Some Studies on Object Tracking using Kalman Filter and Tuning of Process Noise Covariance Parameter for Optimal System Performance

By
SK BABUL AKHTAR
Registration No.: 160211 of 2021 - 2022
Exam Roll No.: M4ETC23003

Under the guidance of:
Prof. P. VENKATESWARAN

Thesis submitted in partial fulfillment of the requirements
for the award of the Degree of Master of Engineering in
Electronics and Telecommunication Engineering

Department of Electronics and Telecommunication Engineering
Jadavpur University
Kolkata 700032.

June 2023

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this Thesis contains a literature survey and original work by the undersigned candidate, as part of his Master of Engineering in Electronics and Telecommunication Engineering.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name of the Candidate: **Sk Babul Akhtar**

Exam Roll No.: M4ETC23003

Thesis Title: **Some Studies on Object Tracking using Kalman Filter and Tuning of Process Noise Covariance Parameter for Optimal System Performance**

Signature of the Candidate: _____

Date:

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

I hereby recommend that this Thesis prepared by **Sk Babul Akhtar** entitled **Some Studies on Object Tracking using Kalman Filter and Tuning of Process Noise Covariance Parameter for Optimal System Performance** under my supervision be accepted for partial fulfillment of the requirements for the award of the Degree of Master of Engineering in Electronics and Telecommunication Engineering.

……..

Dr. P. Venkateswaran
Professor
Dept. of Electronics and Tele-Communication Engineering
Jadavpur University,Kolkata – 700032

……

Dr. ManotoshBiswas
Professor and Head
Dept. of ETCE
Jadavpur University
Kolkata – 700032

……

Dean
Faculty of Engg. & Technology
Jadavpur University
Kolkata – 700032

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL*

The foregoing Thesis is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the Degree for which it has been submitted. It is understood that by this approval, the undersigned don't necessarily endorse or approve any statement made, opinion expressed or the conclusion drawn therein but approve the Thesis only for the purpose for which it is submitted.

Committee on
Final examination for
Evaluation of the Thesis of:
SK BABUL AKHTAR
Exam Roll No: M4ETC23003

……

……

……

Signature of Examiners

* Only in case the Thesis is approved

AKNOWLEDGEMENT

I would like to take this opportunity to express my heartfelt gratitude to my Thesis Supervisor, **Prof. P. Venkateswaran**, Department of Electronics & Tele-Communication Engineering, Jadavpur University. His unwavering confidence in my abilities and his invaluable suggestions and ideas at every stage of the Thesis was instrumental in its success.

I extend my special thanks to **Dr. Supriya Dhabal**, Assistant Professor, Department of Electronics & Communication Engineering, Netaji Subhash Engineering College, Kolkata. His constant guidance, tireless efforts, and timely suggestions have been a source of inspiration throughout my Thesis work.

I am sincerely grateful to **Prof. Manotosh Biswas**, Head, Department of Electronics & Tele-Communication Engineering JU, for his support and motivation throughout my research journey. I also extend my appreciation to all the teaching and non-teaching staff of the department for their invaluable support and assistance.

Lastly, but certainly not least, I would like to express my heartfelt thanks to my family, parents, and friends for their unwavering support and motivation throughout the entire Thesis process. Their encouragement and belief in me have been instrumental in overcoming challenges and reaching this significant milestone.

Computer Networking Lab (T-3-16A)
Dept. of ETCE, Jadavpur University
Kolkata - 700 032, West Bengal
June 2023

……..
SK BABUL AKHTAR

ABSTRACT

Object Detection and Tracking are important and challenging tasks in many computer vision applications such as surveillance, vehicle navigation, and autonomous robot navigation. Video surveillance in a dynamic environment, especially for humans and vehicles, is one of the current challenging research topics in computer vision. It is a key technology to fight against terrorism, crime, public safety and for efficient management of traffic.

The present Thesis work involves designing an efficient video surveillance system in complex environments. The Kalman Filter (KF) is a widely used algorithm for tracking objects in a noisy and uncertain environment. This Thesis explores the application of the Kalman filter in object tracking and presents a comprehensive review of the theory and implementation of the Kalman filter algorithm. The Thesis also investigates different techniques for modeling the motion and measurement of objects, and examines the impact of different model parameters on tracking performance. The Thesis presents experimental results on a variety of object tracking scenarios, including tracking a single object in one and two dimensions, multiple objects, and objects with complex motion patterns. The results demonstrate the effectiveness and versatility of the Kalman filter in tracking objects in challenging environments.

Finally, the Thesishighlights the importance of accurate Process Noise Covariance modeling and discusses the tuning of the Kalman filter using RMS indexwhich is verified using Sensitivity and Robustness metrics.Upon tuning of the Kalman filter using RMS index, the system performance was evaluated, showcasing a significant improvement in its overall effectiveness. In the context of a simulated one-dimensional (1D) tracking system, a comparison between an un-tuned Kalman filter and a properly tuned counterpart revealed notable enhancements. Specifically, the mean error experienced a substantial decrease from -8.51m (for an un-tuned Kalman Filter) to -0.20m (for a tuned KF), while the standard deviation reduced from 11.34m to 0.94m for the un-tuned and tuned systems, respectively. This remarkable transformation resulted in a 17-fold improvement in the system's performance for the 1D tuned tracking system. Overall, this Thesis provides a comprehensive overview of the Kalman filter algorithm and its application in object tracking, and serves as a useful reference for researchers and practitioners in the field of computer vision.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Preamble

Object Tracking plays a crucial role in various applications within the realm of computer vision. With the advancement of powerful computing systems, the widespread availability of affordable high-resolution video cameras, and the growing demand for automated video analysis, Object Tracking algorithms have garnered significant attention. The process of video analysis encompasses three fundamental stages: identification and detection of noteworthy moving objects, continuous tracking of these objects across successive frames, and analysis of the object trajectories to discern their behavioral patterns. As a result, Object Tracking finds relevance in a multitude of tasks, including but not limited to:

- Motion-based recognition [1], that is, human identification based on gait, automatic Object Detection, etc.;
- Automated surveillance, that is, monitoring a scene to detect suspicious activities or unlikely events;
- Video indexing, that is, automatic annotation and retrieval of the videos in multimedia databases;
- Human-computer interaction, that is, gesture recognition, eye gaze tracking for data input to computers, etc.;
- Traffic monitoring [2], that is, real-time gathering of traffic statistics to direct traffic flow.
- Vehicle navigation [2], which is, video-based path planning and obstacle avoidance capabilities.

In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene. In other words, a tracker assigns consistent labels to the tracked objects in different frames of a video. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object. Tracking objects can be complex due to [3]:

- Loss of information caused by projection of the 3D world on a 2D image,
- Noise in images,
- Complex object motion,
- Non-rigid or articulated nature of objects,
- Partial and full object occlusions,
- Complex object shapes,
- Scene illumination changes, and
- Realtime processing requirements.

One can simplify tracking by imposing constraints on the motion and/or appearance of objects. For example, almost all tracking algorithms assume that the object motion is smooth with no abrupt changes. One can further constrain the object motion to be of constant velocity or constant acceleration based on a priori information [4-6].The knowledge about the number and the size of objects, or the object appearance and shape, can also be used to simplify the problem.

1.2 Literature Survey

The authors of [7] proposed a method that dynamically adjusts measurement error covariance based on an error estimation technique. The results were compared with CamShift and Speeded-Up Robust Features (SURF) and itwas seen that the proposed method outperformed other methods in terms of tracking accuracy and Robustness. In the paper [8] the authors proposed a method that uses Kalman Filter and Scale-Invariant Feature Transform (SIFT) feature matching to track objects in a video stream. The results showed that in an event of occlusion, SIFT algorithm fails so that Kalman Filter is used to track the object in the next frames. The paper [9] compares the performance of Kalman Filter and particle filter in tracking moving objects in a video sequence. The results showed that although Particle filter is more robust and accurate, Kalman Filter works fairly in non-complex situations with more computational efficiency. The authors proposed a method in [10] that tuned the Kalman filter's noise covariance using Particle Swarm Optimization. The results indicated a better performance than traditional Kalman Filter. In the article [11] the authors proposed a method that uses Kalman Filter and Hungarian algorithm to track pedestrians. Fuzzy inference system is used to detect foreground and to deal with the illumination changes and shadows. The results showed that the proposed method works

well and tracks accurately even in presence of occlusions. The use of both Kalman Filter and Extended Kalman Filter [12] showed that the Extended Kalman Filter achieved higher accuracy than the Kalman Filter. The authors of [13] suggested a Kalman Filter based method for online video pedestrian. The results showed that the proposed method achieved high tracking accuracy and robustness in challenging scenarios with better performance and faster speed. In the paper [14] the authors proposed a multiple Object Tracking method based on deep matching that uses Kalman Filter. This paper introduces the optical flow information to Kalman Filter for building an accurate object motion model, which can contribute to predicting object position.

1.3 Thesis Motivation

Object Tracking is a fundamental problem in computer vision and has extensive applications in various fields, including video surveillance, robotics, autonomous vehicles, and augmented reality. Accurate and efficient Object Tracking is crucial for tasks such as target recognition, behavior analysis, and object interaction.

The Kalman Filter has long been recognized as a reliable algorithm for Object Tracking due to its ability to handle noisy measurements and uncertainty. However, there is a growing need to optimize the performance of the Kalman Filter and explore enhancements to address the challenges faced in real world tracking scenarios.

The motivation behind this Thesis is to delve into the domain of optimized Object Tracking using the Kalman Filter. By investigating and implementing various optimization techniques, the goal is to enhance the accuracy, robustness, and efficiency of Object Tracking algorithms based on the Kalman Filter. The Thesis also intends to investigate and explore the tuning aspects of the Kalman Filter in Object Tracking and by focusing on the parameter selection and optimization techniques, the goal is to enhance the performance of the Kalman Filter based tracking algorithms in various scenarios.

1.4 Thesis Outline

The Thesis has been systematically organized and efforts have been taken to cover the theoretical aspects in detail about the work and simulations performed in the laboratory.

Chapter 1 discusses the importance of Object Tracking and gives an introduction regarding the various scenarios, where it is needed and implemented.It briefly addresses the complexities associated with Object Tracking and includes a literature survey where different methods and algorithms from multiple references are compared with the Kalman Filter. Additionally, the Chapter presents the motivation behind the Thesis and outlines its structure.

Chapter 2 of the Thesis focuses on various Object Detection and Tracking algorithms. It discusses different ways of representing an object and explores the features that can be used to track an object. The Chapter also includes references to relevant literature on the subject.

Chapter 3 introduces the Proposed Object Tracking Algorithm, providing athorough discussion of the computations and working principle of the Kalman Filter. It provides a comprehensive discussion of the relevant parameters involved. The Chapter also explains the theory behind system tuning, utilizing the RMS index as well as the Sensitivity and Robustness metrics. Additionally, algorithms for calculating these parameters are presented.

Chapter 4 showcases the Simulation Results for various tracking scenarios and their corresponding calculations. It encompasses the Tracking of Objects in one dimension, two dimensions, and real videos with Multiple Object Tracking. Each simulation is extensively discussed, accompanied by plots and theoretical explanations. The Chapter also includes the tuning of the Kalman Filter using the RMS index and the verification of its performance using the Sensitivity and Robustness metrics for one-dimensional tracking. The results of these analyses are presented in this Chapter.

Chapter 5 concludes the Thesis with a positive discussion and scope for future work.

Chapter 2: Review of Object Detection and Tracking Algorithms

2.1 Representation of an Object

An object is simply nothing but an entity of interest. Objects can be represented by their shapes and appearances. For example, boats on the sea, fish in an aquarium, vehicles on a road, planes in the air, people walking on a road may be important to track in a specific domain. So there are various representations of object shapes [15- 19], which are commonly used for and are discussed as follows:

- **Points:** The object is represented by a point, which is the centroid (Fig. 1a) or a set of points (Fig. 1b). The point representation is suitable when it is given more concentration on the object which occupied small regions in an image.
- **Primitive Geometric Shape:** Geometric shape i.e. the object shape is represented by rectangle, ellipse (Fig. 1c, Fig. 1d). Primitive geometric shapes are more suitable for representing simple rigid objects as well as non-rigid objects.
- **Object Silhouette and Contour:** Contour is the boundary of an object (Fig.1g, Fig.1h). The region inside the contour is called the silhouette of the object (Fig. 1i). These representations are suitable for tracking complex non rigid shapes. ˆ
- **Articulated Shape Models:** Articulated objects are composed of body parts that are held together by joints (Fig. 1e). For example, the human body is an articulated object with torso, legs, hands, head, and feet connected by joints. The relationship between the parts is governed by kinematic motion models, for example, joint angle, etc. ˆ
- **Skeletal Model:** Object skeleton can be extracted by applying medial axis transform to the object silhouette. This model is commonly used as a shape representation for recognizing objects. Skeleton representation can be used to model both articulated and rigid objects (Fig.1f).

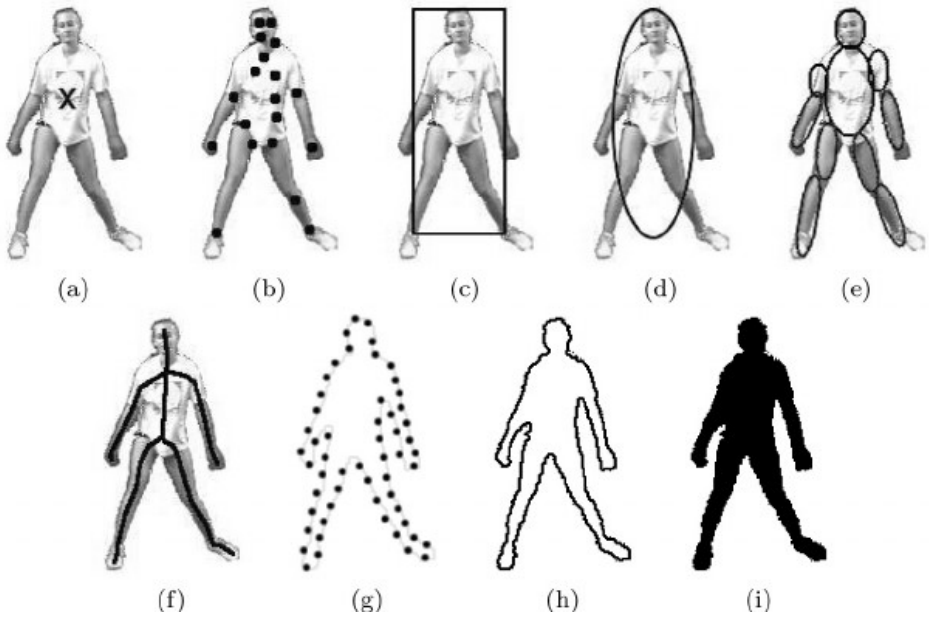

Fig. 1: Different types of Object Representation: (a) Centroid, (b) Multiple points, (c) Rectangular patch, (d) Elliptical patch, (e) Part-based multiple patches, (f) Object skeleton, (g) Complete object contour, (h) Control points on object contour, (i) Object silhouette

Similarly there are a various ways to represent the appearance feature of objects. It should be noted that the shape representation can be combined with appearance representations for tracking. Some common appearance representations in the case of object tracking are described as follows.

- **Probability Densities of Object Appearance:** The probability density estimates the object appearance can either be parameters, such as Gaussian and a mixture of Gaussians, such as Parzen windows and histograms. The probability densities of object appearance features (color, texture) can be computed from the image regions specified by the shape models (interior region of an ellipse or a contour) [3].
- **Templates:** Templates are formed using simple geometric shapes or silhouettes. It carries both spatial and appearance information. Templates, however, only encode the object appearance generated from a single view. Thus, they are only suitable for tracking objects whose poses do not vary considerably during the course of tracking [20].
- **Active Appearance Models:** These are generated by simultaneously modeling the object shape and appearance. Object shape is defined by a set of landmarks. Each landmark, an

appearance vector is stored in the form of color, texture or gradient magnitude. These models required a training phase where both the shapes & its associated appearance is learned from a set of samples [21].

- **Multi-view Appearance Models:** These models encode different views of an object. One approach to represent the different object views is to generate a subspace from the given view. Example of subspace approaches is Principal Component Analysis (PCA), Independent component Analysis (ICA). One limitation of multi-view appearance models is that the appearances in all views have required a lot of time [22].

In general, there is a strong relationship between object representation and tracking algorithms and object representations are chosen according to the application domain.

2.2 Object Detection

Object Detection is a fundamental problem in computer vision that involves identifying and localizing objects in an image or video. There are several different techniques that can be used for Object Detection, including Point Detectors, Background Subtraction, Segmentation, and Supervised Learning [3, 21].

- **Point Detectors:** Point detectors are one of the earliest methods used for Object Detection. They work by detecting salient points or interest points in the image, such as corners or edges, and then matching these points across different frames to detect the object. Point detectors are useful for detecting objects with distinctive features or patterns, such as faces or logos. Some popular point detectors include SIFT, SURF, and Oriented FAST and Rotated BRIEF (ORB) [23- 26].

- **Background Subtraction:** Background subtraction is a method for detecting objects by subtracting the static background from a video stream [27]. This technique assumes that the objects in the video are moving and that the background is static. By subtracting the background, the moving objects can be identified and localized. Background subtraction can be performed using various techniques such as frame differencing, Gaussian mixture models, and ViBe.

- **Segmentation:** Segmentation is a technique for detecting objects by segmenting the image into regions or objects of interest [27]. This method involves identifying and separating the objects from the background by analyzing their color, texture, shape, or other visual features. Some popular segmentation techniques include GrabCut, Watershed, and Convolutional Neural Networks [28].

- **Supervised Learning:** Supervised learning is a method for detecting objects by training a classifier on a large dataset of labeled images. This technique involves extracting features from the images and using them to train a machine learning algorithm to recognize the objects [28, 29]. The classifier can then be used to detect objects in new images or videos. Some popular supervised learning techniques for Object Detection include Support Vector Machines (SVM), Random Forests, and Convolutional Neural Networks (CNNs).

In conclusion, there are several different techniques that can be used for Object Detection, each with its own advantages and limitations. Point detectors are useful for detecting objects with distinctive features, while background subtraction is useful for detecting moving objects. Segmentation can be used to separate objects from the background, and supervised learning can be used to train a classifier to recognize objects. The choice of technique depends on the specific application and the characteristics of the objects being detected.

2.3 Object Tracking

The aim of an Object Tracker is to generate the trajectory of an object over time by locating its position in every frame of the video. Object Tracker may also provide the complete region in the image that is occupied by the object at every time instant. The tasks of detecting the object and establishing correspondence between the object instances across frames can either be performed separately or jointly. In the first case, possible object regions in every frame are obtained by means of an Object Detection algorithm, and then the tracker corresponds to the objects across frames. In the latter case, the object region and correspondence is jointly estimated by iteratively updating object location and region information obtained from previous frames. The model selected to represent object shape limits the type of motion or deformation it can undergo. For example, if an object is represented as a point, then only a translational model can be used. In the

case where a geometric shape representation like an ellipse is used for the object, parametric motion models like affine or projective transformations are appropriate. These representations can approximate the motion of rigid objects in the scene. For a non-rigid object, silhouette or contour is the most descriptive representation and both parametric and nonparametric models can be used to specify their motion.The various tracking techniques are shown in Fig. 2 as a chart and then discussed. Some of the notable works on the relevant tracking techniques are given in Table 1.

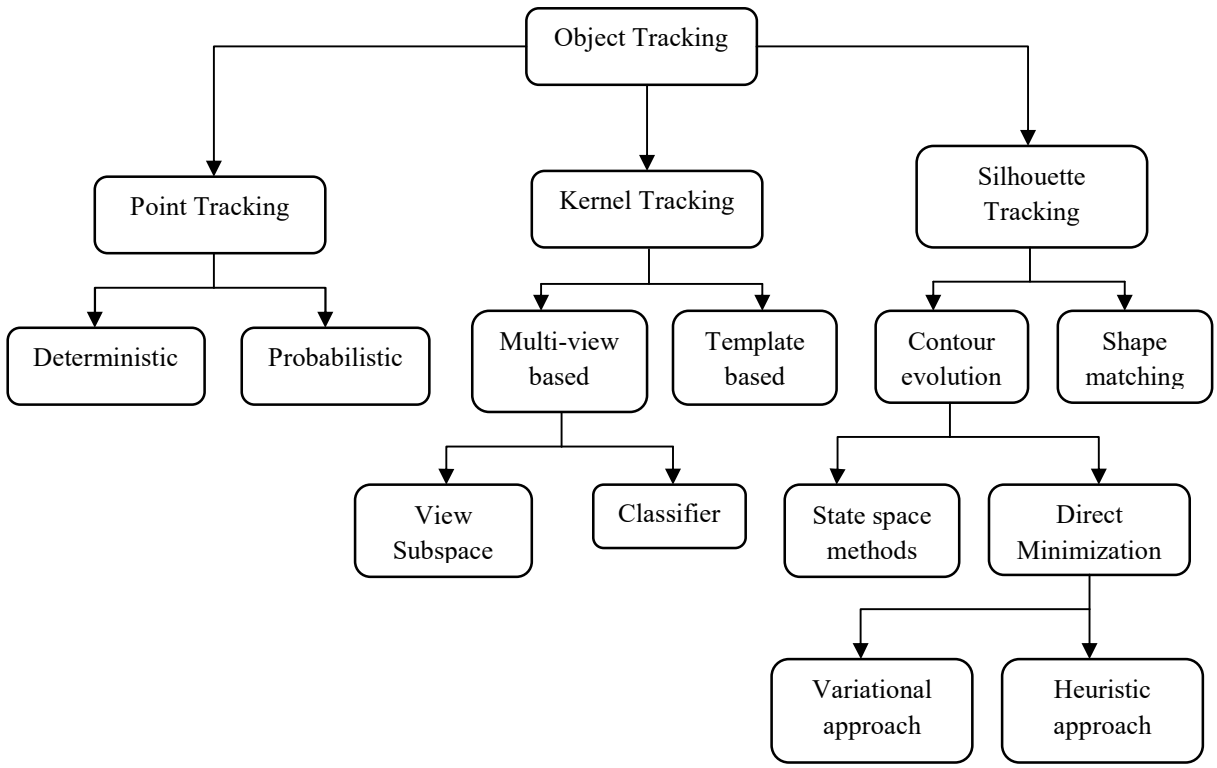

Fig. 2 Taxonomy of Tracking Methods

- **Point Tracking**: Point tracking, also known as feature tracking, is a popular method for tracking objects in computer vision. It works by identifying and tracking specific points in an image or video sequence over time. These points can be corners, edges, or other salient features that are easily recognizable and distinguishable in the images. Point tracking can be divided into two broad categories, i.e. Deterministic approach and Statistical approach [31- 33]. Objects detected in consecutive frames are represented by points, and the association of the points is based on the previous object state which can

include object position and motion. The Kalman Filter is a point tracking strategy which is categorized in statistical approach.

- **Kernel Tracking**: Kernel tracking is a method of Object Tracking in which the object is represented by a kernel or a patch of pixels [34, 35]. The kernel is usually centered on the object and its size and shape are chosen to be appropriate for the object being tracked. For example, the kernel can be a rectangular template or an elliptical shape with an associated histogram. Objects are tracked by computing the motion of the kernel in consecutive frames. This motion is usually in the form of a parametric transformation such as translation, rotation, and affine.Kernel tracking methods are divided into two sub-categoriesbased on the appearance representation used i.e. Template and Density-based Appearance Model and Multi-view appearance model [39- 41].

Table 1: Tracking Categories

Categories	**Methods**	**Representative Work**
Point Tracking	• Deterministic methods	CenterTrack[24] GOA tracker [37]
	• Statistical methods	Kalman filter [31] AMC- JPDAF [38]
Kernel Tracking	• Template and density based appearance models	Mean-shift [39] KLT [40]
	• Multi-view based appearance models	SVM Tracker [42]
Silhouette Tracking	• Contour evolution	State space models [43] Heuristic methods [44]
	• Matching shapes	Histogram [45]

- **Silhouette Tracking:** Silhouette tracking [36] is a computer vision technique used to track the movement of objects in a video sequence. It works by extracting the silhouette of the object from each frame of the video, and then analyzing the changes in the silhouette over time. The process involves three main steps: first, the silhouette of the

object is extracted from each frame using techniques such as background subtraction or segmentation. Second, the silhouette is represented in a suitable format for tracking, such as a binary image or contour-based representation. Finally, the changes in the silhouette are used to track the object's position and trajectory over time. Silhouette tracking is useful for tracking objects that have a distinctive shape and move against a relatively static background, but it can be sensitive to changes in illumination and suffer from errors due to background clutter or noise.Silhouette trackers can be divided into two categories i.e. Shape matching and Contour tracking.

2.4 Feature Selection for Tracking

Choosing the appropriate visual features is a crucial aspect of tracking. Ideally, a visual feature should possess the property of uniqueness, which enables easy differentiation of objects in the feature space. Object representation is closely linked to feature selection, and the choice of feature(s) depends on the method of object representation. For instance, histogram-based appearance representations commonly use color as a feature, whereas contour-based representations often utilize object edges as features. Typically, a combination of these features is used in several tracking algorithms. The following elaborates on some frequently used visual features [3].

- **Color:** The color of an object that we perceive is primarily determined by two physical factors: the spectral power distribution of the illuminant and the surface reflectance properties of the object. When it comes to image processing, the RGB color space is typically employed to represent color. However, the RGB space is not a perceptually uniform color space. In other words, the differences between the colors in the RGB space do not correspond to the color differences perceived by humans. Furthermore, the RGB dimensions exhibit high correlation. On the other hand, $L*u*v$ and $L*a*b$ are examples of perceptually uniform color spaces, while HSV is an approximately uniform color space. Nonetheless, these color spaces are susceptible to noise. To summarize, there is no definitive answer on which color space is most effective in tracking, and therefore a range of color spaces are used in practice.

- **Edges:** Strong variations in image intensities are often caused by object boundaries, which can be detected using edge detection. One notable advantage of using edges as a feature is that they are less susceptible to changes in illumination compared to color features. Therefore, algorithms that track object boundaries frequently utilize edges as the representative feature. The Canny Edge detector is the most widely adopted edge detection method due to its precision and simplicity [46].
- **Optical Flow:** Optical flow refers to a dense field of displacement vectors that specifies the translation of each pixel in a particular region. The calculation of optical flow utilizes the brightness constraint, which assumes that corresponding pixels in consecutive frames have constant brightness [14, 47]. Optical flow is often employed as a feature in motion-based tracking and segmentation applications. For a thorough evaluation of the performance of optical flow techniques, interested readers can refer to the survey by [48].
- **Texture:** Texture is a metric that gauges the variation in intensity of a surface, which can quantify attributes such as smoothness and regularity. Texture necessitates a processing step to generate the descriptors when compared to color. A variety of texture descriptors are available [3], including Gray-Level Co-occurrence Matrices (GLCM's) (a 2D histogram that demonstrates the co-occurrences of intensities in a specific direction and distance), Law's texture measures (twenty-five 2D filters generated from five 1D filters that correspond to level, edge, spot, wave, and ripple), wavelets (an orthogonal bank of filters), and steerable pyramids. Similar to edge features, texture features are less affected by changes in illumination when compared to color.

2.5 Summary

This Chapter provided a comprehensive review of Object Detection and Tracking algorithms. It delves into various methods utilized for Object Detection and Tracking and provides relevant literature for a deeper understanding of the topic. Additionally, it investigates different approaches to representing objects and emphasizes the importance of feature selection in achieving efficient Object Tracking. Overall, this Chapter serves as a valuable resource for gaining insights into the field of Object Detection and Tracking.

Chapter 3: Proposed Object Tracking Algorithm using Kalman Filter

3.1 Kalman Filter

An important part of a tracking system is the ability to predict where an object will be in the next frame. This is needed to aid the matching the tracks to detect objects and to predict the position during occlusion. There are four common approaches to predict the objects' positions:

- Block matching
- Kalman filters
- Motion models
- Particle filter

The Kalman Filter (KF), also known as the linear quadratic estimator, is a mathematical algorithm used for estimating unknown variables in the presence of noise and inaccuracies [9]. It operates on a series of noisy measurements and recursively computes the optimal estimate of the unknown state. This technique is also referred to as the linear state space estimator, as it employs linear equations to model both the state and the measurements. The Kalman Filter provides an efficient means to estimate the true state of a system by accounting for the uncertainties in the measurements and the underlying dynamics [31]. By continually updating its estimate based on new data, the filter can track the state of the system over time with greater accuracy and reliability.The Kalman Filter is a powerful tool used for estimating the state of a system in the presence of linear models and Gaussian noise. It leverages the minimum mean-squares estimation theory to minimize the mean square error and provide optimal estimates of the unknown state. The filter is highly regarded for its effectiveness in linear estimation theory, and it has a strong relationship with the theory of adaptive filters. By understanding the functionality of the Kalman Filter, we can extend our knowledge of adaptive designs and enhance traditional approaches. As we delve deeper into the study of adaptive filters, we can connect the Kalman

Filter with the theory of least squares estimation and demonstrate how it can inspire useful expansions.

The Kalman Filter is a widely used linear state space model that is relatively straightforward to implement when the system and measurement models are linear. However, when the unknown state variablesare recursively estimated over time, there is always some degree of uncertainty in the measurement values due to process noise and measurement noise. It is important to note that these noises must be Gaussian in nature for the Kalman filter to be effective in estimating the unknown state variables. Because of this uncertainty, we cannot model the system entirely deterministically. Therefore, the Kalman filter relies on linear equation systems that include white Gaussian noise as a standard model to estimate the unknown state variables.

Table 2: Kalman Filter Parameters

Terminology	Symbol	Dimension
State vector	x_k	$n \times 1$
True measurement vector	z_k	$m \times 1$
State transition matrix	φ_k	$n \times m$
Observation matrix	H	$m \times n$
Process white noise	ω_k	$n \times 1$
Process noise co-variance matrix	Q	$n \times n$
Measurement noise	v_k	$m \times 1$
Measurement noise co-variance matrix	R	$m \times n$
A priori error co-variance matrix	P_k^-	$n \times n$
A posteriori error co-variance matrix	P_k^+	$n \times n$
Kalman gain	K_k	$n \times m$

The Kalman Filter operates by combining the predicted system state with new measurements using a weighted average technique. The weights assigned to each measurement are based on their respective uncertainties, and the resulting estimate lies between the predicted and measured values, with reduced uncertainty for more accurate results. The filter is highly effective at tracking dynamic systems, particularly when the observations arise from a low-dimensional

linear state-space model. By incorporating the system model and control inputs, along with multiple series of measurements, the Kalman Filter can estimate unknown variables that vary over time with a high degree of accuracy. This is a significant advantage over relying on a single measurement, as the filter can continually update its estimate based on new data, leading to more reliable predictions. Overall, the Kalman Filter is a powerful tool that uses weighted averaging to provide highly accurate estimates of the state of a dynamic system, even in the presence of uncertainties and measurement noise [49- 51]. Table 2 and Fig. 3 provide distinct representations of the terminologies associated with each notation and the operational mechanism of the Kalman Filter, respectively.

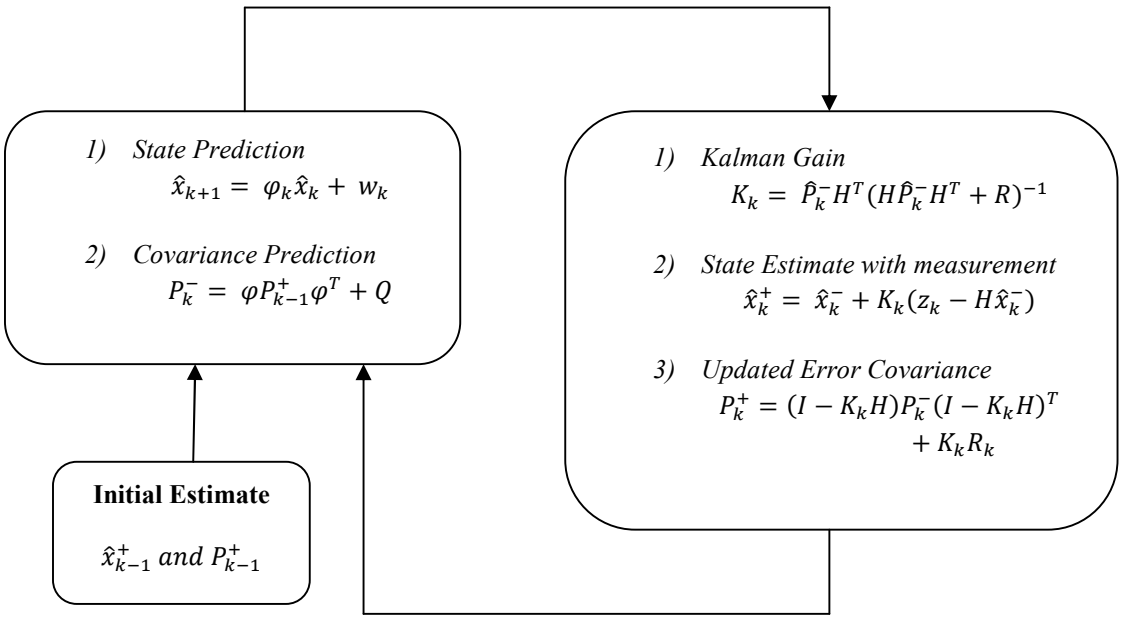

Fig. 3 Kalman Filter Algorithm

3.1.1 The Computational Origins of Kalman Filter

A discrete-time controlled process that is governed by the linear stochastic difference equation [49,52]:

$$x_{k+1} = \Phi_k x_k + \omega_k \tag{3.1}$$

where x_k is the state vector of the process at time k of order $(n \times 1)$, Φ_k is the state transition matrix of the process from the state at k to the state at $(k + 1)$ of order $(n \times m)$, and is

assumed stationary over time; ω_k is the associated white noise process with known co-variance of order $(n \times 1)$. Observations on this variable can be modeled in the form:

$$z_k = Hx_k + v_k \tag{3.2}$$

where z_k is the actual measurement of x at time k of order $(m \times 1)$,H of order $(m \times n)$ is the noiseless connection between the state vector and the measurement vector, and is assumed stationary over time, v_k of order $(m \times 1)$ is the associated measurement noise which is assumed to be a white noise process with known co-variance. It has zero cross-correlation with ω_k. The process noise co-variance matrix $Q = E[\omega_k \omega_k^T]$ and the measurement noise covariance matrix $R = E[v_k v_k^T]$ are assumed stationary over time. However, these two matrices might change with each time step or measurement.

A priori state estimate $\hat{x}_k^-$ at step $k$is defined, given knowledge of the process prior to step k and a posteriori state estimate $\hat{x}_k^+$at step k given measurement z_k. Thus, a priori error estimate is defined as:

$$e_k^- = x_k^- - \hat{x}_k^- \tag{3.3}$$

and a posteriori error estimate:

$$e_k = x_k - \hat{x}_k^+ \tag{3.4}$$

The a posteriori error co-variance matrix at time k is defined as:

$$P_k^+ = E[e_k^- e_k^{-T}] = E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T] \tag{3.5}$$

The a priori error co-variance matrix at time k is defined as:

$$P_k^- = E[e_k e_k^T] = E[(x_k^- - \hat{x}_k^-)(x_k^- - \hat{x}_k^-)^T] \tag{3.6}$$

To derive an a posteriori state estimate $\hat{x}_k$ as a linear combination of an a priori estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a measurement prediction $H\hat{x}_k^-$ results a parameter update equation of the form

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3.7}$$

The difference $(z_k - H\hat{x}_k^-)$ is called the measurement innovation, or the residual. The matrix K_k (known as Kalman Gain) is chosen to be the gain or blending factor that minimizes the a posteriori error covariance P_k^+ . This minimization can be accomplished by substituting the Equation (3.2) into the Equation (3.7) as below.

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(Hx_k + v_k - H\hat{x}_k^-) \tag{3.8}$$

Subtracting the Equation (3.8) from x_k at both side we get

$$\begin{aligned} x_k - \hat{x}_k^+ &= x_k - \hat{x}_k^- + K_k(Hx_k + v_k - H\hat{x}_k^-) \\ &= (x_k - \hat{x}_k^-) + (K_kHx_k - K_kH\hat{x}_k^-) - K_kv_k \\ &= (x_k - \hat{x}_k^-) + K_kH(x_k - \hat{x}_k^-) - K_kv_k \end{aligned} \tag{3.9}$$

Therefore,

$$x_k - \hat{x}_k^+ = (I - K_kH)(x_k - \hat{x}_k^-) - K_kv_k \tag{3.10}$$

Using the above relation in the Equation (3.5) we get

$$P_k^+ = E\{[(I - K_kH)(x_k - \hat{x}_k^-) - K_kv_k][(I - K_kH)(x_k - \hat{x}_k^-) - K_kv_k]^T\}$$

As v_k is uncorrelated with $(x_k - \hat{x}_k^-)$ we can write:

$$P_k^+ = (I - K_kH)E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T](I - K_kH)^T + K_kE[v_kv_k^T]K_k^T \tag{3.11}$$

Using the Equation (3.5) and $R = E[v_kv_k^T]$ we get a posteriori error co-variance matrix

$$P_k^+ = (I - K_kH)P_k^-(I - K_kH)^T + K_kRK_k^T \tag{3.12}$$

Now, let us consider a position vector

$$r = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = [x \quad y \quad z]^T \tag{3.13}$$

For this vector,

$$P_k^+ = E[(r_k - r_k^+)(r_k - r_k^+)^T] = E[[\Delta x_k \quad \Delta y_k \quad \Delta z_k]^T[\Delta x_k \quad \Delta y_k \quad \Delta z_k]] \quad (3.14)$$

where

$$\Delta x_k = x_k - \hat{x}_k^+ ; \ \Delta y_k = y_k - \hat{y}_k^+ ; \quad \Delta z_k = z_k - \hat{z}_k^+ ;$$

So,

$$\hat{P}_k^+ = E\begin{bmatrix} \Delta x_k^2 & \Delta x_k \Delta y_k & \Delta x_k \Delta z_k \\ \Delta x_k \Delta y_k & \Delta y_k^2 & \Delta y_k \Delta z_k \\ \Delta x_k \Delta z_k & \Delta y_k \Delta z_k & \Delta z_k^2 \end{bmatrix} \quad (3.15)$$

The trace of $\hat{P}_k^+$:

$$T\left[\hat{P}_k^+\right] = \Delta x_k^2 + \Delta y_k^2 + \Delta z_k^2 = \|(r_k - \hat{r}_k^+)\|^2 \quad (3.16)$$

Minimizing $\hat{P}_k^+$ or minimizing $T\left[\hat{P}_k^+\right]$ is equivalent to minimizing the distance between true r_k and estimated $\hat{r}_k^+$. From Equation (3.12) we get

$$P_k^+ = (\hat{P}_k^- - K_k H \hat{P}_k^-)(I - K_k H)^T + K_k R K_k^T \quad (3.17)$$

Using the property, the transpose of a product of two matrices is equivalent to the product of their transposes in reversed order: $(AB)^T = B^T A^T$

$$P_k^+ = \hat{P}_k^- - \hat{P}_k^- H^T K_k^T - K_k H \hat{P}_k^- + K_k H \hat{P}_k^- H^T K_k^T + K R K_k^T \quad (3.18)$$

Therefore,

$$P_k^+ = \hat{P}_k^- - \hat{P}_k^- H^T K_k^T - K_k H \hat{P}_k^- + K_k (H \hat{P}_k^- H^T + R) K_k^T \quad (3.19)$$

Then,

$$T\left[\hat{P}_k^+\right] = T\left[\hat{P}_k^-\right] - 2T[K_k H \hat{P}_k^-] + T[K_k H \hat{P}_k^- + K_k (H \hat{P}_k^- H^T + R) K_k^T] \quad (3.20)$$

At minima,

$$\frac{dT[P_k^+]}{dK_k} = -2(H \hat{P}_k^-) + 2K_k (H \hat{P}_k^- H^T + R) = 0 \quad (3.21)$$

$$\text{or, } (H \hat{P}_k^-)^T = K_k (H \hat{P}_k^- H^T + R) \quad (3.22)$$

Hence Kalman Gain,

$$K_k = \hat{P}_k^- H^T (H\hat{P}_k^- H^T + R)^{-1} \tag{3.23}$$

Alternatively,

$$K_k = \frac{\hat{P}_k^- H^T}{H\hat{P}_k^- H^T + R} \tag{3.24}$$

The measurement error co-variance R approaches zero, the gain $K_k$weights the residual more heavily i.e. the actual measurement z_k is trusted more and more, while the predicted measurement $H\hat{x}_k^-$ is trusted less and less. Specifically,

$$\lim_{R \to 0} K_k = H^{-1}$$

$$\therefore \hat{x}_k^+ \approx H^{-1} z_k \tag{3.25}$$

On the other hand, as the a priori estimate error co-variance $\hat{P}_k^-$ approaches zero, the gain K_k weights the residual less heavily i.e. the actual measurement z_k is trusted less and less, while the predicted measurement, $H\hat{x}_k^-$ is trusted more and more. Specifically,

$$\lim_{\hat{P}_k^- \to 0} K_k = 0$$

$$\therefore \hat{x}_k^+ \approx \hat{x}_k^- \tag{3.26}$$

Now from the Equation (3.1) and Kalman predict step,

$$\hat{e}_{k+1}^- = x_{k+1} - \hat{x}_{k+1}^- = (\varphi x_k + \omega_k) - \varphi \hat{x}_k^+$$

$$= \varphi(x_k - \hat{x}_k^+) + \omega_k \tag{3.27}$$

Therefore,

$$\hat{e}_{k+1}^- = \varphi \hat{e}_k^+ + \omega_k \tag{3.28}$$

Then,

$$\begin{aligned}\hat{P}_{k+1}^{-} &= E\left[\hat{e}_{k+1}^{-}\hat{e}_{k+1}^{-\,T}\right] \\ &= E[(\varphi\hat{e}_k^{+} + \omega_k)(\varphi\hat{e}_k^{+} + \omega_k)^T] \\ &= E[(\varphi\hat{e}_k^{+})(\varphi\hat{e}_k^{+})^T] + E[\omega_k\omega_k^T] \\ &= \varphi E[(\hat{e}_k^{+})(\hat{e}_k^{+})^T]\varphi^T + Q \end{aligned} \tag{3.29}$$

Hence a priori error co-variance matrix

$$\hat{P}_{k+1}^{-} = \varphi\hat{\mathrm{P}}_{\mathrm{k}}^{+}\varphi^{\mathrm{T}} + Q \tag{3.30}$$

3.1.2 Filter Parameters and Tuning

In the Kalman Filter, the Q matrix represents the covariance matrix of the process noise. It is used to model the uncertainty or variability in the system dynamics that is not accounted for by the model itself. Tuning the Q matrix is important because it directly influences the filter's estimation accuracy and responsiveness [54, 55]. The Q matrix helps the Kalman filter adapts to different system dynamics by capturing the inherent variability or uncertainty in the underlying process. By adjusting the values in the Q matrix, you can control how much the filter trusts the model predictions versus the actual measurements. Tuning it appropriately ensures that the filter can accurately track the system's behavior. A poorly tuned Q matrix can result in overconfidence or underestimation of the process noise, leading to inaccurate state estimates. On the other hand, an overly conservative Q matrix can make the filter sluggish in responding to changes in the system.

Unlike the measurement noise, which can be estimated or can be obtained from the sensor specifications, determining the exact values for the elements of the Q matrix often requires a deep understanding of the underlying system dynamics and the sources of process noise. In many cases, this knowledge may be limited or unavailable, making it difficult to accurately tune the Q matrix. Tuning the Q matrix is an iterative process that involves trial and error. Adjusting the Q matrix requires evaluating the filter's performance, comparing the estimated states with ground truth or reference data, and iteratively refining the Q matrix based on the observed behavior. This

iterative nature adds complexity and requires a systematic approach to converge on an optimal Q matrix configuration [53].

3.2 RMS Index

The process noise selection problems discussed must be solved to effectively design Kalman tracking filters. Thus, we must properly evaluate the performance of the filter. The effective steady-state performance index was derived and is termed root-mean-squared error index (an RMS index) [54, 55]. This section introduces the RMS index for POM and PVM systems and shows the analytical relationships between the RMS index and Q.

In tracking filtering, the following two functions are required:

- **Function 1**: Reduces random errors caused by measurement noises.
- **Function 2**: Tracks targets with complicated motions (e.g., accurate tracking of an accelerating target is required for the CV model).

The RMS index is proposed for the comprehensive evaluation of the performance of these two functions and is defined as:

$$\varepsilon_p = \sqrt{[(x_{tak} - \tilde{x}_k)^2]} \tag{3.31}$$

where $\tilde{x}_k$is the predicted target position, $E[\,]$ indicates the mean with respect to k, and x_{tak} is the true position of a constant acceleration target which is

$$x_{tak} = a_c(kT)^2/2 \tag{3.32}$$

where a_c is constant acceleration of the target. In the Kalman filter tracker using the CV model, it is assumed that the target velocity is constant during the sampling interval. Thus, for the constant acceleration target, a steady-state bias error occurs because of the difference between the target motion and the assumed dynamic model. Moreover, $\tilde{x}_k$ includes random errors due to measurement noise. Thus, the RMS index ε_p expresses both bias errors and random errors. With the steady-state bias error due to the model/motion difference of e_{ac} and the steady-state standard deviation of the random errors in $\tilde{x}_k$ of σ_p, ε_p is expressed as:

$$\varepsilon_p = \sqrt{{e_{ac}}^2 + {\sigma_p}^2} \tag{3.33}$$

The term,σ_p expresses the performance corresponding to Function 1 and e_{ac} expresses the performance corresponding to Function 2. The smaller these errors are, the better is the tracking filter. Thus, the minimum ε_p achieves the best tracking filter in a steady state.

3.2.1 RMS Index of POM System

One important advantage of the RMS index is that it can be expressed in closed form. The closed form of ε_p for the Position Only Measurement (POM) system was derived in [54]**.** This subsection introduces the RMS index and its relationship to the design parameter Q in the POM system.In the conventional tracking systems, the most commonly used random acceleration (RA) process noise is often selected because it has a better performance. Its Q is

$$Q_{ra} = \begin{bmatrix} T^4/4 & T^3/2 \\ T^3/2 & T^2 \end{bmatrix} {\sigma_q}^2 \tag{3.34}$$

The arbitrary Q is defined as

$$Q_{gen} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \tag{3.35}$$

where $a > 0,\ b > 0, and\ c > 0,$ and the dimensions of $a, b, and\ c$ are $[m^2\ ], [m^2/s],$ $and\ [m^2/s^2\ ]$, respectively.

For example, substituting $(a, b, c\ ) = ({\sigma_q}^2\, T^4/4\, , {\sigma_q}^2\, T^3/2\, , {\sigma_q}^2 T^2)$ into Equation (3.35) gives the Q_{ra} of Equation (3.34) and $b = 0$ leads to the diagonal Q. The analytical relationship between Q_{gen} and ε_p is expressed by the following closed form:

$${\varepsilon_p}^2 = \frac{{a_c}^2 dt^4}{\beta^2} + \frac{2\alpha^2 + 2\beta + \alpha\beta}{\alpha(4-2\alpha-\beta)} B_x \tag{3.36}$$

where α and β are components of the steady-state Kalman gain $K_\infty = (\alpha, \beta/T)^T$ calculated from (a, b, c) using the following equations:

$$\beta = \frac{C + \sqrt{C(16+4A-4B+\ )}}{4} - \sqrt{\frac{C^2(16+4A-4B+\ )}{8\sqrt{C(16+4A-4B+\ )}} + \frac{C(2A-2B+C)}{8}} \tag{3.37}$$

And,

$$\alpha = 1 - \frac{\beta^2}{C} \tag{3.38}$$

where

$$A = {}^{a}/_{B_x}, B = {}^{bdt}/_{B_x}, C = {}^{cdt^2}/_{B_x}$$

The derivation process of these equations is shown in [54]**.** As shown in Equations (3.36) to (3.38), the optimal (a, b, c) is designed minimizing ε_p.

The evaluating function to determine optimal Q is $\varepsilon_p$normalized by B_x, which is defined as:

$$\mu_p = {\varepsilon_p}^2/B_x = \frac{{a_D}^2}{\beta^2} + \frac{2\alpha^2 + 2\beta + \alpha\beta}{\alpha(4 - 2\alpha - \beta)} \tag{3.39}$$

where,

$${a_D}^2 = {a_c}^2 dt^4/B_x \tag{3.40}$$

is the preset parameter for the proposed strategy. Substituting Equation (3.37) and Equation (3.38) into Equation (3.39) we obtain $\mu_p(a, b, c, a_D)$. Using this, the optimal (a, b, c) for the POM system [59] is determined by solving

$$arg \min_{a,b,c} \ \mu_p(a, b, c, a_D) \tag{3.41}$$

$$sub.to. a > 0, b > 0, c > 0, and\ a_D = const.$$

3.2.2 Algorithm for Tuning Q

The proposed method follows the following algorithm to determine Q:

Step 1: First define the parameters that will be constant for the entire operation: dt, $\mathrm{a_c}$, σ_q, $\mathrm{B_x}$.

a. Using these values the ${\mathrm{a_D}}^2$ is calculated
b. The initial process noise covariance matrix $\mathrm{Q_{nom}}$ is populated using the nominal $\mathrm{a, b, c}$ values.

Step 2: For $P = -m$ to $+n$, where m and n are suitable integers, which defines the range through which Q will be varied:

a. Q is varied as$Q = Q_{nom} \times Q_{fac}$, where $Q_{fac} = 10^{P}$
b. Calculate μ_p for each iteration.
c. Retrieve the Q_{fac} for which the μ_p is minimum.

Step 3: The Kalman filter can then use the same Q value which yielded the lowest μ_p value to carry forward with the tracking process.

This same algorithm can be used to determine the process noise covariance,Q to track objects in two dimensions and three dimensions as well by using Kalman filter. In order to do video tracking an extra layer of Object Detection can be used, which is used to gather measurement data. Several Object Detection techniques are discussed in Section 2.2.

3.3 Sensitivity and Robustness Metrics

The Kalman filter is a widely used technique for tracking system state estimation. It takes into account the impact of un-modeled dynamics and measurement noise using covariance matrices. However, if the designer uses incorrect a priori statistics of the noise, the filter can lead to significant estimation errors or even diverge. The issue is that the statistics required for filter design are either unavailable or of questionable accuracy in practical applications. Currently, the process noise covariance matrix Q and the measurement noise covariance matrix R must be provided by the designer through ad hoc procedures, which poses a significant challenge [60]. To overcome the problem, we can look at the innovation and its covariance, which provides a deterministic basis for predicting filter tuning parameters. However, since the innovations q_k are random variables obtained in real-time, the innovation error covariance is used for predictions. Therefore, the innovation covariance, Kalman gain, and other filter covariances are algebraically manipulated to derive the performance metrics for tuning the Kalman Filter.

Sensitivity and Robustness are performance metrics used in various fields to evaluate the behavior and reliability of a system or process [53]. Sensitivity refers to the system's ability to detect small changes in the input and produce a proportional change in the output. In other words, Sensitivity measures the responsiveness of the system to changes in the input. A highly sensitive system will produce a large change in the output in response to a small change in the

input. Sensitivity is an important metric in many applications, such as medical tests, where it is crucial to detect even small changes in a patient's condition.Robustness, on the other hand, measures the ability of a system to maintain its performance even in the face of uncertainty or unexpected variations in the input. A robust system is one that can handle disturbances or uncertainties without significantly affecting its output. Robustness is a critical metric in many real-world applications where the input is subject to noise or variation, such as control systems, manufacturing, and finance.

Both Sensitivity and Robustness are important performance metrics, and the choice of which metric to use depends on the specific application and requirements of the system or process being evaluated. In some cases, a system may need to be highly sensitive to changes in the input, while in other cases, robustness may be more critical to ensure reliable operation in the face of uncertainties and disturbances.

3.3.1 Defining the Metrics

Let two terms A_k and B_k be defined as:

$$A_k = H_k \varphi_{k-1} P_{k-1}^{+} \varphi_{k-1}^{T} H_k^{T} \tag{3.42}$$

$$B_k = H_k Q_{k-1} H_k^{T} \tag{3.43}$$

Then, pre- and post-multiplying Equation (3.30) by $H_k$and H_k^Trespectively, yields

$$H_k P_k^{-} H_k^{T} = H_k \varphi_{k-1} P_{k-1}^{+} \varphi_{k-1}^{T} + Q_{k-1}) H_k^{T} = (A_k + B_k) \tag{3.44}$$

Thus, the innovation covariance $S_k = H_k P_k^{-} H_k^{T} + R_k$ can be expressed as

$$S_k = (A_k + B_k + R_k) \tag{3.45}$$

Thereafter, pre-multiplyingK_k in Equation (3.23) by H_k and using Equation (3.44) and Equation (3.45) yields

$$H_k K_k = H_k P_k^{-} H_k^{T} S_k^{-1} = (A_k + B_k)(A_k + B_k + R_k)^{-1} \tag{3.46}$$

Pre- and post-multiplying the a posteriori state estimation error covariance P_k^{+} , as stated in Equation (3.12), by H_k and H_k^T , respectively, yields

$$H_k P_k^+ H_k^T = (I - H_k K_k) H_k P_k^- H_k^T (I - H_k K_k)^T + H_k K_k R_k K_k^T H_k^T$$

$$= A_k + B_k - (A_k + B_k)(A_k + B_k + R_k)^{-1}(A_k + B_k) \quad (3.47)$$

Thereafter, the resulting first term on the right is shifted to the left and then both sides of the result are pre-multiplied with $(A_k + B_k)^{-1}$ to obtain

$$(A_k + B_k)^{-1}[H_k(P_k^+ - \varphi_{k-1} P_{k-1}^+ \varphi_{k-1}^T) H_k^T]$$

$$= (A_k + B_k)^{-1} B_k + (A_k + B_k + R_k)^{-1} R_k - I_m \quad (3.48)$$

where I_m is an identity matrix of size m, which is the number of measurements. Let $tr(A)$denote the trace of the matrix A. Taking the trace of both sides of Equation (3.48) and rearranging the terms, the two performance metrics $J_{1k}$and J_{2k} are defined as:

$$J_{1k} = trace[(A_k + B_k + R_k)^{-1} R_k] \quad (3.49)$$

$$J_{2k} = trace[(A_k + B_k)^{-1} B_k] \quad (3.50)$$

and

$$J_{1k} + J_{2k} = m - tr(N_k) \quad (3.51)$$

where

$$N_k = (A_k + B_k)^{-1}[H_k(\varphi_{k-1} P_{k-1}^+ \varphi_{k-1}^T - P_k^+) H_k^T] \quad (3.52)$$

It is observed from Equation (3.51) that the value of $(J_{1k} + J_{2k})$ at any instant k deviates from the number of measurements m due to the contribution of the term $tr(N_k)$ at that particular instant of time k. For the evaluation of the overall filter performance, let the performance indices or metrics J_1 andJ_2 be defined in terms of the total horizon N as [53]:

$$J_1 = \frac{1}{N} \sum_{k=1}^{N} J_{1k} \quad (3.53)$$

$$J_2 = \frac{1}{N} \sum_{k=1}^{N} J_{2k} \quad (3.54)$$

3.3.2 Interpretation of the Metrics

The innovation covariance consists of three components: A_k, B_k, and R_k. The term $A_k$represents the a priori state estimation error covariance in the measured output, while B_k signifies the process noise covariance in the measured output. The sum $(A_k + B_k)$ forms the state-dependent component of the innovation covariance S_k. Thus, the measurement noise covariance R_k and the projection of the process noise covariance Q_{k-1} onto the measurement, i.e., B_k, are the two causative factors of the S_k. The third component, A_k is considered the dependent factor, representing the corresponding projection of the a priori state estimation covariance $\varphi_{k-1}P_{k-1}^{+}\varphi_{k-1}^{T}$ onto the measurement [53].

The performance metric J_2 measures the effect of the process noise covariance Q_{k-1} on the sum $(A_k + B_k)$. Thus, J_2 is primarily driven by the causative factor B_k and is known as the Robustness metric. On the other hand, the performance metric J_1shows the effect of the measurement noise covariance R_k on the total sum $(A_k + B_k + R_k)$. Any change in J_1 can be primarily attributed to R_k, and J_1 is termed the Sensitivity metric. If there is a mismatch between the assumed process noise covariance Q_k and the covariance of the true process noise, J_2 is affected, and it indicates modeling errors and/or noise. Therefore, J_2 is an important metric to assess the robustness of the system. In contrast, if there is a mismatch between the assumed measurement noise covariance R_k and the actual measurement noise covariance, it directly affects the a posteriori state estimate, and J_1 becomes a crucial metric to evaluate the sensitivity of the system.

3.3.3 Algorithm for calculating J1 and J2

Step 1: For each sampling interval while running the Kalman filter, calculate $J_{1k}$and J_{2k}.
Step 2: At the end of the tracking process, average out $J_{1k}$and J_{2k} to obtain J_1 and J_2 respectively.
Step 3: Repeat step 1 and 2 for different values of Q, varying Q in the relation: $Q = Q_{nom} \times Q_{fac}$, where $Q_{fac} = 10^{P}$
Step 4: Plot J_1 and J_2vs. P, where P is the power to 10 in determining Q_{fac}.

3.4 Summary

In this Chapter, the theory of Kalman Filter is presented, discussing all the relevant parameters and its derivation. It specifically focused on the significance of Process Noise Covariance (Q) and discussed the tuning of the filter while incorporating the underlying theory. Additionally, this Chapter included the calculation of the RMS index and the Sensitivity and Robustness metrics. Section 3.2 presented the algorithm for calculating the RMS index which is used to determine the appropriate Process Noise Covariance. The interpretation of the Sensitivity and Robustness metrics is discussed, highlighting not only their importance in the tuning of the Process Noise Covariance matrix but also in verifying the system performance. Lastly, Section 3.7 presented the algorithm for calculating these metrics.

Chapter 4: Comparison and Performance Evaluation of Tracking in 1D and 2D using Kalman Filter: Simulation Results

4.1 Introduction

This chapter presents the Simulation Results for different tracking scenarios and their corresponding calculations. The chapter covers the tracking of objects using Kalman Filter in one dimension, two dimensions, and for real videos with Multiple Object Tracking. Each simulation is thoroughly explained, with clear plots and theoretical explanations provided. Additionally, the chapter discusses the calibration of Process Noise Covariance parameter of the Kalman filter using the RMS index and verifies the system performance using the Sensitivity and Robustness metrics for one-dimensional tracking. The chapter concludes by presenting the outcomes of these analyses.

4.2 Tracking in One Dimension using Kalman Filter

The very first tenet that is considered in the usage of a Kalman filter in a system is that there is error in both the prediction and measurement of a system variable that we are concerned with. The Kalman filter will be dealt with, in the context of tracking the position of a certain object. A one dimensional (1D) Kalman filter to track an object moving along the x-axis is implemented in order to gain an understanding. The flow chart for the same implementation is given in Fig. 4.

The Kalman filter model assumes that the state of a system at a time t evolved from the previous state at time $t-1$ according to the equation.

$$X_t = \varphi_t X_{t-1} + Bu_t + \omega_t \tag{4.1}$$

where

- X_t : The state vector and is given by: $[x_t \quad v_t]^T$where x_t and v_t are postion and velocity elements in one dimension.
- φ_t: The state transition matrix defines how the current state depends on the previous and is given by $\begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$
- u_t : The vector holding the control inputs.
- B : The control input matrix that maps the control input to the corresponding State Vector representation.
- ω_t : Process noise with a covariance Q.

For system measurements,

$$y_t = H_t X_t + v_t \tag{4.2}$$

where

- y_t : Vector of measurements
- H_t: Matrix that mapthe state vector to measurement space and is equal to $\begin{bmatrix} 1 & 0 \end{bmatrix}$.
- v_t : Measurement Noise with a covariance R

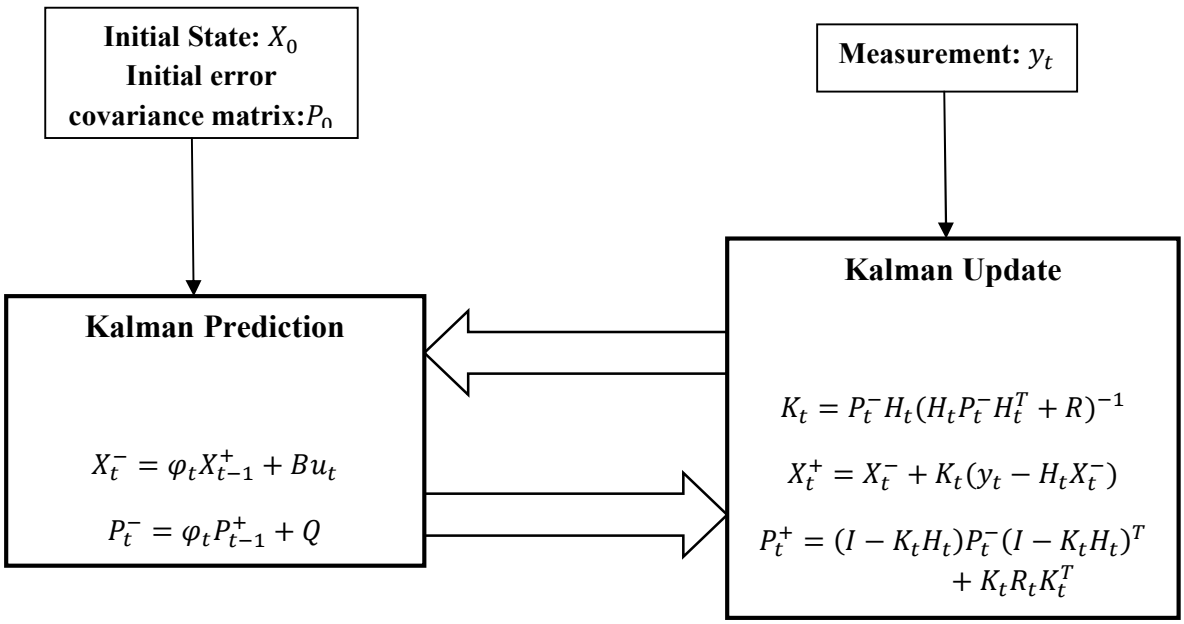

Fig. 4 Kalman Filter for 1D Tracking

For an object whose position on the x-axis varies, the equation for updating $X_t$can be written as**:**

$$x_t = x_{t-1} + \dot{x}_{t-1} + \frac{1}{2}\ddot{x}_t dt^2 \tag{4.3}$$

and $\dot{x}_t$ is given as:

$$\dot{x}_t = \dot{x}_{t-1} + \ddot{x}_{t-1} dt \tag{4.4}$$

and $\ddot{x}_{t-1}$ is a constant in this case.

Thus the state vector, X_t can be written as:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}dt^2 \\ dt \end{bmatrix} \ddot{x}_{t-1} \tag{4.5}$$

So comparing this with the general equation allows us to conclude:

$$\varphi_t = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{4.6}$$

$$B = \begin{bmatrix} \frac{1}{2}dt^2 \\ dt \end{bmatrix} \tag{4.7}$$

Here acceleration,$\ddot{x}_{t-1}$can be thought of as the control input and B is the matrix that maps it to position and velocity.In the measurement module, only the position is measured as:

$$y_t = H_t x_t + v_t = [1 \quad 0] \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} + v_t \tag{4.8}$$

The Q in the system is taken as standard as given by Equation (3.34) and R is given by $\sigma_m{}^2$.

A complex non-linear path is simulated using a mathematical function and the Kalman filter based Object Tracker is run to track the simulated path. The filter parameters discussed are provided as inputs to the filter. Fig. 5 and Fig. 6 show the True path, Measurements, and KF outputs, whereas Fig. 7 shows the deviation of estimate path from true path. It can be seen that the Kalman filter based on the provided input parameters has been able to track the simulated path with a good amount of precision and accuracy. It is also worth mentioning that in the first few iterations, the Kalman filter estimate has a large deviation from the True path, as the provided initial state estimate, X_0 was not close to the true position. However, as the system is

stable enough with a well-established measurement model and state model, the filter output quickly converges into the true state of the system.

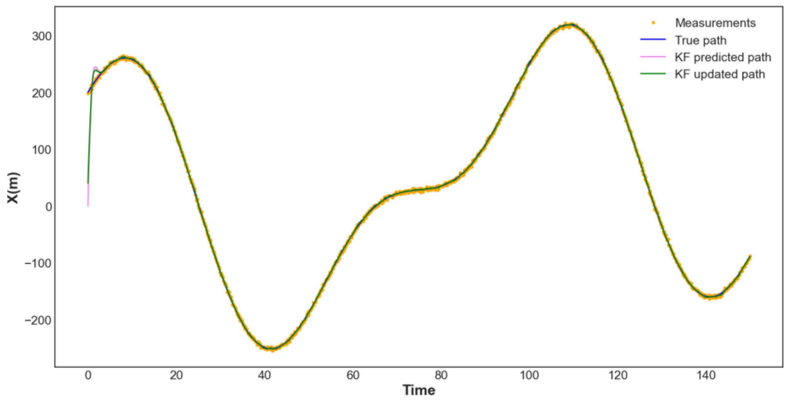

Fig. 5 True Path, Measurements and KF Estimate for full time span

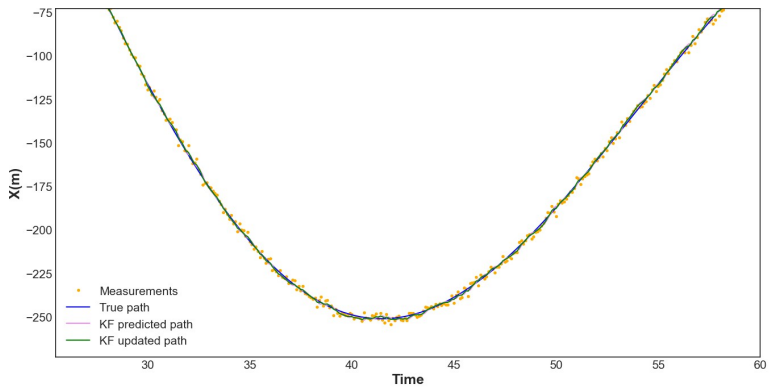

Fig. 6 True Path, Measurements and KF Estimate zoomed in

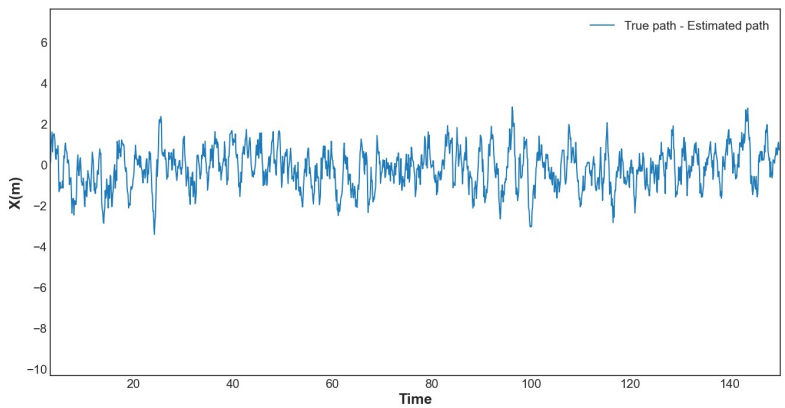

Fig. 7 Deviation of Estimated Path from True Path

4.2.1 Simulation Tool Used

All the prior and subsequent simulations are performed using Python (version 3.9.0). The Integrated Development Environment (IDE) used is Visual Studio Code. The plots presented in the results are generated from matplotlib module of python. In order to achieve Multiple Object Tracking for real video as discussed in Section 4.4 and 4.5, OpenCV module is used for various detection related operations.

4.3 Tracking in Two Dimensions using Kalman Filter

The basic equations should remain the same as the 1D case with additional considerations for the other variable. Considering x_t and y_t to be the concerned variables denoting the position, the state vector should look like:

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} \tag{4.9}$$

And by extension the can be related to the previous state vector as follows:

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}dt^2 & 0 \\ 0 & \frac{1}{2}dt^2 \\ dt & 0 \\ 0 & dt \end{bmatrix} \begin{bmatrix} \ddot{x}_{t-1} \\ \ddot{y}_{t-1} \end{bmatrix} \tag{4.10}$$

which gives

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \; and \; B = \begin{bmatrix} \frac{1}{2}dt^2 & 0 \\ 0 & \frac{1}{2}dt^2 \\ dt & 0 \\ 0 & dt \end{bmatrix}$$

Since we're considering the case where velocity isn't being measured the H matrix will look like:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.11}$$

Also process noise covariance Q will look like:

$$Q = \begin{bmatrix} \frac{dt^4}{4} & 0 & \frac{dt^3}{2} & 0 \\ 0 & \frac{dt^4}{4} & 0 & \frac{dt^3}{2} \\ \frac{dt^3}{2} & 0 & dt^2 & 0 \\ 0 & \frac{dt^3}{2} & 0 & dt^2 \end{bmatrix} \sigma_{\ddot{x}}{}^2 \tag{4.12}$$

The measurement noise R extends itself to two dimensions (2D) as follows:

$$R = \begin{bmatrix} \sigma_x{}^2 & 0 \\ 0 & \sigma_y{}^2 \end{bmatrix} \tag{4.13}$$

where$\sigma_x{}^2$ and $\sigma_y{}^2$ are the variances for measurements in x and y coodinates.

4.3.1 Detection Pipeline

We use basic blob detection to find the circles in the video on which we will test the 2D filter [61, 62]. The exact steps involved in the preprocessing can be understood from the Fig. 8.

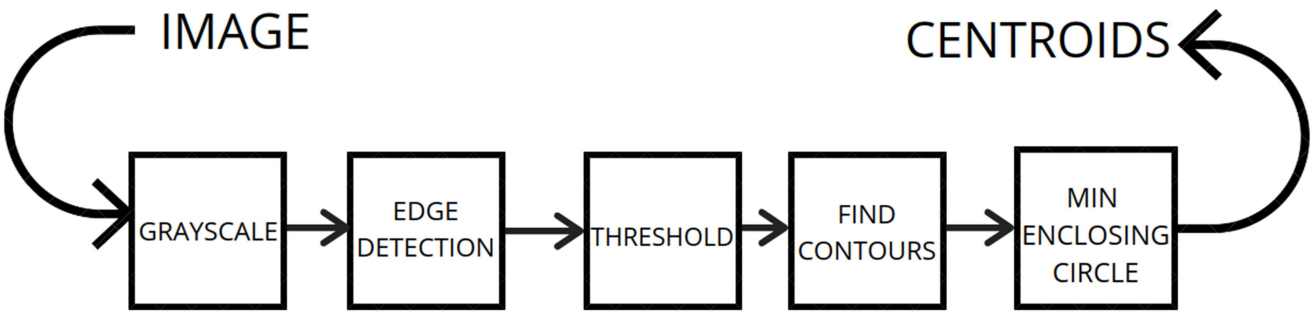

Fig. 8 Blob Detection Flow Chart

Here's a step-by-step explanation of each stage:

- **Grayscaling**: The first step is to convert the image from color to grayscale. This simplifies the image by removing color information and reduces computational complexity.

- **Edge Detection**: Edge detection algorithms, such as the Canny edge detector or the Sobel operator, are applied to the grayscale image. These algorithms highlight areas of significant intensity variation, which often correspond to edges or boundaries of objects.
- **Thresholding**: After edge detection, a thresholding operation is performed to convert the image into a binary representation. This involves setting a threshold value, where pixel intensities above the threshold are considered part of the object, and pixel intensities below the threshold are considered background.
- **Finding Contours**: Once the image is binary, contour detection algorithms, such as the OpenCV function findContours(), are applied. Contours are curves that represent the boundaries of objects in an image. The function extracts the contours from the binary image.
- **Blob Detection**: Finally, detected contours are used to identify blobs or objects. Various criteria can be used, such as contour area, aspect ratio, or compactness, to distinguish between different objects or filter out unwanted contours.

Fig. 9 shows the graphical diagram of a frame illustrating measured position and estimated position. Fig. 10 shows the plot of estimated and measured coordinates of the object moving in two dimensions and Fig. 11 shows the absolute deviation of the object from the true coordinates.

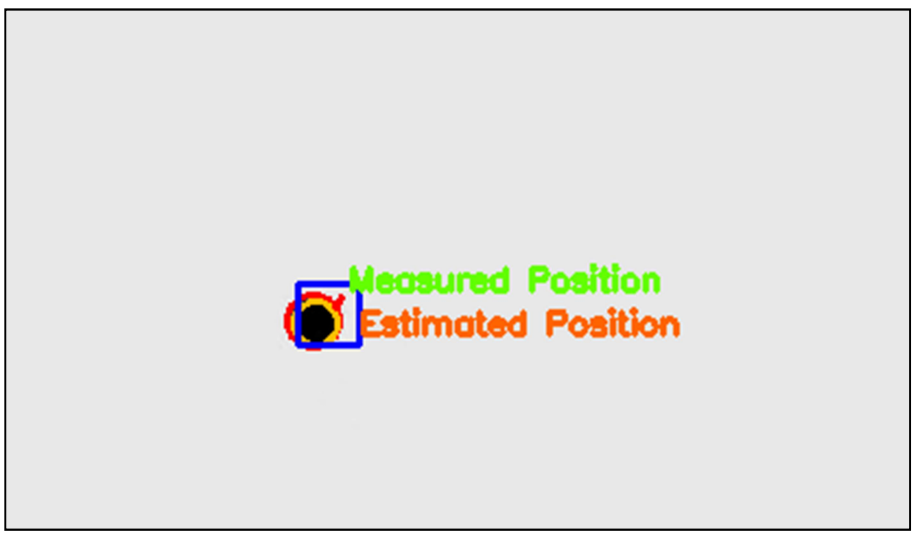

Fig. 9 Graphical Demo of 2D Object Tracking

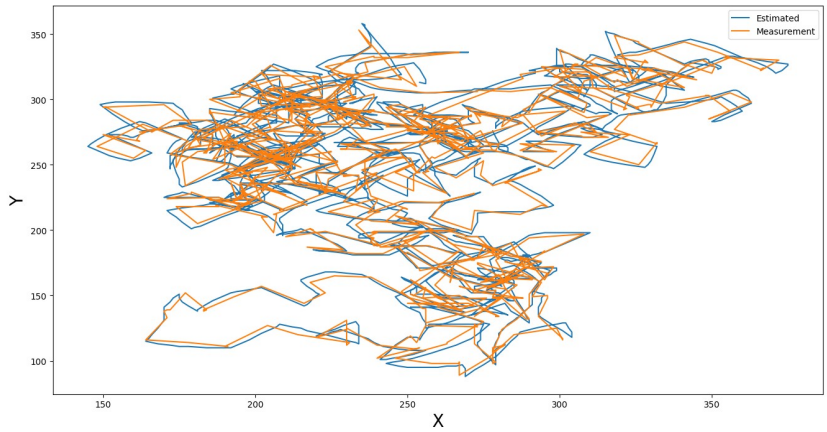

Fig. 10 Plot of Estimate and Measurement for the 2D Tracker

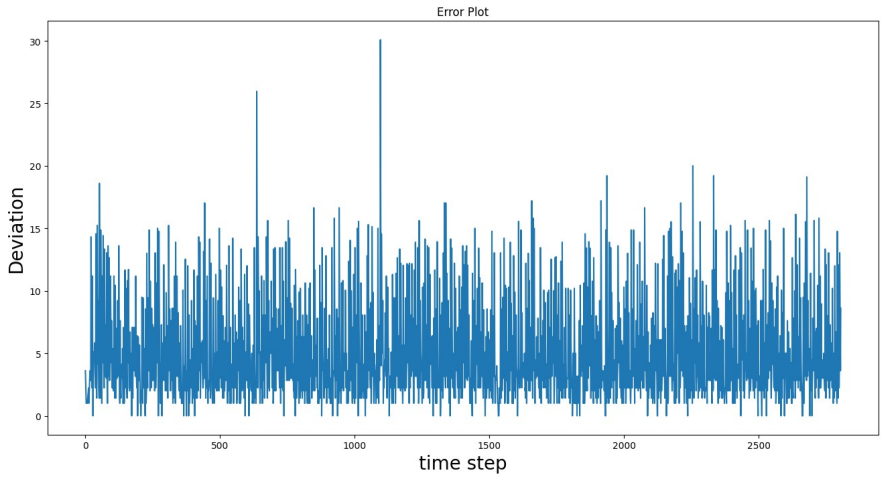

Fig. 11 Deviation of Estimated Path from True Path

4.4 Multiple Object Tracking

Whenever we try to detect multiple objects in a frame the corresponding image processing algorithm treats each frame like a separate problem and outputs the centroids in the order the objects are detected. Since this assumes no previous knowledge of the object's position, the centroids returned need to be mapped to the correct object. In order to solve this problem we need to assign an ID to each object once it is detected. The next problem is to correctly assign the IDs to the objects detected in the next frame and to solve that we need to model this as an assignment problem.

This picture in Fig. 12 shows bipartite graphs where vertices represent objects with their (x, y) co-ordinates. The edges represent the possible matching objects. The edges have a cost which is the Euclidian distance between the two co-ordinates / vertices. The objects on the left are from a previous detection and the objects on the right are the current detections. The minimum weight/cost matching will allow us to correctly ID the objects from left to right.

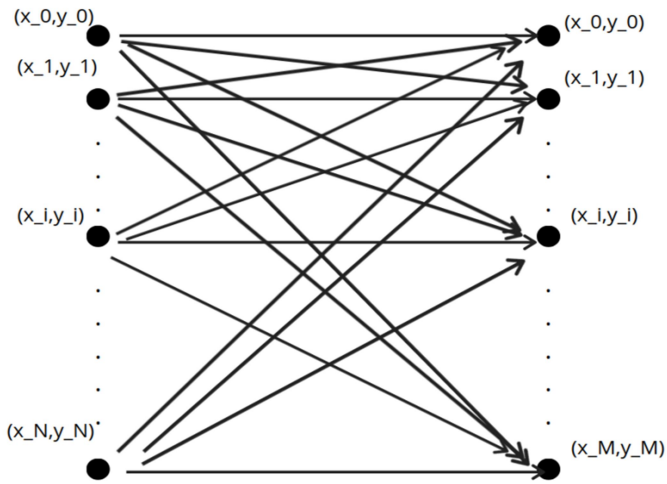

Fig. 12 Multiple Object Representation in two successive frames

The greedy solution to the problem is to check the minimum distance cost comparing with all the possible matching objects and then assign the ID. Considering there are N object IDs and N detections, the number of iterations is:

$$Iterations = N(N-1)(N-2)\dots 1$$

Thus the time complexity becomes:

$$Complexity = O(N!)$$

This is highly undesirable as the assignment needs to be done for every frame and it would eventually make the rendering computationally intensive. The Hungarian algorithm can be used to overcome the particular issue. The bipartite graph can be represented as an adjacency matrix as follows [63]:

$$Adjacency = \begin{bmatrix} dist_{11} & \cdots & dist_{1M} \\ \vdots & \ddots & \vdots \\ dist_{N1} & \cdots & dist_{NM} \end{bmatrix}$$

The Hungarian algorithm to improve the time complexity is given in steps:

Step 1: Find the smallest element in each row and subtract every element in that row with that element.

Step 2: Do the same (as step 1) for all columns.

Step 3: Cover all zeros in the matrix using minimum number of horizontal and vertical lines.

Step 4: *Test for Optimality:* If the minimum number of covering lines is N, N being the number of objects, an optimal assignment is possible and we are finished. Else if lines are lesser than N, we haven't found the optimal assignment, and must proceed to step 5.

Step 5: Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to step 3.

4.5 Real Video Tracking

The diagram in Fig. 13 shows the detailed block diagram to track multiple objects in a video sequence.

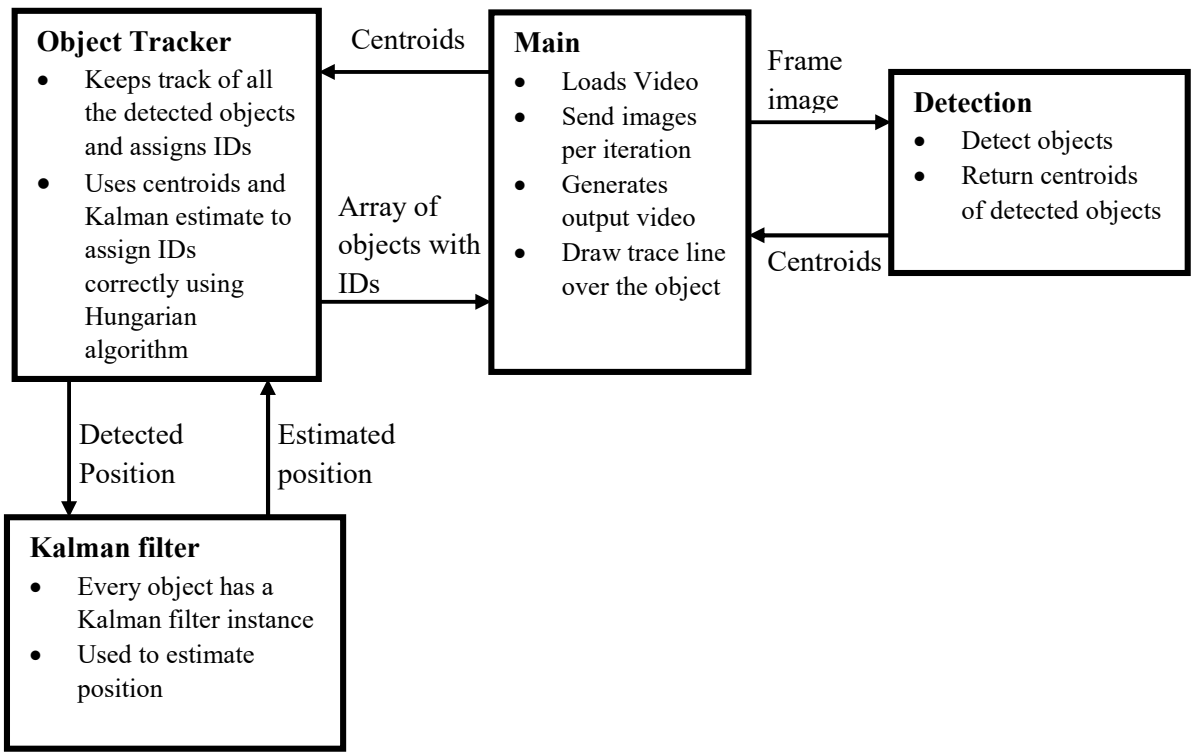

Fig. 13 Block Diagram for Object Tracking

The object tracker is responsible for tracking and logging object IDs using the Hungarian algorithm described in Section 4.4. The Kalman filter is responsible to predict and estimate the position coordinates of each object in each successive frames, based on measurements and model provided to it. The Kalman filter is tuned based on RMS index calculated using the algorithm described in Section 2.6. The detection block as described in Section 4.3 is responsible for detecting the centroids of objects that are to be tracked using edge detection. Since the detection code won't work on a real video the use of pre-trained weights is implemented based on a DNN using OpenCV library of Python.

The following frames are from video tracking using the mentioned algorithm described in Section 4.5:

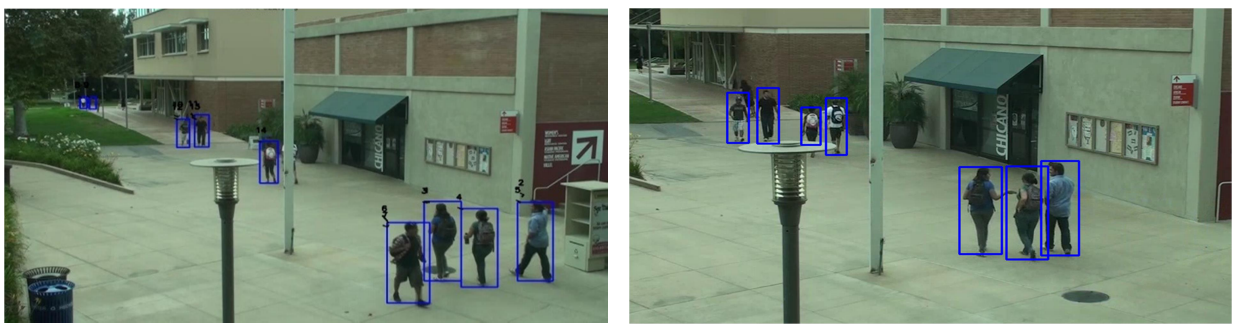

Fig. 14 Real Video Tracking

4.6 Q tuning using RMS index, Sensitivity and Robustness metrics

As discussed in Section 3.1, it is necessary to tune the Kalman filter to get the optimal results, especially the process noise covariance matrix. The algorithms described in Section 3.1 and Section 3.2 are used to determine the optimal Q in all of our results. In this section the simulations and result comparison between a tuned and un-tuned Kalman filter when tracking a 1D object is presented.

Firstly, the traditional Kalman filter with appropriate system parameters to track the motion of an object in one dimension is performed. Then the algorithm in Section 3.2 is used to tune the Kalman filter and P is calculated for which Q happens to give the lowest RMS index, μ_p. In the simulation, the value of $P$is evaluated as 3. Both the results are compared belowwith figures (Fig. 15 to Fig. 20) and table (Table 1).

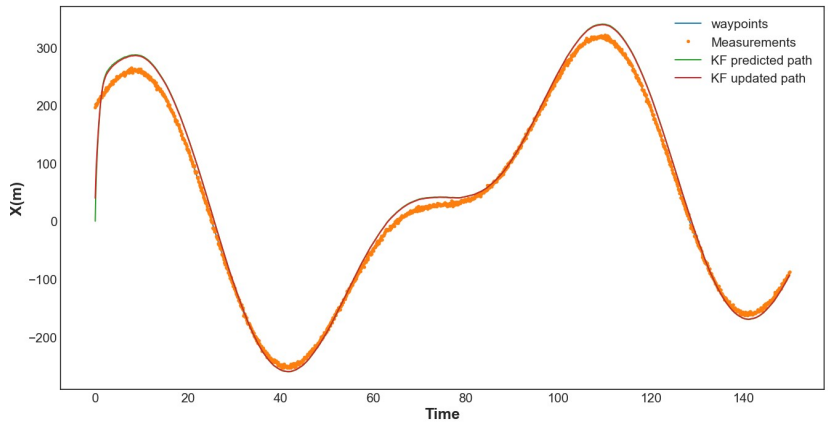

Fig. 15 True Path, Measurements and KF outputs (full time span) for un-tuned KF

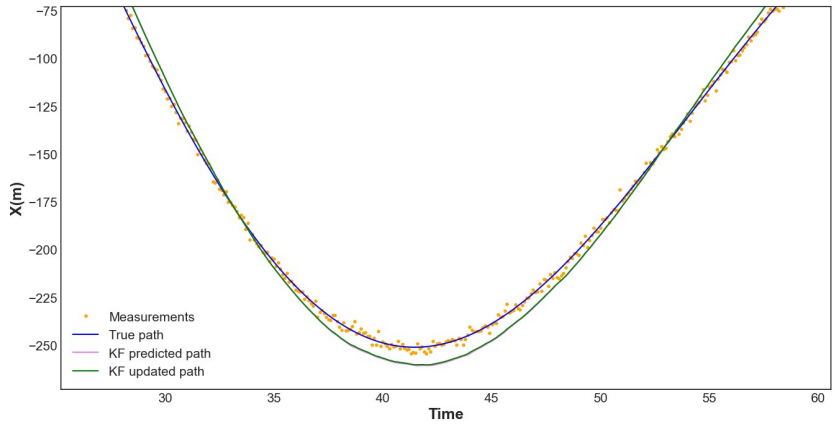

Fig. 16 True Path, Measurements and KF outputs (zoomed in) for un-tuned KF

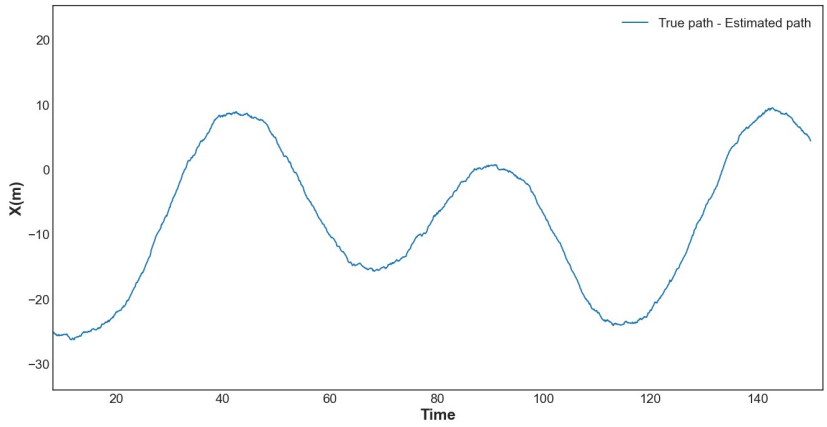

Fig. 17 Deviation of Estimated Path from True Path for un-tuned KF

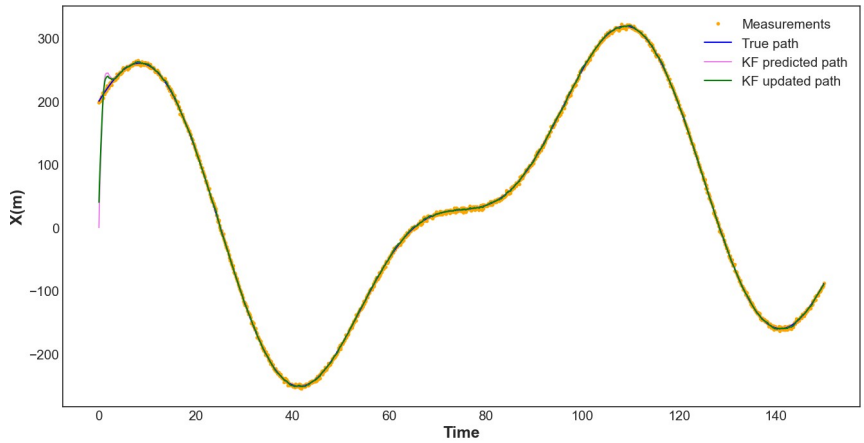

Fig. 18 True Path, Measurements and KF outputs (full time span) for tuned KF

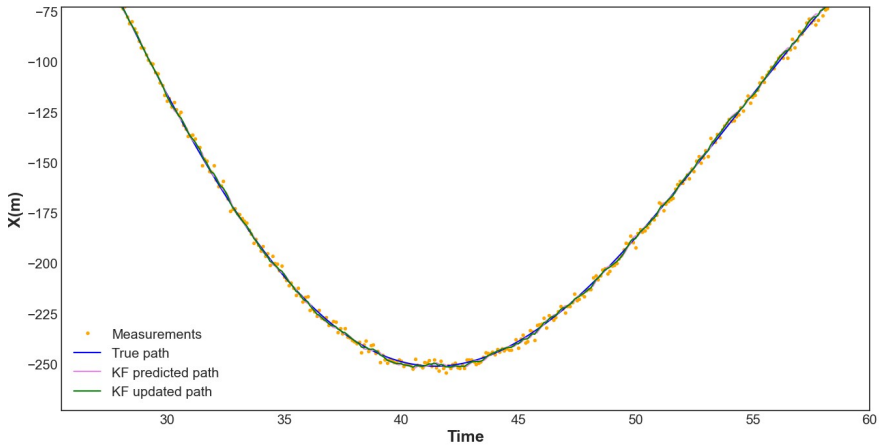

Fig. 19 True Path, Measurements and KF outputs (zoomed in) for tuned KF

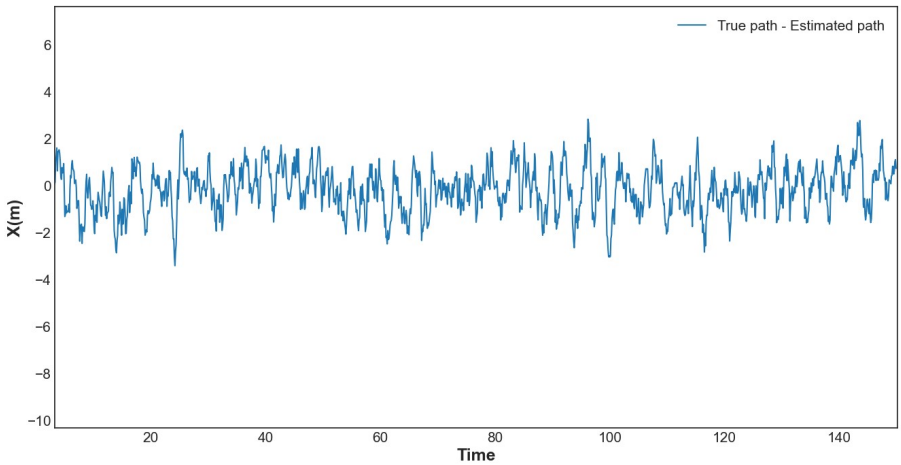

Fig. 20 Deviation of Estimated Path from True Path for tuned KF

4.7 Comparative Study of Tuned and Un-tuned Kalman Filter

The auto tuning algorithm helps to reduce the error without much manual intervention in the system. Fig. 15 and Fig. 18 shows the true path, measurements, KF predicted path and KF updated path for un-tuned and tuned Kalman filter respectively. Fig. 16 and Fig. 19 shows the same by zooming in and it can be very clearly identified that the tuned KF performs better. The zoomed error plots for un-tuned and tuned Kalman filter respectively are presented in Fig. 17 and Fig. 20 which make things more clear as there is seen a great improvement in both the bias and the standard deviation of the error.

In the normal or un-tuned KF, the error plot shows oscillatory behavior, which is an indication of a poorly tuned system. The oscillatory nature is reduced by applying the suitable tuning parameter, derived from the algorithm presented in Section 3.2, and it is seen that the error is greatly reduced to a stochastic nature. Table 1 compares the mean and standard deviation before and after tuning the KF.

Table 3 Comparison between Tuned and Un-tuned KF

	Traditional KF without tuning [51]	**Auto tuned KF**
Error Mean (m):	-8.51	-0.20
Error SD (m):	11.34	0.94

In order to verify that indeed the chosen Q gives the best result, the Sensitivity and Robustness metrics for the system is calculated. Suitable R value is chosen and J_1,J_2 parameters are calculated and plotted in Figure 4, from where it is seen how the system is performing and whether the system is performing in the Robustness zone or in the sensitive zone.

Thus, according to the above mentioned calculations the J_1 and J_2 metrics for the current system is calculated by varying Q as $Q = Q_{nom} \times 10^P$, where P ranges from -10 to +19, and J_1, J_2 vs. P graph is plotted.

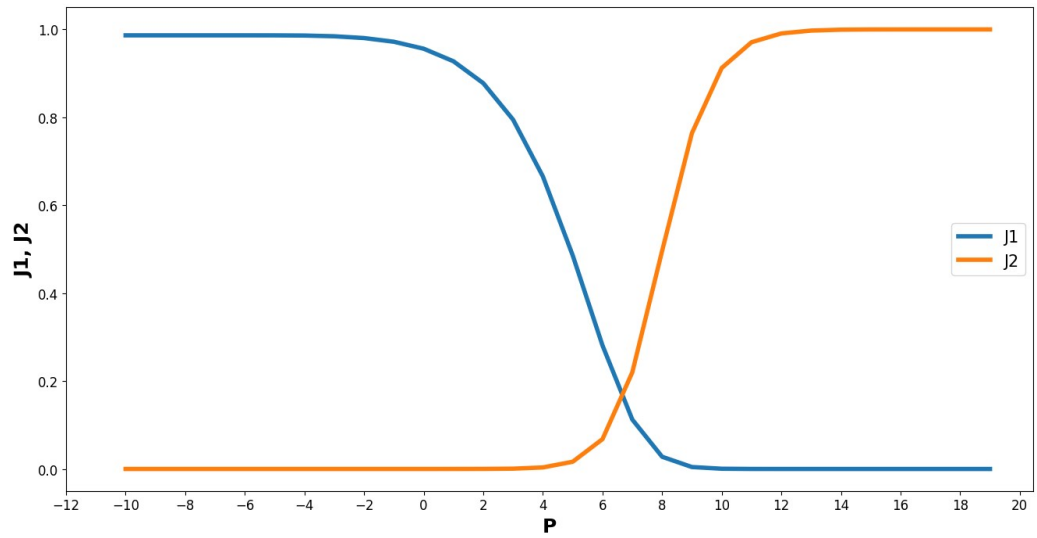

Fig. 21 P vs. Senstitivty (J_1) and Robustness (J_2) metrics

The blue line denotes the J_1 curve and orange line denotes the J_2 curve. As the plot is traversed to the left, where J_1 is close to one and J_2 is close to zero, robust characteristics in the system is experienced. Similarly, as the plot is traversed to the right, where J_2 is close to one, and J_1 is close to zero, sensitive characteristics in the system are experienced.It was clear from the Q estimating algorithm that the system performed the best when Q_{fac} was set to 10^3. Similarly, it is seen that in J_1, J_2 plot at $P = 3$, the system is fairly sensitive and the point is located at the knee of the J_1 curve which tends to give the best performance. J_1value is approximately 0.8 and J_2 is close to zero, which signifies that the system is working in sensitive conditions. In most of the systems it is seen that it works best when operated in robust conditions, i.e., when the J_1 value is near to the knee of the curve [16]. Thus this metric determination establishes the fact that or algorithm worked perfectly to auto-tune the Q, and avoided any kind of tedious manual intervention or manual tuning in order to get the best performance.

4.8 Summary

In this Chapter, simulations are conducted for various scenarios, each accompanied by a description of the relevant input parameters and corresponding output plots. Firstly, the theory for one dimensional tracking using Kalman filter is explained in theory, and the simulation plots for the same are shown. Then, in two dimensional tracking, blob detection algorithm is

introduced, which is used with Kalman filter to detect objects in two dimensions (Section 4.3). As this algorithm cannot be used to track multiple objects at once, a new algorithm is introduced in order to achieve so in Section 4.4. This algorithm made use of detecting each object and assigning it a particular ID, which is significant to track the object over the next frames. Each object is tracked using a separate Kalman filter dedicated to each object's instance. Moreover in Section 4.4, Hungarian algorithm is introduced, which reduces time complexity in order to reach maximum efficiency. The final algorithm discussed in Section 4.5 is used to achieve Multiple Object Tracking for a real video.

The simulation in Section 4.6 included the tuning of the Kalman filter outputs, highlighting the significance of process noise parameter tuning through a comparison between tuned and un-tuned systems. It is observed that, upon proper tuning, a poorly performing system can undergo a substantial improvement in its overall performance. The mean and standard deviation of the error decreased notably after the algorithm in Section 3.2 was used to tune the Process Noise Covariance of the system, reducing from $-8.51m$ to $-0.20m$ and from $11.34m$ to $0.94m$, respectively. Furthermore, the results were validated using Sensitivity and Robustness metrics, which showed that the algorithm used in our system is working optimally. The presented outcome in Table 3 demonstrated the performance disparity between a finely adjusted Kalman Filter and an unadjusted one. It served as evidence that the employed tuning algorithm effectively determines the appropriate Process Noise Covariance matrix.

Chapter 5: Conclusion and Future Scope

5.1 Conclusion

We have analyzed various cases of Object Tracking using Kalman filter. Based on the simulation results obtained, we have arrived at some major conclusions. Kalman filter gives better results if the chosen system model is linear model, assumed noise distribution is Gaussian and the filter is properly tuned. We have made use of Hungarian algorithm to overcome the issue of Multiple Object Tracking.

Tuning the Kalman filter noise parameters is relatively a complex task, and in this Thesis, a way to automatically adjust the process noise covariance matrix, Q is presented. The Initial Error Covariance Matrix, P_0 can be easily predicted based on the initial state and measurement noise covariance. The Measurement Noise Covariance, R can also be set by generally looking at the sensor specifications as specified by the manufacturer or by other experimental means. The Process Noise Covariance,Q on the other hand is not specified and hence an estimation method is necessary to determine Q in order to obtain the best performance. This Thesis presented several techniques to track objects using Kalman filter and also presented a way to automatically tune the Q parameter without much need of manual intervention. It is observed from results in Section 4.6 that, upon proper tuning, a poorly performing system can undergo a substantial improvement in its overall performance. The mean and standard deviation of the error decreases notably, reducing from $-8.51m$ to $-0.20m$ and from $11.34m$ to $0.94m$, respectively, thus improving the system performance by 17 times. The same algorithm (in Section 3.2) to tune Q can be used to improve more complex tracking systems as well. The Sensitivity and Robustness metrics helps to verify that the algorithm is working correctly. It is tested for different systems and significant improvement in result is observed. For future works, this method can be tried and tested and even modified on non-linear and other complex models to establish its wide applicability. Also, this algorithm can also be tested to see its eligibility in different forms of Kalman filter like Unscented Kalman Filter (UKF), Extended Kalman Filter (EKF), etc.

REFERENCES

[1] C. Cédras and M. Shah, "Motion-based recognition: A survey," Image and Vision Computing, vol. 13, no. 2, pp. 129-155, 1995. DOI: 10.1016/0262-8856(95)93154-K.

[2] J. Jiao and H. Wang, "Traffic behavior recognition from traffic videos under occlusion condition: A Kalman filter approach," Transportation Research Record, vol. 2676, no. 7, pp. 55-65, 2022.

[3] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," ACM Comput. Surv., vol. 38, no. 4, pp. 1-45, 2006. DOI: 10.1145/1177352.1177355.

[4] M. F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, et al., "Argoverse: 3D tracking and forecasting with rich maps," in Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8748-8757.

[5] H. Chiu, A. Prioletti, J. Li, and J. Bohg, "Probabilistic 3D multi-object tracking for autonomous driving," arXiv preprint arXiv:2001.05673, 2020.

[6] S. R. Jondhale and R. S. Deshpande, "Kalman filtering framework-based real-time target tracking in wireless sensor networks using generalized regression neural networks," IEEE Sensors Journal, vol. 19, no. 1, pp. 224-233, 2018.

[7] S. Xu and A. Chang, "Robust object tracking using Kalman filters with dynamic covariance," Cornell University, 2014.

[8] P. Mirunalini, S. M. Jaisakthi, and R. Sujana, "Tracking of object in occluded and non-occluded environment using SIFT and Kalman filter," in Proc. of TENCON 2017 - 2017 IEEE Region 10 Conference, 2017, pp. 1290-1295.

[9] C. M. Bukey, S. V. Kulkarni, and R. A. Chavan, "Multi-object tracking using Kalman filter and particle filter," in Proc. of IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017, pp. 1688-1692.

[10] N. Ramakoti, A. Vinay, and R. K. Jatoth, "Particle swarm optimization aided Kalman filter for object tracking," in Proc. of International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2009, pp. 531-533.

[11] S. Vasuhi, M. Vijayakumar, and V. Vaidehi, "Real time multiple human tracking using Kalman Filter," in Proc. of 3rd International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, India, 2015, pp. 1-6.

[12] P. Lal, J. Wei, and K. Goebel, "Comparison of Kalman-filter and extended Kalman-filter for prognostics health management of electronics," in Proc. of 13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, San Diego, CA, USA, 2012, pp. 1281-1291.

[13] F. Yang, H. Chen, J. Li, F. Li, L. Wang, and X. Yan, "Single Shot Multibox Detector With Kalman Filter for Online Pedestrian Detection in Video," in IEEE Access, vol. 7, pp. 15478-15488, 2019.

[14] Y. Chen, D. Zhao, and H. Li, "Deep Kalman Filter with Optical Flow for Multiple Object Tracking," in Proc. of IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 2019, pp. 3036-3041.

[15] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," in Proc. of the IEEE/CVF International Conference on Computer Vision, 2019.

[16] Z. Zhang, B. Sun, H. Yang, and Q. Huang, "H3dnet: 3d object detection using hybrid geometric primitives," in Proc. of Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Part XII, Springer International Publishing, 2020.

[17] S. Ojha and S. Sakhare, "Image processing techniques for object tracking in video surveillance-A survey," in Proc. of International Conference on Pervasive Computing (ICPC), IEEE, 2015.

[18] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele, "Arttrack: Articulated multi-person tracking in the wild," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[19] T. Schubert, A. Gkogkidis, T. Ball, and W. Burgard, "Automatic initialization for skeleton tracking in optical motion capture," in Proc. of IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015.

[20] X. Lan, M. Ye, R. Shao, B. Zhong, P.C. Yuen, and H. Zhou, "Learning modality-consistency feature templates: A robust RGB-infrared tracking system," in IEEE Transactions on Industrial Electronics, vol. 66, no. 12, pp. 9887-9897, 2019.

[21] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," in Artificial Intelligence, vol. 293, p. 103448, 2021.

[22] S. Zhang, X. Yu, Y. Sui, S. Zhao, and L. Zhang, "Object tracking with multi-view support vector machines," in IEEE Transactions on Multimedia, vol. 17, no. 3, pp. 265-278, 2015.

[23] B. Lee, E. Erdenee, S. Jin, M. Y. Nam, Y. G. Jung, and P. K. Rhee, "Multi-class multi-object tracking using changing point detection," in Proc. of Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Part II, vol. 14, pp. 68-83, Springer International Publishing.

[24] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in Proc. of Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Part IV, pp. 474-490, Springer International Publishing, 2020.

[25] Y. Sakai, T. Oda, M. Ikeda, and L. Barolli, "An object tracking system based on sift and surf feature extraction methods," in Proc. of 18th International Conference on Network-Based Information Systems, pp. 561-565, IEEE, September 2015.

[26] J. Chen, Z. Xi, C. Wei, J. Lu, Y. Niu, and Z. Li, "Multiple object tracking using edge multi-channel gradient model with ORB feature," in IEEE Access, vol. 9, pp. 2294-2309, 2020.

[27] A. Tah, S. Roy, P. Das, and A. Mitra, "Moving object detection and segmentation using background subtraction by Kalman filter," in Indian Journal of Science and Technology, vol. 10, no. 19, 2017.

[28] Y. Li, H. Wang, L. M. Dang, A. Sadeghi-Niaraki, and H. Moon, "Crop pest recognition in natural scenes using convolutional neural networks," in Computers and Electronics in Agriculture, vol. 169, p. 105174, 2020.

[29] Q. Wang, Y. Zheng, P. Pan, and Y. Xu, "Multiple object tracking with correlation learning," in Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3876-3886, 2021.

[30] I. Misra, A. Shrivastava, and M. Hebert, "Watch and learn: Semi-supervised learning for object detectors from video," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3593-3602, 2015.

[31] S. E. Li, G. Li, J. Yu, C. Liu, B. Cheng, J. Wang, and K. Li, "Kalman filter-based tracking of moving objects using linear ultrasonic sensor array for road vehicles," in Mechanical Systems and Signal Processing, vol. 98, pp. 173-189, 2018.

[32] C. Veenman, M. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 1, pp. 54-72, 2001.

[33] K. Shafique and M. Shah, "A non-iterative greedy algorithm for multi-frame point correspondence," in Proc. of IEEE International Conference on Computer Vision (ICCV), pp. 110-115, 2003.

[34] S. R. Balaji and S. Karthikeyan, "A survey on moving object tracking using image processing," in Proc. of 11th International Conference on Intelligent Systems and Control (ISCO), pp. 469-474, IEEE, January 2017.

[35] Y. Chen, J. Wang, R. Xia, Q. Zhang, Z. Cao, and K. Yang, "RETRACTED ARTICLE: the visual object tracking algorithm research based on adaptive combination kernel," in Journal of Ambient Intelligence and Humanized Computing, vol. 10, pp. 4855-4867, 2019.

[36] S. R. Balaji and S. Karthikeyan, "A survey on moving object tracking using image processing," in Proc. of 11th International Conference on Intelligent Systems and Control (ISCO), pp. 469-474, IEEE, January 2017.

[37] L. S. K. Vatsavai and K. S. V. Mantena, "Camshift Algorithm with GOA-Neural Network for Drone Object Tracking," in Journal of Ambient Intelligence and Humanized Computing, vol. 28, no. 2, pp. 491-498, 2023.

[38] K. Zarai, I. Ben Abdallah, and A. Cherif, "Improved multi-target tracking crossing paths in MIMO FMCW 8× 16 radar system using a new hybrid AMC-JPDAF algorithm," in Aerospace Systems, pp. 1-9, 2023.

[39] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, pp. 564-575, 2003.

[40] V. Buddubariki, S. G. Tulluri, and S. Mukherjee, "Multiple object tracking by improved KLT tracker over SURF features," in Proc. of Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), pp. 1-4, IEEE, December 2015.

[41] H. Tao, H. Sawhney, and R. Kumar, "Object tracking with Bayesian estimation of dynamic layer representations," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 75-89, 2002.

[42] J. Ning, J. Yang, S. Jiang, L. Zhang, and M. H. Yang, "Object tracking via dual linear structured SVM and explicit feature map," in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4266-4274, 2016.

[43] S. Särkkä and L. Svensson, "Bayesian filtering and smoothing," Cambridge University Press, vol. 17, 2023.

[44] R. Ronfard, "Region based strategies for active contour models," in International Journal of Computer Vision, vol. 13, no. 2, pp. 229-251, 1994.

[45] H. Karunasekera, H. Wang, and H. Zhang, "Multiple object tracking with attention to appearance, structure, motion and size," in IEEE Access, vol. 7, pp. 104423-104434, 2019.

[46] L. Xuan and Z. Hong, "An improved Canny edge detection algorithm," in Proc. of 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 275-278, IEEE, November 2017.

[47] K. Kale, S. Pawar, and P. Dhulekar, "Moving object tracking using optical flow and motion vector estimation," in Proc. of 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), pp. 1-6, IEEE, September 2015.

[48] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," in International Journal of Computer Vision, vol. 12, pp. 43-77, 1994.

[49] G. Welch and G. Bishop, "An introduction to the Kalman filter," in Proc. of the Siggraph Course, vol. 8, January 2006.

[50] P. S. Maybeck, "Stochastic models, estimation and control," Mathematics in Science and Engineering, Academic Press, 1979.

[51] B. Ekstrand, "Some Aspects on Filter Design for Target Tracking", Journal of Control Science and Engineering, vol. 2012, Article ID 870890, pp. 15, 2012.

[52] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in Proc. of 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), pp. 74-77, IEEE, November 2015.

[53] M. Saha, R. Ghosh, and B. Goswami, "Robustness and Sensitivity Metrics for Tuning the Extended Kalman Filter," in IEEE Transactions on Instrumentation and Measurement, vol. 63, no. 4, pp. 964-971, April 2014.

[54] K. Saho and M. Masugi, "Automatic Parameter Setting Method for an Accurate Kalman Filter Tracker Using an Analytical Steady-State Performance Index," in IEEE Access, vol. 3, pp. 1919-1930, 2015.

[55] G. F. Basso, T. Guilherme Silva De Amorim, A. V. Brito, and T. P. Nascimento, "Kalman Filter With Dynamical Setting of Optimal Process Noise Covariance," in IEEE Access, vol. 5, pp. 8385-8393, 2017.

[56] V. Jain and W. D. Blair, "Filter design for steady-state tracking of maneuvering targets with LFM waveforms," in IEEE Transactions on Aerospace and Electronic Systems, vol. 45, no. 2, pp. 765-773, April 2009.

[57] P. Lal, J. Wei, and K. Goebel, "Comparison of Kalman-filter and extended Kalman-filter for prognostics health management of electronics," in Proc. of 13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, San Diego, CA, USA, 2012, pp. 1281-1291.

[58] F. Yang, H. Chen, J. Li, F. Li, L. Wang, and X. Yan, "Single Shot Multibox Detector With Kalman Filter for Online Pedestrian Detection in Video," IEEE Access, vol. 7, pp. 15478-15488, 2019.

[59] K. Saho, "Kalman Filter for Moving Object Tracking: Performance Analysis and Filter Design," 10.5772/intechopen.71731, 2018.

[60] M. Ananthasayanam, "Kalman filter design by tuning its statistics or gains?" in Proc. of Int. Conf. Data Assimilation, pp. 1-102,July 2011.

[61] K. T. M. Han and B. Uyyanonvara, "A survey of blob detection algorithms for biomedical images," in Proc. of 7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES), pp. 57-60, IEEE, March 2016.

[62] R. Reisenhofer and E. J. King, "Edge, ridge, and blob detection with symmetric molecules," in SIAM Journal on Imaging Sciences, vol. 12, no. 4, pp. 1585-1626, 2019.

[63] E. Hamuda, B. McGinley, M. Glavin, and E. Jones, "Improved image processing-based crop detection using Kalman filtering and the Hungarian algorithm," in Computers and Electronics in Agriculture, vol. 148, pp. 37-44, 2018.