TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Vehicle Localization and Tracking for Collision Avoidance System

Behtarin Ferdousi

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Vehicle Localization and Tracking for Collision Avoidance System

Fahrzeuglokalisierung und -verfolgung für das Kollisionsvermeidungssystem

Author:	Behtarin Ferdousi
Supervisor:	Prof. Dr.-Ing. Matthias Althoff
Advisor:	Jagat Rath , Ph.D
Submission Date:	15.05.2020

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Munich, 15.05.2020 Behtarin Ferdousi

Acknowledgments

I want to thank Jagat Rath to help me from ground up for this project. His support and motivation have given me the confidence to work on this topic. Regardless of my countless silly questions, he was always patient and calm in discussions. I am grateful for his time and availability to me.

I am very lucky to have Professor Matthias Althoff for his guidance and advice. I want to express my gratitude for his time and feedback, despite his tremendously busy schedule.

My heartfelt gratitude goes to my husband, who calmed me when I panicked, my parents who constantly encouraged me and believed in me, my sister for sharing laughter and my in-laws for supporting and standing by me. I feel blessed to have such a wonderful company around me and I thank God with all of my heart for the strength and opportunity to study here in TUM.

Abstract

With the current rate of development in autonomous vehicles, the demand for a high-intelligent collision avoidance system is increasing. Due to the inability to determine the inner state of tracked vehicles from Lidar, GPS (Global Positioning System), and radar sensors, researchers have utilized state estimation methods to converge available measurements to the true state of the system. Set-based state estimation methods are used to enclose the true state of the system in a set, in contrast to the stochastic methods which give a point-estimate close to the true state. Encapsulating the true state in a set is important to not allow any fault in the state estimation for safety-critical tasks in autonomous vehicles. The purpose of this thesis is to review and implement multiple set-based state estimation algorithms, using zonotopes as domain representation, on existing datasets of real traffic participants (approx. 10,518 entities). The algorithms implemented are segment intersection methods (optimizing the F-radius, P-radius, and volume) and an interval observer (using H-∞ observer). They are compared in terms of computation time, time to converge, tightness of bound, and accuracy. Due to the impact of vehicle dynamic model on estimation performance, the model is altered among constant velocity, constant acceleration, and the point-mass model to compare the influence. Among the compared estimators, the H-∞-based interval observer is the fastest and the most accurate estimator. However, its performance depends on the initial estimation error bounds, computation of which is not challenging for an automobile collision avoidance system. To summarize, the H-∞-based interval observer with the point-mass model performs the best to locate and track vehicles for a collision avoidance system.

Contents

1 Introduction

Currently, there is steep progress in the research and development of autonomous vehicles. The race to the top of the automobile industry, featuring companies like BMW (Bavarian Motor Works), Tesla, Waymo/Google, requires fast development and vigorous testing of novel technologies. One of the many challenges of this field is to ensure collision avoidance. With no human behind the wheel for Level 5 [1] cars, the vehicle must keep track of roads and surrounding traffic participants (like vehicles and pedestrians) in different circumstances, including rain and fog, to ensure the safety of its passengers. Current collision avoidance systems based on sensors, radar, and camera would be overwhelmed with high computation demands for this purpose. Tolerating error in such a system can cause accidents, including fatality [1].

The collision avoidance system in a car consists of two parts: sensing and tracking and motion planning. The sensing and tracking part is achieved by applying sophisticated algorithms on signals from sensors like radar, camera, and GPS (Global Positioning System). With a decline in the cost for cameras and advancement in technologies in image processing, image analysis, and object detection, sensing and tracking is developing fast. Conversely, each sensor has limitations. Although cameras can classify vehicles, they cannot guarantee measurement in a low-light environment (e.g. night) [2]. In contrast, radar guarantees robustness to weather in exchange for a higher cost. Similarly, there are limitations in GPS, e.g. the inability to function in the urban canyon environment. Thus, one uses sensor fusion to compensate for the shortcomings of specific sensors. After detecting all relevant elements in the environment, a motion planner has to find a collision-free path. The computation of such a path requires certain parameters to predict the tracked vehicle's trajectory. The sensors cannot solely measure all these parameters, hence researchers have turned to state estimation algorithms.

One of the widely-applied state estimation techniques is the Kalman filter [3], which can estimate target dynamics for measurement with additive Gaussian noise. Despite its simplicity, the filter is not suitable for vehicle localization for two reasons. Firstly, statistical noise with known covariance is, unfortunately, not practical. Secondly, the filter provides close point-estimation, relying on which can be safety-critical. These motivate to use set-based state estimation methods.

[1] https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe

The set-based state estimation technique computes a set of state enclosing the true state of the system as long as the dynamics are accurately modeled and the noise and perturbations have known bounds. The main steps are the prediction step and the correction step. The prediction step extrapolates prior estimate, while the correction step improves the extrapolation. Algorithms vary in the approach for the correction step. Another differentiating factor is the choice of geometric shape to represent the estimated set. Zonotope is one of the popular choices, compared to ellipsoid and polytopes, due to a higher accuracy to computation cost ratio. Furthermore, zonotopes have gained fame for state estimation because of wrapping effect control [**Kuhn1998**] (i.e. controlling the increase in size by accumulating noise) and conservative Minkowski sum (i.e. the Minkowski sum of zonotopes is also a zonotope). Therefore, we chose zonotopes to represent the state for all the algorithms in this paper. The following section briefly discusses the related work on zonotopic set-based state estimation.

1.1 Related Work

In 2001, Puig et al [4] used a gain matrix to map input measurement to a zonotopic set of estimation. Following in 2003, Combastel [5] used a singular value decomposition to overapproximate the estimate that is consistent with the input. Although the aforementioned methods are computationally light, they did not focus on the size of the estimated region. In 2005, T. Alamo et al. [6] formulated a convex optimization problem to minimize zonotope size criteria. They focused on two main size criteria: F-radius and volume. F-radius results in fast but conservative estimates, whereas volume computation is heavy, but gives tight bounds. In 2011, T. Alamo et al. [7] optimized P-radius to obtain fair accuracy for a reasonable computation load. Initially, algorithms are developed for single-output linear discrete systems and are later generalized to multi-output and non-linear systems.

Another classification of set-based estimation is the interval observers, where the idea is to design observers such that the error in the estimation is minimal. Unfortunately, the construction of such an observer is not very easy. Hence, the observer design requirements are relaxed in [8], [9]. The relaxation results in conservatism, which led to an interval observer based on H-∞ with reachability analysis [10].

1.2 Paper Outline

The algorithms evaluated in this paper are the segment intersection methods (minimizing the F-radius, P-radius, and volume) and one interval observer (based on H-∞). The prerequisite step before applying state estimation algorithms is to model the tracked vehicle with a well-defined mathematical model. Although there are complex models that can be used to represent a vehicle state [11], not all can be used due to the unavailability of parameters like wheelbase, velocity, etc. accessible to the ego vehicle. Hence, the models used in this paper to compare are the simplest, yet complete enough to determine the properties of the tracked vehicle for trajectory prediction: Constant Velocity, Constant Acceleration, and the Point-Mass Model.

A high degree of accuracy and guarantee is the necessity of the collision avoidance system, hence we chose to compare the set based state estimation algorithms for different scenarios involving dynamic traffic participants from a dataset collected from intersections using drones and fixed cameras. [12] has encouraged many sections in this paper and compares a superset of algorithms covered here; however, the algorithms were compared on simulated data, in contrast to this paper.

The paper is organized as follows. Chapter 2 builds up the vehicle localization problem to be solved by the state estimation algorithms. The following chapter 3 discusses the zonotope-based state estimation algorithms to be compared. In chapter 4, we evaluated the algorithms and discussed the results. Finally, chapter 5 concludes with a summary and a discussion of possible future works.

2 Vehicle Localization : The Guaranteed Estimation Problem

2.1 Preliminaries

The following standard notations are maintained in this paper.

- $\mathcal{R}^n$ and $\mathcal{R}^{n\times m}$ denote the n and $n \times m$ dimensional Euclidean space, respectively.
- I^n represents the n-identity matrix. If n is missing, then appropriate dimension is assumed.
- For a matrix A, A^T, A^{-1}, A_i and A^j denote its transpose, inverse, i^{th} row, and j^{th} column, respectively . $rs(A)$ is the row sum of A, and $det(A)$ the determinant.
- $|.|$ is the absolute value and $||.||_x$ is the x-norm.
- With a vector $a \in \mathcal{R}^n$, $diag(a)$ is a diagonal matrix of dimension n.
- With a vector a, n_a is its dimension.
- For a real symmetric matrix, $P \in \mathcal{R}^{n\times n}, P \prec 0 (P \succ 0)$ implies P is a negative (positive) definite.

2.1.1 Zonotopes: Definitions and Properties

The following definitions and properties are essential for this paper.

Definition 1 ***Interval*** *An interval* $[a,b]$ *is defined as the set* $\{x : a \leq x \leq b\}$.

Definition 1.1 *The unitary interval, denoted by* $\mathbf{B}$, *is [-1,1].*

Definition 1.2 *A box(* $([a_1,b_1],..[a_n,b_n])^T$ *) is an interval vector.*

Definition 1.3 *An unitary box in* $\mathcal{R}^n$ *is denoted by* $\mathbf{B}^n$, *and is a box with n unitary intervals.*

Definition 2 *The Minkowski sum of two sets,*$\mathcal{X}$ *and* $\mathcal{Y}$, *is defined by:*

$$\mathcal{X} \bigoplus \mathcal{Y} = \{x + y : x \in \mathcal{X}, y \in \mathcal{Y}\} \tag{2.1}$$

Definition 3 ***Zonotope***: *An affine transformation of a hypercube, $\boldsymbol{B}^m$ is called an m-ordered zonotope, denoted by $\mathcal{Z} \in \mathcal{R}^n$:*

$$\mathcal{Z} = \langle p, H \rangle = \{p + Hz : z \in \boldsymbol{B}^m\} \tag{2.2}$$

where $p \in \mathcal{R}^n$ is the center of $\mathcal{Z}$, and $H \in \mathcal{R}^{n \times m}$ is called the generator of $\mathcal{Z}$.

Property 1 *For two zonotopes, $\mathcal{Z}_1 = \langle p_1, H_1 \rangle$ and $\mathcal{Z}_2 = \langle p_2, H_2 \rangle$, the following equations hold:*

$$\begin{aligned} \mathcal{Z}_1 \bigoplus \mathcal{Z}_2 &= \langle p_1 + p_2, [H_1 \quad H_2] \rangle \\ L\mathcal{Z}_1 &= \langle Lp_1, LH_1 \rangle \end{aligned} \tag{2.3}$$

Property 2 *[13] A box of an m-zonotope , $\mathcal{Z} = \langle p, H \rangle \in \mathcal{R}^n$, is an n-interval vector, over-approximating the zonotope such that:*

$$\lceil \mathcal{Z} \rceil = box(\mathcal{Z}) = [p - \Delta H, p + \Delta H], \quad \Delta H = \sum_{i=1}^{m} |H^i| \tag{2.4}$$

Property 3 ***Zonotope Reduction*** *[6], [5]: An m-zonotope ($\mathcal{Z} = \langle p, H \rangle \in \mathcal{R}^n$) can be reduced to an s-zonotope, s.t. $n < s < m$, by first sorting the columns of H in decreasing order of Euclidean norm ($\hat{H} = [\hat{h}_1 ... \hat{h}_m]$, with $||\hat{h}_i||_2 \geq ||\hat{h}_{i+1}||_2$). Then, let's split $\hat{H}$ into $\hat{H}_A$ (the first $s - n$ columns) and $\hat{H}_B$. Then the following inclusion is obtained:*

$$\mathcal{Z} \subseteq p \bigoplus [\hat{H}_A \quad rs(\hat{H}_B)] \boldsymbol{B}^s \in \mathcal{R}^n \tag{2.5}$$

It is denoted by $\mathcal{Z}_{\downarrow s}$ in this paper.

Property 4 *[6] Given a zonotope $\mathcal{Z} = p \oplus H\boldsymbol{B}^m \in \mathcal{R}^n$, a strip $\mathscr{S} = \{x \in \mathcal{R}^n : |cx - y| \leq \phi\}$, there exists a vector $\lambda \in \mathcal{R}^n$ such that a family of zonotopes parameterized by λ contains the intersection of the zonotope and the strip s.t.:*

$$\mathcal{Z} \cap \mathscr{S} \subseteq \mathcal{Z}(\lambda) = p(\lambda) \bigoplus H(\lambda) \boldsymbol{B}^{m+1} \tag{2.6}$$

with $p(\lambda) = p + \lambda(y - cp) \in \mathcal{R}^n$
and $H(\lambda) = [(I - \lambda c)H \quad \phi\lambda]$

The construction, reduction, and interval calculation of zonotopes are already implemented in the Matlab® toolbox CORA (COntinuous Reachability Analyzer) [14].

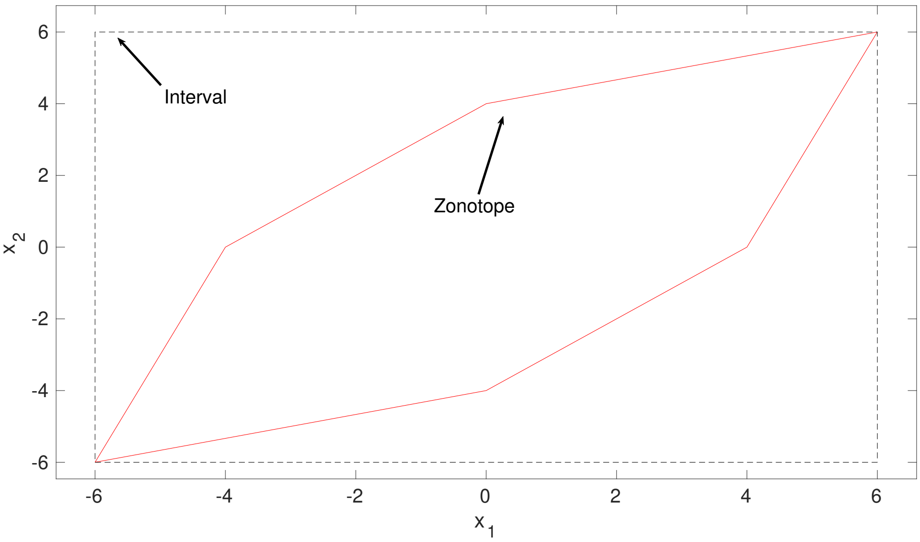


Figure 2.1. An illustration of a zonotope and its interval hull in 2-D

2.2 Problem Formulation

Let us denote the state of the vehicle to be tracked at time k as x_k and the measured state as y_k. Eq. (2.7) formuates a multi-output discrete-time linear system for the tracked vehicle, where $A, E \in \mathcal{R}^{n_x \times n_x}$, $C \in \mathcal{R}^{n_y \times n_x}$, and $F \in \mathcal{R}^{n_y \times n_y}$ are matrices defined by the vehicle system model; w_k and v_k are process noise and measurement noise at time k, respectively.

$$\begin{aligned} x_k &= Ax_{k-1} + Ew_k \\ y_k &= Cx_k + Fv_k \end{aligned} \tag{2.7}$$

Assuming w_k and v_k have known bounds ($\overline{w_k}$ and $\overline{v_k}$, respectively), we can define $\mathcal{W}$ and $\mathcal{V}$ such that $w_k \in \mathcal{W}$ and $v_k \in \mathcal{V}$ as (2.8).

$$\mathcal{W} = \langle 0, H_w \rangle, \quad \mathcal{V} = \langle 0, H_v \rangle \tag{2.8}$$

The dimension of x_k (n_x) varies across vehicle models; however, the measurement vector (y_k) is the measured position in the x and y-direction (2.9).

$$y = [s_x \quad s_y]^T \tag{2.9}$$

Given a vehicle model (discussed in the next section), the problem of set-based state estimation is to compute an outer bound of the state (x_k) containing all the possible values of the true state of the system consistent with the uncertain vehicle model and the measurements.

2.3 Vehicle Model

A major performance-influencing factor is to choose the right model for the tracked vehicle. Three linear systems are implemented in this paper to compare the state estimation algorithms. Although there exist highly precise vehicle models for ego vehicles, the simplest models are used here to represent the tracked vehicle, because complex vehicle models require parameters which are non-acquirable for tracked vehicles. In particular, physical dimensions, like wheelbase or side-slip, cannot be measured directly. Another reason is that adding some parameters, e.g. steering angle and yaw rate, makes the system non-linear and hence does not suit all the algorithms presented. Hence, the following models are investigated:

- **Constant Velocity (CV) Model**
- **Constant Acceleration (CA) Model**
- **Point-Mass (PM) Model**

2.3.1 Constant Velocity Model

The vehicle is assumed to travel in constant velocity [15]. The state of the system (x_k), state transition matrix (A), and the measurement matrix(C) is shown in (2.10).

$$
\begin{aligned}
x &= \begin{bmatrix} s_x & s_y & v_x & v_y \end{bmatrix}^T \\
A &= \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{2.10}
$$

2.3.2 Constant Acceleration Model

Although the constant velocity model is easy to implement, it is unrealistic to assume constant velocity. Acceleration model takes care of changing velocity and assumes constant acceleration [15]. Hence, the estimation errors for position and velocity are expected to be relatively smaller, when the velocity is constantly changing. The state of the system (x_k), the state transition matrix (A), and the measurement matrix(C) are shown in (2.11).

$$
\begin{aligned}
x &= \begin{bmatrix} s_x & s_y & v_x & v_y & a_x & a_y \end{bmatrix}^T \\
A &= \begin{bmatrix} 1 & 0 & \Delta T & 0 & \frac{1}{2}\Delta T^2 & 0 \\ 0 & 1 & 0 & \Delta T & 0 & \frac{1}{2}\Delta T^2 \\ 0 & 0 & 1 & 0 & \Delta T & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
C &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\tag{2.11}
$$

2.3.3 Point-Mass Model

It is trivial to note that vehicles might have varying acceleration, which is not satisfied in the previous models. This brings us to the point-mass model [11], which is similar to the constant acceleration model, except that the acceleration here can strike up to a certain limit. This model treats the tracked vehicle as a point mass, ignoring wheel-base, slip-angle, etc. The state transition and measurement matrices are the same as the constant acceleration model (Eq. (2.11)). The acceleration bounds are set as $11.5m/s^2$ [11] in both x and y-direction for this paper.

3 Zonotope-based guaranteed state estimation

With the essential knowledge on zonotope and vehicle models from the previous chapter, this chapter digs deeper into the state estimation algorithms in terms of vehicle dynamics. With the vehicle dynamics represented by (3.1) (refer to (2.7) for details), we look into segment intersection techniques (minimizing F-radius, volume and P-radius) and an H-∞-based interval observer.

$$\begin{aligned} x_{k+1} &= Ax_k + Ew_k \\ y_k &= Cx_k + Fv_k \end{aligned} \tag{3.1}$$

3.1 Segment Intersection

Algorithm 1 State estimation using segment intersection

Input y_k

Output $\overline{x_k}$, $\underline{x_k}$

1: $\overline{\mathcal{X}_k} \leftarrow \textsc{predict}(\hat{\mathcal{X}}_{k-1})$
2: $\hat{\mathcal{X}}_k \leftarrow \overline{\mathcal{X}_k} \cap \overline{\mathcal{X}_{y_k}}$
3: $[\overline{x_k}, \underline{x_k}] \leftarrow \textsc{interval}(\hat{\mathcal{X}}_k)$
4: $\hat{\mathcal{X}}_k \leftarrow \hat{\mathcal{X}_{k\downarrow m}}$

Alg. 1 shows the overview of segment intersection technique. It starts by predicting the state at time k as $\overline{\mathcal{X}_k} = \langle p, H \rangle$, an r-zonotope. Line 2 finds the intersected segment between the prediction and the state in consistent with the measurement input. One way to do this is to iteratively find the intersected segment for each measurement in a multi-output system. The set to represent the i^{th} input measurement at time k is a strip, denoted by $\mathscr{S}_i = \{x \in \mathbb{R} : |C_i x - y_{k/i}| \leq \overline{v}_{k/i}\}$. With $\hat{\mathcal{X}}_{k/1} = \overline{\mathcal{X}}_k \cap \overline{\mathscr{S}}_1$, $\hat{\mathcal{X}}_{k/i}$ is intersected with $\mathscr{S}_i$ for $i = 2$ to n_y. Using Property 4, the $\hat{X}_{k/i}$ can be parameterized by a vector $\lambda_i \in \mathbb{R}^n$ s.t. (3.2) holds.

$$\begin{aligned} \hat{\mathcal{X}}_{k/i} &= \hat{p}(\lambda_i) + \hat{H}(\lambda_i)\mathbf{B}^{r+1} \\ \textit{where} \quad \hat{p}(\lambda_i) &= p + \lambda_i(y_{k/i} - C_i p) \\ \textit{and} \quad \hat{H}(\lambda_i) &= [(I - \lambda_i C_i)H \quad \overline{v}_{k/i}\lambda_i] \end{aligned} \tag{3.2}$$

The motive of all segment intersection methods is to find the value of λ such that the intersected segment is compact. Once the intersected segment is computed, the upper and lower bounds of the estimation can easily be derived using the Property 2.

For every iteration, the order of the zonotope, and hence the computation overhead, increases. It can be prevented by reducing the zonotope to a maximum order of m as shown in Line 4.

3.1.1 F-radius

Algorithm 2 Segment minimization

Input: y_k

Output: $\overline{x_k}$, $\underline{x_k}$

1: $\overline{\mathcal{X}_k} \leftarrow$ PREDICT$(\hat{\mathcal{X}}_{k-1})$
2: $\langle p, H \rangle \leftarrow \overline{\mathcal{X}}_k$
3: **for** $i \leftarrow 1$ to n_y **do**
4: $\quad \lambda_i \leftarrow$ CALCULATE_LAMBDA$(H, C, \overline{v}_{k/i})$
5: $\quad p \leftarrow p + \lambda_i(y_{k/i} - C_i p)$
6: $\quad H \leftarrow [(I - \lambda_i C_i)H \quad \overline{v}_{k/i}\lambda_i]$
7: **end for**
8: $\hat{\mathcal{X}}_k \leftarrow \langle p, H \rangle$
9: $[\overline{x_k}, \underline{x_k}] \leftarrow$ INTERVAL$(\hat{\mathcal{X}}_k)$
10: $\hat{\mathcal{X}}_k \leftarrow \hat{\mathcal{X}}_{k\downarrow m}$

One approach to minimize the intersected segment is to minimize the F-radius of the resulted zonotope. The F-radius of a zonotope is the Frobenius-norm of its generators. Alg 2 presents the algorithm to implement this approach.
In Line 4, the λ_i, corresponding to the segment with minimum F-radius, is computed. To derive the λ_i, let us rewrite $\hat{H}_i(\lambda_i)$ from Eq. 3.2 as $A + \lambda b^T$ such that $A = [H \quad 0]$ and $b^T = [-C_i H \quad \overline{v}_{k/i}]$. Thus, the F-norm of the generators of a zonotope can be calculated using the formula (3.3). Refer to [6] for proof.

$$\begin{aligned} ||H||_F^2 &= ||A + \lambda b^T||_F^2 \\ &= 2\lambda^T A b + (b^T b)\lambda^T \lambda + tr(A^T A) \end{aligned} \tag{3.3}$$

$$\lambda^* = \frac{-Ab}{b^T b} = \frac{HH^T(C_i)^T}{C_i HH^T(C_i)^T + \overline{v}_{k/i}^2} \tag{3.4}$$

The λ^*, that corresponds to the minimum F-radius of the intersected zonotope, is calculated using the formula (3.4) for each measurement. With the λ, the intersected segment is computed as shown in Line 5 and 6 in Alg. 2. The remaining steps find the bounds and reduce the zonotope order.

This approach is used when a fast real-time state estimation is needed. However, sharper bounds of estimation can be obtained by optimizing the volume.

3.1.2 Volume

The volume of $\hat{\mathcal{X}}$ for the i^{th} measurement state is [6]:

$$\begin{aligned} Vol(\hat{X}(\lambda)) = & 2^n \sum_{j=1}^{N(n,r)} |det[(I - \lambda C_i)A_j]| \\ & + 2^n \sum_{j=1}^{N(n-1,r)} |det[(I - \lambda C_i)B_j \quad \overline{v}_{k/i}\lambda]| \end{aligned} \tag{3.5}$$

where $N(n,r)$ denotes the number of combinations of r elements from a set of n elements. A_j and B_j denote each of the different matrices generated by choosing n and $n-1$ columns from H respectively.

The algorithm is same as F-radius (Alg. 2), with an exception in the λ calculation in Line 4. The `zonotope.volume` function provided by CORA along with `fmincon` solver in Matlab® can be used to find the value of λ, parameterizing the intersected segment with minimum volume.

Although volume minimization significantly improves the intersected zonotope, the computations are extremely heavy. Therefore it works best for use-cases that are not time-sensitive, e.g. fault diagnosis and fault-tolerant control systems [16].

3.1.3 P-radius

The P-radius of a zonotope can be calculated with the formula (3.6) where P is a positive definite matrix [6].

$$\max_{z \in Z}(||z - p||_P^2) = \max_{z \in Z}((z-p)^T P (z-p)) \tag{3.6}$$

The λ, to parameterize the segment with non-increasing P-radius, can be computed off-line by solving the LMI (Linear Matrix Inequality) (3.7). The $\beta \in (0, 1]$ can be found using binary search, and the λ can be solved using Mosek solver in Matlab®.

$$\begin{bmatrix} \beta P & 0 & 0 & A^T P - A^T C_i Y^T \\ * & F^T F & 0 & F^T P - F^T C_i Y^T \\ * & * & \overline{v}_{k/i}^2 & Y^T \overline{v}_{k/i} \\ * & * & * & P \end{bmatrix} \succeq 0, where \quad Y = P\lambda_i \tag{3.7}$$

Due to off-line computation, this method is expected to be substantially faster in comparison to the previous methods. Consequently, the over-approximation parameter, λ, does not correct itself with measurements. Therefore, it has been used in lower accuracy-prone systems like secure monitoring of cyber-physical systems against attacks [17].

3.2 Interval Observer

Interval observers require an accurate design of observers to minimize the error in the estimation. For the system in (3.1), (3.8) defines the observer, where L is the observer gain to be designed. The design of such observers is not very easy. The following section discusses a method, which uses H-∞ observer combined with reachability analysis.

$$x_{k+1} = Ax_k + L(y_k - Cx_k) \tag{3.8}$$

3.2.1 H-∞-based Observer

The interval observer, proposed in [10], computes the observer gain as $L = P^{-1}Y$ with P, a positive definite matrix with dimension $n_x \times n_x$, and Y, a matrix with dimension $n_x \times n_y$, both solution to the optimization problem in (3.9).

$$\min_{\gamma_2} s.t. \text{ (3.10)} \tag{3.9}$$

$$\begin{bmatrix} I_{n_x} - P & * & * & * \\ 0 & -\gamma^2 I_{n_w} & * & * \\ 0 & 0 & -\gamma^2 I_{n_v} & * \\ PA - YC & PE & -YF & -P \end{bmatrix} \prec 0 \tag{3.10}$$

With L derived using a Mosek solver on (3.9) in Matlab®, the estimator is initialized with the following parameters:

$$\begin{aligned} \mathcal{W} = \langle 0, H_w \rangle, &\quad \mathcal{V} = \langle 0, H_v \rangle \\ \mathcal{D}_w = E\mathcal{W}, &\quad \mathcal{D}_v = -LF\mathcal{V} \\ \mathcal{S}_x = \langle 0, H_0 \rangle, &\quad \mathbf{S}_w = \emptyset, \quad \mathbf{S}_v = \emptyset \end{aligned} \tag{3.11}$$

where $H_w = diag(\overline{w})$ and $H_v = diag(\overline{v})$

For every measurement, y, the estimator estimates using the Alg. 3.

Algorithm 3 Estimation using H-∞-based interval observer

Input y

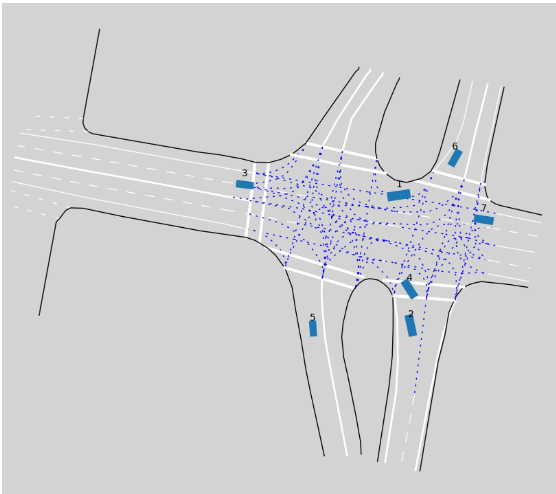
Output $\overline{x}$, $\underline{x}$

1: $[\overline{e}, \underline{e}] \leftarrow \textsc{interval}(\mathcal{S}_x) \bigoplus \mathbf{S}_w \bigoplus \mathbf{S}_v$
2: $\overline{x} \leftarrow \hat{x} + \overline{e}$
3: $\underline{x} \leftarrow \hat{x} + \underline{e}$
4: $\hat{x} \leftarrow A\hat{x} + L(y - C\hat{x})$
5: $\mathcal{S}_x \leftarrow (A - LC)\mathcal{S}_x$
6: $\mathbf{S}_w \leftarrow \mathbf{S}_w \bigoplus \textsc{interval}(\mathcal{D}_w)$
7: $\mathbf{S}_v \leftarrow \mathbf{S}_v \bigoplus \textsc{interval}(\mathcal{D}_v)$
8: $\mathcal{D}_w \leftarrow (A - LC)\mathcal{D}_w$
9: $\mathcal{D}_v \leftarrow (A - LC)\mathcal{D}_v$

The beauty of this interval observer lies in its off-line computation and the absence of run-time complex set operations. Consequently, interval observers have gained fame in control theory and can be used in stabilization, optimal control, fault detection, and circuit systems [18].

4 Evaluations

Figure 4.1. Simulation of the dataset



The INTERACTION Dataset [1] is used to evaluate the algorithms. The dataset contains multiple scenarios in different locations, captured using drones or fixed cameras over a variable amount of time. Each scenario consists of multiple traffic participants, identified by an ID, and each frame per 0.1s has a set of vehicles and their position and velocity in the x and y-direction. Over a choice of multiple videos, the location with all videos summing up to a total length of 259.43 minutes is chosen for this paper. A simulated scenario is demonstrated in Fig. 4.1. There are 60 recorded files in this location with a total of 10,518 vehicles. The position in the x and y-direction is used as a measurement input to the algorithms, whereas the velocity in the x and y-direction is used to calculate the error and evaluate the estimates.

With every participant modeled as (2.7), the initial state is set as (4.1). The matrices E and F are I_{n_x} and I_{n_y} respectively. All zonotopes are limited to a maximum order of

[1] https://interaction-dataset.com/

20.

$$\begin{aligned} \mathcal{X}_0 &= \langle 0, diag([1000 \quad 1000 \quad 10 \quad 10 \quad 10 \quad 10]^T) \rangle \\ \overline{w_k} &= [0.1 \quad 0.1 \quad 0.4 \quad 0.4 \quad 0.1 \quad 0.1]^T \\ \overline{v_k} &= [0.1 \quad 0.1]^T \end{aligned} \tag{4.1}$$

All the evaluations are carried out by single-threaded scripts run on an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz machine with MATLAB® 2019b. The CORA toolbox is used for set computations; the Mosek solver in YALMIP toolbox is used to solve the optimization problems. Templates to implement model and estimator in MATLAB® is attached in the Appendix B.

4.1 Computation Time

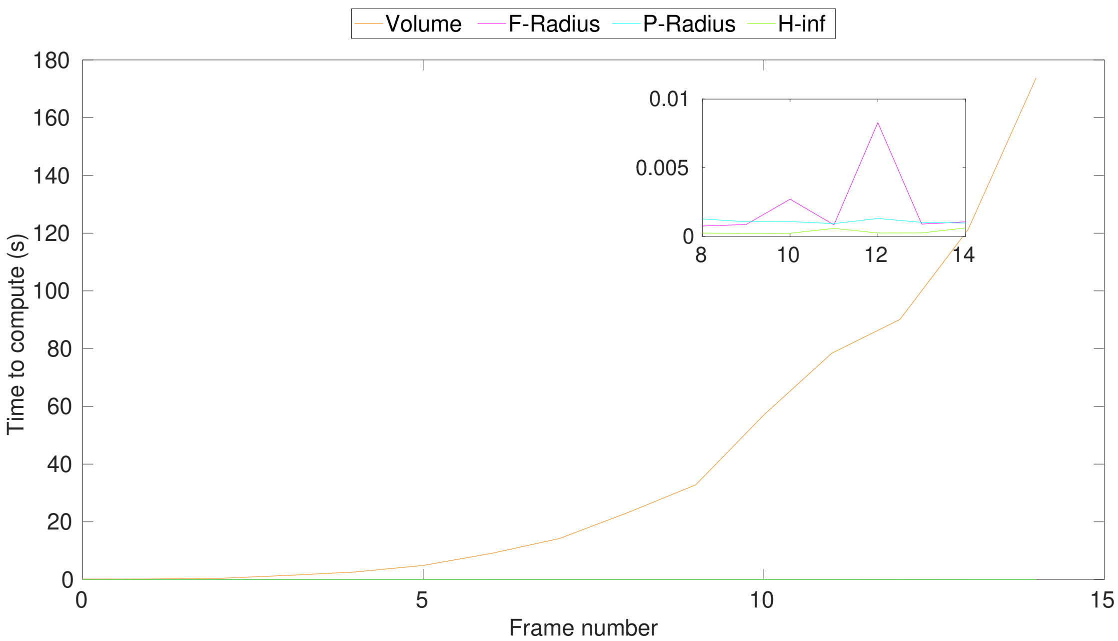


Figure 4.2. Computation time for each method to estimate using the CV model

Fig. 4.2 illustrates that the computation time for volume minimization rises exponentially over time. Due to the choice of limiting zonotope order to $m \leq 20$, volume computation time is expected to steeply rise till the 16^{th} frame for the cv model. As seen from the figure, the computation time has reached 160,000% of the time step. Such an outcome makes volume minimization futile for a collision avoidance system; hence this method is avoided in the rest of the paper.

Table 4.1. Comparison of computation time (ms)

	Average Computation Time (ms)		
Method	CV model	CA model	PM model
F-Radius	0.396	0.375	0.621
P-Radius	0.312	0.319	0.544
H-∞ approximation	0.145	0.147	0.144

Tab. 4.1 shows that the computation time for the other methods is negligible compared to the frame rate, i.e 100ms. Furthermore, the H-∞-based interval observer has almost half the time taken by the segment intersection methods, which can be explained by the absence of zonotope reduction operation in the former method.

In addition, Tab. 4.1 highlights how the choice of vehicle model affects the performance. The H-∞-based observer is not much affected by models, whereas, the segment minimization methods have a significant decline in performance with the point-mass model. This observation can be explained by how the constraints in the point-mass model are applied. In segment minimization methods, the point-mass model applies constraints on the zonotope for the state, whereas, the H-∞-based observer only constraints the upper and lower bound.

4.2 Time to Converge

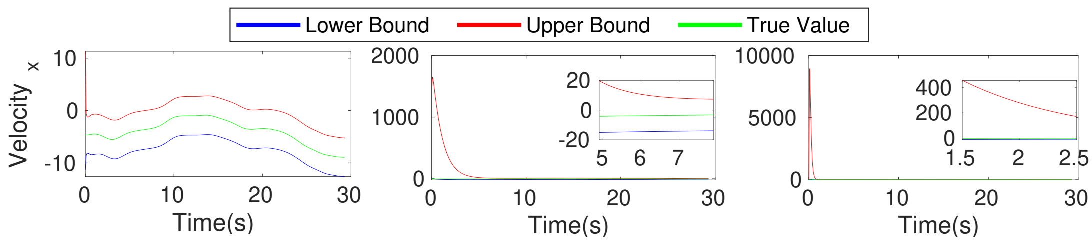
Table 4.2. Comparison of approximate time (in s) to converge

	Velocity			**Acceleration**	
Method	**CV model**	**CA model**	**PM model**	**CA model**	**PM model**
F-Radius	0.2	0.2	0.2	0.6	4.3
P-Radius	4.8	3.7	3.7	5	5.2
H-∞ approximation	1.5	2.9	2.8	3.2	3.6

For this paper, the time to converge is calculated as the time taken for the rate of change of estimated bounds to approach zero. Tab. 4.2 shows the approximate time for the unmeasured estimates, $velocity_x$ and $acceleration_x$, to converge. It is trivial to note that the P-radius technique demonstrates the worst performance, which can be explained by its fixed pre-computed parameter for the intersected segment. In contrast, the F-radius technique, which adjusts the aforementioned parameter in run-time for every measurement, performs the best, taking at most six time steps (compared to fifty time steps for P-radius minimizer) to converge both velocity and acceleration. An exception appears in the pm model, which can be explained by the disturbance in model linearity due to its constraint in acceleration.

In general, the point-mass model delays the time to converge. Interestingly, the H-∞-based observer exhibits a rather consistent time to converge, even for an increase in the state dimension (e.g. ca model) or new constraints (e.g. pm model). This behavior makes H-∞-based observer apt for complex models. An extended result of the rate of change of bounds of all the variables over time is available in the Appendix A.2.

4.3 Bounds



(a) F-radius minimizer (b) P-radius minimizer (c) H-∞-based observer

Figure 4.3. The $velocity_x$ bounds using pm model

As seen in Fig. 4.3, all methods enclose the true state in the mentioned setup for a chosen participant in the dataset. Estimation for all combinations of model and algorithm is presented in Appendix A.1. Tab. 4.3 tabulates the average bounds for all the vehicles in the scenario. Bounds using constant velocity model is tighter compared to constant acceleration and point-mass model. This is an indication that better results are obtained for a model with fewer unmeasured states. Bounds from constant acceleration and point-mass are identical in all variables, except the acceleration where point-mass performs better. In general, the F-radius minimizer has better bounds for the unmeasured states.

As the point-mass model provides estimation for both velocity and acceleration, and outperforms constant acceleration model, this model is selected to compare the techniques in the next section.

Table 4.3. Comparison of bounds of estimation

Method	s_x	s_y	v_x	v_y	a_x	a_y
			Constant Velocity			
F-Radius	0.441	0.441	5.686	5.686	-	-
P-Radius	0.4606	0.459	13.45	13.45	-	-
H-∞ approximation	0.9867	0.937	6.177	6.177	-	-
			Constant Acceleration Model			
	s_x	s_y	v_x	v_y	a_x	a_y
F-Radius	0.5713	0.5075	8.461	8.461	15.86	15.97
P-Radius	0.4598	0.4523	16.39	16.39	16.61	17.42
H-∞ approximation	1.5	1.5	9.414	9.414	16.42	16.35
			Point-Mass Model			
	s_x	s_y	v_x	v_y	a_x	a_y
F-Radius	0.5713	0.5075	8.461	8.461	15.79	15.78
P-Radius	0.4598	0.4523	16.39	16.39	16.43	16.18
H-∞ approximation	1.5	1.5	9.414	9.414	16.11	16.24

4.4 Accuracy

Accuracy is compared using the root mean square error (RMSE) of the estimation. Since, the dataset does not have the measurement for acceleration, the accuracy of acceleration estimation cannot be evaluated.

The initial estimation before convergence gives an extreme error which affects the result, hence the estimations after convergence (i.e. after 50 time-steps) are allowed in the evaluation. The RMSE is then computed as a percentage from the maximum measurement in the time frame.

The boxplot of RMSE for $velocity_x$ using the point-mass model for all the techniques are shown in Fig. 4.4. The segment minimization using P-radius has a high range of extremes and the mean error is also greater than the other methods. The range of error is the lowest for H-∞ observer with a mean of 1.9048 and a standard deviation of 1.9776 for $velocity_x$. A detailed report of the mean and standard deviation of each of the methods can be found in Tab. 4.4. The error expectation from segment intersection optimizing F-radius is similar to H-∞-based interval observer, however, the error in the measured state is slightly lesser in the F-Radius minimizer compared to the H-∞ observer.

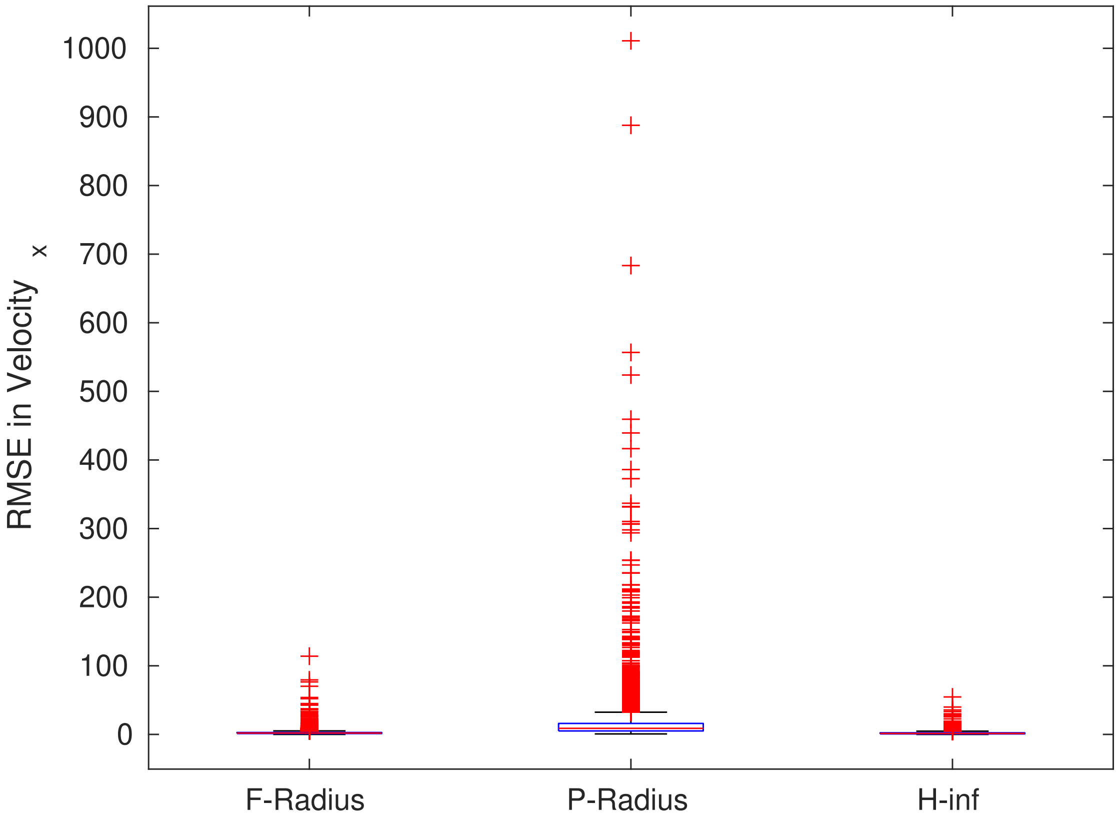


Figure 4.4. Comparison of RMSE(Root Mean Squared Error) in $velocity_x$ using pm model

Table 4.4. Comparison of RMSE with pm model

	Mean ± SD			
Method	s_x	s_y	v_x	v_y
F-Radius	0.0007 ± 0.0004	0.0004 ± 0.0003	2.3412 ± 2.7092	2.4184 ± 1.5641
P-Radius	0.0016 ± 0.0020	0.0008 ± 0.0017	13.4470 ± 26.2465	13.7642 ± 54.6724
H-∞ approximation	0.0007 ± 0.0004	0.0006 ± 0.0005	1.9048 ± 1.9776	2.1230 ± 2.1946

4.5 Discussion

The results in the previous sections show that every estimator has pros and cons. Besides, the output of the techniques depends heavily on the vehicle model used. In this section, we try to choose an estimator and a model, best suited to the automobile collision avoidance system.

Firstly, from the presented models, the point-mass model is the most accurate model for tracked vehicles, because it gives better estimates of the acceleration, an unmeasurable state. Secondly, comparing the estimators, the volume minimizer lost in computation time, whereas the P-radius minimizer gave terrible bounds. In contrast, the H-∞-based observer computes the fastest, and the F-radius minimizer gave the tightest bounds with fewer measurements. However, the computation time for F-radius rises with the dimension of the state. Moreover, adding constraints in F-radius also increases computation time. Although the computation time in Tab. 4.1 for F-radius does not seem much in comparison to the time step (0.1s), the load would grow with the number of vehicles. In contrast, the H-∞-based interval observer not only takes the shortest time, but also does not change with the increase in state dimension. Furthermore, it gives a lower range of error in estimation. However, one challenge with the H-∞-based interval observer is to initialize the estimated state appropriately to ensure true state enclosure. This can be done in the automobile collision avoidance system by adding the location of the ego vehicle and the maximum radius of the sensors. In conclusion, the H-∞-based interval observer with point-mass model is the best choice for a collision avoidance system.

5 Conclusion

A demand for intelligent collision avoidance system is timeless. State estimation algorithms can be used to track vehicles with assurance to predict a collision-free path. One significant factor is to fit the behavior of traffic participants to a mathematical model. Besides, higher the number of measured states, sharper is the enclosure. In comparison to the segment intersection techniques, the H-∞-based interval observer does not worsen much with model complexity. This makes the interval observer attractive to be used for complex models. In addition, the H-∞-based inverval observer is the fastest estimator, and its estimated bounds can be tuned using parameters. In conclusion, the H-∞-based interval observer along with the point-mass model is the best choice for a collision avoidance system. This paper can be a starting point to implement higher-defined models of the tracked vehicle and compare the performance of the state estimation methods. The state estimation methods can further be evaluated by the effect of the initial estimated state. Further developments can be to use non-linear state-estimation algorithms on complex vehicle models and compare the performance.

A Extended Result

A.1 Set Estimation

Estimation using the techniques with different models are illustrated using data for one particular vehicle in the dataset in this chapter. Results show that the true state is always bounded by the set of estimation. For acceleration, there is no true measurement because the acceleration of the tracked vehicle is absent in the dataset.

A.1.1 Segment Minimization using F-Radius

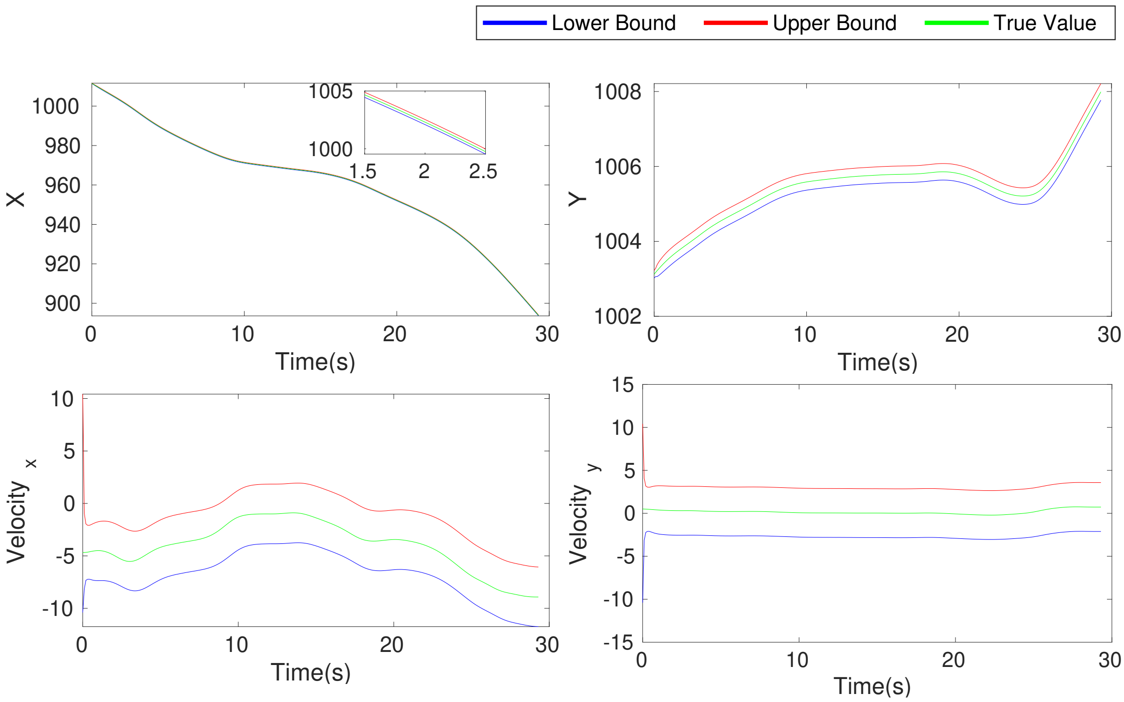


Figure A.1. Estimation using the F-radius and the constant velocity model

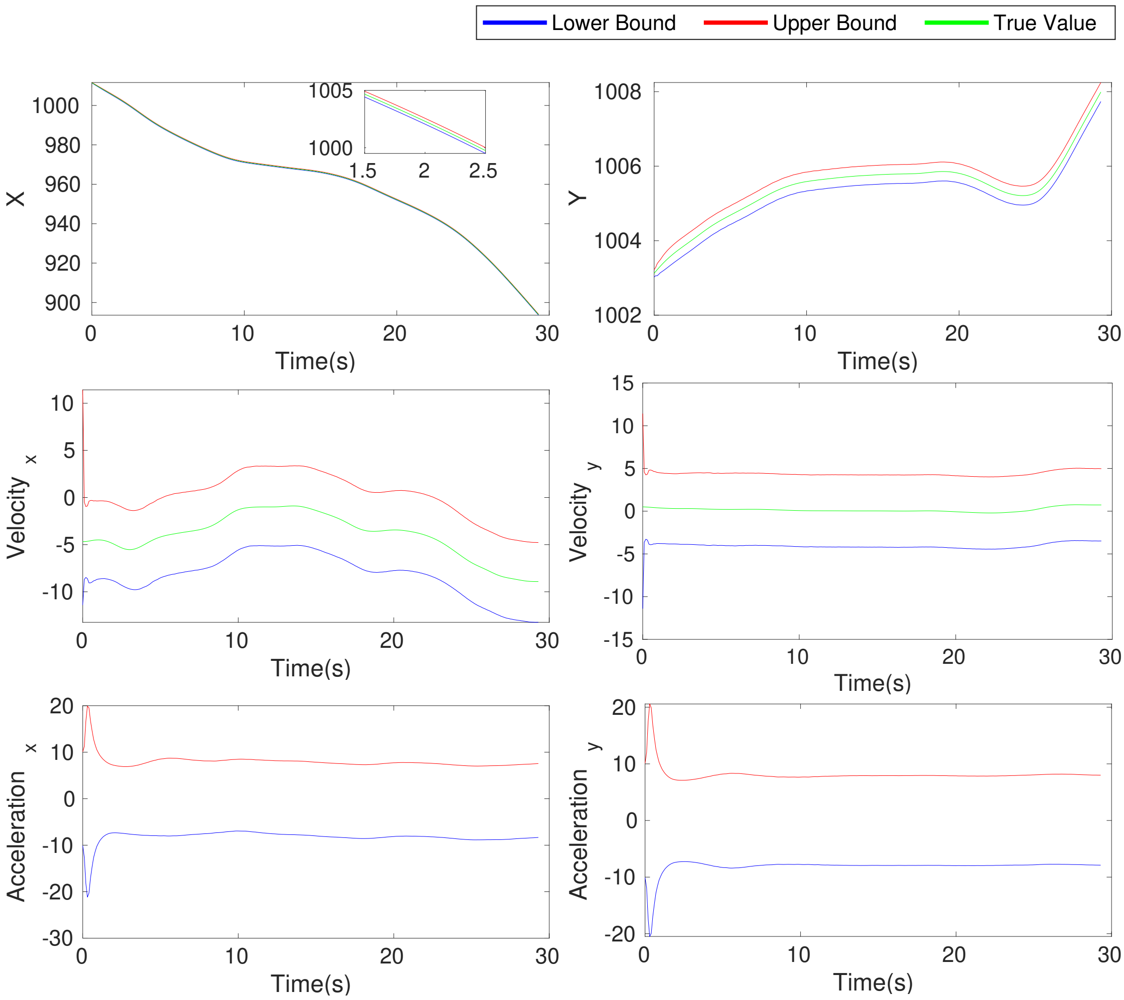


Figure A.2. Estimation using the F-radius and the constant acceleration model

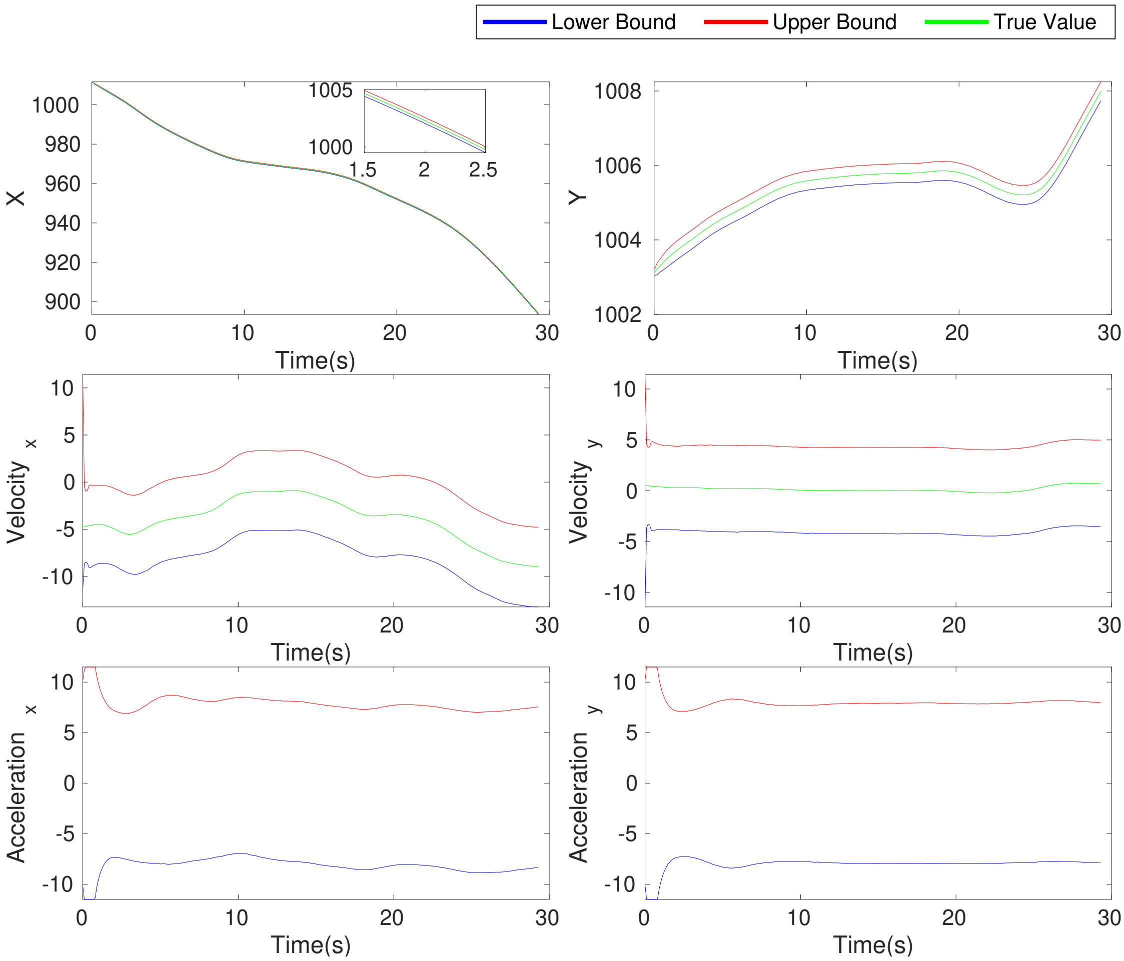


Figure A.3. Estimation using the F-radius and the point-mass model

A.1.2 Segment Minimization using P-Radius

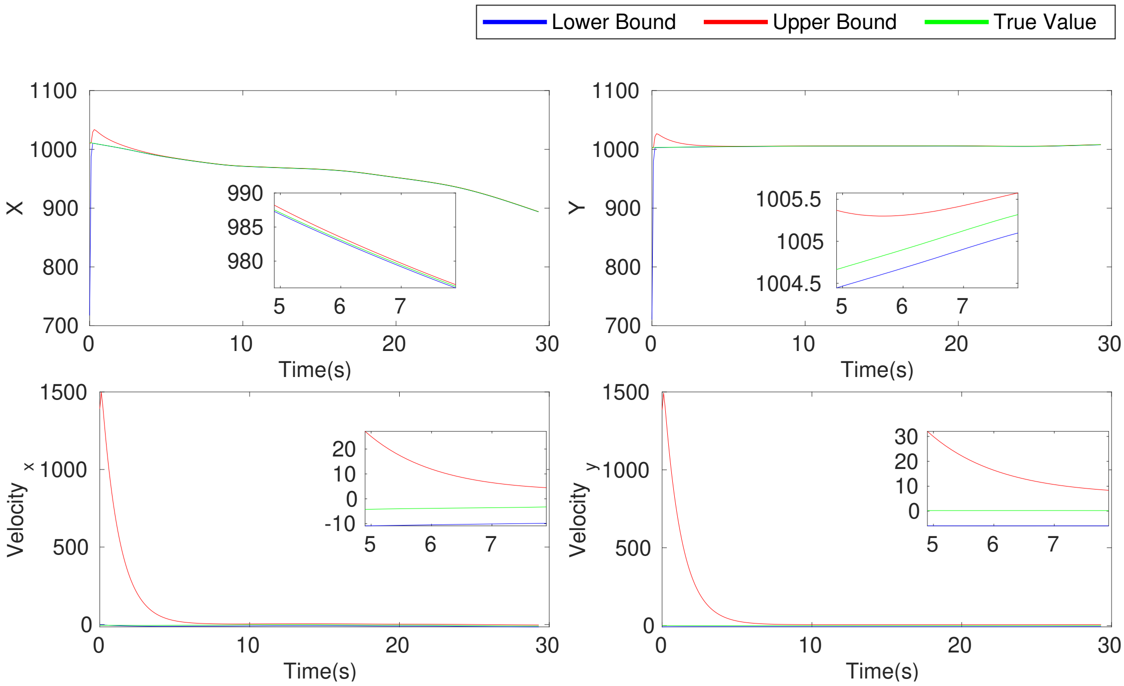


Figure A.4. Estimation using the P-radius and the constant velocity model

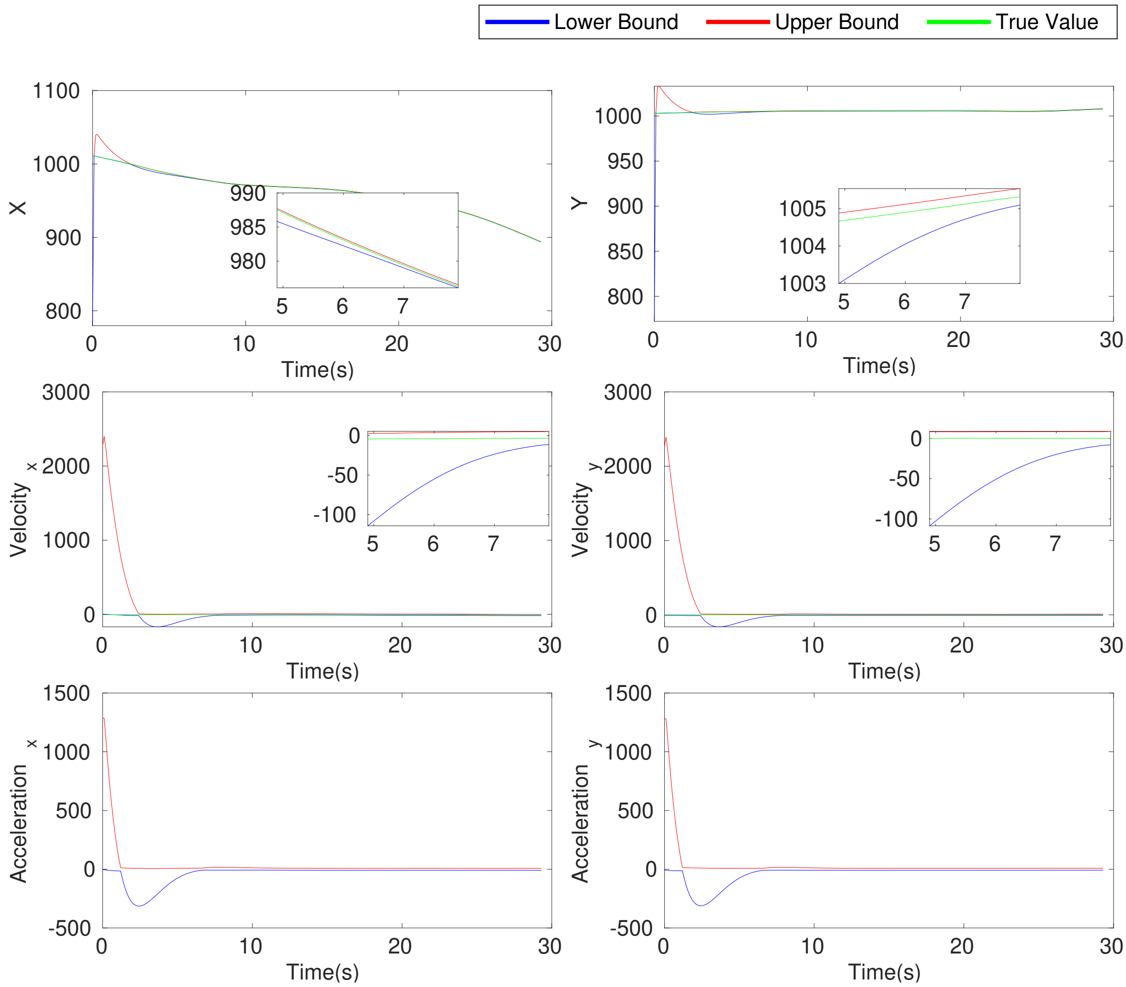


Figure A.5. Estimation using the P-radius and the constant acceleration model

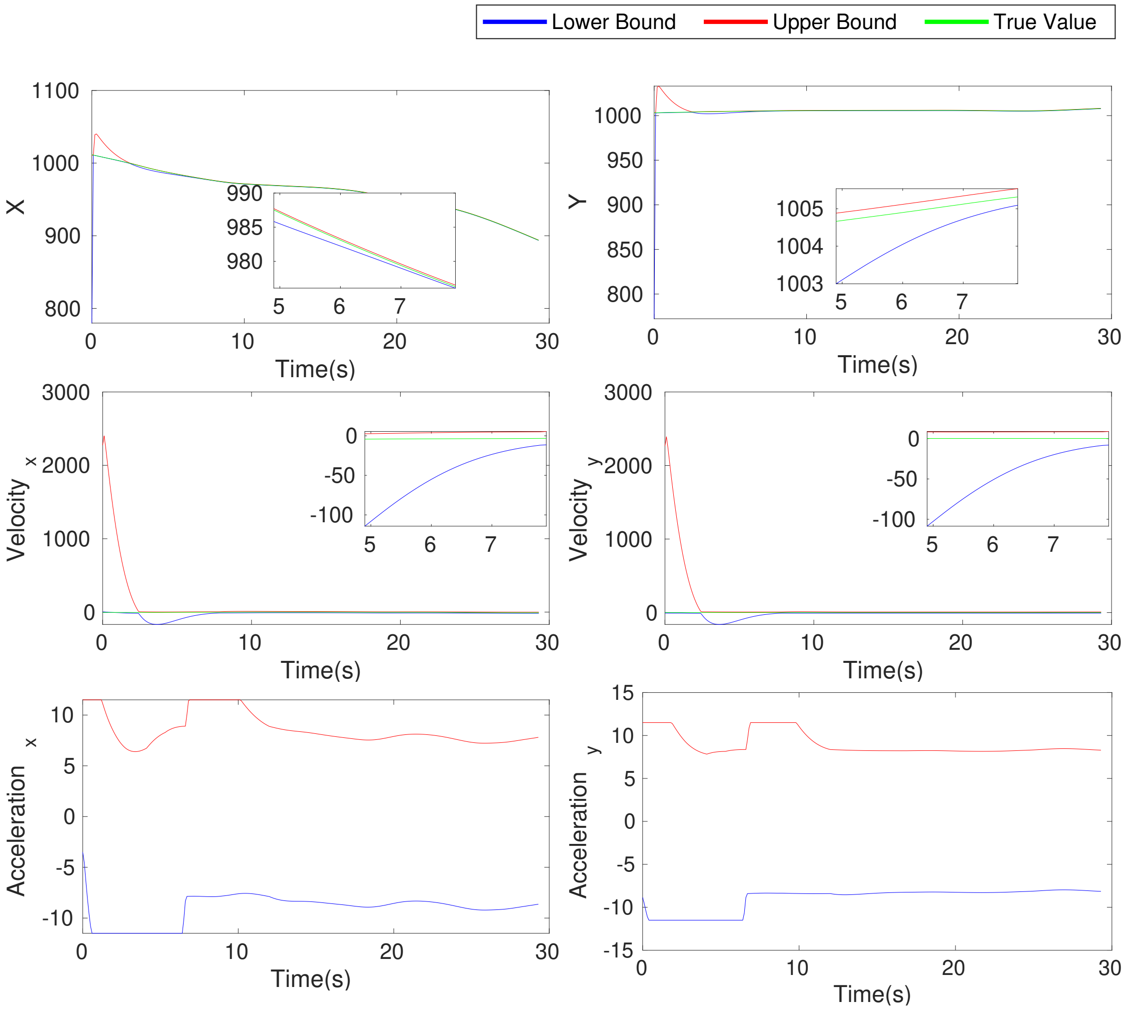


Figure A.6. Estimation using the P-radius and the point-mass model

A.1.3 Interval Observer using H-∞

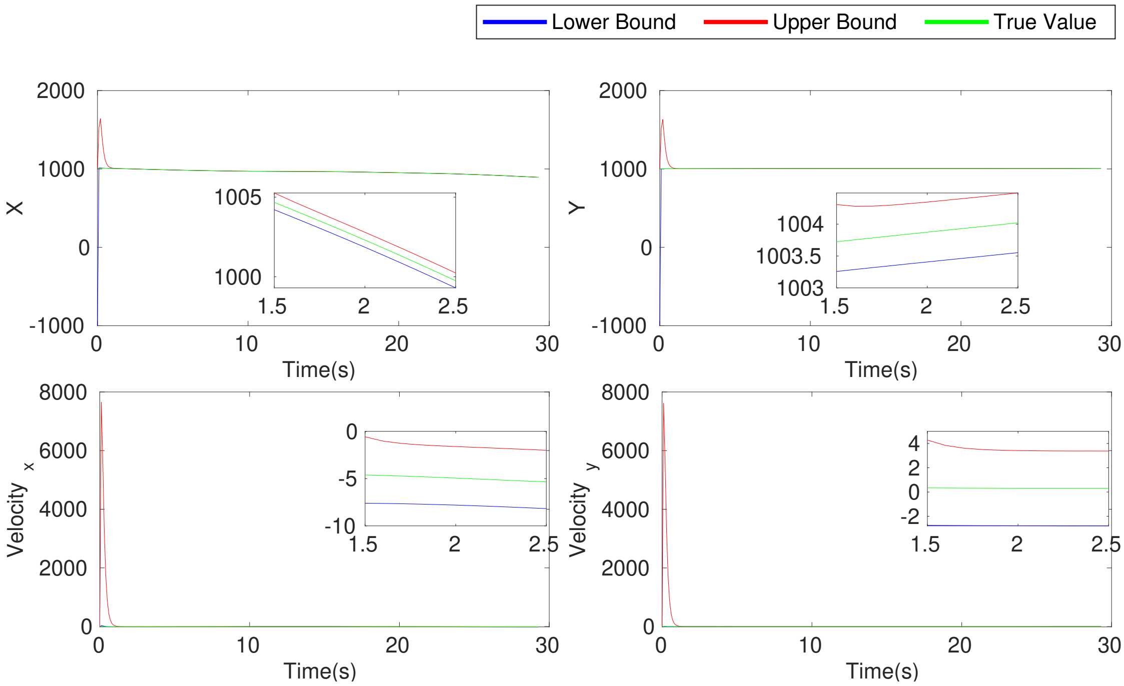


Figure A.7. Estimation using H-∞ observer and the constant velocity model

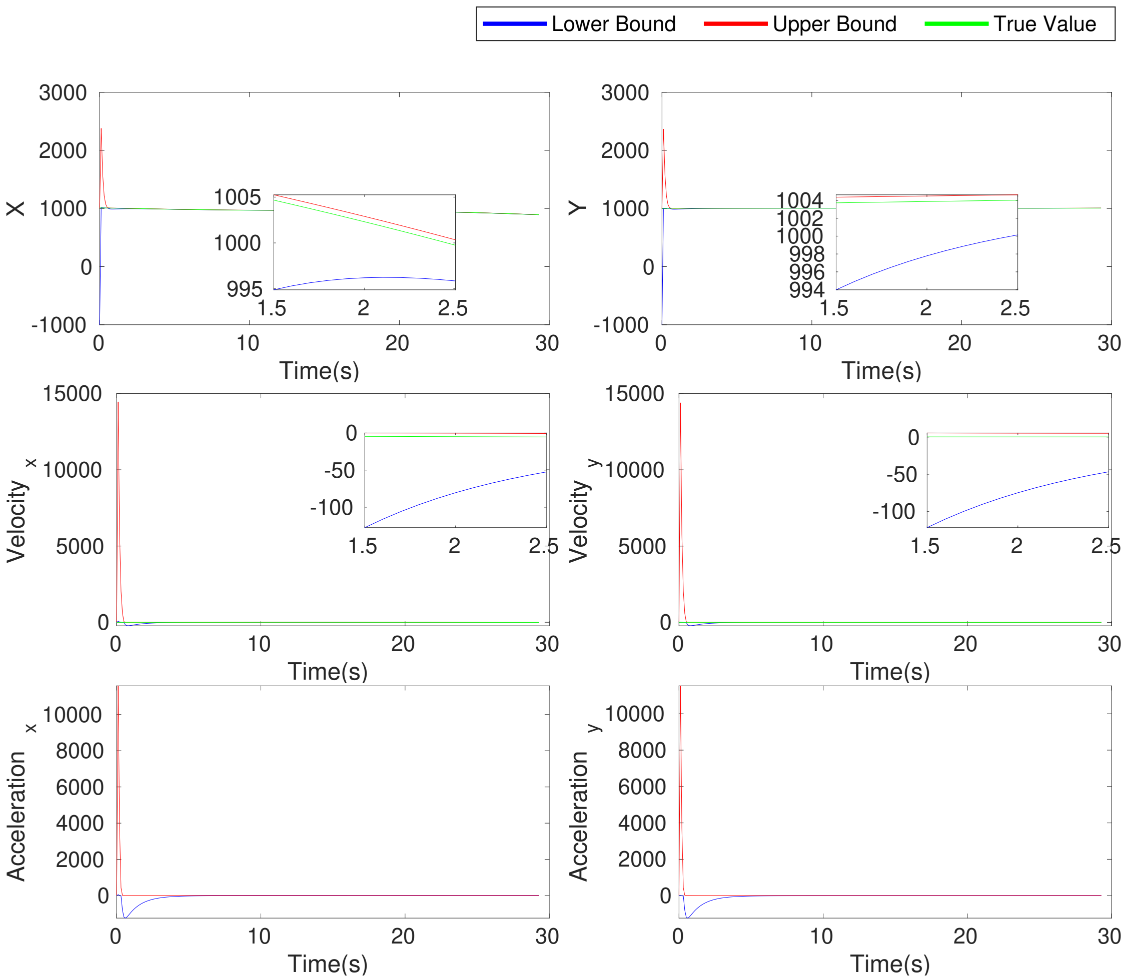


Figure A.8. Estimation using H-∞ observer and the constant acceleration model

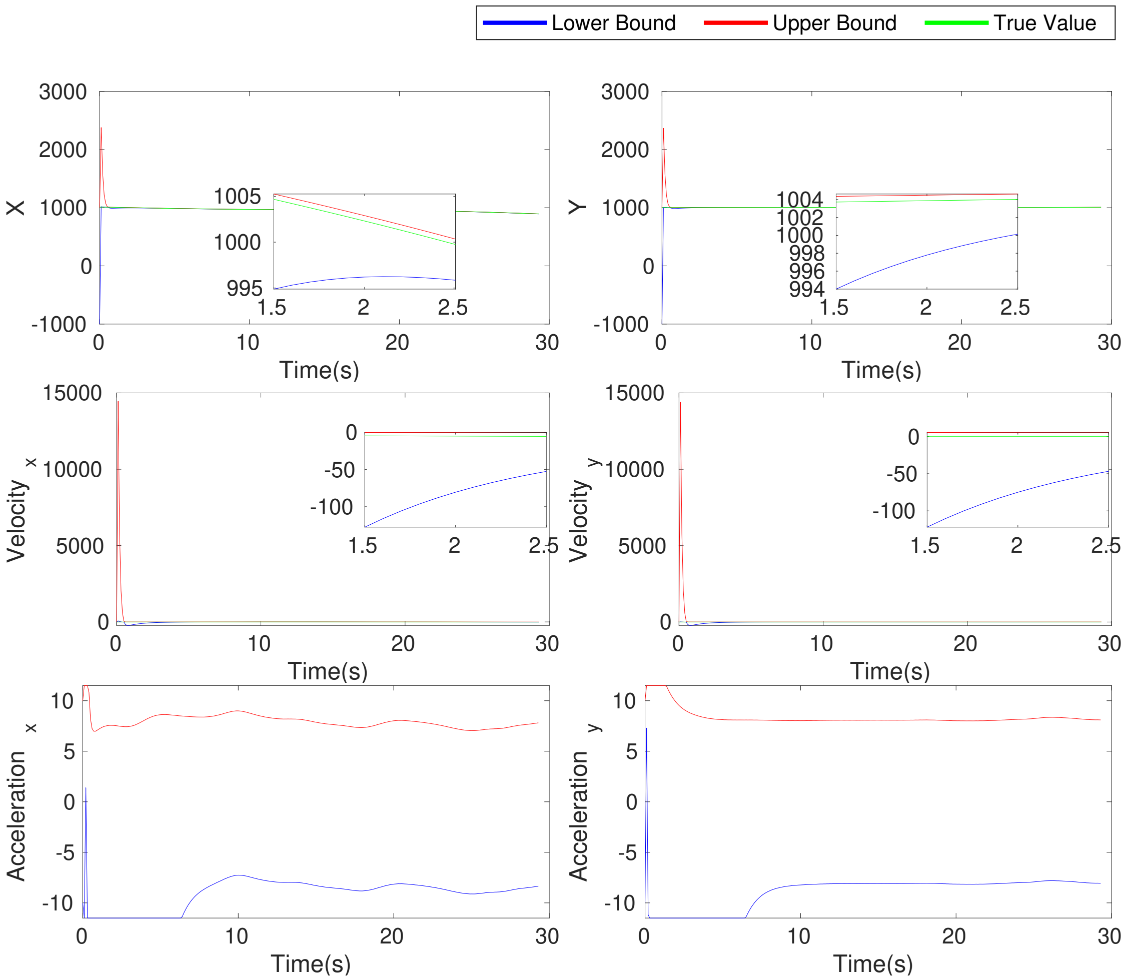


Figure A.9. Estimation using H-∞ observer and the point mass model

A.2 Rate of Change of Bounds

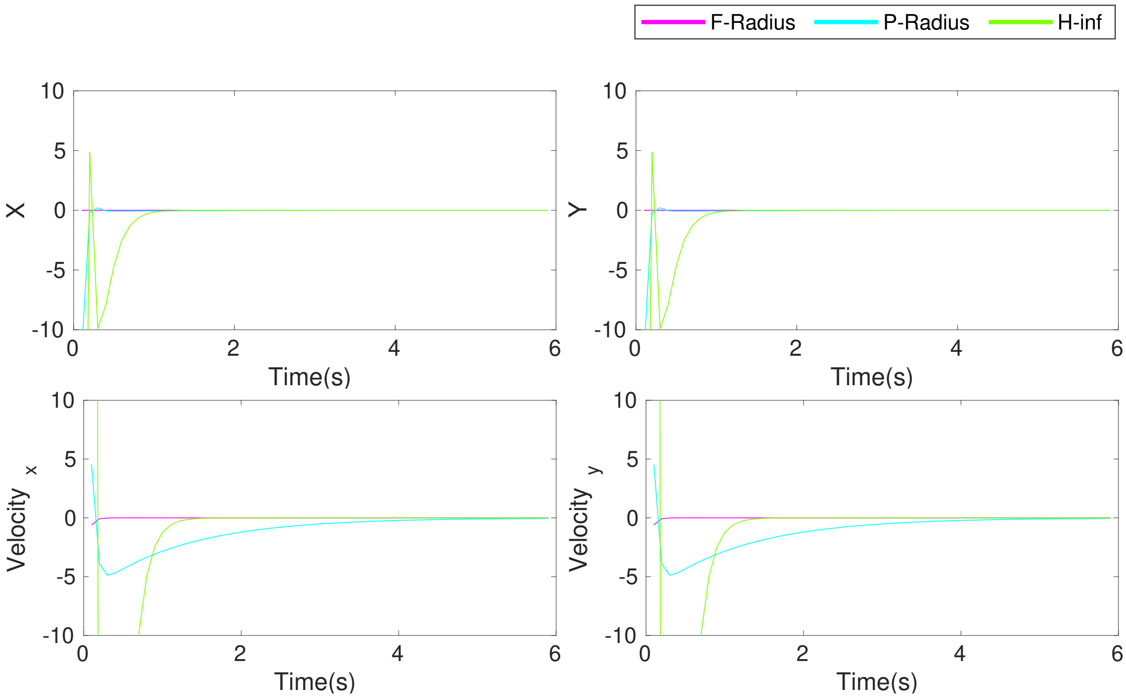


Figure A.10. Rate of change of bounds using the constant velocity model

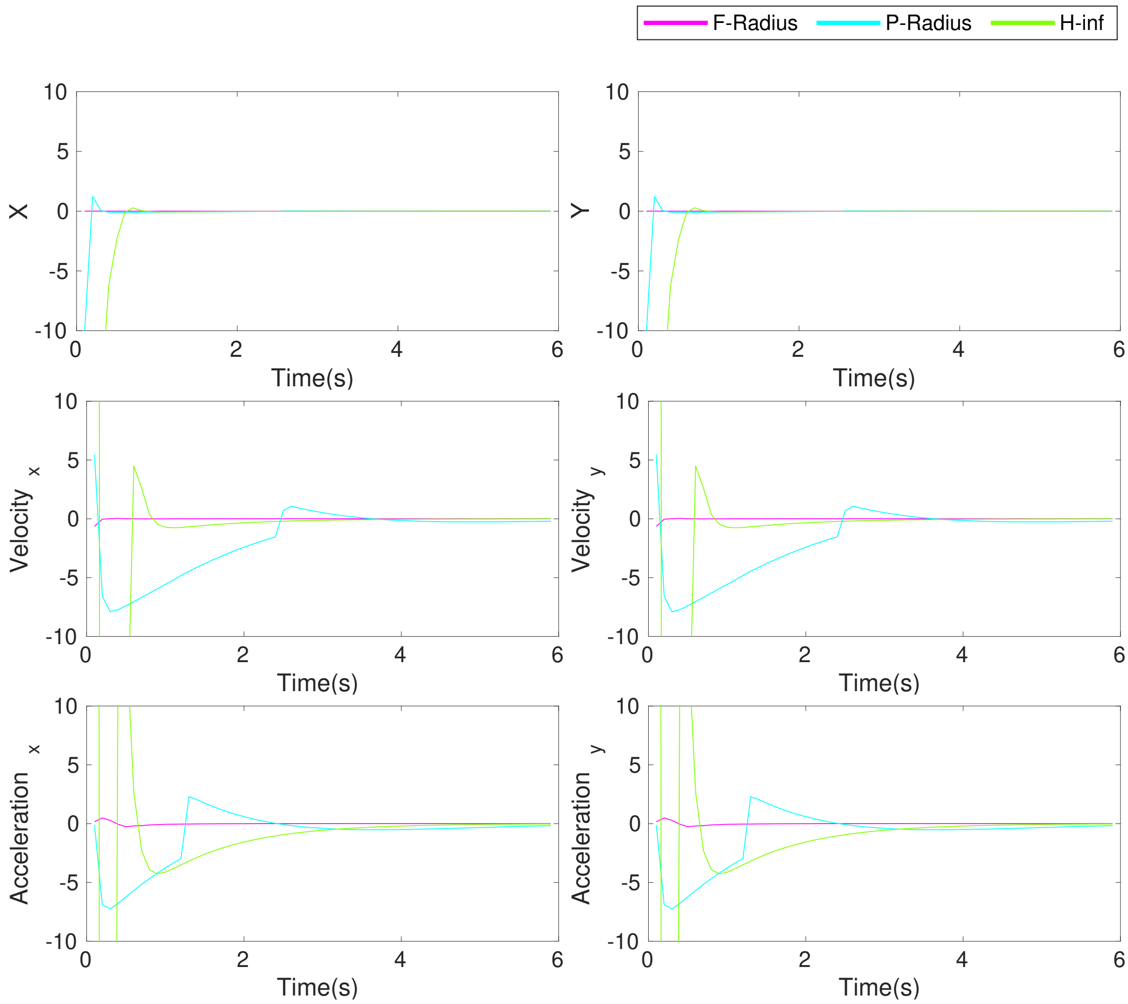


Figure A.11. Rate of change of bounds using the constant acceleration model

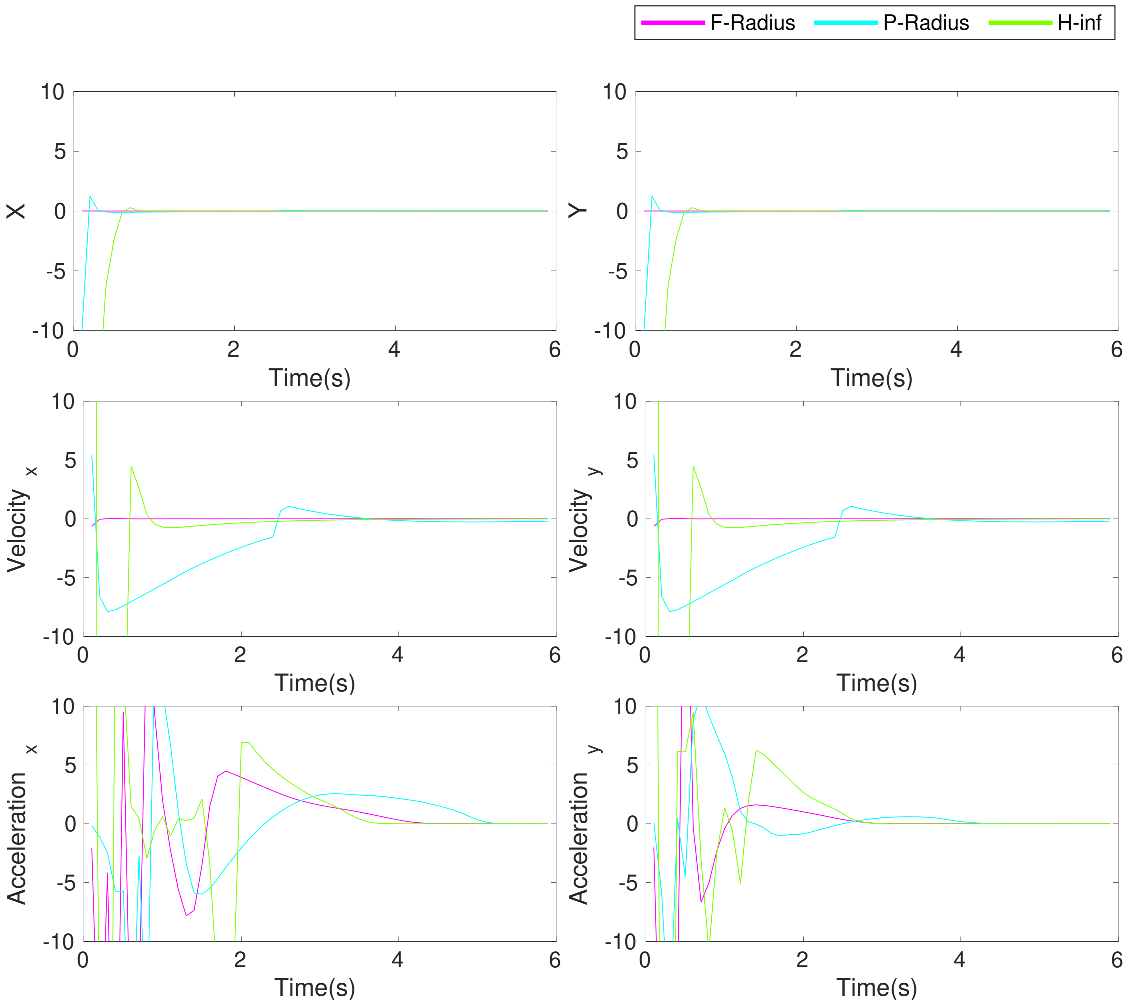


Figure A.12. Rate of change of bounds using the point-mass model

B Template Code

B.1 Template for Model

```
classdef Model
    %MODEL Model Template
    %   delT: time step

    properties
        A
        C
        W
        V
        initial
        dim_x; % estimate dimension
        dim_y; % measurement dimension
        delT = 0.1;
    end

    methods
        function obj = Model(delT)
            %MODEL Construct an instance of this class
            %   Initializes the model variables
            obj.delT = delT;

            % Assign parameters here
        end
    end
end
```

B.2 Template for Estimator

```
classdef Estimator < handle
    %ESTIMATOR Estimator template
    %   model: model to represent vehicle dynamics

    properties
        model
        E
        F
        x_estimated
        index
    end

    methods
        function obj = Estimator(model)
            %ESTIMATOR Constructor
            % Initialize variables
            obj.model = model;
            obj.index = 1; % keep track of measurements

            % initialize variables here

            % implement initial computation
        end

        function [x_upper, x_lower] = estimate(obj, y)
            %ESTIMATE estimate from the measurement

            % implement estimator algorithm on measurement, y

            % return bounds
            x_upper = 0;
            x_lower = 0;
        end

    end
end
```

List of Figures

List of Tables

Bibliography

[1] S. (2014). "Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems." In: *SAE International* (2014).

[2] M. Hirz and B. Walzel. "Sensor and object recognition technologies for self-driving cars." In: *Computer-Aided Design and Applications* 15.4 (2018), pp. 501–508. ISSN: 16864360.

[3] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems." In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223.

[4] V. Puig, P. Cugueró, and J. Quevedo. "Worst-case state estimation and simulation of uncertain discrete-time systems using zonotopes." In: *2001 European Control Conference (ECC)* (2001), pp. 1691–1697.

[5] C. Combastel. "A state bounding observer based on zonotopes." In: *European Control Conference, ECC 2003* (2003), pp. 2589–2594.

[6] T Alamo, J. M. Bravo, and E. F. Camacho. "Guaranteed state estimation by zonotopes." In: (2005).

[7] V. T. Le, T. Alamo, E. F. Camacho, C. Stoica, and D. Dumur. "Zonotopic set-membership estimation for interval dynamic systems." In: *Proceedings of the American Control Conference* (2012), pp. 6787–6792. ISSN: 07431619.

[8] F. Mazenc and O. Bernard. "Interval observers for linear time-invariant systems with disturbances." In: *Automatica* 47.1 (2011), pp. 140–147. ISSN: 0005-1098.

[9] T. Raïssi, D. Efimov, and A. Zolghadri. "Interval State Estimation for a Class of Nonlinear Systems." In: 57.1 (2012), pp. 260–265.

[10] W. Tang, Z. Wang, Y. Wang, and T. Ra. "Interval Estimation Methods for Discrete-Time Linear Time-Invariant Systems." In: *IEEE Transactions on Automatic Control* 64.11 (2019), pp. 4717–4724.

[11] M. Althoff. "CommonRoad : Vehicle Models." In: (2016).

[12] J. J. Rath and M. Althoff. "Evaluation of Set-Based Techniques for Guaranteed State Estimation of Linear Disturbed Systems." In: (2020).

[13] M. Althoff. “ur Steuerungs- und Regelungstechnik Technische Universit ¨ at M ¨ unchen Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars.” In: (2010).

[14] M. Althoff, N. Kochdumper, and C. Arch. “CORA 2018 Manual.” In: (2018).

[15] R. Schubert, E. Richter, and G. Wanielik. “Comparison and evaluation of advanced motion models for vehicle tracking.” In: *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008* 1 (2008).

[16] V. Puig. “Fault diagnosis and fault tolerant control using set-membership approaches: Application to real case studies.” In: *International Journal of Applied Mathematics and Computer Science* 20.4 (2010), pp. 619–635. ISSN: 1641876X.

[17] X. Ge, Q.-L. Han, X.-M. Zhang, D. Ding, and F. Yang. “Resilient and secure remote monitoring for a class of cyber-physical systems against attacks.” In: *Information Sciences* 512 (2020), pp. 1592–1605. ISSN: 0020-0255.

[18] J. Huang, X. Ma, H. Che, and Z. Han. “Further Result on Interval Observer Design for Discrete-time Switched Systems and Application to Circuit Systems.” In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 7747.c (2019), pp. 1–1. ISSN: 1549-7747.