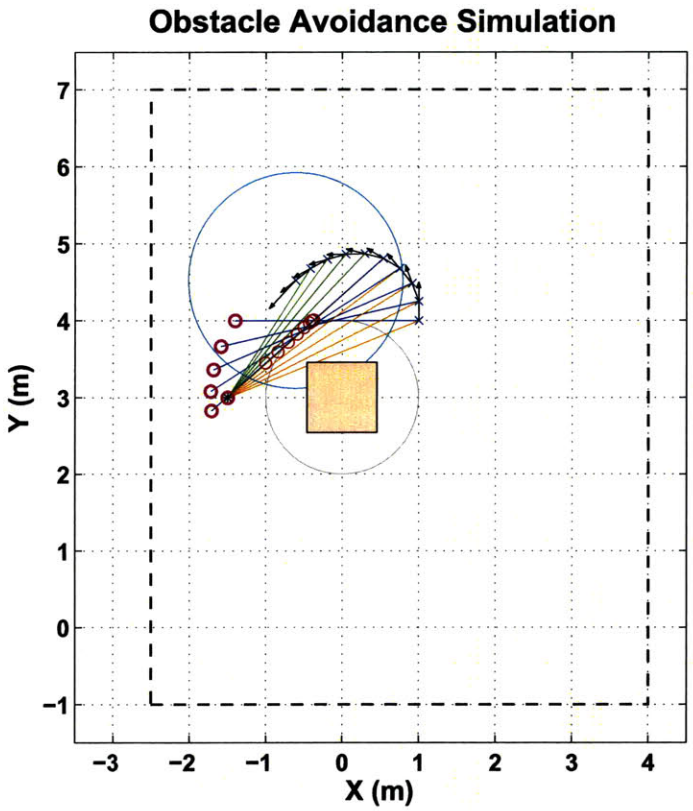


Figure 5-5: Reference Path: Segment #1

some authors prefer to call "carrot point") which always lie on the reference path. These target points provide great insight as to how the reference path continuously changes. As before, the cyan circle of radius L_1 is centered at the vehicle location and is used to determine arrival to a waypoint. In this particular engagement, the vehicle starts with a heading of 90^0 measured from the x-axis, and moves at a constant speed of 2.5 m/sec.

Fig. 5-5 shows the evolution of the engagement during the first segment of the reference path. Notice that when the vehicle's line of sight to the waypoint is blocked, a new temporary waypoint is assigned and the vehicle follows a modified reference path; however, as soon as the original waypoint is unobstructed, the desired reference path, which connects the unobstructed vehicle to the first waypoint, is established and the vehicle now heads for the original waypoint. (Notice also how the target points (red circles) follow this reference path.) Once the vehicle completed the first segment of the path, the simulation marks its location with a red circle (indicating

arrival to the waypoint) and the planner plots a course to the next waypoint; however, the vehicle's line of sight to this new waypoint is blocked so a temporary path must be computed, as shown in Fig. 5-6.

Note how the target points (red circles) trace a circle as the planner attempts to steer the vehicle around the buffer zone and towards the next waypoint. Again, as soon as there is clear line of site between the vehicle and the next waypoint, the vehicles heads towards it and the target point converges to the line connecting the previous waypoint, WP_1 with the next waypoint, WP_2. Upon completion of the second segment of the reference path, the planner plots a course to the final waypoint by simply connecting WP_2 with WP_3 (the final waypoint.) This is shown in Fig, 5-7. Notice that for this final segment, there is no obstruction and the vehicle heads toward the catching location.

Chapter 6 will show a visualization of the obstacle avoidance algorithm from an actual engagement.

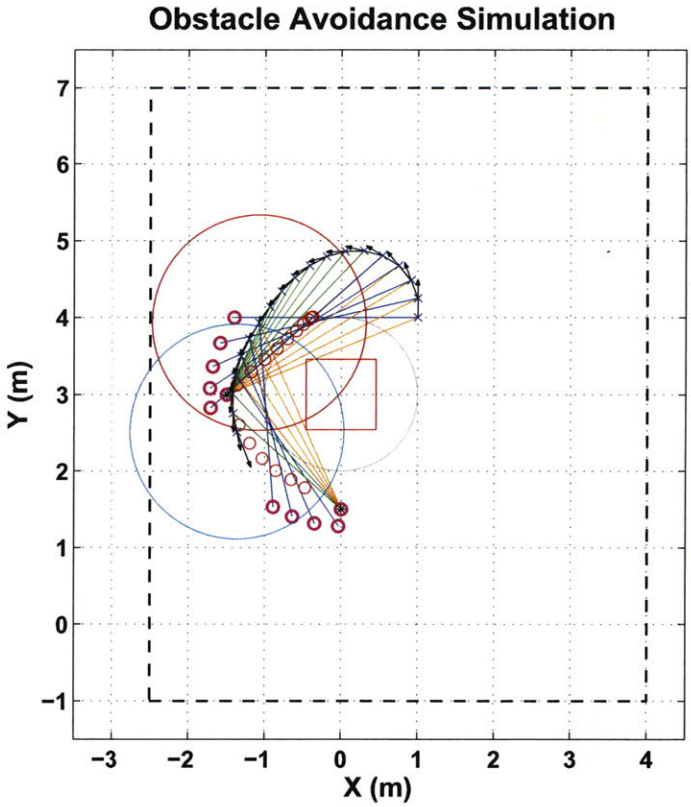


Figure 5-6: Reference Path: Segment #2

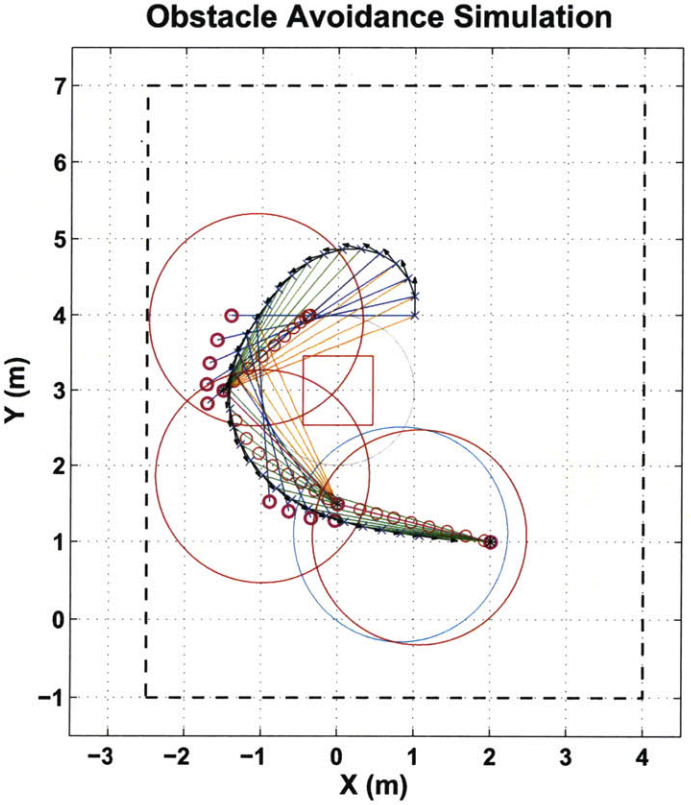


Figure 5-7: Reference Path: Segment #3

Chapter 6

Results

6.1 Introduction

The previous chapters described in detail the various strategies selected to address the different technical aspects of the intercept problem. Each chapter also presented either test results or simulation results that support either the validity or the merit of the method selected. The goal of this chapter is integrate all the the proposed solutions and present results at the system level. The results are presented in two sections one of which handles nominal obstacle free engagements and the other which addresses static obstacle avoidance. Each section includes a series of plots that help visualize how the integrated system operates. At the end of the section, a table is included that shows the catching success percentage as well as plots that quantify the prediction error in catch location computed at the time the interceptor catches the ball or, in the case of a miss, at the point of closest approach.

6.2 Open Field Intercepts

The open field intercept engagements are categorized as either Type I or Type II. In a Type I engagement the vehicle's initial heading is kept within $\pm$ 45 degrees of the ball's initial trajectory. This is technically the least challenging engagement in that the vehicle is almost lined up with the ball's initial trajectory. This, of course, does

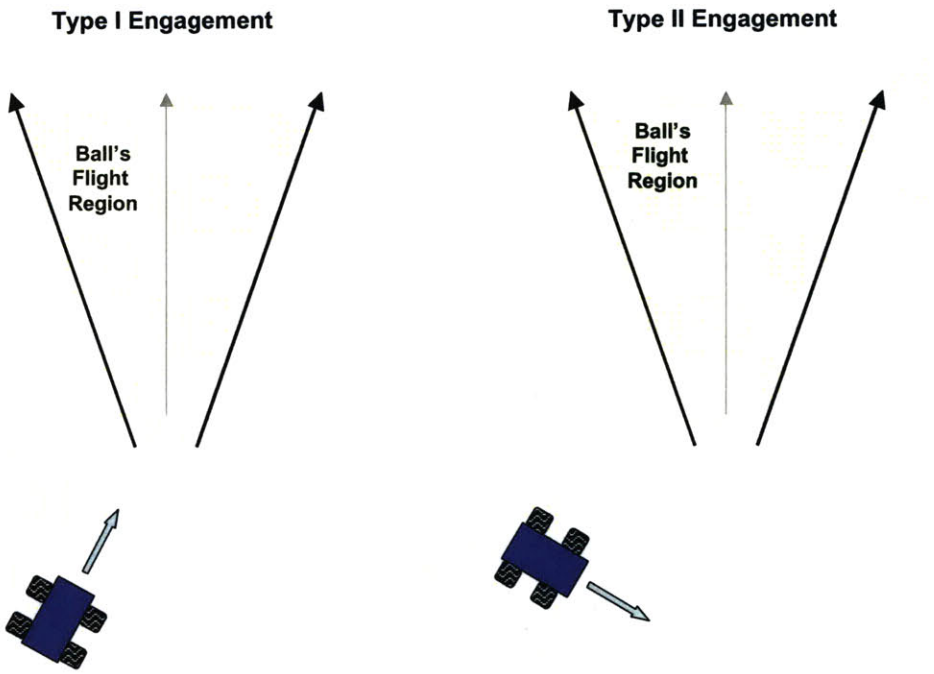


Figure 6-1: Intercept Engagements

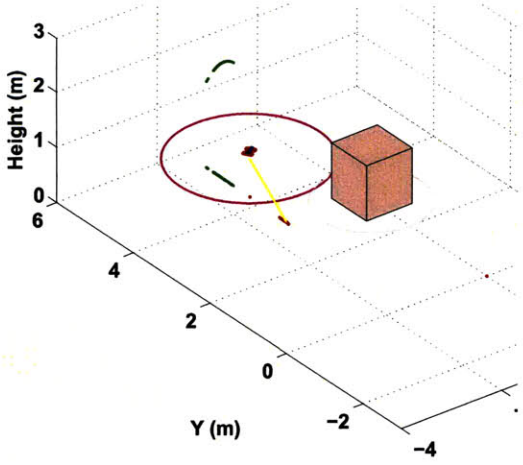
not detract from the complexity of the problem, especially when the ball's trajectory can randomly change at each bounce. In a Type II engagement, the relative angle between the vehicle's initial heading and the ball's initial trajectory is at greater than 45 degrees and the vehicle is also displaced a larger distance away from the ball's release point. This added requirement increases the complexity of the problem in two ways: first, the vehicle must maneuver to align it's trajectory to the ball's trajectory (e.g. reference path) and second, given the vehicle's non-holonomic constraints, this maneuver may induce skidding or drifting. Either way, both of these problems have a direct impact on the problem's timeline. Both engagement types are depicted in Fig. 6-1. Prior to data collection, a number of runs are performed in order to tune the gains of the steering and position controllers. The tuning of the steering controller is done by tasking the vehicle to follow a series of straight paths and comparing the vehicle's actual trajectory to the desired trajectory. The goal is to minimize the lateral error computed as the difference between the actual and desired trajectory

during the last segment of the reference path[1]. The tuning of the position controller requires the timely arrival of the vehicle to a desired location in a prescribed amount of time. Once the gains are selected, data collection commences simply by throwing the ball and having the vehicle catch it. The ball is released with and without spin in order to test the vehicle's robustness to flight path changes.

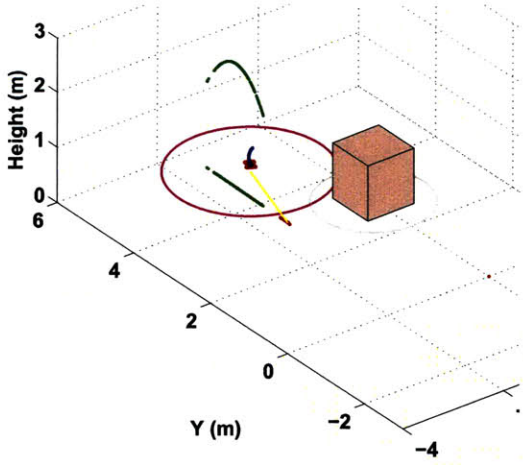
In order to obtain a better understanding of how each of the pieces contribute to the overall solution of the problem, let's examine Fig. 6-2. The figure is composed of six different snapshots of the engagement. The snapshots are organized into three groups each of which contains two plots showing the progress of the engagement prior to the first and second bounce and prior to catching the ball. Each panel depicts the engagement in three dimensions where the x and y axis are the cross-range and range respectively, and the z-axis is the height. The ball's trajectory is shown in green, and includes both the 3D motion and the motion on the plane (i.e. x-y plane). The car is shown in red with the relative heading as recorded by the tracking system. The vehicle's trajectory is shown in blue, while the yellow line connects the vehicle to the next waypoint. This is done primarily to shown the end point of the current reference path segment. The magenta circle around the car represents the look ahead (L_1) distance used for path tracking. The red dots represent the impact prediction locations (i.e. where the ball will impact the floor) for the first two bounces and the catch location for the last bounce. Finally, the red box represents a physical obstruction (a column) that is present in the RAVEN room and the grey circle is a buffer zone used for obstacle avoidance. All the plots shown in this figure were constructed using data from an actual engagement and, therefore, represent actual vehicle performance during the run.

This engagement is a Type II engagement as can be seen by the vehicle's placement and heading in Fig. 6-2(a). The vehicle is headed toward the first impact point which can be seen to change as the vehicle approaches. The change in impact location is a direct consequence of the Kalman Filter state estimation process which changes as

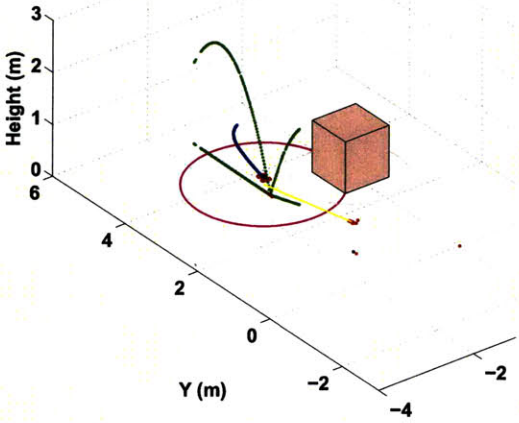
[1]Though the path is straight and all the segments that form the path are collinear, the vehicle is only required to align with the last segment, mainly because the vehicle is allowed to maneuver into position in the same manner as it would during an actual engagement.



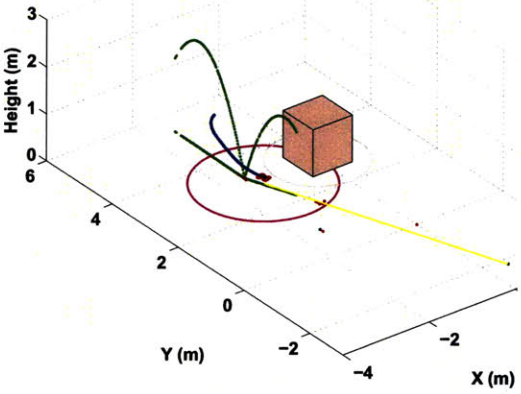
(a) Panel 1: Prior to First Bounce



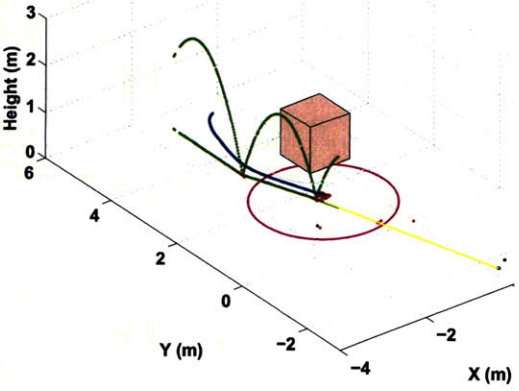
(b) Panel 2: Prior to First Bounce



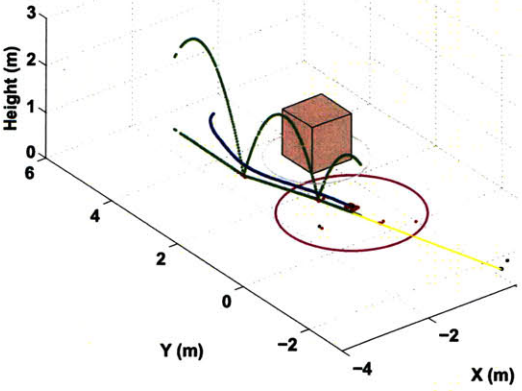
(c) Panel 3: Prior to Second Bounce



(d) Panel 4: Prior to Second Bounce



(e) Panel 5: Prior to Catch



(f) Panel 6: Prior to Catch

Figure 6-2: Run 23 Visualization: Type II Engagement

new ball information is made available to the filter. In the next snapshot (see Fig. 6-2(b)), the vehicle is starting to turn to line up with the reference path and is very close to arriving at the current waypoint. It is important to mention that the vehicle may arrive at a designated waypoint prior to the ball impacting the ground. In this case, the vehicle proceeds to the next waypoint and follows the new reference path segment. If the ball were to impact the ground prior to the vehicle's arrival to the waypoint, the reference path is updated and the process continues. Fig. 6-2(d) is a good example of the vehicle arriving to the waypoint prior to the second bounce. In this case, the reference path is updated and the vehicle heads to the estimated catch location. The accuracy of this estimated location will depend on the amount of deflection in the ball's trajectory after impacting the ground for the second time. Therefore, after the second bounce (or for that matter after each bounce), the estimate is expected to change. This can be seen in Fig. 6-2(e), where the final waypoint has moved slightly right of the last estimate. The vehicle is now well positioned to catch the ball, which indeed it does. A top view of the engagement (see Fig. 6-3) shows how well the pure pursuit controller was able to track the reference path and position the vehicle on a tail engagement (which resulted in a catch.)

The experiment was repeated 39 time, of which 29 were Type I engagements and 10 were Type II engagements. The results were grouped into three categories:

A. The ball landed in the cup.

B. The ball impacted the rim of the cup and bounced off.

C. The engagement was a miss.

For each category, the total number of occurrences were tabulated along with the success rate of each category computed as the ratio of the number of occurrences to the total number of runs. The results are compiled for each engagement type along with an overall success percentage across all engagements as shown in Table 6.1. Interestingly, the results for Type II engagements appear to be better than those for Type I engagements but this may be a consequence of the total number of runs in

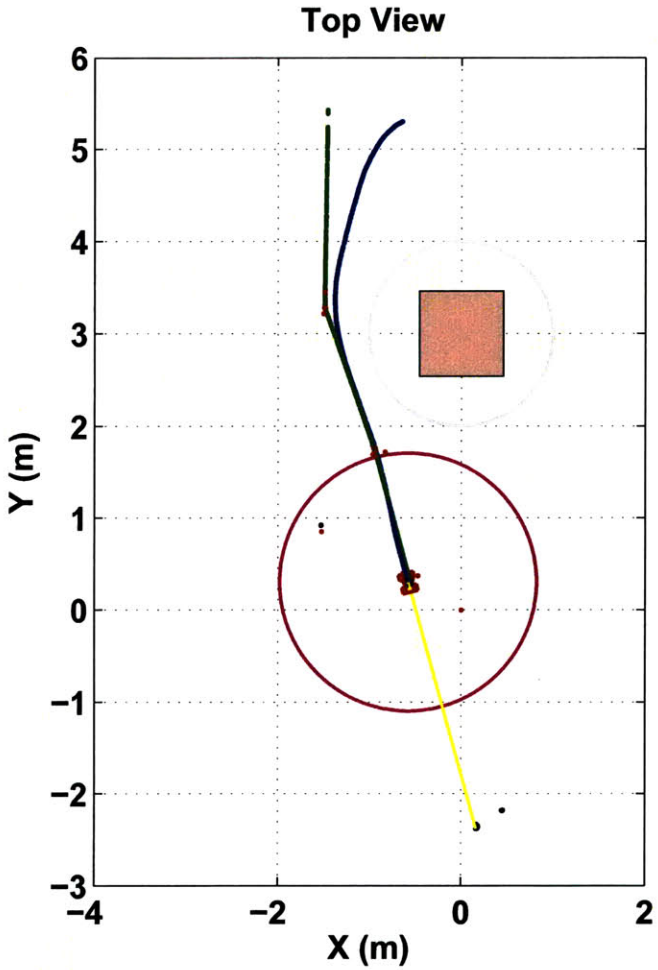


Figure 6-3: Run 23: Top View of the Engagement

Table 6.1: Results of Obstacle Free Engagements

	Type I		Type II		Overall
Category	Occurrences	Rate	Occurrences	Rate	Rate
A	20	69%	9	90%	74%
A+B	26	90%	10	100%	92%
C	3	10%	-	-	8%
Total	29		10		

each category. The overall catching percentage is just under 75% but if one were to count the engagements that hit the rim of the cup, then the percentages go as high as 92%. These percentages show that the position controller is doing a good

job at driving the vehicle to the intercept location in time. All that remains is to determine how well the impact predictor estimated the catching location. This was accomplished by generating a scatter plot of the actual and predicted ball locations at the instant the ball was caught or, in the event of a miss, the point of closest approach. Fig. 6-4 show the results of the scatter plot, where the green lines are used to join pairs of prediction and actual locations; the solid red line is the outline of the catching device and the dotted red line represent half of the radius of the ball (to be used to determine those engagements where the ball hit the rim of the cup). One important observation is that, in the pure sense of interception (i.e. collision of the interceptor and target), the vehicle successfully collides with the target in all the cases.

As mentioned in Chap. 3, the RMS error of the catching point's prediction (see Eq. 3.25) is a time varying quantity. At the end of the engagement, it is useful to determine the total error at the time, t_c, when the vehicle caught the ball or, in the event of a miss, at the time the ball physically crossed the catching plane. Numerically, the desired prediction error is just:

$$\epsilon(t_c) = ||\mathbf{P}_{Prediction}(t_c) - \mathbf{P}_{actual}(t_c)||_2 = ||\delta P(t_c)||_2 \tag{6.1}$$

The prediction error is shown in Fig. 6-5 where the mean and variance of the error are 1.36 and 0.86 cm respectively.

Fig. 6-6 show a series of snapshots that were extracted from the video recorded during a test run. From the first panel in easy to see that this was a Type II engagement. Though hard to discern, the vehicle lined up with the ball's trajectory in route to a catch as shown in Panel 6.

6.3 Obstacle Avoidance

Obstacle avoidance (OA) presented a challenge to the intercept problem in that the path planner needed to modify the reference trajectory in order to avoid known sta-

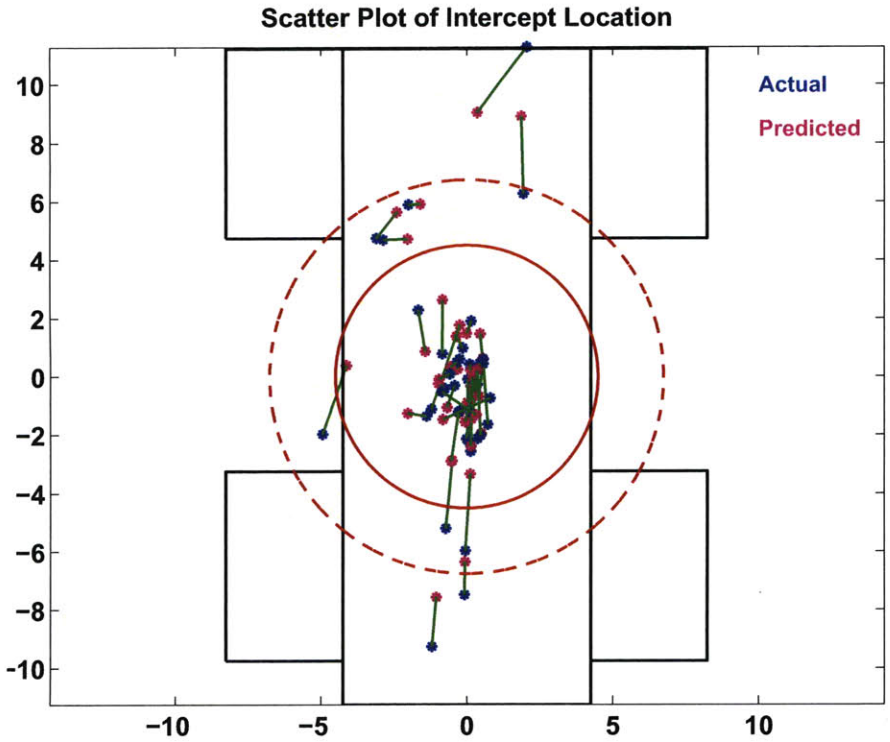


Figure 6-4: Scatter Plot of the Intercept Locations Measured at Time of Catch ($T_{c_3}^G$)

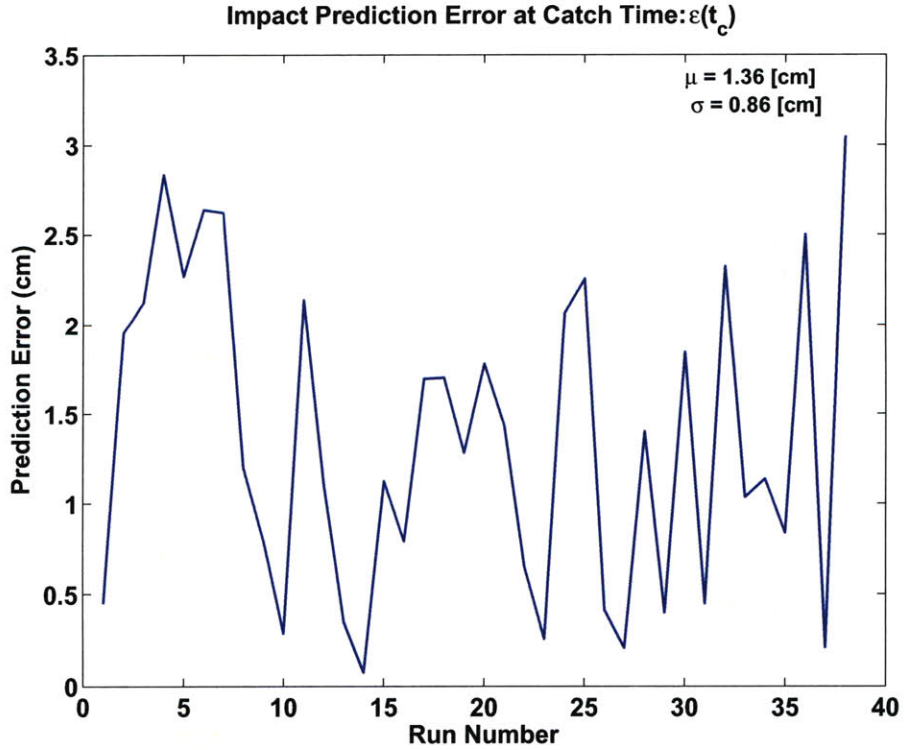


Figure 6-5: Error in the Prediction of the Intercept Point at the Time of a Catch

(a) Panel 1

(b) Panel 2

(c) Panel 3

(d) Panel 4

(e) Panel 5

(f) Panel 6

(g) Panel 7

Figure 6-6: Type II Engagement

tionary obstacles. The challenge was not just in path planning but also in maintaining the vehicle in an operational regime where slip and drift were minimized. In order to illustrate the OA algorithm, it is best to analyze the plots shown in Fig. 6-7. The contents of the plot are exactly the same as those described in the previous Section ??. Of particular interest are the red box and gray circle that surrounds the box. The red box is an actual column that is physically located inside the room, while the gray circle is a buffer zone that the vehicle must avoid during an engagement.

At the beginning of the engagement, the reference path is a straight line that connects the vehicle's initial position to the first waypoint which represents the predicted location of the first impact point. The path is shown as a yellow line in Fig. 6-7(a). Note that the reference path actually penetrates the buffer zone which will result in a collision unless the path is modified. The path planner identifies this conflict and selects a new reference path that is tangential to the buffer zone as described in Chap. 5. The vehicle starts tracking the new path until the moment where the vehicle has clear line of sight to the current impact location. When this happens, the reference path is re-computed and the vehicle is redirected to the current impact location. It is possible that prior to the vehicle arriving at the current waypoint, the ball impacts the ground for the first time and a new waypoint is identified. This situation is shown in Fig. 6-7(b) where a new reference trajectory is computed to direct the vehicle to the new waypoint (i.e. the second impact point.) Once again, the obstacle avoidance software determines if the new reference path penetrates the buffer zone, and if this is the case, the path is modified once again. The vehicle continues to drive tangentially to the buffer zone as shown in Fig. 6-7(d) until it clears the obstacle completely as shown in Fig. 6-7(e). All that is left is the end game, where the reference path is dynamically modified to ensure that the vehicle passes through the catch point. A top view of the entire engagement is shown in Fig. 6-8 in order to appreciate the effects of modifying the reference path to ensure that the interceptor avoids the obstacle's buffer zone. Notice that the interceptor successfully navigates around the buffer zone but the resulting path is not overly aggressive in that the vehicle does not travel around the circumference of the buffer zone. This is a direct result

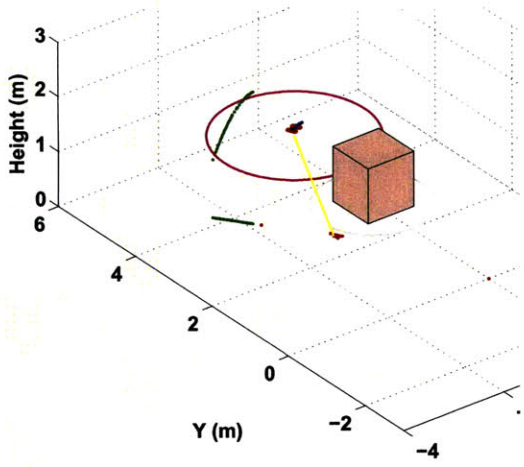
of the conservative approach chosen to navigate around obstacles (as explained in Chap. 5). Also note that the ball continuously changes direction after every bounce. This is where the end game path planning strategy demonstrates its capability by continuously changing the reference path in order to ensure the arrival of the vehicle to the catching point. The vehicle's final trajectory clearly shows that, depending on the vehicle's initial position and the ball's trajectory, the car may need to make very aggressive end game maneuvers which may result in degraded performance.

Twenty-four runs were conducted to test system level performance with obstacle avoidance capability. The results are shown in Table 6.2 where, as before, the outcome of each test was grouped into one of three categories: A, B or C (see previous section for description). These results are slightly lower than those obtained in the previous

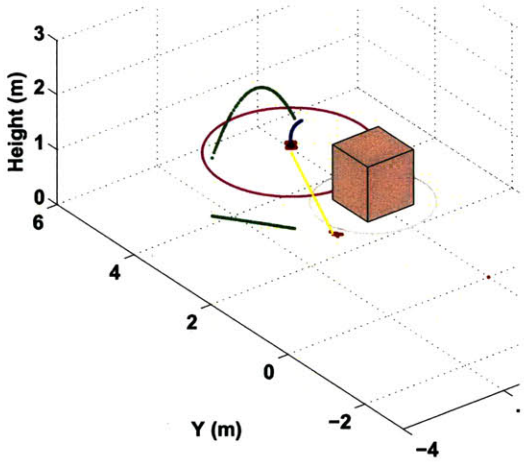
Table 6.2: Results of Obstacle Avoidance Engagements

	OA Results	
Category	Occurrences	Rate
A	16	67%
A+B	20	83%
C	4	17%
Runs	24	

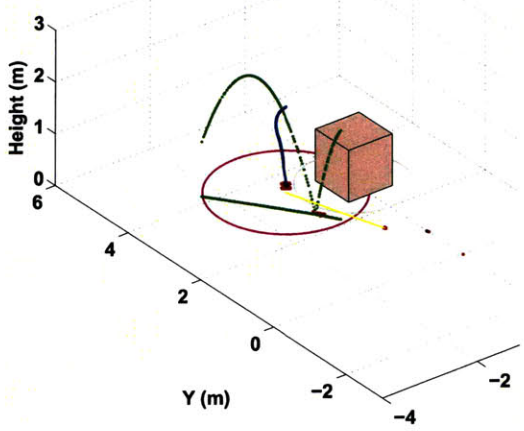
section (see Table 6.1). There are two reasons for this small decrease in performance. First, the obstacle avoidance engagements are more dynamically demanding than the Type I and II engagements, which causes the vehicle to push its performance envelope. In many cases, slipping was observed at the beginning of the engagement and during obstacle avoidance maneuvers. Secondly, these dynamic engagements cause the battery to drain faster than when performing Type I and II testing. As the battery's power is depleted, the vehicle's performance and the overall results are affected. For comparison purposes, a scatter plot comparing the predicted and actual catch locations is shown in figure 6-9, while the prediction error is shown in Fig. 6-10. The average impact prediction error is the same as the one computed for Type I and Type II engagements. Clearly the predictive aspect of the problem is quite sound and the results are robust; however this is to be expected because



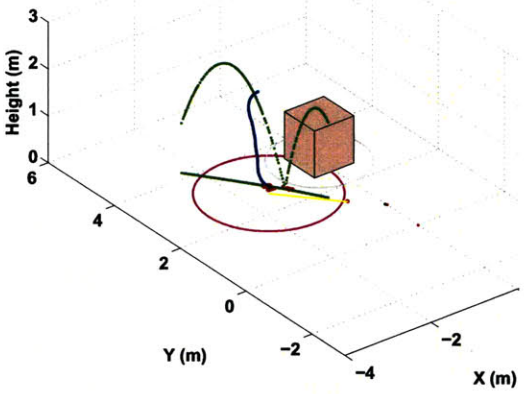
(a) Panel 1: Prior to First Bounce



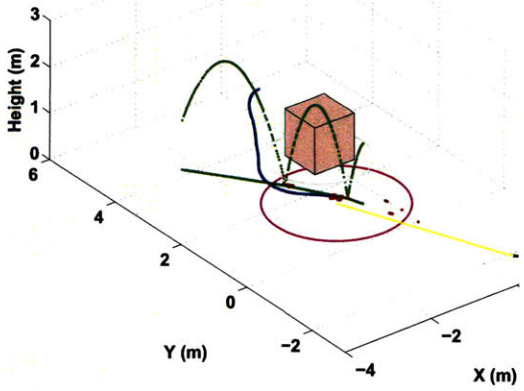
(b) Panel 2: Prior to First Bounce



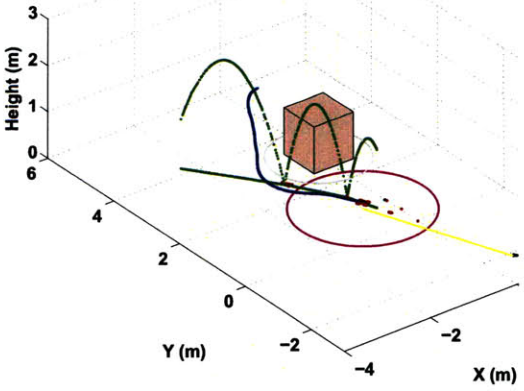
(c) Panel 3: Prior to Second Bounce



(d) Panel 4: Prior to Second Bounce



(e) Panel 5: Prior to Catch



(f) Panel 6: Prior to Catch

Figure 6-7: Run 31 Obstacle Avoidance

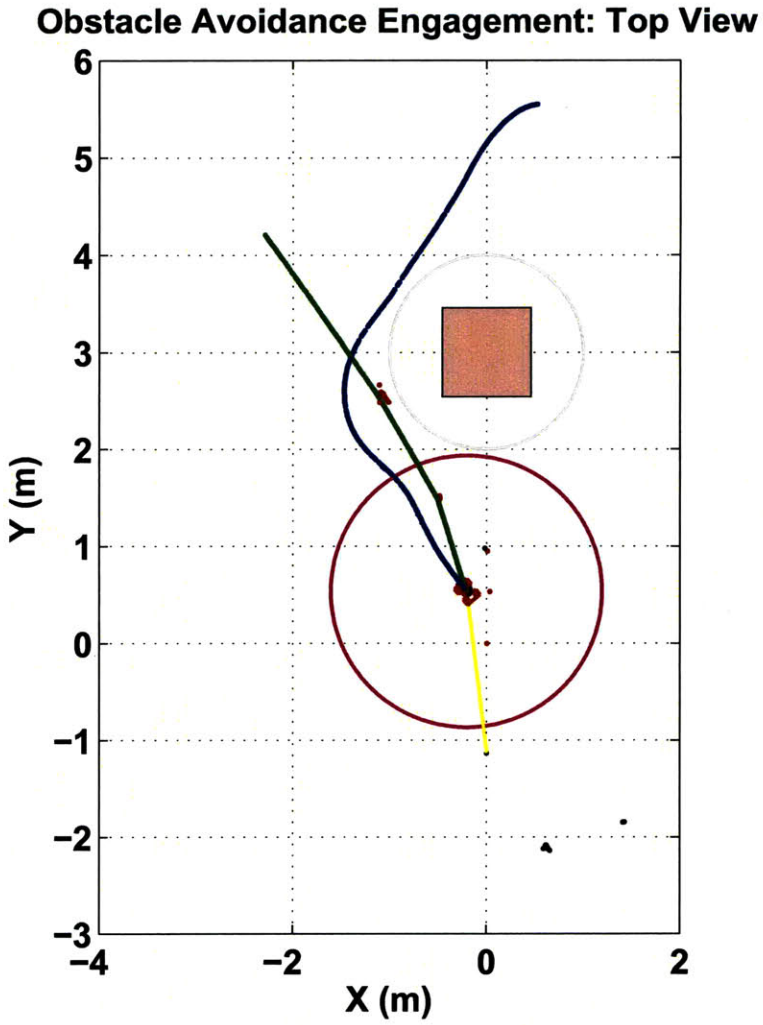


Figure 6-8: Run 31: Top View of Obstacle Avoidance Engagement

the tracking system is quite robust and reliable. The apparent variability is in the control aspect of the problem where variables such as battery life, vehicle skidding and drifting can adversely affect vehicle performance, which translates into degraded system performance.

Finally, Fig. 6-11 shows several snapshots from video collected during one of the runs. Panel 1 shows the vehicle's initial position (hiding behind the column) and heading (approximately 180^o from the ball's initial trajectory. Clearly, these are very challenging initial conditions. Panels 2 - 6 show the vehicle performing the equivalent

to a U-turn in order to intercept the ball, while Panels 7 and 8 show the vehicle returning to base with the ball inside its catching device.

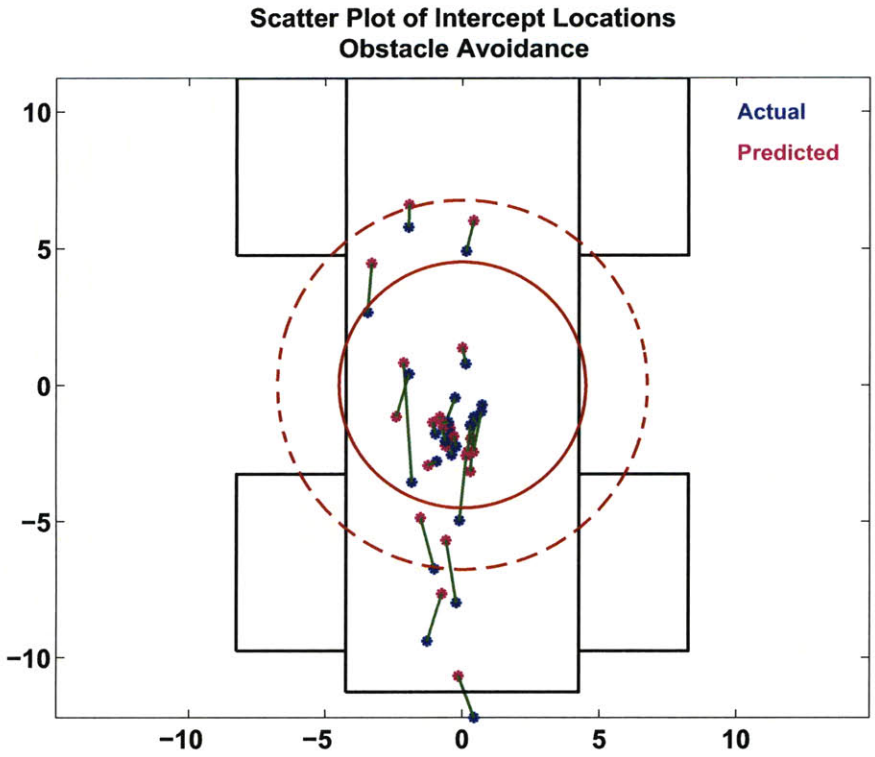


Figure 6-9: Scatter Plot of the Intercept Locations Measured at Time of Catch ($T_{c_3}^G$)

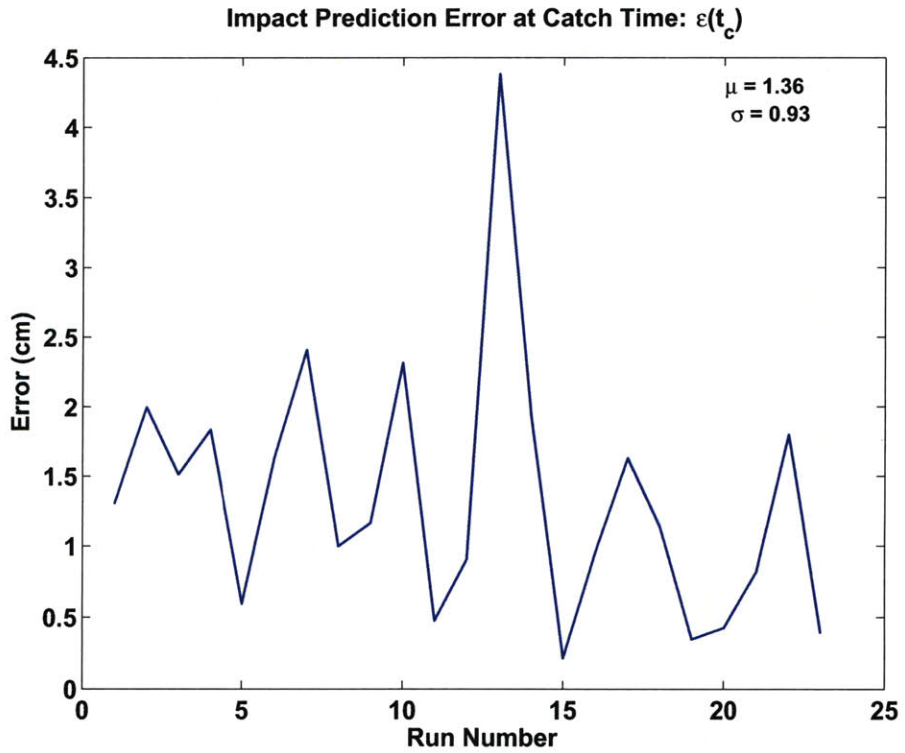
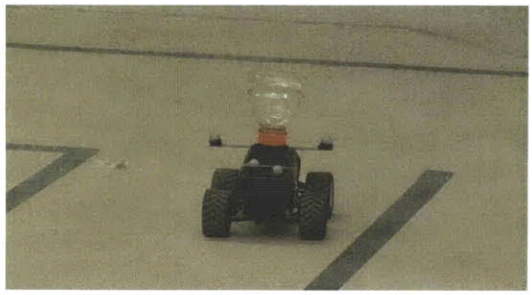


Figure 6-10: Error in the Prediction of the Intercept Point Measured at the Time of Catch (OA Engagements)



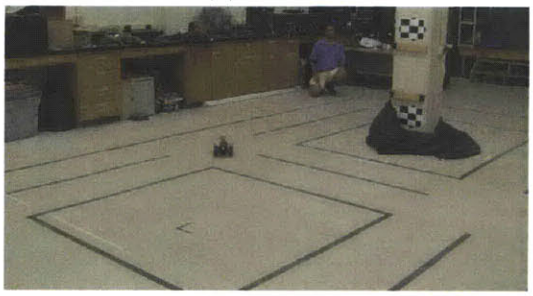
(a) Panel 1

(b) Panel 2

(c) Panel 3

(d) Panel 4

(e) Panel 5



(f) Panel 6

(g) Panel 7

(h) Panel 8

Figure 6-11: Obstacle Avoidance Engagement

Chapter 7

Conclusion

This thesis presents a solution strategy to the intercept problem that borrows from the strengths of proven methods used to tackle path planning, path tracking, target tracking, estimation, and intercept problems. The problem was solved in a piece-wise manner by focusing on each aspect of the problem somewhat independently and then integrating the solution. The problems considered were: detection and tracking, prediction and estimation, guidance and control, and obstacle avoidance.

The detection problem was addressed using track before detect which allows for the rejection of false targets thus reducing the number of tracks that must be maintained, especially when the problem demands the tracking of multiple objects. The use of pre-history in conjunction with track before detect is well known in the field of missile warning and integrated air defense systems since it allows for the recording of information that can readily be used by a discrimination algorithm upon track initiation. For this application, pre-history was used to compute the initial (a-priori) covariance and state estimates for use by a Kalman filter. The Kalman Filter was used to estimate the unknown (but constant) parameters that define the ball's motion in space and time. The impact predictor used the state estimates to predict future ball trajectory. Results showed that the impact predictor excelled at estimating the ball's bounce locations, as well as the final intercept location, with the caveat that the estimation accuracy of any particular bounce location, b_i depends primarily on the data collected after the last bounce, b_{i-1}.

The guidance and control problem required the generation of a collision course that would allow for the intercept *and retrieval* of the dynamic object. The path planning strategy is based on the concept of pure pursuit in which the interceptor's velocity vector is continuously pointed at the target, which results in the vehicle approaching the target in a rear aspect. Results show that the use of pure pursuit algorithms in conjunction with modified path planning in the endgame successfully positioned the interceptor in a rear aspect collision course with the target. The retrieval of the dynamic object depended on the accurate arrival of the interceptor to the catch location. This was achieved with a position controller that uses feedforward to bring the interceptor close to the catching point and feedback to null any error in the position estimate. Results showed that this control strategy proved to be successful at placing the interceptor at the catching point at the right time.

The intercept problem was augmented to include obstacle avoidance. The strategy for navigating around obstacles was base on research done in [9] with some additional modifications to account for vehicle dynamics and, more importantly, the geometry of the problem. Results show that modifying the path planning strategy to drive the vehicle around the obstacle while maintaining the vehicle in "pure pursuit" was sufficient to yield good intercept results, albeit at slightly lower success rate than the one obtained in an obstacle free engagement.

In summary, this thesis shows that the use of a pure pursuit controller for steering in conjunction with a position controller for timely arrival to a goal point proved to be a very viable solution to the intercept problem. Clearly, success in target interception hinged on the ability to modify the path planning strategy in the end game to allow more aggressive maneuvering and proper guidance of the interceptor to the target.

7.1 Future Work

During the course of this works, there were other solution strategies that merit further exploration, including modifications to the current control strategy and interesting extensions to the general problem of path tracking and target interception. These

ideas are presented below.

7.1.1 Battery Life Model

As mentioned in Chapter 2, there is some variability in the performance of the battery pack that powers the Duratrax RC vehicle. In addition, when the vehicle is required to perform highly dynamic maneuvers, as is the case for obstacle avoidance, the battery can drain rather quickly and the vehicle performance can be drastically impaired. This impact in performance can reflect directly in the ability of the vehicle to intercept targets successfully. An adaptive controller that could compensate for battery variability (within reason) could be a potential implementation to increase robustness in performance.

7.1.2 Dynamic Obstacle Avoidance

The results presented in this thesis were limited to stationary obstacles with known location. A more challenging path planning problem would be the addition of dynamic obstacles whose location is readily available through measurements. This is a very challenging problem given the non-holonomic constraints of the vehicle and the problem's timeline restrictions.

7.1.3 Multiple Agents - Multiple Targets

A very natural extension to this work is the tracking and interception of multiple targets. In this problem, the vehicle either can be required to intercept a single target or both targets. In the single target case, the vehicle must decide which engagement would yield the highest probability of intercept, while the multiple target problem would require the vehicle to plot a path that would enable it to catch the multitude of objects. Another variation would be the addition of a second (or multiple) intercept vehicle and the creation of a single master path planner that would not only determine which target is assigned to each vehicle but also would plot an intercept course for each vehicle. Of course, obstacle avoidance is inherent to this problem since the

vehicles must not collide with each other on their way to intercepting their respective target.

7.1.4 Emulation of Additional On-Board Sensors

It is reasonable to consider the case where the intercepting vehicle carries on-board sensors or additional instrumentation that allows for autonomous detection and tracking, similar to the manner in which passive homing missile detect and track intended targets.

Bibliography

[1] G. Ambrosino, M. Ariola, W. Ciniglio, F. Corraro, A. Pironti, and M. Virgilio. Algorithms for 3d uav path generation and tracking. In *IEEE Conference on Decision & Control*, 2006.

[2] O. Amidi and C. Thorpe. Integrated mobile robot control. In *Proceedings of SPIE*, volume 1388, pages 505–523, 1990.

[3] Omead Amidi. Integrated mobile robot control. Technical Report CMU-RI-TR-90-17, Robotics Institute, Pittsburgh, PA, May 1990.

[4] S. Badal, S. Ravela, B.A. Draper, and A.R. Hanson. A practical obstacle detection and avoidance system. In *WACV94*, 1994.

[5] A. Balluchi, A. Bicchi, A. Balestrino, and G Casalino. Path tracking control for dubin's cars. In *IEEE Intenational Conference in Robotics and Automation*, 1996.

[6] T.E. Bar-Shalom, Y. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[7] S. Blackman. *Multiple-Target Tracking with RADAR Applications.* Artech House, Inc, 1986.

[8] R. G. Brown and Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, 1997.

[9] L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[10] D.L. Hall and J. Llinas. *Handbook of MultiSensor Data Fusion.* CRC Press, 2001.

[11] J.P. How, A. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vechile test environment. *IEEE Controls Systems Magazine*, 2008.

[12] T. Junge. One dimension robot juggler. Master's thesis, Swiss Federal Institute of Technology in Lausanne, 2008.

[13] Y. Kuwata, G.A. Fiore, J. Teo, E. Frazzoli, and J.P. How. Motion planning for urban driving using RRT. In *IEEE/RSJ International Conference on Intelliget Robots and Systems, 2008. IROS 2008.*, 2008.

[14] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J.P. How. Motion planing in complex environments using closed-loop prediction. In *AIAA Conference on Guidance, Navigation, and Control*, 2008.

[15] P.H. Lewis and C. Yang. *Basic Control Systems Engineering.* Prentice Hall, 1997.

[16] P. Mazurek. *Automation and Robotics.* InTech Education and Publishing, 2008.

[17] K. N. Murphy. Analysis of robotic vehicle steering and controller delay. In *Fifth International Symposium on Robotics and Manufacturing (ISRAM)*, pages 631–636, Aug 1994.

[18] Paul J. Nahin. *Chases and Escapes.* Princeton University Press, 2007.

[19] A. Naik. Arc path collision avoidance algorithm for autonomous gournd vehicles. Master's thesis, Virginia Polytechnic Institute ad State University, 2005.

[20] A. Ollero and G. Heredia. Stability analysis of mobile robot path tracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 461–466, 1995.

[21] E. Park, W. Lee, and J. Kim. Practical study about obstacle detecitng and collision aviodance algorithm. In *International Condrence on Control, Automation, and Systems*, 2003.

[22] A. L. Rankin, C. D. Crane, and D. Armstrong. Evaluating a pid, pure pursuit, and weighted steering controller for an autonomous land vehicle. In *Proceedings of SPIE*, volume 3210, pages 1–12, 1997.

[23] J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2), 1990.

[24] S. Skogestad and I Postlethwaite. *Multivariable Feedback Control, Analysis and Design.* John Wiley & Sons, 1996.

[25] P. Soueres and J.P. Laumond. Shortest path synthesis for a car-like robot. In *European Control Conference*, 1993.

[26] Motion Analysis System. MA raptor-4 system. Available at `http://www.motionanalysis.com`, June 2008.

[27] Vicon. Vicon MX systems. Available at `http://www.vicon.com/products/viconmx.html`, July 2006.