[1]Ayanda Nkosi
[2]Mbuyu Sumbwanyambe
[3]Tlotlollo Hlalele

Simulation of a Radar-Based 2D Object Tracking and Velocity Estimator in the X-Y Plane

Journal of Electrical Systems

Abstract: - This study tackles the significant challenge of augmenting accuracy in the two-dimensional (2D) tracking of mobile objects, particularly in environments plagued by high noise and data uncertainty. Traditional Kalman filter-based methods often struggle under these conditions, failing to deliver the needed precision. Consequently, the primary objective of this research is to devise, implement, and validate a refined linear Kalman filter approach. This approach aims to significantly diminish estimation errors in tracking object positions and velocities, addressing the identified limitations of existing methodologies. The approach involves an innovative adaptation of the linear Kalman filter technique, rigorously evaluated through detailed simulations and analysis to ascertain its effectiveness in enhancing tracking accuracy under challenging conditions. The results show that a linear Kalman Filter can be used to accurately estimate the position of mobile objects, with root mean square errors of less than 0.1%. We demonstrate marked improvements in accuracy and reliability for 2D object tracking, showcasing the method's potential applicability in real-world scenarios such as autonomous navigation, wildlife and automobile tracking systems. While recognizing the persistent challenges posed by noise variations, this research paves the way for future exploration into nonlinear applications and update processes. Such investigations are anticipated to further refine object tracking methodologies, building on the foundational work presented here.

Keywords: Kalman filters, Estimating, Object velocity, Position, Radar-based systems.

I. INTRODUCTION

From wildlife monitoring to autonomous navigation, object tracking in two-dimensional spaces is a crucial element across various engineering fields. Traditionally, studies have relied on remote measurements to track objects without their own sensing capabilities [1,2].

As we move forward in the digital age, there's a shift toward a method known as probabilistic data fusion. Imagine trying to pinpoint a location in a bustling city based on several uncertain tips; this method similarly combines different data sources to enhance the precision of object tracking, despite the inherent uncertainties in each data point [3,4].

To better understand how we track a vehicle's movement, consider this simplified analogy: Just as weather forecasts use probabilities to predict the likelihood of rain, we use similar methods to estimate a vehicle's position. In the real world, measurements like those from a car's GPS aren't always perfect due to various factors like signal interference or obstacles.

Thus, we combine GPS data with other sensor information to improve our predictions, treating the vehicle's position and movement as probabilities rather than certainties. This approach, rooted in a statistical method known as Bayesian inference, is widely applicable, from helping self-driving cars navigate to enhancing military tracking systems. At the heart of our study are two key elements: how accurately we can estimate where the vehicle is (its position) and how fast it's going (its velocity) within a two-dimensional space, like a flat map [4].

Imagine trying to predict where a ball will land while it's still in the air, using only its past movements to guide you. This is like the task we face in tracking objects, where we continuously predict and then update our predictions based on new data. In this back-and-forth process, our certainty about the object's location can fluctuate. Here, Kalman filters become essential tools. They help us refine our predictions using a mathematical approach that combines past data with new information, much like updating your guess of the ball's landing spot with every bounce it takes. This method is efficient enough to work even on devices with limited computing power, making it widely applicable. [5].

[1] *Corresponding Author 1: University of South Africa, Department of Electrical & Smart Systems Engineering (Email: 64261425@mylife.unisa.ac.za)
[2] Author 2: University of South Africa, Department of Electrical & Smart Systems Engineering
[3] Author 3: University of South Africa, Department of Electrical & Smart Systems Engineering

This study is centered around the purpose of designing, developing, and critically evaluating a 2D Tracking Filter. With the help of remote radar measurements, this instrument assesses the velocity and positional attributes of mobile entities, specifically in the context of South Africa. Among the key objectives of this task are: understanding current object tracking methods, collecting radar sensor data, designing 2D Tracking Filters, simulation-based evaluations, improving the filter for optimal efficiency, evaluating its broader impacts, and sharing results with academics and stakeholders.

The scope of this study must, however, be clearly defined. Within a 2D framework, it emphasizes objects moving at consistent velocities. In addition, most of the research relies on simulations, which are limited by a specific time frame. Although the study covers a broad spectrum, it avoids going into exhaustive detail about the economics and the environment.

Likewise, by deliberately excluding complex factors like radar jamming and extreme weather conditions to maintain analytical focus. Radar jamming, an intricate electronic countermeasure, and extreme weather conditions, which can significantly affect radar signal quality, require specialized analyses beyond this study's probabilistic and mathematical scope.

By isolating the Kalman filter's intrinsic performance from these variables, we aim to provide clear insights into its core capabilities. However, acknowledging these exclusions highlights essential future research directions, including the integration of electronic countermeasures and meteorological impacts in tracking system analyses to develop more robust and comprehensive tracking solutions.

This research marks a significant advancement in simulation methodologies, closely mirroring the complexities encountered in real-world scenarios. It bridges a notable gap in existing literature by offering solutions that are specifically designed for challenges prevalent in South Africa. By refining our models to minimize position and velocity errors to remarkably small margins, we can vastly improve the efficiency of tracking systems.

This precision is crucial for applications like wildlife monitoring, where it enables the accurate tracking of animal movements, and in combating crime, particularly in disrupting operations of criminal syndicates involved in poaching or vehicle theft. The insights derived from this focused, simulation-based study are expected to have far-reaching effects on enhancing radar-based tracking technologies, providing a robust framework that can be adapted to various practical contexts beyond the initial focus on South African challenges.

This paper is arranged as follows Section (I) deals with the Introduction, Section II provides a Problem Statement which precisely defines the issue at hand. Section (III) provides the Background and Related Work highlights industry challenges and a selective literature review. The Section IV outlines the Theoretical Background which presents essential concepts and formulae. Section (V) deals with the Methodology, which covers system design, mathematical formulations, component derivations, and simulation tools. Section (VI) provides the Decision Matrix and Solution Selection, which briefs on the decision process and the chosen solution based on techno-economic factors. Section (VII) outlines the Impact Assessment, which evaluates the design's broader implications. Section (VIII) deals with the Uncertainty and Risk, which deliberates on potential project pitfalls. The narrative concludes with Summary and Conclusions (IX), followed by a comprehensive list of References.

II. Problem Statement

With South Africa's vast and diverse terrain, tracking objects' movements becomes very important for transportation, wildlife monitoring, and other applications. Given that position data is acquired via radar-based sensors, and that objects move at a consistent speed in a 2D X-Y plane, it's imperative to have a reliable way to estimate both their position and velocity. In this paper, we develop a 2D Tracking Filter that is capable of accurately and efficiently estimating the position and velocity of moving objects based on remote radar measurements.

III. Literature Review

Various scholars have explored various strategies to address the 2D tracking challenge. For example, U. B. Gohatre et al. have adopted a computer vision methodology to tackle the 2D estimation issue [6]. Their focus on

real-time tracking through stereo vision and predictive analytics achieves a significant frame rate, highlighting potential value for time-critical applications.

However, the reliance on stereo vision introduces limitations in depth perception [7], a challenge frequently faced in real-world environments with complex 3D geometry. Moreover, such computationally intensive approaches might face challenges in scenarios with limited hardware resources. Our research explores an alternative, resource-efficient approach based on Kalman filtering, aiming to balance accuracy and computational demands specifically for object tracking within a 2D plane.

Noise is a pervasive challenge in image processing, impacting the accuracy and reliability of 2D tracking systems. Techniques like median filtering are commonly employed to mitigate noise effects, enhancing image clarity and object tracking precision. However, such techniques introduce constraints, particularly in rapid deployment scenarios where continuous camera calibration and stereo setup add complexity and inflexibility.

The limitations of rapid image processing deployment are well-documented, including computational overhead and the need for frequent model updates in dynamic environments [8]. Furthermore, environmental factors can significantly impact image quality, posing challenges for traditional approaches [9]. Despite the real-time tracking capabilities of algorithms employing median filters, their performance benchmarks can be difficult to align with more established methods, notably the Kalman filter, which is renowned for its efficacy in various tracking contexts [1, 4, 7].

The existing literature indicates a gap in addressing noise impacts with enough flexibility and speed, especially in dynamic environments requiring quick adaptation without extensive recalibration [8, 9]. Our study contributes to this discourse by proposing a method that combines the robustness of median filtering against noise with the adaptability and computational efficiency of the linear Kalman filter. By integrating these approaches, we aim to offer a novel solution that maintains high tracking accuracy while accommodating the fast-paced requirements of real-world applications, such as autonomous navigation systems and dynamic surveillance, where rapid and reliable object tracking is paramount.

C. Zhang et al. introduce an innovative methodology to the realm of 2D object tracking by integrating 2D and 3D computer vision techniques, enhancing robustness and adaptability across various environmental conditions [10]. Their approach, leveraging Zhang's calibration and RANSAC algorithms for plane extraction, aims to elevate the precision in generating 3D point clouds. To boost the reliability of the homography matrix, they employ the SIFT algorithm alongside the MSAM technique, ensuring accurate keypoint identification and thus, enhancing the model's robustness [10, 11]. However, the algorithm's heavy reliance on detailed human movement patterns and substantial computational demands could limit its real-time application in resource-constrained settings.

The literature acknowledges the potential of combining 2D and 3D visions to surmount the limitations of purely 2D methods, especially in complex motion scenarios [12, 13]. Yet, the computational intensity and potential inaccuracies in unpredictable motion contexts highlight a significant gap — the need for an efficient, less resource-intensive solution capable of maintaining high accuracy without extensive computational demands. This study seeks to address this gap by proposing a less computationally intensive alternative that retains the benefits of 3D enhancements in 2D tracking, potentially impacting a variety of real-world applications where rapid, accurate tracking is crucial yet computational resources are limited.

Zhi Jin et al. introduced an innovative ground plane detection algorithm utilizing depth maps, termed the Depth-map Driven Planar Surface Detection (DDPSD) method. This technique advances the field by growing a plane from the largest contiguous area in the depth map with consistent depth values, which is assumed to be the ground plane [14]. By integrating dynamic threshold adjustments and seed patch growing techniques, DDPSD significantly improves planar surface detection, offering a strategic solution to the common issue of over-segmentation in image and video data processing [14, 15].

Such advancements are crucial in applications where precise planar detection is vital, such as in augmented reality interfaces where accurate anchoring of virtual elements is essential [16, 17] or the development of navigation

systems for autonomous vehicles, where ground plane estimation is fundamental for path planning and obstacle avoidance [18, 19]. The method's reliance on depth maps for primary data not only enhances its detection accuracy but also expands its utility in detecting semi-planar surfaces, a capability that broadens the spectrum of its applicability.

However, the DDPSD's two-tiered detection process, which depends heavily on seed patch validation, introduces certain complexities. These include potential computational burdens when managing extensive depth maps and difficulties in segmenting intricate planar formations like multi-layered staircases. This study delves into these limitations by exploring optimizations of the DDPSD method to enhance its computational efficiency. The goal is to refine its suitability for real-time applications where resource constraints are a crucial factor alongside the need for accurate planar detection.

D. Y. Kim's study provides valuable insights into the cost-effectiveness of the Kalman Filter (KF) in multi-object tracking systems that utilize a variety of physical sensors [20,21]. His research is particularly notable for introducing an innovative data fusion method that leverages affordable radar modules and CCD cameras to amalgamate data from these disparate sources effectively [22]. This fusion technique underscores a pivotal advancement: systems that integrate data from multiple sensors manifest a substantial improvement in tracking accuracy compared to those reliant on a singular sensor source.

This finding is crucial as it directly addresses a gap in existing literature—how to enhance tracking system accuracy in a cost-effective manner without relying on expensive or complex hardware. The real-world implications of this research are significant, particularly for applications where budget constraints are as pivotal as performance requirements, such as in public security, wildlife tracking, or even consumer-grade navigation systems.

Kim's approach, aligning the Kalman Filter's computational efficiency with the practical necessity for affordable tracking solutions, paves the way for broader application of advanced tracking systems in sectors where cost concerns might otherwise be a prohibitive factor. Our study builds upon this foundation, aiming to further refine data fusion methods to optimize both accuracy and cost-efficiency, thereby extending the practical reach of these systems in various industries.

This literature review has revealed a critical need for computationally efficient and robust tracking solutions that can tackle noisy real-world data while maintaining accuracy. Systems using a fusion of observations from more than one sensor significantly outperform those using just one sensor alone in terms of accuracy. Due to its demonstrated efficacy and efficiency, the KF was chosen in this study as the method for fusing data due to its demonstrated efficacy and efficiency.

IV. Decision Matrix and Solution Selection

Considering the emphasis on real-time performance for dynamic systems highlighted in our literature review, a crucial factor in our filter selection will be computational efficiency and the ability to provide reliable state estimates within strict time constraints. The Decision Matrix that follows is designed to methodically compare the potential candidates – Kalman Filters (KF), Extended Kalman Filters (EKF), and Unscented Kalman Filters (UKF) – against a set of criteria that are crucial for effective state estimation in dynamic systems. This comparative analysis is grounded in our foundational understanding of the tracking challenges identified, ensuring that our selection is closely aligned with the specific needs highlighted in our review.

A. Decision Matrix

Choosing the right filter for vehicle state estimation in intricate settings is pivotal. The main contenders are KFs, EKFs, and UKFs [17-20]. The selection often hinges on project-specific needs. An objective matrix, displayed in Table 1, assists in comparing these filters using predefined criteria, intending to highlight their pros and cons [28]. The evaluation metrics include:

- Application: Versatility across varied problems.
- Estimation Accuracy: Precision in predicting/correcting states.

- Computational Complexity: Resources needed, with higher scores favouring real-time applications.
- Nonlinearity Handling Technique: The efficiency in addressing nonlinearities.
- Real-world Implementation Ease: Ease of practical implementation.
- Integration with Lidar/GPS/Gyro: Sensor data fusion capability.
- Non-linearity Handling: Performance in non- linear system dynamics.
- Noise Resiliency: Stability against noisy inputs.
- Complexity of Tuning: Ease of parameter adjustments.
- Resource Requirement: Processor and memory demand, significant for embedded systems.
- Scalability: Handling growth in system states or measurements without much added complexity.

B. Solution Selection

The Decision Matrix evaluates each filtering method against criteria derived from our literature review, allowing us to identify strengths, weaknesses, and position our study within the research landscape. We recognize the inherent trade-offs between accuracy and computational demands, particularly when it comes to the KF's limitations in nonlinear systems versus the increased complexity and resource requirements of the EKF and UKF. For instance, while the Kalman Filter is recognized for its real-time efficiency and accuracy in linear systems, it may not be the best fit for systems with significant nonlinearities, where EKF or UKF could offer advantages.

While the EKF extends the KF's utility to nonlinear systems, it introduces approximation errors and increased computational load. Our analysis provides insights into the trade-offs between accuracy and computational demands, offering a nuanced perspective on when EKF's increased precision outweighs its higher resource requirements.

Similarly, the UKF offers improved handling of nonlinearities without the need for linearization. However, its computational burden is significant. Our study assesses the UKF's applicability in scenarios where accuracy in nonlinear state estimation is non-negotiable, identifying contexts where its benefits justify the computational costs.

This study contributes to the field by demonstrating how, despite its inherent limitations in handling nonlinearity, the KF can be optimized for certain real-world applications where computational resources are constrained, thereby extending its applicability beyond traditional domains. filtering methods. Among the reasons the Kalman Filter (KF) stood out are:

- Through its linear framework, this application is efficient in real-time.
- The ability to accurately estimate linear systems, especially when approximated as linear with Gaussian noise.
- Compared to both EKF and UKF, this group performed better.

V. THEORETICAL BACKGROUND

Kalman filters rely on the assumption that all probability distributions associated with state and measurement can be modeled as Gaussian distributions. This simplifies the Bayesian computations extensively. The filter's efficiency in numerous real-world applications positions it as a cornerstone in data fusion methodologies in various industries [17, 18].

The Kalman filter operates in two main phases:

Prediction Step:

The state and covariance are predicted using the process model. For a state x and covariance P, the prediction is given by equations 1 and 2 respectively [17, 18]:

$$x_k^- = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} \quad (1)$$

$$P_k^- = F_{k-1}P_{k-1}F_{k-1}^T + L_{k-1}Q_{k-1}L_{k-1}^T \quad (2)$$

Where F is the state transition model, G is the control input model, L is the process noise sensitivity matrix, and Q is the covariance of the process noise [17, 18].

Update:

Upon receiving a measurement, the state and covariance estimates are corrected using equations 3 to 5 respectively:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1} \quad (3)$$

$$x_k = x_k^- + K_k(z_k - H_k x_k^-) \quad (4)$$

$$P_k = (I - K_k H_k) P_k^- \quad (5)$$

Where H is the measurement model, M is the measurement noise sensitivity matrix, R is the covariance of the measurement noise, K is the Kalman gain, and z is the actual measurement. The discrete-time nature of the Kalman filter allows it to work effectively on computational systems like microprocessors [17, 18].

In a continuous system, variations of the state are both smooth and continuous, but in the discrete realm, progression happens in increments. Noise assumptions form the bedrock of the filter's operation. Both process and measurement noises are assumed to emanate from Gaussian distributions with zero mean. Crucially, Q and R represent their respective covariances. Furthermore, no temporal correlation is assumed among these noise variables, and independence between process and measurement noise is a staple assumption.

VI. METHODOLOGY

A. System Formulation

In a two-dimensional tracking filter, distal measurements are used to determine both the position and velocity of a mobile object. In contrast to internal sensors on the target, these measurements are derived from external sources, rendering the target essentially uncooperative.

Radars and various other sensors are commonly used for providing these external measurements. As a result of this filter's versatility, you can use it for radar-based aircraft tracking, optical tracking in autonomous vehicles, object tracking in image sequences, as well as surveillance, military operations, robotics, and GPS-assisted navigation [26-28].

Figure 1 visually represents the two-dimensional motion tracking scenario, where the X and Y axes define the plane of motion. The 'Dynamics' graph demonstrates the theoretical trajectory and velocity of an object in motion, represented by a vector originating from the point (p_x, p_y), which marks its current position, and extending to indicate the velocity components (v_x, v_y) at that position.

In contrast, the 'Measurements' graph depicts a scatter of data points; each signifies an independent position measurement (p_x, p_y) of the object obtained from external sensors. The relative sparsity and arrangement of these points reflect the inherent inaccuracies and noise associated with real-world sensor data. [17-20].

The workflow illustrated in Figure 2 explains the two essential stages of the Kalman filter algorithm: the “Time Update” and the “Measurement Update”. Starting from the “a posteriori” estimates of the previous time step, represented as $\hat{x}_{k-1}^+$ and $\hat{P}_{k-1}^+$ for the mean and covariance respectively, the filter proceeds to the prediction phase. In this stage, the filter uses a predefined linear system model to anticipate the state of the system at the next time step, generating a priori estimates $\hat{x}_k^-$ and $\hat{P}_k^-$.

Following the prediction, the Measurement Update phase commences, wherein new observational data, depicted as z_k, along with its measurement covariance R_k, is incorporated to refine the predictions. This phase is critical as it adjusts the predicted state by considering the actual measurements, culminating in the updated, or “a posteriori”, estimates $\hat{x}_k^+$ and $\hat{P}_k^+$. These refined estimates then form the basis for predictions in the next cycle, demonstrating the recursive nature of the Kalman filter.

It is imperative to note that the elegance of the Kalman filter resides in its recursive nature, wherein it leverages previous post-measurement estimates to forecast current premeasurement estimates, subsequently refining them using present measurements.

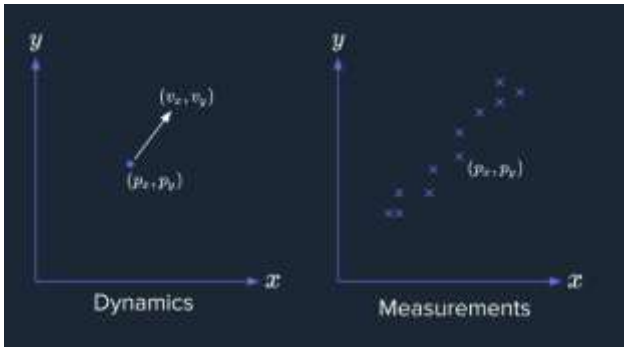

Figure 1. Shows a graphical description of the problem [28].

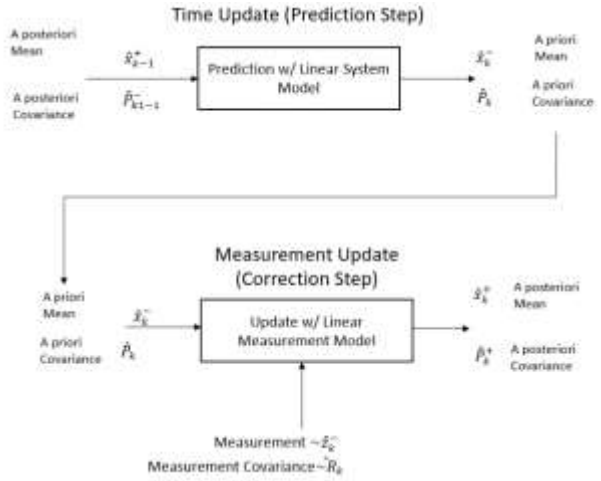

Figure 2. Shows a system model diagram of the workings of a KF [28].

B. Mathematical formulation of subsystems

The 2D tracking system model, as depicted, is formulated based on the classical equations of motion. Initially, acceleration a is defined as the second derivative of position with respect to time, as given in equation 7. Under the assumption of constant acceleration, the relations between velocity v and position p in relation to their initial values v_0 and p_0 are represented in equations 8 and 9 respectively.

For a clearer representation, these equations of motion can be transcribed into matrix form as shown in equation 9. To account for the discrete nature of most digital systems, the continuous time representation is translated to a discrete time format, where t is replaced by Δt, as detailed in equation 10.

Key to understanding the system behavior is the state vector x_k, which consolidates the position and velocity components in both x and y dimensions, as articulated in equation 11.

Moreover, external disturbances or uncertainties in the system, such as sensor noise, are encapsulated within the noise vector w_k, where a_x and a_y are Gaussian noises with zero mean and variances σ_x^2 and σ_y^2 respectively, as presented in equation 12.

The discrete process model in equation 13 characterizes the evolution of the system state over time, integrating the state and noise vectors. This equation forms the foundation for the prediction step in equation 14, which anticipates the next state based on the current state and control inputs.

Moving forward to the update phase, the state is corrected based on new measurements. The prediction error covariance P_k^- is computed using equation 15, providing a measure of the estimated accuracy of the state prediction. The updated or corrected state, represented as $\hat{x}_k^+$, is then computed using equation 16 where K_k is the Kalman gain, a factor that determines the weightage of the measurement update.

This gain is derived using equation 17 and serves to minimize the estimation error. The updated error covariance, P_k^+, as detailed in equation 18, provides a post-corrected estimate of state accuracy. To further enhance

this correction, S_k, the measurement prediction covariance, is computed using equation 19 which integrates the effect of the system's inherent noise.

This mathematical construct efficiently combines prediction and update steps to estimate the position and velocity of an object in a 2D plane, effectively accounting for uncertainties and delivering a refined state estimate.

2D Tracking System Model:

- **Derive the System Process model from the Equations of Motion:**

$$a = \frac{dv}{dt} = \frac{d^2p}{dt^2} \quad (6)$$

- **Assuming constant acceleration, the relations are found:**

$$v = v_0 + at \quad (7)$$

$$p = p_0 + tv_0 + \frac{1}{2}at^2 \quad (8)$$

- **Write the Equations of Motion in Matrix Form:**

$$\begin{bmatrix} p \\ v \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}t^2 \\ t \end{bmatrix} a \quad (9)$$

- **Convert to Discrete Time: $t = \Delta t$**

$$\begin{bmatrix} p \\ v \end{bmatrix}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix} [a] \quad (10)$$

$$\text{State Vector: } x_k = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix} \quad (11)$$

$$\text{Noise-Vector} w_k = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \quad \begin{matrix} a_x \sim N(0, \sigma_{a_x}^2) \\ a_y \sim N\left(0, \sigma_{a_y}^2\right) \end{matrix} \quad (12)$$

- **Discrete Process Model:**

$$x_k = \mathbf{F}x_{k-1} + \mathbf{L}w_{k-1}$$

$$\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_k = \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_{k-1} + \underbrace{\begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}}_{\mathrm{L}} \begin{bmatrix} a_x \\ a_y \end{bmatrix}_{k-1} \quad (13)$$

- **Prediction Step:**

$$\hat{x}_k^- = \mathbf{F}_{k-1}\hat{x}_{k-1}^+ + \mathbf{G}_{k-1}u_{k-1} \quad (14)$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \quad (15)$$

- **Update Step:**

$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k(z_k - \mathbf{H}_k\hat{x}_k^-) \quad (16)$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^- \quad (17)$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{S}_k^{-1} \quad (18)$$

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k \quad (19)$$

VII. Computer Simulation

Prediction Step:

The initial phase of designing a 2D tracking filter involves a prediction step. As shown in equation 14, the process model has historically been represented as shown in equation 15. When coupled with state and noise vectors, Equation 15 presents the state transition matrix taking the noise into account. Considering the system's noise properties, the random vector adheres to a Gaussian distribution with covariance Q as expressed by equation 15 [17, 18].

There are distinct noise variances for X and Y accelerations in this distribution, illustrated as a diagonal matrix. In the case of non-additive noise, this method encounters a challenge in Kalman filter's process noise prediction,

which requires converting variance from process model units to state units through the EL transformation represented by equation 12.

The Kalman filter becomes ill-conditioned over time if this transformation fails to yield a valid Gaussian distribution, which necessitates modifying it. A single variance is posited for both X and Y accelerations, resulting in a 1D Gaussian distribution for the noise. A scalar EL matrix facilitates additive noise. Equation 17 illustrates the resulting covariance. As a result, equation 12 represents the new covariance matrix for the random vector W.

In this research, we employ the revised version to ensure the system's robustness and stability. Implementing Kalman filter prediction equations in Python, as outlined in previous literature, is the immediate objective. To begin, we need to initialize the state and covariance, consider initial positions and velocities, and then construct the matrices F and Q. In subsequent steps, the Kalman filter prediction step will be implemented, as shown in equations 14 and 15.

In subsequent steps (steps 1-7), detail the underlying mechanisms and simulation results will be discussed in greater depth.

- **Step 1)** Open the python file 'prediction.py', that is presented by code in Appendix 1.
  a. Run the simulation as is. See that the object starts at the origin (px,py) $= (0,0)$ and moves at a 45 deg angle at $10\ \mathrm{m/s}(\mathrm{vx},\mathrm{vy}) = (7.07,7.07)$

- **Step 2)** Setup the initial state and covariance
  a. Assume initial position is $(0,0)$ and initial velocity is $(7.07,7.07)$
  b. Assume no initial uncertainty (Zero matrix)

$$x = [0,0,7.07,7.07]^T$$
$$P = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- **Step 3)** Setup the model F and Q Matrices
  a. Use the time step and define the F process model matrix
  b. Define the Q matrix as a function of a variable accel_std
  c. Assume the process model noise acceleration stdev is zero initially

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\mathbf{Q} = \sigma_a^2 \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix}$$

- **Step 4)** Implement the Kalman Filter Prediction Step Equations
  a. State Prediction
  b. Covariance Propagation
- **Step5)** Run the Simulation
  a. Check that the Prediction follows the truth closely.

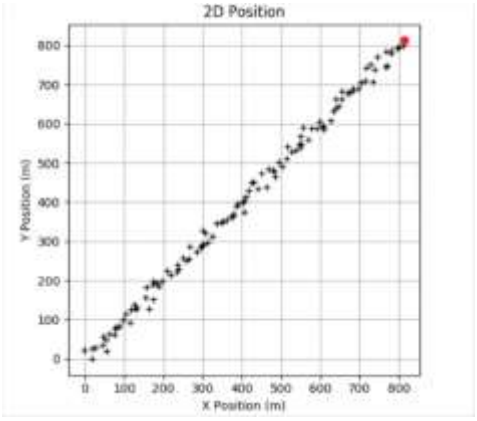

Figure 3. Resulting simulation from initial test.

```
Position Mean Squared Error: 0.010959819397519322 (m)^2
Velocity Mean Squared Error: 2.2804443600974467e-06 (m/s)^2
```

Figure 4. Resulting position and velocity errors from simulation test.

Following the implementation of the Kalman filter, it is observed, as per Equation 14 and 15 [11], that the prediction aligns notably well with the true trajectory. Given the precise knowledge of the initial conditions—both the starting position and velocity—the anticipated outcome is a minimal mean error, particularly in terms of the resultant velocity error post-simulation. Should there be significant deviation from this expectation, it may signal an erroneous implementation. Therefore, the congruence between the predicted state and the actual trajectory serves as a validation of the filter's operational accuracy. Figures 3 and 4 show the results of the simulation.

- **Step 6)** Check the Position Covariance Prediction is Working Correctly

  a. Set the initial position x and y covariance to be $(5)\text{^}2$

  b. Run the simulation and see that the (3 Sigma) position uncertainty stays at approximately $+/-15$ m.

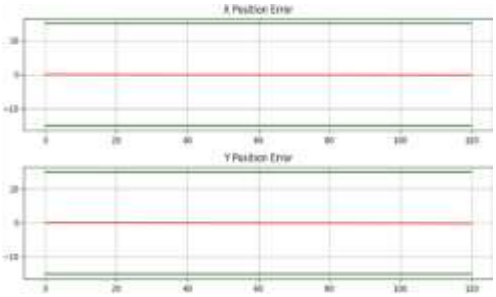

Figure 5. Resulting observed three-sigma position uncertainty remains consistently within an approximate range of ±15 meters.

Upon executing the simulation once more, the observed three-sigma position uncertainty remains consistently within an approximate range of ±15 meters, as stipulated by Equation 15 and 16. This observation serves as an empirical affirmation that the position covariance prediction, as integrated within the Kalman filter framework, is functioning as intended. As illustrated by Figure 5.

- **Step 7)** Check the Velocity Covariance Prediction is Working Correctly

  a. Set the initial state to be all zero.

  b. Set the initial position covariance to zero and the initial velocity x and y covariance to be $(7/3)\text{^}2$.

  c. Run the simulation and see that the (3 Sigma) error position uncertainty grows at the same rate as the position changes.

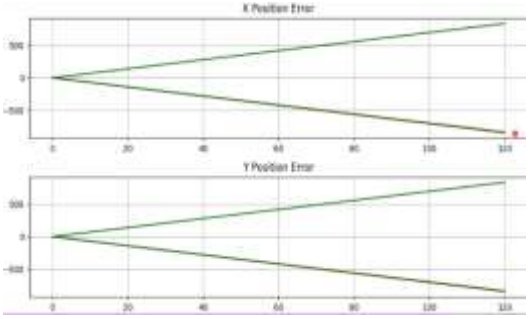

Figure 6. velocity covariance prediction aligns closely with the genuine velocity, specifically in relation to the error's rate of change.

In evaluating the performance of the KF, it is evident from Figure 6 that the velocity covariance prediction aligns closely with the genuine velocity, specifically in relation to the error's rate of change. Such an observation underscores the accuracy and robustness of the KF in estimating system states, further highlighting its reliability in tracking applications.

- **Step 8)** Check the Acceleration Covariance Prediction is Working Correctly

a. Set the initial state back to the original value.
b. Set the initial covariance to be all zero.
c. Set the process model accel_std to be 0.1.
d. Run the simulation and see that the (3 Sigma) velocity uncertainty grows quadratically with time.

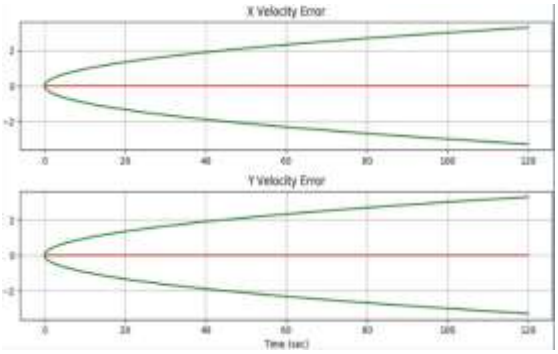

Figure 7. Shows that the three sigma velocity grows quadratically through time as expected.

Figure 7 shows that the three-sigma velocity grows quadratically through time as expected. It appears that the velocity uncertainty starts at zero and has a quadratic shape that grows with time until it reaches a value almost at three sigma.

Update Step:

Python script titled Update Mechanism was written to gain a deeper understanding of the update mechanism A single update was performed with "One_Update". As soon as the script is executed, an object is shown moving from the origin with variable velocity components, and, both of which remain undetermined. The purpose of this exercise is to estimate these parameters more accurately using the Kalman update step. In the script's initialization segment, matrices , , and are delineated as shown in Equations 1 and 2. Furthermore, the measurement precision is assumed to be 10 standard deviations, thus affecting the result.

Within the Python script's "Update_Step" function (Equation 3), the Kalman filter update equations are integrated.

Simulations run after Kalman filter implementation reveal that the state estimates are actively altered over time by the Kalman filter. Nevertheless, a static filter estimate is produced by setting the initial uncertainty to zero. As a result, the initial uncertainty of velocity is adjusted to 10 in order to compensate. Object tracking is demonstrated in the revised simulation as a result of shrinking and converging covariance around the object, as time progresses. Moreover, minimized errors in mean position and velocity squared are further evidence of effective tracking.

In subsequent steps (steps 1-4), detail the underlying mechanisms and simulation results will be discussed in greater depth.

- **Step 1)** Open the python file 'update.py'

Run the simulation as is. See that the object starts at the origin $(px, py) = (0,0)$ but it moves in a random direction and speed, so (vx,vy) are unknown. Each simulation run movies with a different initial velocity.

- **Step 2)** Setup the H Matrix and R Matrix

a. Assume the position measurement std is 10.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_{\text{meas}}^2 & 0 \\ 0 & \sigma_{\text{meas}}^2 \end{bmatrix}$$

- **Step 3)** Implement the Kalman Filter Update Step Equations

def update_step(self, measurement).

- **Step 4)** Run the Simulation
  a. See that the Kalman Filter Estimate does not change or be updated. (Initial Uncertainty is zero)
  b. Change the initial velocity std to 10 and re-run the simulation.

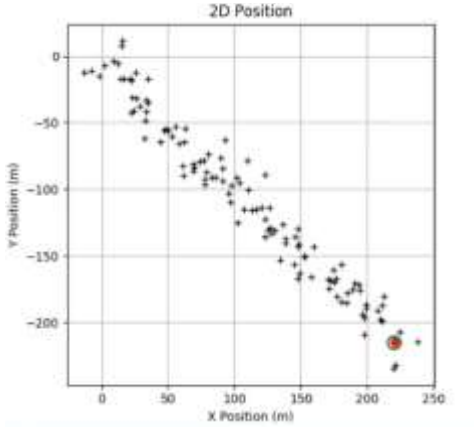

Figure 8. Shows the position graph from the update step, the green circle shows the innovation.

```
Position Mean Squared Error: 13.598641307721596 (m)^2
Velocity Mean Squared Error: 1.0297190032573558 (m/s)^2
```

Figure 9. Shows the new position and velocity error once the update step is activated.

Summaries for Each Simulation Step:

The following gives an overview, summary and the key findings of each of the experiments that have been conducted thus far, as well a rationale of each of the steps taken.

Step 1 Summary: The initial simulation run confirms the filter's capability to track an object's trajectory starting from the origin, moving at a predicted 45-degree angle. The velocity vectors observed align with the theoretical predictions, laying a strong foundation for subsequent, more complex simulations.

Step 2 Summary: With the state and covariance matrices initialized, the filter's predictions for position and velocity closely mirror the initial assumptions. This step validates our setup and is crucial for accurately reflecting the system's initial state with minimal uncertainty.

Step 3 Summary: Defining the process model and noise matrices is pivotal for capturing the system dynamics. The assumption of zero process noise as a starting point facilitates the evaluation of the filter's prediction capability under ideal conditions.

Step 4 Summary: The implementation of the Kalman Filter prediction equations yields results that are consistent with the expected trajectory. This consistency supports the reliability of our prediction model in linear system estimation.

Step 5 Summary: Upon simulation, the prediction aligns closely with the actual trajectory, suggesting the initial conditions are well understood and the model accurately reflects system dynamics.

Step 6 Summary: Adjusting the position covariance and observing the consistent range of three-sigma position uncertainty provides empirical evidence of the prediction's reliability and the filter's robustness against defined uncertainty levels.

Step 7 Summary: Setting the initial state and covariance to assess velocity predictions reveals that the filter accurately captures the rate of error growth, which is critical for dynamic tracking in systems where velocity is a primary variable.

Step 8 Summary: Introducing acceleration noise into the system and observing the quadratic growth in velocity uncertainty validates the filter's capacity to adapt predictions based on changing process noises, an essential aspect of real-world system tracking.

Update Step Summary: The update mechanism, when engaged, demonstrates the Kalman Filter's adeptness in refining state estimates with new measurements. The visible convergence of covariance around the object's trajectory and minimized positional and velocity errors are indicative of an effective tracking system.

VIII. RESULTS AND DISCUSSIONS

Filter State Prediction Model Check (State Transition):

Simulations that closely represent the system's dynamics are essential for building an effective and robust modeling system. It is recommended to validate the model in a noise-free environment with a known starting point and run the Kalman filter prediction step concurrently, as shown in equations 18 and 19. In this way, the Kalman filter model is validated against the dynamics of the system under observation.

The update step of the Kalman filter should be run separately from this step. Since the correction step within the update phase may mask discrepancies or inaccuracies inherent in the model, incorrect conclusions may result. By using such a methodology, the intrinsic prediction step can be finely tuned to the system model being estimated. Delving into the code will reveal this empirically, as shown Appendix 2.

Based on the state behavior in the simulation, when the process model noise is set to null and the state and covariance are initialized to zero, the system exhibits a trajectory inclined at 45° and maintains a constant velocity of 10 meters per second, as show by Figures 10, 11 and 12, respectively and this is consistent regardless of how many time the simulation is run, also note that the error is very large.

However, when the simulation is given an initial velocity of $7 m.s^{-1}$, the blue dot and red dots overlap which is in contrast with the previous result, as shown by Figures 13 and 14 respectively. The reason for this is that, is because we started with known conditions for velocity and position, note the errors given by L are very low.

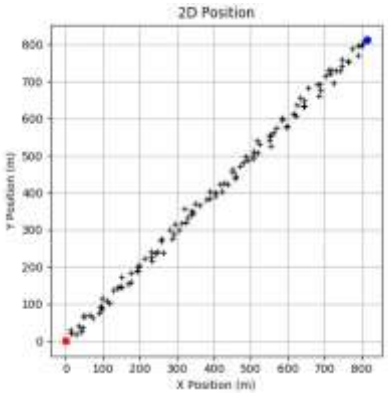

Figure 10. Shows the system exhibits a trajectory inclined at 45° and maintains a constant velocity of 10 meters per second, when then there is no initial velocity, note the lag between the blue dot and red dot.

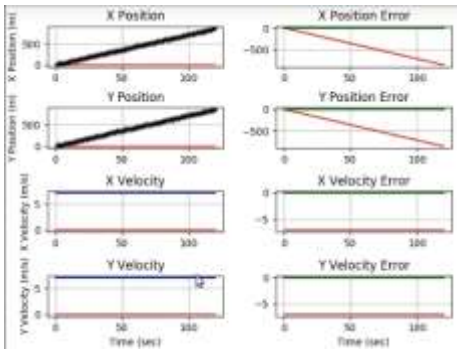

Figure 11. Position velocity graphs of the first test along with the resulting errors, for the transition state test.

```
X Position Measurement Innovation Std: nan (m)
Y Position Measurement Innovation Std: nan (m)
Position Mean Squared Error: 480600.1666666733 (m)^2
Velocity Mean Squared Error: 100.0 (m/s)^2
```

Figure 12. Resulting console out of the transition estimate test, note that the error is very large, when there is no initial velocity given.

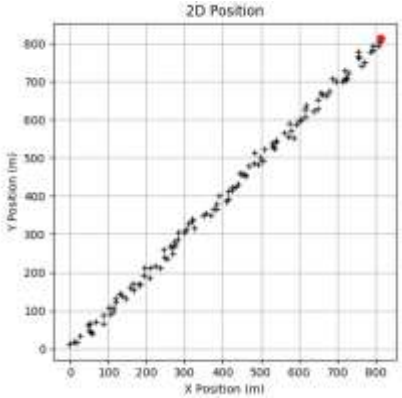

Figure 13. Shows the system exhibits a trajectory inclined at 45° and maintains a constant velocity of 10 meters per second, when initial velocity of 7m.s-1, the blue dot and red dots overlap which is in contrast with the previous result.

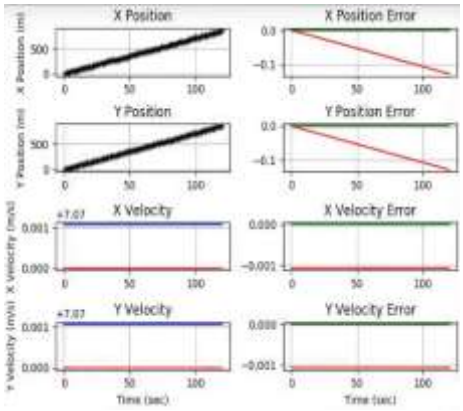

Figure 14. Position velocity graphs of the first test along with the resulting errors, for the transition state test.

```
X Position Measurement Innovation Std: nan (m)
Y Position Measurement Innovation Std: nan (m)
Position Mean Squared Error: 0.010959819397519322 (m)^2
Velocity Mean Squared Error: 2.2804443680974467e-06 (m/s)^2
```

Figure 15. Resulting console out of the transition estimate test, note that the error reduces significantly, when the initial velocity given.

Check that the Position Uncertainty is operating as Expected:

When the state transmission model is validated as being capable of accurately predicting the true state response from a defined initial condition, the focus shifts to how well it manages uncertainty transformations. Ideally, there should be no growth in the system's uncertainty when the process model noise is zero. Rather, it should be

transformed in accordance with the model of the system. In order to assess the position uncertainty's performance, an initial position uncertainty with a non-zero value is set while the velocity uncertainty remains at zero, this is done through the code in Appendix 2. In the proposed approach, the position standard deviation is set to five, yielding a variance of 25 for both X and Y axes.

An uncertainty ellipse is generated around the estimate after the developed code is executed, as shown in Figure 16. The graphical representation, in Figure 17 of the position velocity graphs, confirms that the model behaves as expected, with constant three-sigma bounds (green trajectories) for position errors. It is apparent that this behavior does not reflect an escalation or a decrease in uncertainty, but rather its maintenance throughout the simulation. As a result, the covariance prediction for position uncertainty appears to be accurate.

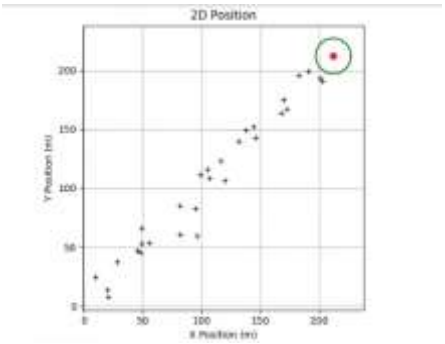

Figure 16. Shows how the uncertainty ellipse is generated around the estimate after the developed code is executed.

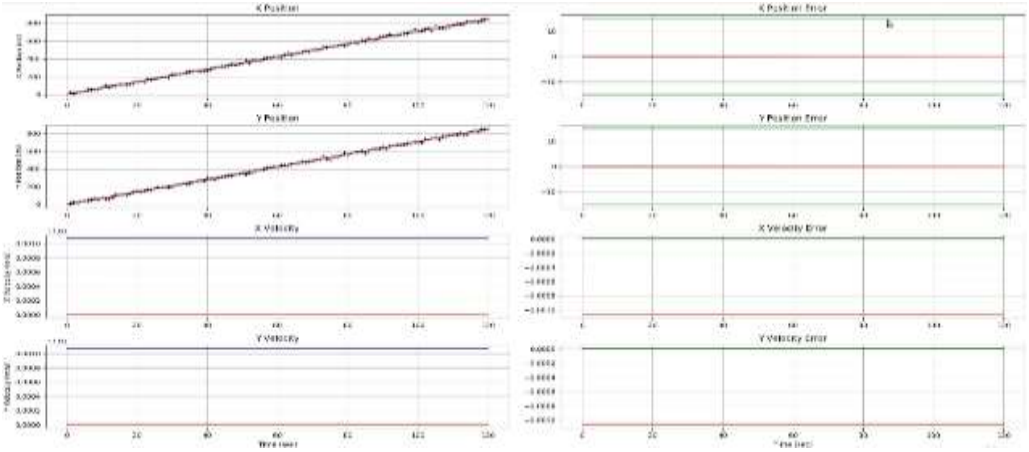

Figure 17. Position/ velocity graphs resulting from position uncertainty and is operating as expected.

Check that the Velocity Uncertainty is operating as Expected:

It is crucial that a filtering system maintains and propagates uncertainty in a way that is consistent with the dynamics of the system. For the filter's predictions to remain accurate, this is a prerequisite. The uncertainty in velocity can impact the uncertainty in position over time if the only dynamic is the integration of velocity. The explanation for this can be found in an experiment which starts with zero position uncertainty but introduces non-zero velocity uncertainty later in the experiment. Two hypotheses are proposed:

1. Because no external factors or model dynamics are introduced that could perturb velocity uncertainty, this bound should remain constant over time.
2. A linear growth pattern should be observed as the position uncertainty decreases over time. Based on the dynamics of the system, integrating a constant velocity results in a linearly changing position.

When the velocity uncertainty is initialized with a standard deviation of one in a simulation environment, the results confirm the hypothesis, shown in Figure 18. As a result of the filter's consistent handling of this parameter, the velocity uncertainty remains within three sigma bounds, as illustrated by Figure 19. As a result of integrating a constant velocity with time, the position uncertainty exhibits a linear growth pattern. The position uncertainty (captured by the three-sigma bound) at 120 seconds into the simulation reaches 360, which is expected based on the dynamics of the system.

It is evident from this empirical evidence that the filter is effective at handling uncertainties and propagating them. An effective uncertainty propagation mechanism ensures that the filter remains both consistent and reliable

by harmonizing with the system's inherent dynamics. When used in critical applications, where estimation errors can have severe consequences, such demonstrations are critical to building trust in the filter's performance.

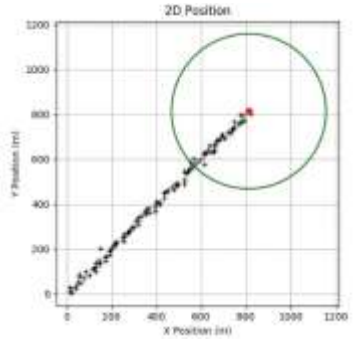

Figure 18. Shows the result when no external factors or model dynamics are introduced that could perturb velocity uncertainty.

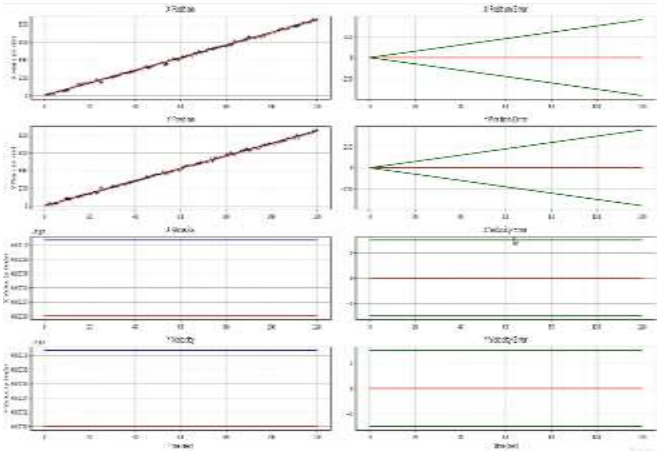

Figure 19. Show the resulting velocity/ position graphs no external factors or model dynamics are introduced that could perturb velocity uncertainty.

Check the Process Model Noise (Acceleration uncertainty):

The Q-matrix of the Kalman Filter (KF) estimation plays a pivotal role in the uncertainty equation of Kalman Filter (KF) estimation. Acceleration uncertainty is a key component of this noise. An experiment was designed where the initial uncertainties for position were set to zero, while an acceleration standard deviation was introduced to ensure that the Q-matrix was not zero.

The cumulative system uncertainty will progressively increase over time when the KF is executed with the parameters. It is directly related to the inherent dynamics of the system: as position is the second integral of acceleration, velocity uncertainty should grow linearly, while position uncertainty should grow quadratically. In Appendix 2, you will find a detailed description of how this analysis is implemented computationally.

A standard deviation of 0.1 was determined for acceleration in the present simulation. Throughout the simulation period, the system's uncertainty increased consistently. Specifically, the three-sigma bounds expanded, as predicted by theory. It was validated by examining the covariance matrices that velocity uncertainty increased linearly and position uncertainty increased quadratically.

Randomness or the Q-matrix becomes marginally significant in situations with deterministic system dynamics, like an object moving with consistent velocity from a predefined origin. Since there is no inherent uncertainty in the system, position and velocity estimates are the only sources of uncertainty.

In a broader sense, Q-matrix or process noise serves as a compensatory mechanism for potential discrepancies in KF process models. The Q-matrix adjusts by elevating internal system uncertainty if the model does not accurately represent the actual system dynamics. The KF remains consistent within its uncertainty bounds thanks to this adaptive mechanism. Process noise is intrinsically linked to measurement noise, and their combined influence on the filter's performance is crucial.

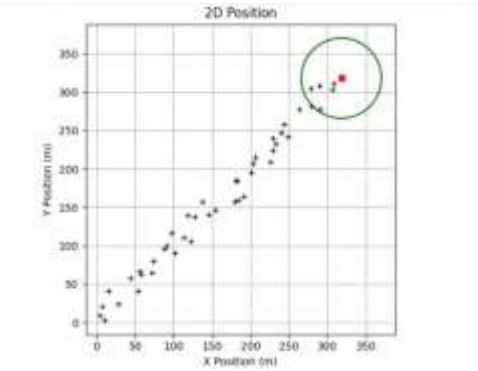

Figure 20. Resulting trajectory from the Process Model Noise (Acceleration uncertainty).

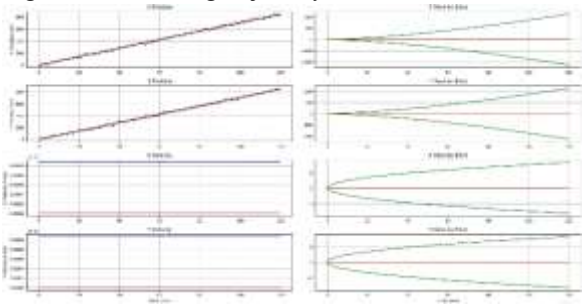

Figure 21. Resulting position/ velocity graphs Process Model Noise (Acceleration uncertainty) test.

Final Simulation with optimal KF Parameters

The code snippet in Figure 22 shows how to track a two-dimensional (2D) object with a recursive Kalman filter. Among other initialization conditions, this simulation includes several defining parameters, such as end time, measurement rate, and motion type. Kalman filters are also initialized with parameters like acceleration standard deviation and measurement standard deviation.

Measurement shown in Figure 23, indicates the Innovation charts for both X and Y dimensions illustrate how the actual measurements differ from the filter's predictions. blue lines here represent the innovations for each of the update steps inside the KF. While the green dotted line here represents one sigma bounds of the covariance uncertainty for the innovations.

In order to understand anomalies or significant deviations from expected values, this distinction is crucial. The position and velocity estimation charts provide insight into the Kalman filter's accuracy in tracking 2D object motion. Based on the data, it appears that KF predictions are largely within the error bounds provided, suggesting an expected margin of error, as shown by Figure 23.

As can be seen in the 2D trajectory visualization, the object's path is random at first, in Figure 24 (this is illustrated by the red and blue dot far apart), and eventually converging towards a more consistent trajectory as speed and heading parameters are randomized, as in Figure 25 (this is illustrated by the red and blue dot overlapping). As a result, the dynamic nature of the system is reaffirmed. As a result of the simulation setup and resulting charts, the Kalman filter appears to be an effective solution for tracking 2D objects, effectively handling the inherent dynamics and uncertainties.

In Figure 26 the graphs below illustrate the relationship between X-position versus Y-position as well as X-velocity versus Y-velocity. The blue trajectory represents the true state of the system. As shown on the X-axis, the time progression provides insight into the object's positional evolution over time. In a similar manner, the trajectory reveals the object's time-dependent position on the Y-axis.

The black markers on these plots delineate the position measurements for both X and Y dimensions. The red trajectory represents the estimated positions in the respective axes, as determined by the Kalman filter. In the lower panel, the velocity states for both dimensions are shown, demonstrating that the true state maintains a steady velocity. The X-axis moves at approximately one meter per second, while the Y-axis remains nearly stationary.

Discrepancies between the Kalman filter's estimation and the true state of the system are shown on the adjacent panel detailing errors. As a result of the Kalman filter, a zero-error would mean perfect alignment with the true states. In the red trajectory, we see the error caused by the Kalman filter, while in the green trajectory, we see the uncertainty around this error—namely, the three-sigma bounds derived from P's covariance matrix. These three-sigma constraints indicate effective filter operation when observed errors are within them.

```
import numpy as np
from kfsims.tracker2d import run_sim
from kfsims.kftracker2d import KalmanFilterModel

# Simulation Options
sim_options = {'time_step': 0.01,
               'end_time': 120,
               'measurement_rate': 1,
               'measurement_noise_std': 10,
               'motion_type': 'straight',
               'start_at_origin': False,
               'start_at_random_speed': True,
               'start_at_random_heading': True,
               'draw_plots': True,
               'draw_animation': True}

kf_options = {'accel_std':0.0,
              'mass_std':10,
              'init_on_measurement':True}

# Run the Simulation
run_sim(KalmanFilterModel, sim_options, kf_options)
```

Figure 22. Shows the code snippet of the final Kalman filter.

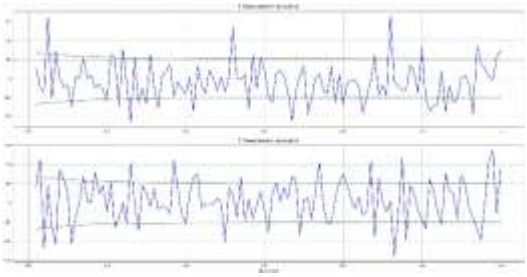

Figure 23. Shows the innovation of the Kalman filter.

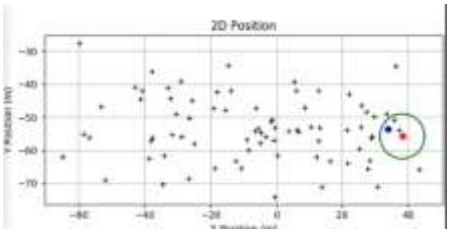

Figure 24. Shows the 2D trajectory visualization, the object's path is random at first, in Figure 24 (this is illustrated by the red and blue dot far apart).

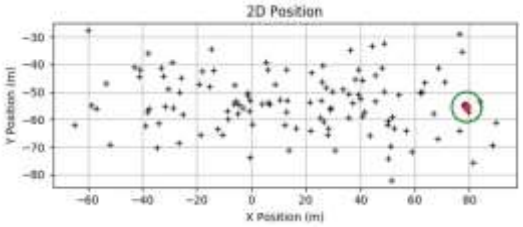

Figure 25. Shows the particle converging towards a more consistent trajectory as speed and heading parameters are randomized, as in Figure 25 (this is illustrated by the red and blue dot overlapping).

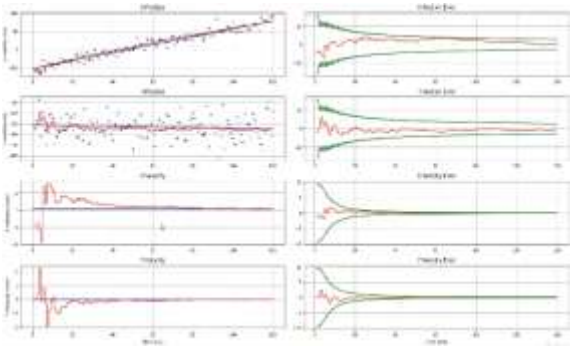

Figure 26.Illustrates the relationship between X-position versus Y-position as well as X-velocity versus Y-velocity.

IX. Critical analysis and Recommendations

Data estimation can be improved using the Kalman filter, based on probability density functions. However, numerical inaccuracies in computational systems, especially when representing floating-point numbers, cause issues when transferring theory to practice. Covariance matrix distortions can cause system failures due to such errors. A more robust Kalman filter is available with Joseph stabilization, but it is computationally intensive. Errors in modeling or incorrect assumptions about noise can also pose practical challenges.

Strategies for mitigating risk include:

1. For enhanced computational precision, 64-bit numbers are used.
2. Stability is achieved using square root Kalman filters.
3. Initiating the covariance matrix appropriately and ensuring its symmetry.
4. The fading memory filter technique is applied.
5. In order to counteract modelling errors, the Q matrix needs to be adjusted.

The variance of the filter captures the uncertainty associated with the estimate; it is important to portray it accurately. Unreliable estimates can be produced by filters with inconsistencies, affecting downstream applications. Unless the state transmission model is accurate, filtering can result in propagated errors or system failures. Reliability is measured by the consistency of the filter, which is essential for the reliability of a system. There should be future studies that assess the effects of non-linear systems on the filter and test the update step.

In the context of the Kalman Filter, the three-sigma position uncertainty can be understood as a quantitative measure of the state estimate's uncertainty. The significance of observing three-sigma position uncertainties that remain consistent can be multi-fold:

Three-sigma position uncertainties:

The three-sigma rule is based on common knowledge in statistical process control and Kalman filtering. This rule is also known as the empirical rule or 68-95-99.7 rule, which is a statistical principle that asserts that nearly all values lie within three standard deviations of the mean in a normal distribution. In the context of the Kalman Filter, the three-sigma position uncertainty can be understood as a quantitative measure of the state estimate's uncertainty. The significance of observing three-sigma position uncertainties that remain consistent can be multi-fold as observed in the results of this paper:

Reliability: The consistency of the three-sigma range, approximately ±15 meters as shown in Figure 5, indicates that the state estimates provided by the Kalman Filter are dependable. The actual position of the object, remaining within this range, demonstrates that the filter is statistically sound and that the model used is capturing the system's dynamics with a high degree of accuracy.

Confidence: By maintaining the three-sigma interval, as observed with the position uncertainty staying consistently within ±15 meters, the filter communicates the expected accuracy of its predictions. This allows users to assess the confidence level they can place on the system's outputs and informs the reliability of subsequent actions based on these predictions.

System Stability: Stability of the system is indicated by the three-sigma uncertainty bounds not expanding unpredictably, suggesting the system is stable and the filter parameters are well-tuned. The observed stability, as indicated by consistent position covariance predictions, is crucial for dynamic systems to prevent loss of tracking or incorrect predictions.

Noise Handling: The Kalman Filter's effectiveness in managing noise is demonstrated by maintaining a consistent three-sigma range, even when initial velocity uncertainty is introduced and set to $(7/3)^2$ as seen in the velocity covariance predictions in Figure 6. This shows the filter's capability to separate actual system changes from measurement noise and process disturbances.

Predictive Quality: For applications like navigation and tracking systems, where the Kalman Filter is utilized, the predictable three-sigma uncertainty range indicates that the predictions are of high quality. For instance, in the step involving acceleration uncertainty, where a standard deviation of 0.1 leads to a quadratic growth in the three-sigma bounds of velocity uncertainty, the users can anticipate the potential variance in position estimates, facilitating effective planning around these predictions.

Conclusion

The purpose of this research is to explore two-dimensional (2D) tracking of mobile entities in South Africa's expansive terrains, in order to highlight challenges and to demonstrate the utility of Bayesian data fusion. Especially for radar-based systems, the Kalman filter plays a crucial role in estimating the velocity and position of 2D objects. Based on probability theories and differential equations, the study demonstrates the cost-effectiveness and integration of GPS data with forecasts of weather conditions. As a result, it is also ideal for microprocessor computations and vehicle state estimations due to its simplicity and efficiency.

Although the filter effectively tracks 2D objects in a Python environment, addressing position uncertainty, it is not without flaws, including noise deviations and computational challenges. Despite this, 64-bit representations and Joseph's stabilizations enhance the robustness of the method. Studying nonlinear systems and refining the filter's update processes should be the focus of future research.

ACKNOWLEDGMENT

The author would like to thank Prof. Mbuyu Sumbwanyambe, and Dr. Tlotlollo Hlalele for their encouragement throughout this project. The author would also like to acknowledge the back-end files “kfsim”, which was contributed by Dr. Steven Dumble.

REFERENCES

[1] N. Senel, K. Kefferpütz, K. Doycheva, and G. Elger, "Multi-Sensor Data Fusion for Real-Time Multi-Object Tracking," Processes, vol. 11, no. 501, 2023, doi: 10.3390/pr11020501.

[2] B. Shahian Jahromi, T. Tulabandhula, and S. Cetin, "Real-Time Hybrid Multi-Sensor Fusion Framework for Perception in Autonomous Vehicles," Sensors, vol. 19, no. 4357, 2019, doi: 10.3390/s19204357.

[3] A. Khattak, N. Akbar, M. Aazam, T. Ali, A. Khan, S. Jeon, M. Hwang, and S. Lee, "Context Representation and Fusion: Advancements and Opportunities," Sensors, vol. 14, pp. 9628-9668, 2014, doi: 10.3390/s140609628.

[4] M. Wahbah, M. Chehadeh, M. Hamandi, L. Seneviratne and Y. Zweiri, "Real-Time Adaptive Dynamics Based State Estimation Scheme for Unmanned Aircrafts," IEEE Sensors Journal, vol. 22, no. 14, pp. 14397-14414, 15 July 2022, doi: 10.1109/JSEN.2022.3183187.

[5] S. J. Godsill, J. Vermaak, W. Ng, and J. Li, "Models and Algorithms for Tracking of Maneuvering Objects Using Variable Rate Particle Filters," Proceedings of the IEEE, vol. 95, pp. 925-952, 2007, doi: 10.1109/JPROC.2007.894708.

[6] U. B. Gohatre and V. P. Patil, "Estimation of velocity and distance measurement for projectile trajectory prediction of 2D image and 3D graph in real time system," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 2017, pp. 2543-2546, doi: 10.1109/ICECDS.2017.8389912.

[7] A. Sai Suneel, "Person or object tracking and velocity estimation in real time videos," Publications of Problems & Application in Engineering Research Paper, vol. 4, special issue 01, 2013.

[8] J. Au, D. Reid and A. Bill, "Challenges and Opportunities of Computer Vision Applications in Aircraft Landing Gear," 2022 IEEE Aerospace Conference (AERO), Big Sky, MT, USA, 2022, pp. 1-10, doi: 10.1109/AERO53065.2022.9843684.

[9] Z. Chen, Y. Cao, Y. Liu, H. Wang, T. Xie, and X. Liu, "A comprehensive study on challenges in deploying deep learning based software," in Proc. 28th ACM Joint Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE 2020), 2020, pp. 750–762, doi: 10.1145/3368089.3409759.

[10] C. Zhang and S. Czarnuch, "Perspective Independent Ground Plane Estimation by 2D and 3D Data Analysis," IEEE Access, vol. 8, pp. 82024-82034, 2020, doi: 10.1109/ACCESS.2020.2991346.

[11] J. Arróspide, L. Salgado, M. Nieto, and R. Mohedano, "Homography-based ground plane detection using a single on-board camera," IET Intelligent Transport Systems, vol. 4, no. 2, p. 149, 2010, doi: 10.1049/iet-its.2009.0054.

[12] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby and A. Mouzakitis, "A Survey on 3D Object Detection Methods for Autonomous Driving Applications," IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 10, pp. 3782-3795, Oct. 2019, doi: 10.1109/TITS.2019.2892405.

[13] C. Zhu et al., "Road Scene Layout Reconstruction based on CNN and its Application in Traffic Simulation," 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 2019, pp. 480-485, doi: 10.1109/IVS.2019.8813836.

[14] Z. Jin, T. Tillo, and F. Cheng, "Depth-map driven planar surfaces detection," in Proc. IEEE Vis. Commun. Image Process. Conf., Dec. 2014, pp. 514–517, doi: 10.1109/VCIP.2014.7051607.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM, vol. 24, no. 6, pp. 381–395, Jun. 1981, doi: 10.1145/358669.358692.

[16] I. Rabbi and S. Ullah, "A survey of augmented reality challenges and tracking," ACTA GRAPHICA, vol. 24, pp. 29-46, 2013.

[17] L. Ma, C. Kerl, J. Stückler and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM," 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 1285-1291, doi: 10.1109/ICRA.2016.7487260.

[18] L. K. Bui, J. Awange, and D. T. Vu, "Precipitation and soil moisture spatio-temporal variability and extremes over Vietnam (1981–2019): Understanding their links to rice yield," Sensors, vol. 22, no. 5, p. 1906, 2022, doi: 10.3390/s22051906.

[19] C. -H. Lin, S. -Y. Jiang, Yueh-Ju Pu and K. -T. Song, "Robust ground plane detection for obstacle avoidance of mobile robots using a monocular camera," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 3706-3711, doi: 10.1109/IROS.2010.5653055.

[20] M. Y. Yang and W. Forstner, "Plane detection in point cloud data," in ¨ Proceedings of the 2nd int conf on machine control guidance, Bonn, vol. 1, 2010, pp. 95–104.

[21] J. Schwan, A. R. Dhamija and T. E. Boult, "I-MOVE: Independent Moving Objects for Velocity Estimation," 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 2020, pp. 1090-1099, doi: 10.1109/WACV45572.2020.9093460.

[22] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 5524-5532, doi: 10.1109/ICCV.2017.589.

[23] D. Y. Kim and M. Jeon, "Data fusion of radar and image measurements for multi-object tracking via Kalman filtering," Information Sciences, vol. 278, pp. 641-652, 2014, doi: 10.1016/j.ins.2014.03.080.

[24] Z. Zhou, Y. Li, J. Liu, and G. Li, "Equality constrained robust measurement fusion for adaptive Kalman-filter-based heterogeneous multi-sensor navigation," IEEE Transactions on Aerospace and Electronic Systems, vol. 49, no. 4, pp. 2146-2157, 2013, doi: 10.1109/TAES.2013.6621813.

[25] Q. Li, R. Li, K. Ji and W. Dai, "Kalman Filter and Its Application," 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, China, 2015, pp. 74-77, doi: 10.1109/ICINIS.2015.35.

[26] M. A. Skoglund, G. Hendeby and D. Axehill, "Extended Kalman filter modifications based on an optimization view point," 2015 18th International Conference on Information Fusion (Fusion), Washington, DC, USA, 2015, pp. 1856-1861, doi: 10.1109/ICIF.2015.7266807.

[27] C. Qian, "Unscented Kalman Filter and Its Implementation in Digital Image Correlation," 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 2023, pp. 1056-1060, doi: 10.1109/ICPECA56706.2023.10075979.

[28] M. Roth et al., "Nonlinear Kalman Filters Explained: A Tutorial on Moment Computations and Sigma Point Methods," Journal of Advances in Information Fusion, vol. 11, pp. 47-70, 2016.