SIF Tracking: Novel Estimation Filter for Object Detection and Tracking

by

Alexander Moksyakov

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements
for the degree of

Master of Applied Science

in

Engineering + Artificial Intelligence

Guelph, Ontario, Canada

ABSTRACT

SIF TRACKING: NOVEL ESTIMATION FILTER FOR OBJECT DETECTION AND TRACKING

Alexander Moksyakov
University of Guelph, 2021

Advisor(s):
Dr. Andrew Gadsden

Object detection is currently one of the most heavily researched topics with its potential for object tracking. The research presented here will serve as an introduction to the novel Sliding and Extended Sliding Innovation filter for object tracking. The experiment successfully implements YOLOv4 into Keras to replicate the neural network while integrating the respective code for each filter to perform object tracking and detection on surveillance video. The experiment conducted provided a benchmark for comparing the different filters while introducing a method on integrating new ones for future work. The results demonstrate that the Sliding Innovation Filter had the lowest reported RMSE (root mean square error) out of the 4 filters compared.

DEDICATION

I would like to dedicate this thesis to my family and friends. Thank you all for your support.

TABLE OF CONTENTS

LIST OF TABLES (IF ANY)

LIST OF FIGURES (IF ANY)

1 Introduction

1.1 Project Overview

Artificial intelligence (AI) is a continuously growing field of research that can be implemented into many different disciplines due to its versatility[1],[2]. AI as an application can be used for tasks such as data acquisition, classification, trend prediction and many more, while the concept of AI can even branch into disciplines like philosophy and psychology [3]. Due to the vast number of applications that branch from AI, there are many different ways to define AI. In general, the main goal of AI is to understand and replicate the intelligent behavior of human beings such as reasoning and learning [1]. The more scientific and computational definition of AI is a machine or process that has the ability to develop complex programs by responding to the given environment to produce a model which is optimized to obtain maximum performance [4].

The conceptual beginning of AI began in the mid to late 1940s where the main inspiration of AI-based methods spanned from the philosophy of neuroscience and the biological characteristics of intelligent beings [1],[5]. The amount of resources available for AI-based tasks or application have significantly increased mainly due to the technological advancements throughout the past years. With the transition to a more technologically dependent era, the increase of AI-related resources allowed for the production of highly reliant and complex algorithms [3],[4]. Not only does the availability of data allows for more innovations in AI technology, the use of AI can be also used to produce more information [6]. AI has been revolutionized in ways that no one expected back in the 1940s and expanded into regions originally thought were only obtainable for humans [5], [7]. Benefits of the dramatic advances in AI allows for increased productivity while providing more precision and accuracy [8]. Having these intelligent systems work in tandem with humans allows the allocation of human resources for more significant and less repetitive tasks [3]. This recent resurgence has benefited many different cases

around the world, but one in particular is the newly discovered capabilities of object detection and surveillance [9].

Object detection has been a diverse area of research within the world of machine learning and artificial intelligence, especially with recent technological innovations [10]–[12]. In the instance of recognizing specific dynamics in video surveillance, there are many different algorithms and methods that all hold high accuracies. One such method involves the use of long short-term memory (LSTM) deep models to accurately identify dynamics of a group based on the individuals representing the activity [13]. Object detection is one of the main aspects in research for machine learning. It aims to find target objects with precise localization in a given image and assign each object instance a corresponding label. Currently, there are two main groupings of object detection frameworks: 1) two stage - Region-based CNN and 2) one-stage detectors [14]. R-CNN significantly improved the detection performance in 2014 compared to previous methods [15]. R-CNN's are composed of three primary sections: proposal generation, feature extraction and region classification. For each image that is processed, the R-CNN generates a set of proposals which is designed to reject regions which can be identified as background objects. The proposals are then flattened into a feature vector by a Deep-CNN. This led to the development of a real time detector called YOLO (You Only Look Once) by Redmon et al [16]. This algorithm spatially divides the whole image into a fixed number of cells and each cell now is considered a proposal to detect any objects of interest. Redmon et al [17] then developed YOLOv3 which significantly improved the detection performance and maintained real-time inference speed. This improvement saw a more powerful CNN which was trained on much higher resolution images from ImageNet.

1.2 Motivation and Objectives

The research conducted in this thesis uses the YOLOv4 algorithm for initial object detection. The weights that were used in the model are the standard coco weights [16]. The weights were chosen and kept as-is to avoid any potential research bias for what was used as inputs. Image tracking on the other hand, sets its focus on not only detecting and classifying, but associating a unique number with the detected object and tracking it, frame-by-frame. Seen as an extension of detection, tracking uses other tools besides the neural network for predicting what the next state of the tracked object may be for the next frame. There have been previous frameworks that have used YOLO and their own tracking logic to successfully classify and detect humans, cars, scooters and other items via surveillance footage environments [18]. The work featured here attempts to create unique logic for every respective filter used in the comparison. Estimation Theory, with the usage of these filters can be seen as an extension of YOLO and are pivotal for the research of tracking.

The focus of this study is to introduce a novel estimation theory filter that can leverage the capabilities of artificial intelligence for detection and tracking purposes. This study can also be seen as a benchmark to compare the primary four filters previously discussed (Kalman Filter, Extended Kalman Filter, Sliding Innovation Filter, Extended Sliding Innovation Filter). Currently, there have been multiple experiments that were conducted using the different filters, and the source-code of YoLo implements a Kalman Filter. However, since the Sliding Innovation Filter is novel, there has not been a comparison or even implementation of the filter within object detection and tracking. The research here is directed towards how the Sliding Innovation Filter may be a strong candidate to replace, or use along-side the Kalman filter, and hopefully open the doors for future research done with the SIF. Additionally, this project paves the way on how to implement future tracking techniques with a YoLo detection architecture. This will be open-source and freely available for others to use and branch, as they see fit.

1.3 Thesis Organization

The thesis is organized with 5 chapters and a conclusion with future work. Chapter 1 deals with an overall introduction of the subject, with the motivation behind it and some of the novelty introduced in this work. The goal of chapter one is to familiarize the reader to the current state of object detection and tracking within estimation theory. The second chapter deals with literature review of convolutional neural networks, estimation theory filters and the differences within YoLo and the different versions released through the years. This chapter focuses on what current research uses for implementing computer vision tracking and other pieces of work that have similar use cases with different estimation theory filters. Chapter 3 focused on the backbone of the research for object detection, YoLo v4. Chapter 4 focuses on the proposed method within the research presented, as well as the novelty of the entire source code repository. Lastly, chapter 5 presents the results of experimentation and discussion revolving the results.

2 Literature Review

2.1 Overview

Understanding the role and purpose of artificial intelligence within applications is key. But by understanding how the inner workings of how artificial intelligence works, it allows engineers and practitioners to fully utilize its capabilities within specific domains. Artificial intelligence may have seen a new resurgence in the last 5-6 years, however the core studies of neural networks and deep learning have been around for almost 30 years [19]. There are number of different institutions that are releasing new studies and research with breakthrough technologies that the realm of artificial intelligence is accelerating at a rapid pace. In order to fully understand the complexities involved with object detection and then further, object tracking, detailed research must be done on the existing information surrounding different algorithms and approaches. Chapter two of this thesis will present specific research done around object detection within artificial intelligence, defined object detection algorithms, as well as the core framework used for this study and many others, YoLo.

2.2 Estimation Theory Background Information

Much like the interdisciplinary capabilities of AI, Estimation Theory can be integrated into multiple different fields by utilizing different estimation techniques such as the Kalman Filter (KF), Unscented Kalman Filter (UKF) and the Extended Kalman Filter (EKF) to optimize unknown parameters to predict numerical values. The Sliding Innovation Filter (SIF) and Extended Sliding Innovation Filter (ESIF) are newly introduced filters that create outputs based on bounding properties created in initialization. This bounding feature makes both of these filters great candidates for tracking the movement of people in a surveillance setting [20], [21].

The purpose of implementing estimation techniques is the predictive quality they can provide. The filters allow for object detection algorithms to predict the position of the detected bounding boxes by calculating the position and velocity of the traveling path

[22]–[25]. The predicted bounding box allows for easier calculations during the current step by correcting the estimation using the obtained measurements instead of fully locating the object [26]. Estimation strategies at their core, extract useful information from sensors with noisy measurements [139]. The most popular and well-studied estimation strategy is the Kalman filter. As the basis of modern estimation strategies, the Kalman filter is formulated as a predictor-corrector estimator. The initial states are first predicted using the knowledge of the system and inputs from the system. The initial states are values from the previous time step. Through an iterative process these estimates are then updated using a gain which is based on the state error covariance and observed measurement errors [139].

There are many challenges with tracking objects through a live video feed. However, there has been previous research done using the EKF for the purpose of tracking [134]. Some of these problems are when the object of interest becomes occluded as the detection algorithm will not be able to locate the object as well as when the object splits or merges when occluded. Sensor noise such as Gaussian noise or salt and pepper noise will result in an inaccurate detection and cluttering of objects may result in the algorithm detecting the incorrect object once dispersed [11], [26]–[29]. Implementation of Estimation Theory allows for object detection algorithms to disregard most of these problems. There have been multiple experiments done with estimation techniques that have improved the tracking of the objects as concluded in [21]–[24], [28]–[31]. Another benefit with implementing the previously discussed estimation techniques is the ability to register separate identities to the detected object to allow for true tracking of objects [23], [27].

The KF, UKF, EKF, SIF and ESIF all produce improvements to the system by estimating the trajectory of the tracked object, but each filter differs slightly due to certain computational properties. The Kalman Filter uses a set of predicted and update equations shown in [11], [25], [29] to minimize the mean of the squared error to estimate the potential outcome of the bounding box [23], [28]. The KF is one of the most used filters due to its simplicity and robustness towards linear systems but due to this

property, the KF tends to lose accuracy when induced to a nonlinear system [27], [29], [32]. The UKF produces sigma points that are calculated by the covariance of the estimated system, then applying linear regression on the sigma points to bypass the need of applying techniques introduced in the EKF [33]. The EKF is the nonlinear version of the KF and linearizes the nonlinear model by applying the Jacobian to the system. Though, it has been concluded by multiple sources that EKF is considered a difficult filter to implement and tune [27], [31], [32]. The equations to implement the EKF can be found in [30]. The SIF and ESIF use similar techniques from the linear KF and nonlinear EFK, respectively, but also uses boundary layer techniques from the Smooth Variable Sliding Function (SVSF) to control the range of the outputted prediction update estimator [20], [131]. There is an opportunity to combine the possibilities of different filters together as previously done with the Kalman and SVSF for robust nonlinear estimation strategies [135], [138].

2.2.1 Kalman Filter

The Kalman Filter is the most basic out of the filters listed in sections III and IV [32]. Though it is a lot simpler, the bases of the EKF and even the SIF and ESIF are formulated from the computational natures of the Kalman Filter and built off it to provide more complex filtering techniques that introduce their own advantages and disadvantages given the problem. Specifically, the Kalman Filter works best for linear cases [27]. The linear properties present in the Kalman Filter allows for the filter to predict measurements of linear system to a high degree of success [27], [34], [35].

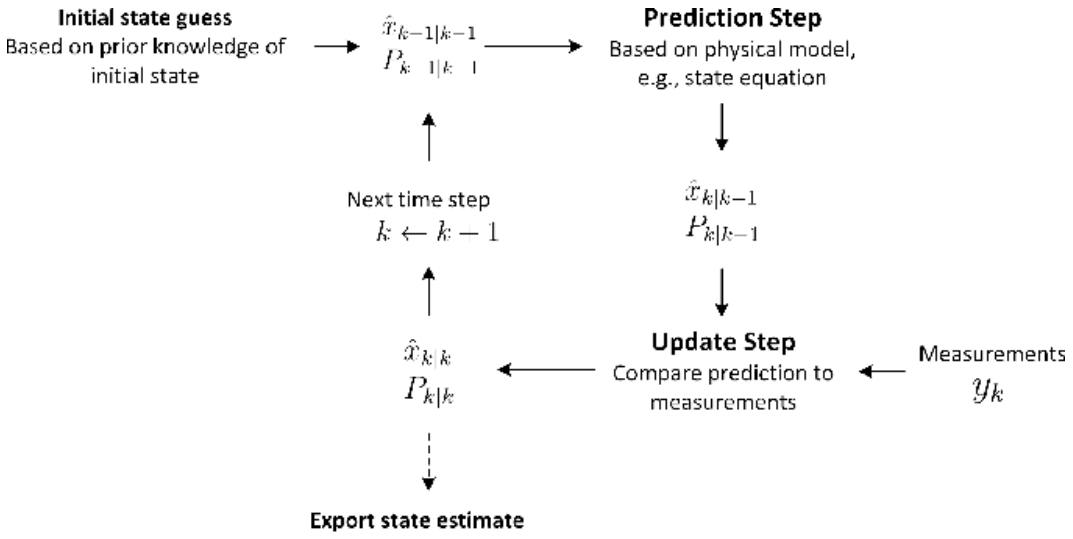

Figure 1: General Structure of Kalman Filter [32]

The main structure that all these filters follow is the prediction and update stage, shown in Figure 1. Once a detection is found, via the object detection algorithm used, in this case YOLO, a bounding box is created. The filters then take the information provided by the detection algorithm and computes the centroid of the box and estimates the new object's position in the next frame, this is the prediction stage. In the update stage, the predicted measurement is then corrected to fit the new measurements obtained. In general, the main purpose is to minimize the squared error between the prediction and the actual [21], [23], [24], [26], [36].

The equations involved in the prediction stage of the Kalman Filter are as followed:

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k \quad 2.1$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \quad 2.2$$

Where $\hat{x}_{k+1|k}$ and $P_{k+1|k}$ are the predicted mean and covariance of the system, respectively, in Equations 2.1 and 2.2. Additionally, F_k represents system matrix, G_k represents the input gain matrix, u_k represents the input to the system, P_k represents the state error covariance matrix and lastly Q_k represents the system noise of the system. $\hat{x}_{k+1|k}$ represents the next state of the system after a time period of k+1 (the next iteration in the system). The update stage then takes the $\hat{x}_{k+1|k}$ and $P_{k+1|k}$ values and inputs them in the update equations shown below in equations 2.3, 2.4, 2.5 and 2.6

$$S_{k+1} = H_{k+1}P_{k+1|k}H_{k+1}^T + R_{k+1} \quad 2.3$$

$$K_{k+1} = P_{k+1|k}H_{k+1}^T S_{k+1}^{-1} \quad 2.4$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - H_{k+1}\hat{x}_{k+1|k}) \quad 2.5$$

$$P_{k+1|k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T + K_{k+1}R_{k+1}K_{k+1}^T \quad 2.6$$

The innovation covariance, S_{k+1} and Kalman gain, K_{k+1} are used to create the updated mean and covariance that will be outputted by the Kalman Filter [25], [29], [36], [37]. H_{k+1} is the system measurement matrix, R represents the measurement noise of the system, while z_{K+1} is the noisy measurements. Note that k refers to the time step of the system, k|k represents the updated values at the previous iteration and k+1 refers to the predicted values at time k+1.

As stated before, due to the Kalman Filter's strong linear properties, the implementations of the Kalman Filter on a nonlinear system will result in poor results compared to the other filters listed in sections III and IV. The main cause of this is because the Kalman Filter ignores the higher order terms in the Taylor series expansion leading to instabilities in the system [38]. Due to most systems being nonlinear, it is required to test the capabilities of the implementation of nonlinear filtering techniques as

it should theoretically improve the predicted outcomes drastically [27], [36]. The use of the Kalman filter has been implemented in many different applications, such as dynamic modelling and motion control of a robotic manipulator, modelling the uncertainties of systems and state estimation [127],[128],[129],[130].

2.2.2 Extended Kalman Filter

The Extended Kalman Filter is another filter with the ability to process information where the source is a nonlinear system. The Extended Kalman Filter makes use of the linearized form of the nonlinear system and measurement functions. By using the Jacobian of the nonlinear system, the linearized version of the system can be found [31], [32]. It has been reported that the implementation of the Extended Kalman Filter can be difficult to optimize. Though the benefit of using the Extended Kalman Filter compared to the KF is that the filter performs well against nearly linear systems as it is based on the first order Taylor series approximation [32], [38].

The structure of the Extended Kalman Filter is very similar to the Kalman Filter with minor differences due to the linearization of the state equations by the use of the Jacobian. The following is the adjusted prediction stage for the Extended Kalman Filter:

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k) \tag{2.7}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \tag{2.8}$$

Just like the KF, the output for the prediction stage present in equation 2.8 are the predicted mean and covariance, $\hat{x}_{k+1|k}$ and $P_{k+1|k}$, respectively. Additionally it is to be noted that *f* refers to the nonlinear system function, F_k refers to the linearized version of the system matrix F (Jacobian matrix or first-order Taylor series expansion.

The $\hat{x}_{k+1|k}$ was calculated using a function of $\boldsymbol{f}()$. This variable represents the Jacobian which linearizes the system of motion. Represented as 'fx' in the python code, the function being called is shown in equation 2.9 below:

$$\begin{aligned} &\text{def } fx(x, dt): \\ &\quad A = \begin{bmatrix} 1.0 & dt \\ 0.0 & 1.0 \end{bmatrix} \\ &\quad return\ np.dot(A, x) \end{aligned} \tag{2.9}$$

The function fx needs two variables, x and dt, where x is the previous predicted mean and dt is delta time for the filter which is a user-defined variable. For this experiment, dt was set to 0.005 to produce a small time-step. Each iteration the Jacobian is called to adjust based on the new predicted mean. The predicted mean is multiplied by the state transition matrix using the numpy [39] function, np.dot(A, x) which conducts the dot product of a 2x2 matrix, A and a 2x1 matrix, x . This results in a linearized predicted mean that was calculated even if the system is nonlinear. For the case of this experiment, as the system is assumed to be linear, the Jacobian is just the linear system.

The update stage equations for the Extended Kalman Filter are shown in equations 3.0, 3.1, 3.2, 3.3 below. When observed, the presented equations look very similar to the KF update stage. The difference is with the calculation of the new updated mean which uses a function of h instead of a value.

$$S_{k+1|k} = H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1} \tag{3.0}$$

$$K_{k+1} = P_{k+1|k} H_{k+1}^T S_{k+1}^{-1} \tag{3.1}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - h(\hat{x}_{k+1|k})) \quad 3.2$$

$$P_{k+1|k+1} = (I - K_{k+1}H_{k+1})P_{k+1|k}(I - K_{k+1}H_{k+1})^T + K_{k+1}R_{k+1}K_{k+1}^T \quad 3.3$$

2.2.3 Sliding Innovation Filter

The Sliding Innovation Filter is a unique filter compared to the KF and EKF as it uses a technique that involves the implementation of a boundary area around the object of interest's motion trajectory. This allows for the system to become more robust with the involvement of multiple techniques to improve tracking responses [20]. The SIF is the linear variant between the SIF and ESIF. Similarities in the formulation structure with the KF can be observed as this filter uses the same prediction update stage approach. The main difference between the SIF and the KF is the implementation of the SIF gain present in the update stage of the estimator. This SIF gain, represented as K_{k+1} uses a δ term to bound the prediction to a fixed width with respect to the saturation term, resulting in a value between -1 and +1, shown in Equation. The representation of the characteristics of the SIF and ESIF can be observed in Figure 2. Once the initial trajectory is identified, the δ term produces a boundary limiting the estimator to a specific width. The saturation term then alternates to keep the prediction as close to the true state as possible.

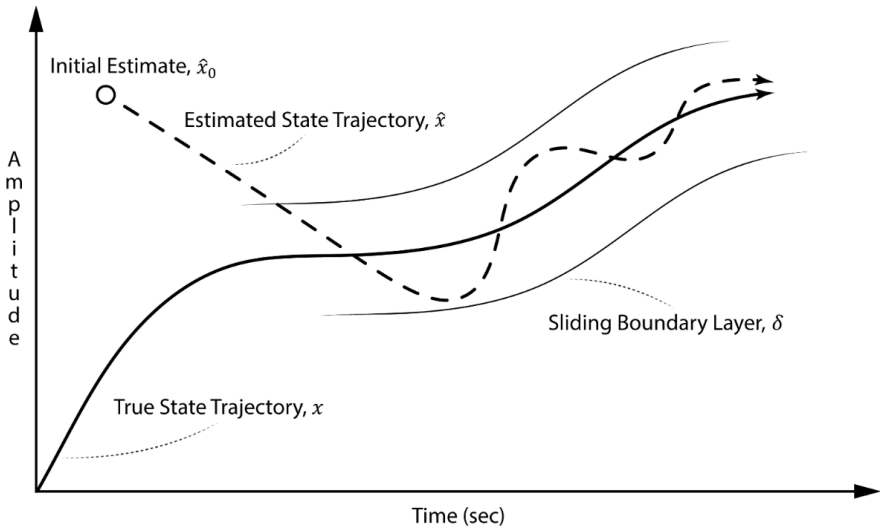

Figure 2: Representation of the Sliding Boundary Layer [20]

The prediction stage equations for the Sliding Innovation Filter are shown in equations:

$$\hat{x}_{k+1|k} = F_k \hat{x}_{k|k} + G_k u_k \tag{3.4}$$

$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q_k \tag{3.5}$$

$$\hat{z}_{k+1|k} = z_{k+1} - C\hat{x}_{k+1|k} \tag{3.6}$$

Identical to the prediction stage of the KF, the outputs in equation 3.4 and 3.5 include the mean and covariance $\hat{x}_{k+1|k}$ and$P_{k+1|k}$ with the addition of the measurement term, $\hat{z}_{k+1|k}$, respectively.

$$K_{k+1} = C^{+}\overline{sat}(|\tilde{z}_{k+1|k}|/\delta) \quad 3.7$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}\hat{z}_{k+1|k} \quad 3.8$$

$$P_{k+1|k+1} = (I - K_{k+1}C_{k+1})P_{k+1|k}(I - K_{k+1}C_{k+1})^{T} + K_{k+1}R_{k+1}K_{k+1}^{T} \quad 3.9$$

The output for the Sliding Innovation Filter includes the updated mean and covariance, as well as the SIF gain, previously explained above. Variables that need to be noted: C^{+} refers to the pseudoinverse of the measurement matrix, $\overline{sat}$ refers to the diagonal of the saturation term, sat refers to the saturation of a value (yielding either -1 or +1), $\hat{z}_{k+1|k}$ refers to the absolute value of the innovation δ refers to the sliding boundary layer, and I refers to the identity matrix (of dimension n-by-n where n is the number of states).

2.2.4 Extended Sliding Innovation Filter

Unlike the SIF, the ESIF builds on the EKF structure, resulting in an estimator capable of providing accurate predictions when introduced to nonlinear systems. The ESIF requires the same techniques as the EKF, such as the implementation of the Jacobian to produce a linearized system of equations [20]. The update stage is where the ESIF differs slightly but is more like the SIF and KF. The introduction of the saturation term and the bounding layer term, δ allows for accurate predictions of a nonlinear system capable of reacting to uncertainties and disturbances.

The prediction stage equations for the Extended Sliding Innovation Filter are shown in equations 4.0, 4.1 and 4.2:

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k) \tag{4.0}$$

$$P_{k+1|k} = F_{k+1} P_{k|k} F_{k+1}^T + Q_{k+1} \tag{4.1}$$

$$\tilde{z}_{k+1|k} = z_{k+1} - h(\hat{x}_{k+1|k}) \tag{4.2}$$

The output of the ESIF prediction stage are identical to those of the SIF prediction stage in equations 3.4, 3.5 and 3.6, but the method to calculate the mean and measurement, $\hat{x}_{k+1|k}$ and $\tilde{z}_{k+1|k}$ include the nonlinear system and measurement functions, $f(\hat{x}_{k|k}, u_k)$ and $h(\hat{x}_{k+1|k})$, respectively.

$$K_{k+1} = H_{k+1}^{+} \overline{sat}(|\tilde{z}_{k+1|k}|/\delta) \tag{4.3}$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1} \tilde{z}_{k+1|k} \tag{4.4}$$

$$P_{k+1|k+1} = (I - K_{k+1} C_{k+1}) P_{k+1|k} (I - K_{k+1} C_{k+1})^T + K_{k+1} R_{k+1} K_{k+1}^T \tag{4.5}$$

The update stage shown in equation 4.5 includes the same gain calculation with the involvement of the saturation and δ terms. The following $\hat{x}_{k+1|k+1}$ and $P_{k+1|k+1}$ terms are used for the next iterations of the prediction update estimator.

2.3 Machine Learning and Object Detection

In recent years, artificial intelligence has seen an incredible technological advancement within different fields, particularly in the field of medicine and speech recognition [40], [41]. The concept of utilizing machine learning within computer vision has been around for a long time and has been the center of breakthrough research for years [42]. However, there needs to be a key distinction made between the capabilities of machine learning techniques and artificial intelligence. Machine learning can be defined as the use of statistical data analysis algorithms for medium-scale supervised and unsupervised problems [43]. It can be defined as a tool that is used to carry-out and automate tasks in an efficient manner, while reducing cost and optimizing results [4][44]. Artificial intelligence on the other hand involves a much deeper understanding and can ultimately be defined as mimicking the understanding of core neuron traits in intelligent beings to becoming the process of learning and reasoning [3], [45]. Artificial intelligence is capable of extending the reach of machine learning by developing a process that can dynamically respond to new information and generalize a reaction (output) [4], [45].

The history of artificial intelligence may have begun within the world of fantasies and imagination, however it is now becoming a very core part of our world [46]. With the use of open-source tools, a lot of novice engineers and software developers are capable of implement their own neural network within 100 lines of code [47]. TensorFlow is one of the frameworks developed by Google that is a machine learning system that can operate on large scale data. The robust architecture is able to provide flexibility to the developer by enabling them to experiment with novel optimizations, loss functions and training algorithms [48]. Another popular framework is PyTorch, which was developed by Facebook. PyTorch is a machine learning library that is designed to support the Pythonic programming style while remaining efficient in utilizing hardware accelerators [49]. Tools such as Tensorflow, PyTorch and Keras have made the possibility of utilizing neural networks for different use-cases extremely easy and even reducing heavy technological overhead [48], [50], [51].

In the past, artificial intelligence was studied and practiced by a small amount of practitioners due to such obstacles of creating robust neural networks, processing power and sheer data size [52]. However, now with open source the world has opened up and with the aid of previously mentioned frameworks. Exceptional research has been done within the studies of chatbots for android applications to interact with text and voice response to even small satellite applications to provide more intelligent computing on space development platforms, utilizing TensorFlow [53].

The nature of open-source has allowed the complexity of artificial neural networks to expand and become further developed as more and more engineers and practitioners have access to tools and technologies [52]. This rapid change has been evident within different fields and research of AI. One particular breakthrough that has been witnessed within AI is the availability and quality of data that is used within model training [6]. O'Leary et al. explains that when a neural network has more data, and the data is of higher quality, the algorithm can generalize results more accurately. As the technological advances helped produce more and better data, the popularity of AI-based systems also increased as the equipment to conduct such experiments become more accessible [43]. This opened the door for particular research to be done within the study of object detection. Some of the earliest research within object detection involved training a general framework based on wavelet representation of an object-class derived from a statistical analysis, utilizing a support vector machine classifier [54]. This earlier approach of object detection was also used within unconstrained, cluttered scenes and had success detecting face, people, and cars [55]. These earlier systems can now be classified as machine learning algorithms, as oppose to artificial intelligence networks as they do not involve the use of a deep neural network. One of the earliest cases of integrating a deep neural network within object detection was in 2013 when Szegedy et al [56] were able to present a powerful formulation of object detection as a regression to object masks. This breakthrough was taken even a step further, as the previously mentioned neural network proposed by Szegedy relied on labeled data to perform making it a supervised problem, Ramik et al [57] presented an intelligent

machine vision system that was able to learn autonomously individual objects present in real-world environments, embracing the ability to be self-learning. These systems relied on deep neural networks that were tuned to specific parameters depending on their interest domain. In 2017 Lin et al [58] introduced RetinaNet, a one-stage detector that demonstrates high degrees of detection and efficiency [59]. RetinaNet was one of the very first frameworks that provided incredibly high-quality results, while maintaining a lightweight build and complex network architecture [58], [60].

The development within AI-based systems lead to more higher quality frameworks and technologies developed. As previously mentioned, every new study opens doors for improvements, even in sub-categories of AI, such as object detection. These improvements also lead to more research done with estimation theory within the realm of AI [132],[133].

2.3.1 Common Strategies and Methods

Within machine learning and artificial intelligence, there are multiple different sub-categories where research is being done to improve the usability of different technologies. Object detection is on the forefront of being one of the most sought out fields within machine learning, as the possibilities of utilizing different computer vision technologies are endless [61]. Many different autonomous, robot-based applications depend on computer vision as it is the only current method to mimic eyes from a human being [62]. The purpose of computer vision and object detection is to successfully process information captured through a lens in real-time which can then be fed into a model for analysis and processing [63].

Within object detection itself, there are multiple sub-components which are necessary. First, the image has to be processed frame-by-frame from the recording channel [64]. Secondly, the object needs to be detected and lastly it needs to be classified [65]. The latter two steps work in sequence, as first an object within the image needs to be detected, via feature extraction of a particular method and/or algorithm, and then classified in to x amount of possible categories that those features best align with

[66]. Feature extraction is the fundamental process that makes object detection so interesting, difficult and rewarding. Feature extraction typically depends on a convolutional neural network as it plays a vital role in accurately determining what is a distinctive feature versus what is not [67]. A CNN typically gets fed values in the form of integer arrays, which represent vector inputs of pixelated images [68]. Depending on the required dimensional representation or what is an important feature, the output can vary significantly depending on what the system identifies as an object. As previously stated, object detection use cases such as video surveillance, there are many different methods and algorithms applied that hold high accuracies. One of the oldest methods is through the use of long short-term memory (LSTM) deep models to accurately identify dynamics of a group based on the individuals representing the activity [69]. For example, when analyzing a video of players playing volleyball, a naïve approach would be to use a dataset of volleyball videos, however the frames would be dominated by features of volleyball courts. Ibrahim et al [70] explored the idea of differentiating between different classes of activities by the spatio-temporal relations between the individuals. This methodology forces a deep model to focus on the relations between people. Thus, with a set of tracker people a temporal deep network (LSTM) can be run to analyze each person. These LSTM's are then used over the people in a scene into the next layer of the deep temporal model [70].

Though, this approach may be used to explore different use-cases, it is still cumbersome, yet it did highlight one of the biggest challenges in video detection: background subtraction. With the sheer amount of data available in a video, it is important to be efficient processing relevant information. This is challenging as all videos contain redundant information (volleyball courts from earlier). Babaee et all [71] identified a Convolutional Neural Network (CNN) to perform segmentation on video frames for background subtraction. Their approach involved an image generating algorithm, a CNN for background subtraction and a post-processing median filter. In real-time video processing, it is difficult to perform pixel-wise temporal filtering as the background model will provide poor results due to rapidly moving objects in the video.

Thus, they relied on SuBSENSE [72] for a more solid pixel-level feedback loop to dynamically adjust the method's parameters. Their model showed significant results by comparing them to the CDnet 2014 [73] metrics. The facility management (FM) score of this CNN was the best overall among considered background subtraction algorithms.

As object detection continues to evolve, the challenge for gaining the most accurate and dependable feature extractions from a given frame remains an obstacle. Currently, there are two main groupings of object detection frameworks: 1) two stage - Region-based CNN and 2) one-stage detectors [14]. R-CNN [15] significantly improved the detection performance in 2014 compared to previous methods. R-CNN's are composed of three primary sections: proposal generation, feature extraction and region classification [74]. For each image that is processed, the R-CNN generates a set of proposals which is designed to reject regions which can be identified as background objects. The proposals are then flattened into a feature vector by a Deep-CNN [74]. Lastly, bounding box regressors are learned from the extracted features as inputs to increase accuracy in bounding objects. This second stage processing is used to learn the proposal generation. The second stage is used to classify the regions based on those inputs.

One-stage detection on the other hand do not divide the detection pipeline into two parts [15]. Instead, they consider all positions of the image as a potential object and try to classify each region of interest as either background of a target object [75]. This detection led to the development of a real time detector called YOLO (You Only Look Once) by Redmon et al [16]. This algorithm spatially divides the whole image into a fixed number of cells and each cell now is considered a proposal to detect any objects of interest. Redmon et al [76] then developed YOLOv2 which significantly improved the detection performance and maintained real-time inference speed. This improvement saw a more powerful CNN which was trained on much higher resolution images from ImageNet.

Both one and two stage detectors are currently explored in production and research-based applications [77]. As with any methods, there are pros and cons to each. The one-stage detector is more lightweight, and is faster than the two-stage detector due to the fact that they pass the image through the neural network once to create and classify bounding boxes [58], [77]. This is possible via the sliding window technique [16], [17], [76], [78], where a window of a certain dimension passes along the image, grid by grid to extract the necessary information.

2.3.2 You Only Look Once

The object detection framework that has been at the center of computer vision with artificial intelligence is You Only Look Once. Initially released in 2016 by Joseph Redmond et al [16], YoLo has gone through multiple different release stages and the current version is YoLoV4, however this newer version of YoLo was not developed by the original author Joseph Redmon [16], [17], [76], [79]. Prior to the release of YoLoV1, the author worked on a ImageNet [9] classification using binary convolutional neural networks [80]. This initial work proposed two efficient approximations to standard convolutional neural networks with the Binary-Weight-Network and the XNOR-Network. Their results were able to boost the operation times of convolutional neural networks by 58 fold [9], [80]. This initial deep dive within the inner workings of convolutional neural network for ImageNet was what lead to the inspiration behind YoLo V1. YoLo is a one-stage feature extractor detection algorithm which uniquely handles detections as regression problem [16], [17], [76], [81]. YoLo is a unified real-time object detection algorithm that instead of attempting to select key interest regions of the image, the algorithm attempts to predict classes and bounding boxes of the whole image through a single run and then detect multiple images using the neural network [16], [17], [76], [82]. The framework is able to train the model by dividing the still-frame image into an SxS grid which is passed through the neural network where features for detection of the bounding boxes and predictions are scanned [83]. Once the image is divided, each grid undergoes a classification and localization. The classification creates predicted bounding boxes from multiple different prediction values of the localized coordinates

(x,y,w,h). Each grid is given an associated confidence score which associates if an object was detected or not. If multiple grids predicted the same object, the localized center point of the object is located and the grid that contains said center point is taken as the ground truth [84].

One of the earliest issues experienced with YoLoV1 came from detecting multiple objects within the single grid instance [16]. Meaning if there multiple different classed objects, than only one is given priority over the remaining grids where lingering objects can reside [16]. YoLo V1 produced state-of-the art benchmarks as it was able to produce fast results in real-time, processing 45 frames a second [16]. This original framework only faced issues detecting small objects due to the spatial restrictions of the bounding boxes and the class prioritization. Afterwards, Joseph Redmon et al. released two new versions of YoLo : YoLo 9000 and YoLo V3 [17], [76]. YoLo 9000 was introduced as a state-of-the-art object detection system that made significant improvements, primarily on being able to predict detection for more than 9000 different object categories [76], [85]. This second version of YoLo was able to predict a large amount of classes by proposing a method that jointly trained the COCO detection dataset [86] with the ImageNet classification dataset. This allowed YoLo 9000 to make predictions on detections for object classed that have yet to be labelled [16], [76], [81], [82]. YoLo v3 upon release demonstrated ground-breaking results compare to the previous two versions of YoLo and all existing other object detection frameworks [17], [82], [87],[88], [89]. The main difference between YoLo v3 and the previous versions of YoLo is the change of what feature extraction convolutional neural network was used [17], [82],[68], [81]. YoLo900 was utilizing a 19-layer feature extraction CNN called darknet-19, which was one of the primary causes for the struggle within smaller objects [90]. YoLo v3, on the other hand has a 53-layer neural network trained on ImageNet detection set [17]. This allowed detections to be made at three different scales by applying a 1x1 detection kernel on feature maps of three different size, all at different locations in the network [17]. Detections at different layers directly addresses the issue of overlapping when attempting to detect smaller objects. Lastly, the loss function used

for YoLo v3 was changed to a cross-entropy loss function [91] for classification of objects [17]. This newer loss function replaced the older softmax loss function, which classifies the scored and then takes the class with the maximum scored to be the detected class [92]. Now, each class is predicted using logistic regression and a threshold is used as a benchmark to predict multiple labels for a potential object. If a class is detected with a higher score than the threshold, then they are assigned the bounding box [17]. These core changes to the algorithm were able to accelerate the YoLo algorithm and was able to achieve incredible results within accuracy, speed and improving on previous downfalls of YoLo 9000. It should be noted that certain two-stage detectors, such as R-CNN [93] were able to provide more accurate results [17], however the speed of YoLo being a one-stage detector [16], [17], [76] while being lighter in size allowed YoLo to be one of the most favorited object detection frameworks to use.

There have been multiple variations of object detection frameworks proposed for specific domain problems. One specific modification was proposed on the actual DarkNet-53 and replace it with DenseNet [94]. DenseNet, coupled with YoLo V3 would further improve the accuracy of detecting smaller objects within denser, crowded environments [95]. One clear opportunity to further improve YoLo v3 is to reduce the size it requires for computational efficiency. With the nature of a CNN, YoLo v3 requires substantial processing power to train and utilize. Huang et al [96] proposed YOLO-LITE in their research, which is a real-time YoLo framework that is able to run portable devices that lack a Graphical Processing Unit. Although faster and lighter, the overall performance of the framework was lost. Tiny-Yolo-V3 is a variant of YoLo v3 [97] that is able to land good performance on accuracy and speed while being a much smaller, more compact model size that can run on embedded devices [98]. During the creation of this project, a newer version of YoLo was released by Bochkovskiy et al [79]. This version did not include the original author of YoLo, however it did make strides in improving the accuracy and speed of the framework [79]. The main differences within YoLo v4 are the new methods for training the algorithm. The authors included a Self-

Adversarial training method as well as a Mosaic training method to improve the ability of YoLo v4 to generalize it's detection patterns [79]. This improvement also increased the speed of training as the newer Mosaic pattern is able to take the inputs of 4 separate training images as a single input, which allows YoLo's original grid localization to work better for feature extraction and object detection [16], [17], [76], [79]. The new YoLo v4 architecture utilized new features such as a Mish activation network, Mosaic data augmentation, DropBlock regularization with the original YoLo v3 head to achieve state-of-the-art results for the MS COCO dataset at a real-time speed of 65 framers per second [79]. As of the writing of this thesis, there currently does not exist a tiny or lightweight version of YoLo v4.

3 You Only Look Once Version Release 4

As previously mentioned, Yolo is a real-time object recognition system that is capable of recognizing multiple objects within a single frame. It is currently able to perform at the highest level of performance in terms of precision and speed [17]. From its earlier development it is currently able to predict up to 9000 [99] classes and even further unlabeled classes, that were previously unseen. These detections are identified with a bounding box and the framework can be deployed and used in production settings [16], [17], [76].

The YoLo framework utilizes a single convolutional neural network, which is capable of dividing an image into separate regions while simultaneously making predictions with bounding boxes. YoLo was initially developed by Joseph Redmon et al [16] in 2016. Since then, the real-time object recognition system has led itself to further improvements for newer computer vision algorithms and related research [100]. Joseph Redmon publicly stated he no longer wanted to develop YoLo due to the potential misuse of technology. He ceased his research, and subsequently evolution of YoLo because the ethical issues were growing and becoming impossible to ignore [101]. Thus, Alexey Bochkovskiy et al [79] developed the newest version of YoLo.

Yolo V4 was released in April of 2020 [79]. The team mentions that it is a significant speed performance from the original Yolo build. According to the research article create by Bochovsky et al [79] – the newer version of YOLO demonstrated a 10% improvement in accuracy calculation for detection and classification, while also providing a 12% increase in speed and optimization. The structure of Yolo v4 can be divided into three separate portions: the backbone, neck and head. The backbone of the architecture is where the dense block of convolutional neural networks, specifically CSPDarknet53 [102]. The neck of the model is where the extracted features from the CNN get passed down for further processing [79]. The neck consists of a Feature Pyramid Network, a Spatial Pyramid Pooling Layer and a Spatial Attention Module. Lastly, the primed and clean features are passed into the head of the network, which is

still the same architecture Yolo v3. Since the release of YoLo v4, there have already been numerous amounts of released research papers that have utilized the new technology for compute vision processing and analysis. For instance, the field of agriculture is constantly researching new methods to accurately detect different features within plants, vegetables, and fruits. Wu et al [103] utilized the Yolo v4 deep learning algorithm for real-time apple flower detection in natural environments. The incredible fact from their work, besides the results, was the fact that the model was only trained on 2230 labeled apple flower images. Relative to the world of neural networks, this a smaller training batch. Wu et al [103] even verified their findings by comparing the results from the Yolo v4 model with other algorithms such as Faster R-CNN [104], Tiny-YoLo v3 [98] and SSD 300 [105]. They concluded that the results achieved with the YoLo v4 algorithm scored a mean average precision of 97.32%, which was the highest of all the algorithms and a score of 72.33 frames per second. None of the other algorithms were able to match the detection speed that YoLo v4 had presented. Another example of research that came with using the improve YoLo v4 algorithm was presented by Zhu et al [106] where they successfully evaluated the sound source localization of sound imaging instruments. Their work showed that they were able to score a mean average precision of 96.3% with detection speeds reaching up to 34.6 frames per second. Due to how new YoLo v4 is, there is still some time that new research is going to be released utilizing the technology, while many current researches may be exploring ways to migrate from the former Yolo v3 version onto the v4. The areas of research YoLo v4 is able to contribute is not limited, as Deng et al [107] used YoLo v4 within the realm of detecting iron surface cracks for material science applications, while Mahto et al [108] used the refined yolo framework for increased accuracy in vehicle detection.

3.1 Schema: How it Works

Prior to analyzing the composition of YoLo v4 and the architecture, it is important to make the distinction between two machine learning concepts known as bag of freebies (BoF) and bag of specials (BaS). Bag of freebies is everything that can be tuned within the model that does not interact with the technological architecture, such as data augmentation, class imbalance, cost functions and other tuning parameters [79]. On the other hand, bag of specials which impacts inference speeds as well as performance. These changes may include feature integration tactics, FPN, and other additions to a neural network [79]. The key aspect with YoLo v4 is that if focused intensively on adding bag of specials improvements to the existing YoLo v3 algorithm.

3.1.1 Backbone

The first aspect of the neural network that was change was the backbone, where ultimately the neural network was completely revamped. The improved accuracy of YoLo V4 is a result of designing a deeper network that has the capability of extending receptive fields and increase model complexity [79], [109]. The dense block contains multiple convolution layers where each layer is composed of batch normalization, a ReLU activation function followed by convolution [79].

YoLo v4 utilizes a CSP connection with the Darknet-53 CNN as the backbone in feature extraction [110],[79]. A Cross-Stage-Partial (CSP) connection is used to separate input features of the Dense blocks into two different parts. The first separation part actually skills the Dense block but acts as the input to the next transition layer, while the second separated portion will go through the Dense block as normal flow. The hybrid of utilizing CSP and Darknet53 can be referred to as CSPDarknet53. The architecture can be seen below of the new backbone CNN. The CSPDarknet53 model achieves a higher accuracy within object detection compared to other ReSNet designs [17].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Figure 3: CSPDarkNet53 Architecture [79]

3.1.2 Neck

After the backbone, the algorithm moves into the neck, where further tuning for feature extraction occurs. The main purpose of the neck is to enrich the information that feeds into the head. This is achieved by feature maps coming from the bottom – up stream, as well as the feature maps form the top-bottom stream are combined together element-wise and then fed into the head [17], [79], [103].

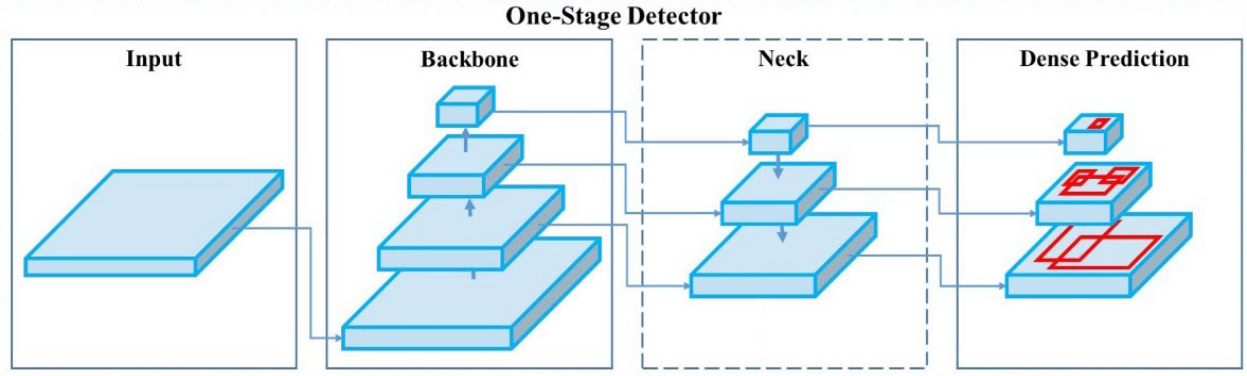

Figure 4: One-stage detector visualized [79]

In Yolo v3, the neck utilizes a Feature Pyramid Network (FPN) in making object detection predictions at different scales. The purpose of a FPN is to make a prediction at a particular kernel scale and combine it with neighboring layers predictions. The result is then passed into a convolutional filter to reducer sampling and create the resulting feature maps for the head input [16], [17], [76]. However, in Yolo v4, the FPN is modified into what is known as a Spatial Pyramid Pooling (SPP) layer.

SPP applies a different strategy in detecting objects at different kernel scales within the neural network. Instead, the SPP replaces the last pooling layer in the convolutional layer. The new passed in feature maps are spatially divided into m by m grids [79], [108], [111]. As previously mentioned, YoLo v4 specializes in detection features at different dimensions and scales. This is due to utilizing the SPP to retain the output spatial dimension of the extracted feature maps. A maximum pooling layer is applied to the sliding kernel of different sizes to preserve the spatial dimension [79]. Then, the feature maps from all the different kernel sizes are concatenated together as the output. The last component of the neck is the Spatial Attention Module (SAM). In SAM, a maximum and average pool are applied separately to input feature maps from the SPP [112]. The results are then fed into a convolutional layer which is activate by a sigmoid function to create spatial attention.

3.1.3 Head

All the detection occurs in the head of the algorithm once all the spatially separated features are extracted. This part is responsible for classification and regression of the bounding boxes. These accepted values contain the predicted bounding box dimensions (x,y,h,w) and the probability of the classes. Since this is a one-stage detector, YoLo applies the head network to each anchor as the objected detectors are anchor-based (meaning which weight has the biggest value corresponding to the highest probability of correctly classifying the detected image).

4 Proposed Strategy

The research proposes a novel approach of utilizing the YoLo v4 framework and estimation theory filters for the purpose of accurately tracking individuals. Combining both elements of estimation theory and YoLo v4 involves separating the two entirely and then combining both near the end of the process. The principal components needed to run this algorithm involved Keras, Tensorflow, OpenCV and the latest CNN of YoLo v4 which is then needed to be replicated within the codebase. The main scripts needed to run this process involve the individual estimation filter files, which is found in the object_tracker subdirectory, the actual model.py file which contains the composition of the YoLo v4 CNN and lastly a main.py file that is responsible for some data processing, communication between both technologies and the main output.

As a quick start, YoLo v4 requires weights files which are trained from the readily available COCO data set [86]. Since YoLo v4 takes in weight files formatted in .h5, a conversion script has been made available to transform the .weight files into .h5 files. With the correct formatted files, YoLo now has the capability and knowledge to properly identify and classify each object it may detect within an image or within a video through still frame processing. Since the release of Tensorflow 2.0.0, Keras is not automatically integrated as the backend for TF, meaning all TF imports utilizing TF 2.0.0 come with the entire Keras library [113]. Keras is an integral component for the project as Keras backend is used to build the YoLo V4 defined model. A detailed breakdown of the YoLo V4 layer composition can be seen in the following sections, however the primary components imported from Keras include the Conv2D layer, the ZeroPadding2D layer, UpSampling2D layer, Concatenate layer and a MaxPooling2D layer. Additionally, LeakyRelu is used as the primary activation function and regular BatchNormalization is used – all imported from the Keras library. The initial Keras build of YoLo v4 was done by GitHub user Ma-Dan – who established the core repository of YoLo v4 which has since received 273 stars and 120 forks [114]. Lastly, OpenCV is used as the main tool to process videos and break them down into their respective frames, while also allowing to create an output video in mp4 format from resulting frames [115]. These technologies

act as the main backbone of the entire project. User's should be warned that due to the nature of these technologies, a list of what specific version used was added, as well as a requirements.txt file as well, to reduce dependency issues.

The following section will breakdown the usage of each specific estimation theory filter, but it is good to have a brief understanding of how these filter work in conjunction to everything, and the overall process flow of the project. First, a virtual environment is spun up to not affect the local machine. Once a video is selected and all path variables have been taken care of, the main YoLo function starts from the main executable file. This function utilizes the OpenCV2 framework to process the video file, then breaks it down into each frame. From the frame, CV2 extracts the height and width of the entire image. While the CV2 is processing all the possible frames from the video, each current frame is then passed into the YoLo v4 detect image function. The detect function transforms the image from RGB pixels into its array matrix representation which is then processed through the YoLo v4 neural network to capture dimension of the grid bounding box, the confidence score of the individual bounding box and what class predicted. This is the critical part and the utilization of YoLo within the entire architecture. After getting the raw outputs from the YoLo model, they are further processed to be used in demonstration and tracking purposes. For example, in the context of this project, all classes that are not detected as 'person' are discarded and no bounding box will be built. Otherwise, the bounding box output is then taken and processed with the Python library Pillow to draw the surrounding bounding box. Lastly, the detected class label is printed on the top of the bounding box itself, with the confidence integer.

Following this process, the bounding box integers and confidence score is passed through a function that calculates the center of the object detected. Since tracking people in a crowd at different depths poses challenges, this function is used to calculate the approximate center point based on the bounding box outputs from YoLo. The main function is fairly straightforward as it just takes the average of the left and right dimensions, while also taking the average of the top and bottom dimensions and placing

it as [1,2] Numpy array. These Numpy arrays contain the center co-ordinate of each person in a frame is passed into the core trackerDetection function, where the estimation theory manipulation takes place. A detailed breakdown of the estimation theory portion can be read below, however the output from this function is the predicted co-ordinates for the next possible frame.

The estimation theory trackerDetection function is iterative, meaning it takes the previously defined ‘tracker’ variable as input to compare the accuracy of the measurements from the previously frame to the actual results of the next. Essentially, the tracker is predicting the center position of the object detected in the n+1 frame. It should be noted that the very first ‘tracker’ co-ordinate pair is a random value that is picked arbitrarily. The nature of the filters are able to adjust themselves based on the results from the predicted values to the actual values [10], [22], [116], [117]. The remaining functions involve creating the visuals that are seen in the output video, such as the bounding box, and the tracking line, which are processed frame by frame and then put together as a video through OpenCV2 and calculating the RMSE values between the predicted and actual co-ordinates. This research also focused on the tracking accuracy for one individual, thus within the footage the results are based on just a detected person class, where there is a maximum count of one present.

4.1 YoLov4 + Estimation Theory Filters

Estimation theory has been a key subject to study within engineering for many years [118]. With the recent boom in popularity within machine learning and artificial intelligence, a lot of research has been made to incorporate said topics within engineering, and other core applications. Estimation Theory has incorporated machine learning in many different applications, such as creating novel computer vision frameworks, object tracking, system modeling and creating neural networks from scratch using principal estimation theory concepts [119]–[121]. In this research we cover two of the core estimation theory filters, the Kalman filter an the Extended Kalman filter. However, there are other types and variations from the original Kalman filter,

depending on the purpose and application, such as the unscented Kalman filter, used for non-linear estimation purposes [122].

YoLo, the popular detection framework discussed throughout this thesis implements a version of the Kalman filter within their C code, in order to keep frame by frame detections accurate for the object initially detected, and to not mix up detections of other objects detected in the same class [16], [17], [76]. From the sheer power that estimation theory provides within the realm of tracking multiple objects, research have been focusing on utilizing this strength for computer vision applications. One such example is the novel Complexer-Yolo, created by Simon et al [123]. Their work was an integration of YoLo V3 for the purpose of detection and semantic segmentation for autonomous driving. Their work focused on integrating a custom-built neural network with YoLo V3 to establish real-time recognition, however within their paper they stated that they utilized a Bayesian filter to define where the state of a individual measurement is and that state is calculated using a Kalman filter to update according to the measurement model. Then, the prediction for the next state is done using an unscented Kalman filter with nonlinear coordinates [122], [123]. Another similar example of how the Kalman filter is now a common tool to be used within object tracking comes from the work of Kim et al [124]. Their work primarily focused on the analysis of using deep learning algorithms for image recognition. They were able to propose a method of combining Kalman filters into a deep learning-based detection network for improved detection and tracking in video. Their detection was built off Yolo V2 and their results showed significant improvement in tracking accuracy over the existing Yolo V2 [124].

DeepSort is another popular framework that was built off of the mathematical principals from the Kalman filter [18]. This framework is actually one of the newer methods released that is completely separate from the YoLo family. DeepSort is also a CNN that is focused on extracting key features from segments of an image. Similar to YoLo, the application of the Kalman filter is to take the current state as the outcome from the detected image and then attempt to predict the next state and re-adjust itself based on the accuracy. There have been many applications of utilizing image

recognition with DeepSort as the backbone, such as the work presented by Hou et al [119] where they attempt to reduce the effect of unreliable detections by incorporating low confidence track filtering with DeepSort.

Most of the previous work that implements object detection with object tracking focuses on using the Kalman filter for linear systems, or even within nonlinear systems as well. In our work we are focusing on presenting a novel filter that can be used within these existing frameworks in hopes of increasing performance, reliability and overall usage of estimation theory within computer vision applications. The Sliding Innovation Filter (SIF) mimics the behavior of the Kalman filter however it differs in one key aspect which is handling irregularities or views that get obstructed. The SIF handles robustness and has a margin for allocating the proper prediction result [20].

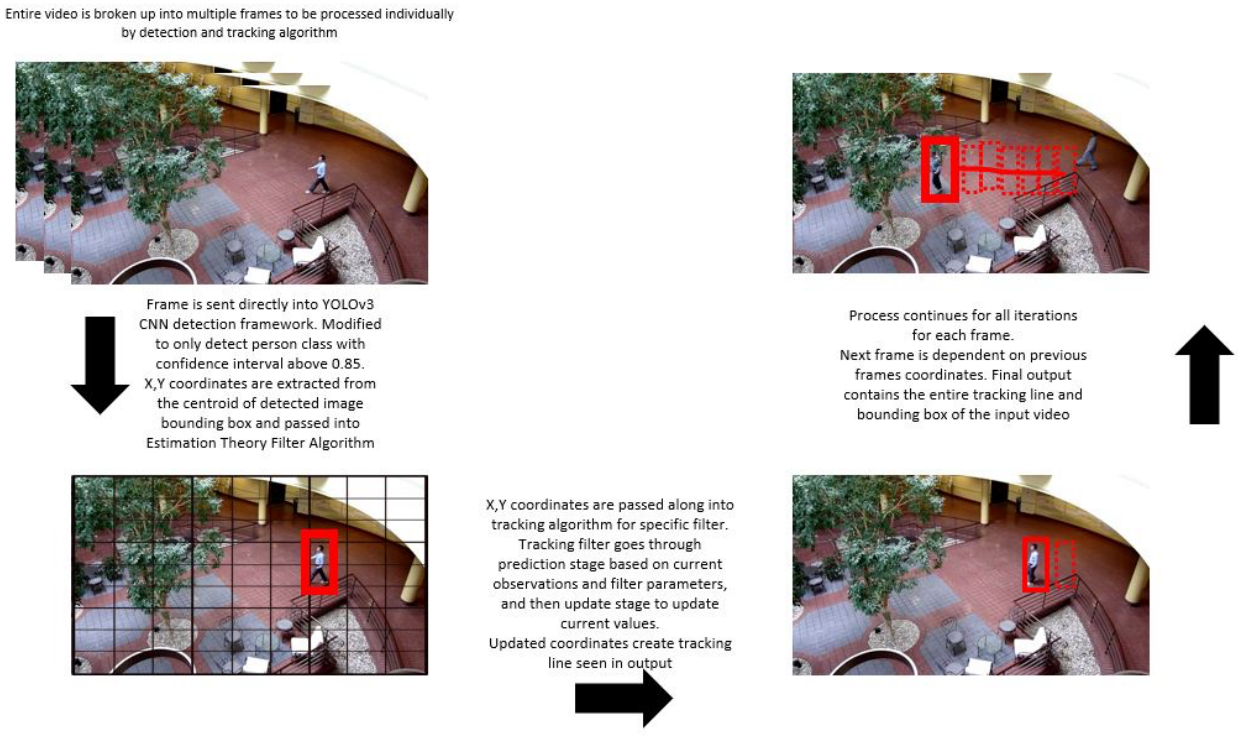

Figure 5: Process of YoLoV4 + Estimation Theory Filters

4.2 Architecture

The proposed architecture of the method follows-up on the previously mentioned process of how an image is pipelined from raw input format, all the way to creating the bounding box and center output coordinates. As the single-frame image is processed through the YoLo V4 neural network, the main outputs received are the center points of all the objects detected within that frame. At this point, excluding the use of a Kalman filter, YoLo V4 would be able to continuously detect objects detected within each frame passed into the neural network [79]. This may serve a shallow purpose of simply identifying objects, however the core usage of a Kalman filter is providing enough support for these detections to assign a unique ID to each object. When a new frame is processed, if the new output coordinates match the prediction stage of the Kalman filter (within a defined margin), then it is the same image. This allows for the process of tracking objects within multiple frames.

The architecture proposed has enabled the usage of four different filters. 2 existing, and 2 novel ones introduced by Gadsden et al [20]. This architecture is able to dynamically process any of the four different filters based on the way the program is written. Even with the Kalman filter, which is already used within many different tracking frameworks [117], [121], [124], [125], the Kalman filter we propose has key differences in the mathematical principles which is further highlighted in the next chapter. As for how each of these filters work within the processing of images, the center co-ordinates are passed into a trackerDetection function called inside the main project script. Depending on which filter is turned on within the script, the trackertDetection will pass the center co-ordinates into that respective filter. Inside the actual filter logic, there is a main class which is responsible for the filter initialization, the prediction, and the update. The chosen filter class keeps track of the estimated state of the system and the variables or uncertainty of the estimate. The predict and the correct methods implement the functionality. The initialization of the filter depends on the dimension of the data that is passed into it. Via the configuration and output from the YoLo algorithm, we set all the dimensions to match the co-ordinate outputs from the images that are processed.

Within initialization, many different variables are declared within the class so that they can be readily available throughout the class declaration. The measurement function, state of the system and the vector of observations are all declared initially with zeros matching the expected shape of the images passed. Then, specific attributes are declared such as the uncertainty covariance, state transition matrix alongside the process uncertainty and state uncertainty. The last variable declared is called lastResult, which keeps the last predicted result from the filter to be used later for comparing and calculation steps. It should be noted that the process and state uncertainties may be tuned. The values chosen for this research and results were chosen over a course of grid selection tuning.

The first real processing step of the filter class is to predict the next values based on the current ones passed in. This step is broken down into multiple functions but the core usage of the prediction sate is to the vector x and variance of uncertainty (covariance) [126]. This part of the code is different from one filter to the other as the linear or nonlinear formulas used to predict, and update state are different. The last step of processing for the filters occurs in the update phase. This phase is responsible for adding a new measurement to the filter, with the primary focus of saving posterior state and updating the current one. The update portion of the script focuses on calculating the filter gain and updating the state estimates and state error covariances. Again, this process and the calculations used are different from one filter to the other to handle the linear versus nonlinear cases. Within the update phase of the filter logic is where the RMSE is calculated. The process for calculating the error, or more specifically the delta is by taking the n+1 image frame with its newly tracked coordinates from the CNN and comparing them with the previous n predicted coordinates from the filter. The difference in these values corresponds to the error calculated and allows for troubleshooting and benchmarking. Additionally, for this particular research case the error is only calculated for a single entity that is seen visible on the frame for a given time. Calculating the RMSE for all the objects identified, regardless of class or number of items causes a skewed answer for the RMSE graphs.

4.3 Benefits of proposed methodology

The proposed methodology has many different benefits for the realm of research and future development. There are existing code repositories that introduce ways of integrating Kalman and Extended Kalman, however the code introduced in this project was created from scratch to mimic the actual mathematical principles of each filter. One key difference is that our methods do not rely on the Hungarian method in the prediction or update phase. To start, it introduces and implements two new novel filters that can be used for tracking and potentially replace the existing Kalman and Extended Kalman filters which are already used in most computer vision tracking frameworks. The utilization of these filters may be used, or at the very least added to the continuation of exploring the most optimal methods for detection and tracking. In our research and experimental set-up, the SIF provided the best results compared to all the other filters. The Sliding Innovation Filter is able to add new constraints for image tracking that the previous Kalman filter was not able to, such as the bounding layer that is built within the SIF. For instance, the SIF creates boundary for the object tracked which allows for a greater margin for detection. This is especially the case when an object is initially capture and seen in a frame, but then is obstructed by going behind another object for a set of frames. In typical cases and depending on how many frames the initial object is obstructed; tracking loses sight and as a result issues a new unique ID when the object reappears. However, with the SIF the object has a greater opportunity to be recognized after reappearing from the obstructed view. RMSE also demonstrated that the SIF provided a lower error than the Kalman filter, which is a promising result for CCTV applications. This value may be tuned even further by modifying some of the gains and values within the initialization step. The last true benefit this research hopefully brings into the world of computer science is the flexibility of using different filters for comparison and benchmarking. With the code all available and open source, it can be seen how 4 completely separate estimation theory filters may be used within the application of a YoLo convolutional neural network. Lastly, script performance can mimic the same performance as YoLo V4 in terms of frames per second.

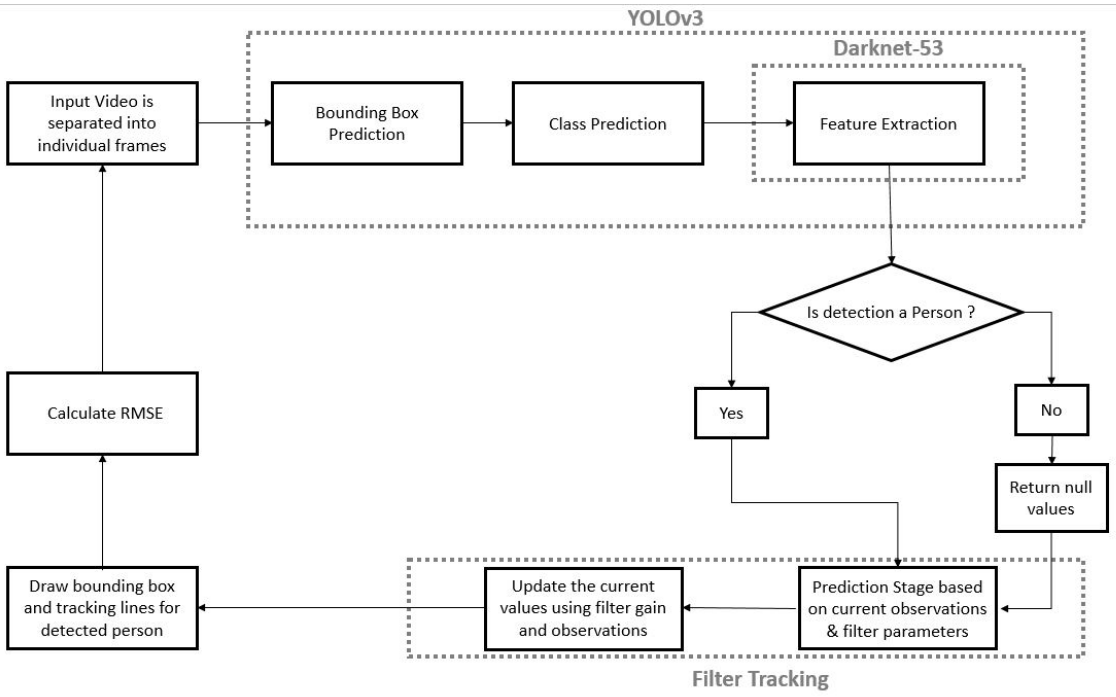

Figure 6: Architecture Flow of Research Project

5 Computer Experiment: Setup and Results

5.1 Experimental Setup

As discussed in the previous section, the implementation of the different Estimation Theory techniques with respect to object tracking will be tested. An existing framework created by GitHub user MattZheng "keras-yolov3-KF-objectTracking" will serve as the boiler plate for the project as it initially encapsulates the YoLo V3 algorithm. This framework created, builds off another repository from srianant's "kalman_filter_multi_object_tracking". The difference between srianant's repository and mattzheng's is the conversion from python 2.7 to python 3.5.2. For this reason, mattzheng's framework was used as the bases of this project. However, further novel modifications were needed to make the experiment successful, such as using Python 3.8, upgrading the machine learning backend to TensorFlow 2.2, and recreating the boiler plate to leverage YoLoV4 instead of YoLoV3. The current framework only consists of an implementation of YoLo using the pre-created weights which can be obtained from pjreddie.com, who are the creators of the YoLo algorithm. As the purpose of this report is to test the differences between the impact of each filter, the only original content created will consist of the respective frameworks that involve the implementation of such filters.

The experiment focused on leveraging YoLo v4's detection algorithm in order to detect weight-trained objects, such as people, chairs, airplanes, etc. The detection algorithm creates a bounding box around each detected object with the percent confidence at the top left corner. For the sake of presenting result findings, only bounding boxes around human detection were made visible. The footage chosen for the experiment is a stock 15-second clip of students walking in an atrium, shown in Figure 7. This video is chosen due to a single shot camera that is stationary, as well as multiple different students are seen in the footage. The latter is used to demonstrate the distinguishing capabilities of the algorithm even from objects that fall under the same category but are individually unique.

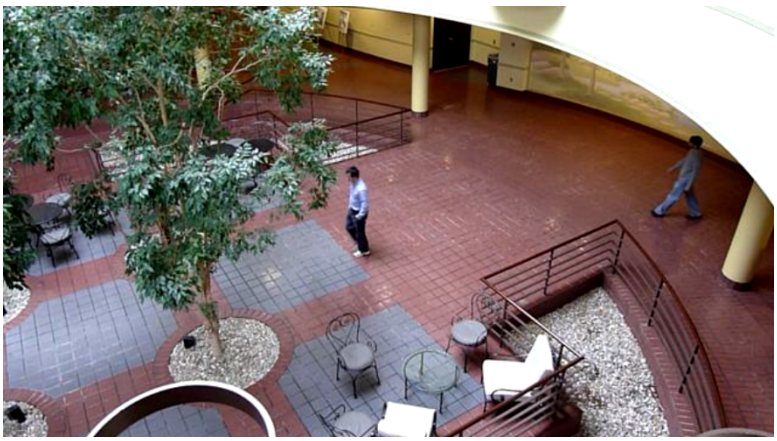

Figure 7: Still frame of Atrium footage used in experiment [136]

5.1.1 Experiment Dependencies

To execute the source code, the dependencies are shown in Table 1:

Table 1: Code Source Dependencies

Tool	Version
YoLo	Version 4
Python IDE	Any applicable
TensorFlow	2.2.0
Keras	2.5.2
Python	3.8
Keras Backend	Tensorflow

5.1.2 Estimation Process

To properly connect the detection algorithm with a filter, multiple separate scripts were written for the 4 filters mentioned in this thesis. The initial detection algorithm creates a bounding box around the detected image. From this bounding box, it calculates the center value of the object detected. This yields a pair of X and Y values. These values that are passed from the detection stage into the filtering stage. The filtering stage itself has three separate components. The first is the initialization of the unique filter. This phase initiates the following parameters shown in Table 2:

Table 2: Variable Representation

Variable	Representation
dt	Delta time for filter
Dim x	Dimension of state variable
Dim z	Dimension of measurement vectors
C	Measurement Function
x	State variable
b	Vector of observations
P	Uncertainty Covariance
A	State transition Matrix
Q	Process Uncertainty
A	State Uncertainty

It should be noted that each filter may have different initialization values. Due to the 2D shape of the images captured, the X and Z dimensions are set as 2 throughout the entire experiment. The source code has been created to allow dynamic data that may require different dimensions for state variables of measurement vectors.

The second phase of the Estimation Theory filter process is to predict the values based on the inputs. As previously mentioned, the inputted values from the detection algorithm are the X and Y coordinates of the object detected. The prediction function predicts the state vector and variance of uncertainty and returns the vector of predicted state estimate. Each respective prediction step equation can be found in the respective filter's sub-section. The prediction section involves calculating the predicted state estimates and using that to calculate the predicted state error covariance. The prediction function returns the newly calculate predicted values and are then fed into the update function. The last phase involves updating all the existing variables from the output of the prediction phase for the iteration. Updating involves calculating the respective filter gain, updating the state estimate based on the actual input from the detections and then updating the state error covariance. Again, these equations can be seen in the respective section of each filter used in the research.

This process continues iteratively for each frame of the footage. The 15 second footage used creates an output of 483 frames. These frames are individually processed, one-by-one within the filter procedure. The output from this process creates 483 images into a separate folder. Each image contains the bounding boxes, percent confidence of the detected object and the unique tracking trail of the object. The tracking trail has a unique color pertaining to the unique identity of the object and shows the entire tracking procedure from entering frame to exiting. Lastly, the images are all put together into a video which is outputted at the very end of the entire combined algorithm.

Figure 8: Still Frame Image with Kalman Filter Tracking [136]

The variables described in the Estimation Theory process are shown and referenced in Table 2

The accuracy and result of each filter were determined by the quality of the outputted video. Cleanliness, proper tracking, and overall quality of the tracking lines determine which video does a better job. Besides relying on a video and human judgement, the root means square error (RMSE) was calculated.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{d_i - f_i}{\sigma_i})^2} \quad 4.6$$

This step occurs in the update phase mentioned above. D_i represents the current true measurement of the frame, f_i describes the previously saved prediction estimate. The mean squared error is calculated between the true values (the actual detected values) and the predicted values (output from the prediction phase of the Estimation Theory filter). Each mean squared error is appended to an empty list and continues until all the frames are complete. The last sequence is to take the sum of all the appended

results, divided them by the total number of results and take the square root. Each result provides deeper insights into the accuracy of the filters and creates a ground for comparison. Some margin of error in this process is when the detection algorithm detects multiple images per frame. This multi-detection cause difficulty to correspond the correct filtered output with the corresponding detected input.

5.2 Chosen Videos

The video for the experiment was chosen as it replicated a CCTV video camera the closest. The footage was stock footage from a University atrium which shows different individuals walking past the captured area. And element that allows this footage to be essential in result comparison is that it allows for error calculations for both tracking a single individual, and a group of individuals all in the same time. Different instances of when individuals entered the frame of reference from random locations was able to provide dynamic flexibility when analyzing the results of the applied filter. It was discussed that having a single video with the same entry point for the object detected and tracked would provide shallow insights into how the actual algorithm is performing.

Another element that this footage provides is within the detection portion of the algorithm. Since the YoLoV4 model was trained on the classical coco data set, it was the capability of recognizing objects such as chairs, trees, animals, and other common items. As such, the stock neural network was capturing all these elements within the frame of refence and initial results were skewed due to not tracking only a single person, or only a human object. The backend processing logic was modified to only be able to recognize and process one object at a time, and only if it is a person detected.

A second video was also chosen as it demonstrated similar criteria. Stationary video camera recording traffic of cars, trucks and motorcycles. For the experiment, the code was modified to only process and detect cars driving along the highway. Further analysis can be done on different still frame cameras to process and compare results. One note to make is that there is a performance gap when dealing with a large crowd of

people versus dealing with a smaller crowd. Calculating the RMSE for a large group of individuals poses many different problems and challenges and is a less straight-forward process than comparing to a single individual.

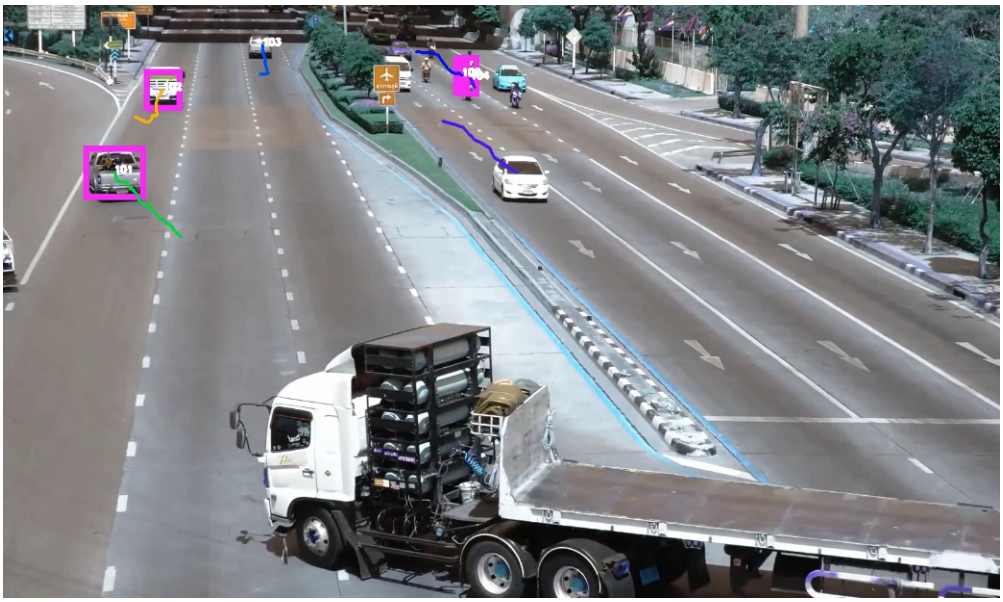

Figure 9: Video Footage with Traffic [137]

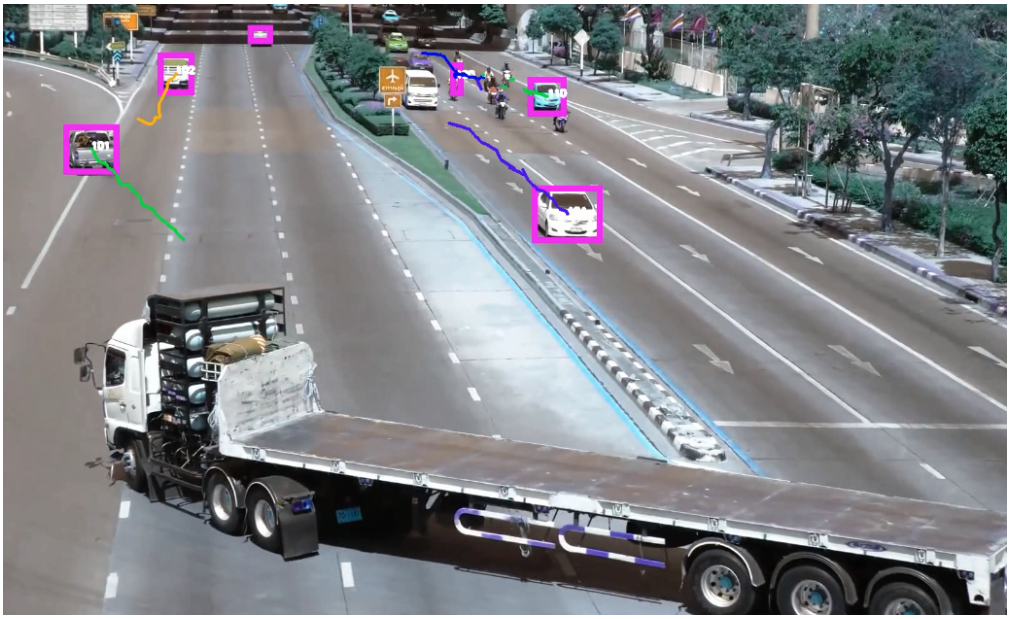

Figure 10: Another Video Footage of Traffic [137]

5.3 Results

This section will highlight the result findings of each filter. To judge and compare the performance of each filter, we will use the RMSE calculate, the image to demonstrate the capabilities of the tracker and lastly an image demonstrates the RMSE graph for the first 45 iterations.

Table 3: RMSE Values for Different Filters

Filter	RMSE Value (Max Pixel Distance)
Kalman Filter	2.2519
Extended Kalman Filter	2.3186
Sliding Innovation Filter	1.2357
Extended Sliding Innovation Filter	1.6392

The following assumptions have been made to simplify the given problem: The equation of motion was a linear system. The video output was 2D meaning depth was not considered. The information provided by the YOLO algorithm was the only data used (x- and y-coordinates). As only x- and y-coordinates were considered, color and shape were ignored resulting in cases where tracking was lost due to nonlinear trajectories while object of interest is occluded and the resulting RMSE values were an accumulation of detected objects in the video. This included people, chairs, and the tree, meaning that a filter may have tracked a moving object such as the people correctly but failed to predict the trajectory of stationary objects, resulting in a higher overall RMSE.

For the traffic video footage, a similar experiment was used to compare the different filters and their results. Instead of tracking an individual person, a group of cars were tracked and the RMSE was calculated based on the results from this.

Table 4: RMSE for car tracking

Filter	RMSE Value (Max Pixel Distance)
Kalman Filter	3.4492
Extended Kalman Filter	3.4623
Sliding Innovation Filter	1.4283
Extended Sliding Innovation Filter	1.6646

5.3.1 Kalman Filter

The Kalman Filter yielded stable results for tracking. It is used for linear systems of equations.

Figure 11: Still Frame with Kalman Filter Tracking

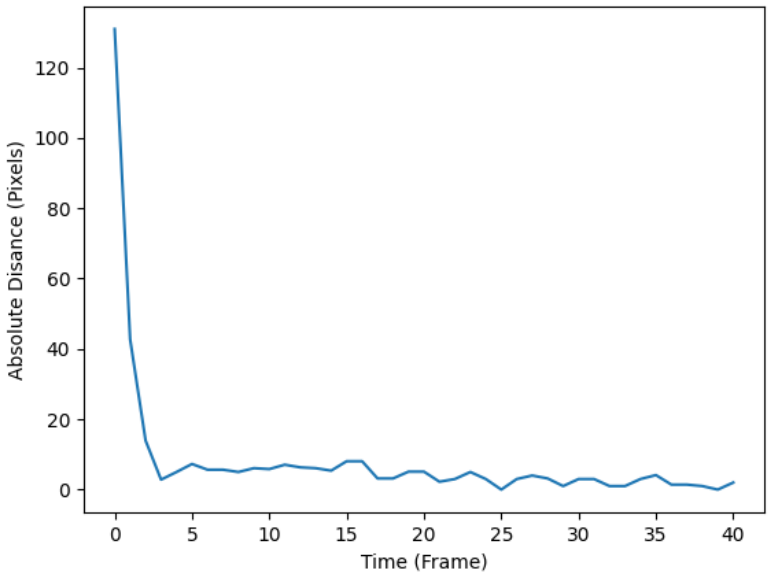

Figure 12: Kalman Filter RMSE for X Iterations

The calculated RMSE value was 2.2519. The Kalman Filter outputs a fairly stable tracking video. Figure 9 shows a distinct blue line tracking the individual from start to finish and begins tracking the second person with another line. There are some lingering lines due to the algorithm tracking and predicting the location of the chairs. The blue lines show a bumpy characteristic which demonstrates the attempts are predicting the pattern of the person detected

5.3.2 Extended Kalman Filter

The Extended Kalman Filter shows poor results due to the nature of the EKF. The EKF is used for nonlinear systems, where the equation of motion is known and can be computed. This was not the case for the experiment.

Figure 13: Still Frame Image with Extended Kalman Filter Tracking

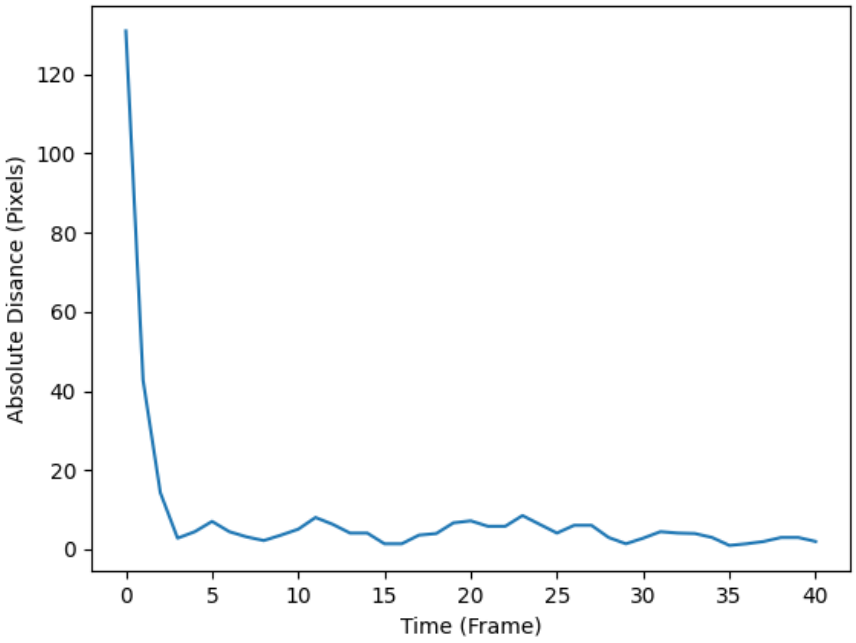

Figure 14: Extended Kalman Filter RMSE for X iterations

The calculated RMSE value was 2.31863. The EKF does significantly worse than the KF for a number of reasons. The first being the EKF works best for nonlinear systems where the equation of motion is known. Secondly, it appears that the EKF struggles when it deals with multiple detected images in a single frame. This is made evident in Figure 11 where there are numerous uncertainty lines in the detection of the stationary chairs.

5.3.3 Sliding Innovation Filter

The Sliding Innovation Filter is a novel filter introduced for object tracking. It is referenced and introduced in [20]. The SIF goes hand-and-hand with the Kalman Filter by being a filter used for linear systems. The novel SIF provides incredible results and has the lowest RMSE with the cleanest picture.

Figure 15: Still Frame Image with Sliding Innovation Tracking

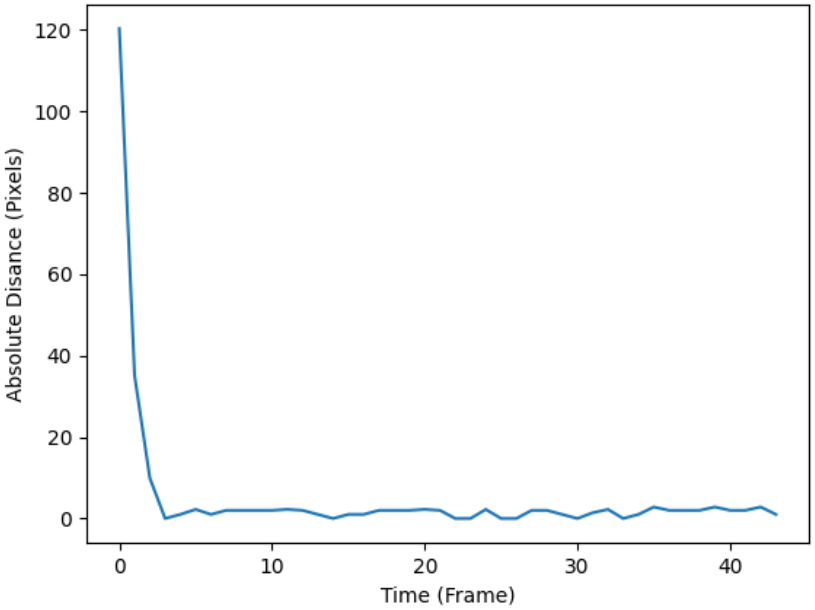

Figure 16: Sliding Innovation Filter RMSE for first the first few

The calculated RMSE value was 1.2357. The SIF has the lowest RMSE values. Figure 14 shows the values of the graph bounded from 0 to 5. The Sliding Innovation Filter does a remarkable job of tracking the object in question. It accurately tracks the center of each object, which is seen in the second person detected within Figure 13. Lastly, the detected chairs are still present, however there is no uncertainty lines as the detection line is a stationary point for the frame.

5.3.4 Extended Sliding innovation Filter

The ESIF is similar to the EKF. It behaves well with nonlinear systems. The ESIF is also a novel filter that was introduced in [20].

Figure 17: Still frame with extended sliding innovation filter tracking

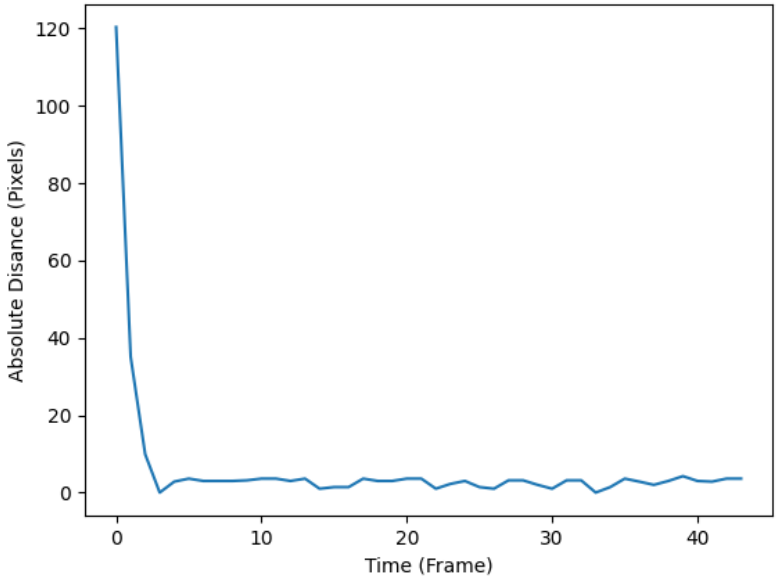

Figure 18: Extended Sliding Innovation Filter RMSE for first few iterations

The calculated RMSE value was 1.6392. The ESIF faces many of the same problems that the EKF faced, in terms of dealing with nonlinear systems without the

precise equation of motion. The ESIF yielded a fairly high RMSE value in comparison to the other five. Figure 15 also shows that the tracking line is jagged, demonstrating the sliding portion of the filter. The chairs are stationary and yield no uncertainty lines.

5.3.5 Computational Timing

To add another element of comparison between the four filters, we added a simple timing function to measure the computational requirements for each filter. The timer begins at the start of the first image and finishes at the end when the last iteration is complete. To get a fairly accurate estimate of each filter and the computational each filter requires, their respective script were ran 5 times and the average between those 5 was chose as the estimated time for completion.

Filter	Time Completed (seconds)
Kalman Filter	19.261
Extended Kalman Filter	19.197
Sliding Innovation Filter	19.136
Extended Sliding Innovation Filter	18.694

5.3.6 Disturbance Cases

Object tracking and detection works almost flawlessly when the correct environment and domain is set up for experimentation. However, in practical settings, a lot of the time tracking becomes difficult for edge cases that were not anticipated within development. Some of these issues include when the object that is being tracked goes

behind another bigger object and the tracker loses sight of the detected class. This occurs frequently within surveillance settings as the target can typically blend in with either crowds of others or hide behind a building. This can cause issues when attempting to follow the movement of a certain individual. However, the Kalman filter and the different varieties that spawn from the Kalman filter have capabilities of re-capturing the object tracked if lost sight for a brief moment. This is what is important about estimation filters as they have enough robustness to account for sudden drops or loss of focus.

To account for disturbances from real-time footage, an experiment was created to compare how the Sliding Innovation Filter and Kalman Filter handle situations when the video was interrupted. This was done by artificially injecting interruptions for batches of frames from the main function of the program. A batch of 20 frames were interrupted from different instances of the video. For example, at the beginning of the video, after the filter had a chance to capture the moving person, a batch of 20 frames were removed from processing, essentially interrupting the flow of the process. When the frames resumed, the filter had to re-engage, detect and resume tracking of the initially detected person. This experiment occurred at three separate points of the video to analyze true differences. These three instances are labelled ‘break 1’, ‘break 2’ and ‘break 3’. Break 1 interrupts the frames from 42 and frame 62, break 2 interrupts the frames from frame 62 to 82, and lastly break 3 interrupts the frames from 82 to 100.

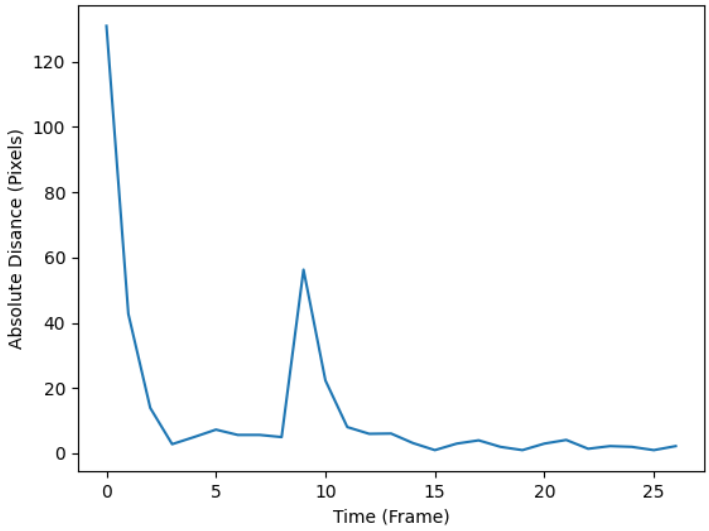

Figure 19: RMSE for KF on Break 1

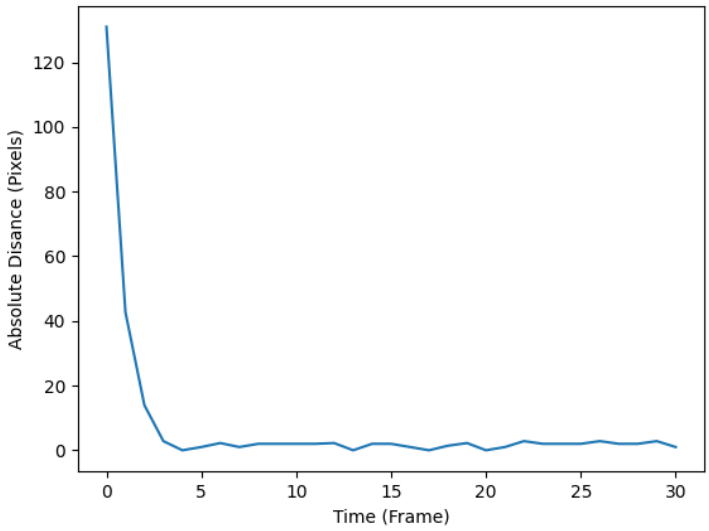

Figure 20: RMSE for SIF on Break 1

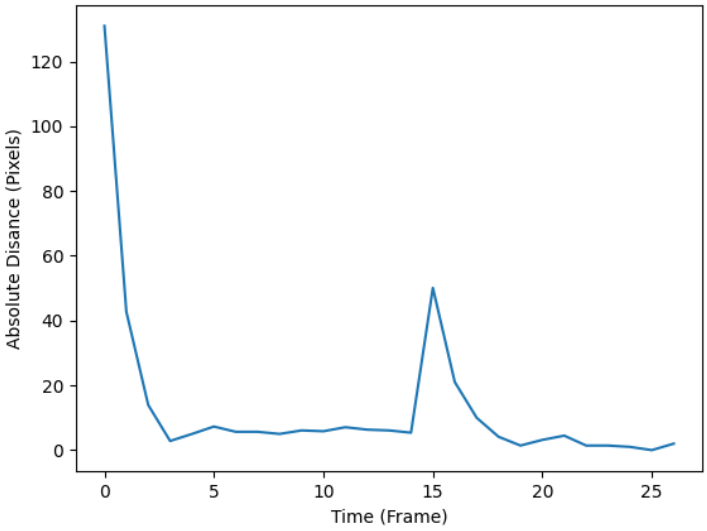

Figure 21: RMSE for KF on Break 2

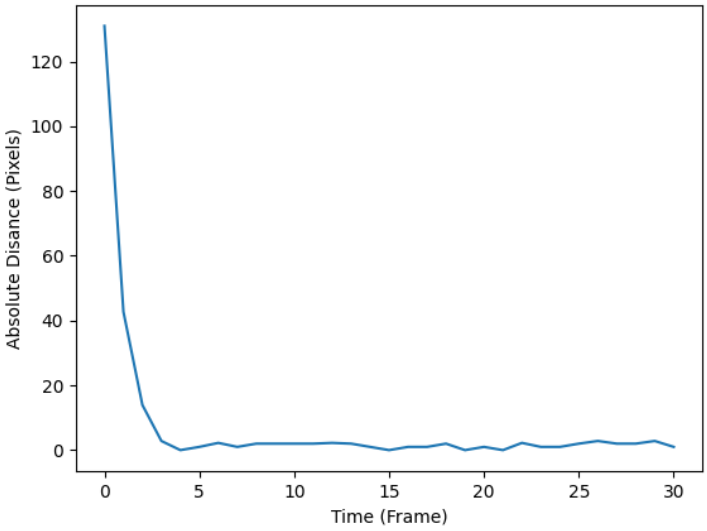

Figure 22: RMSE for SIF on Break 2

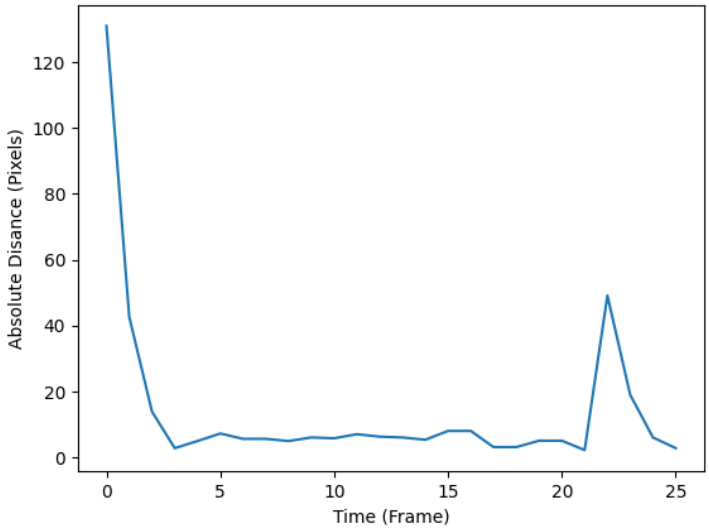

Figure 23: RMSE for KF on Break 3

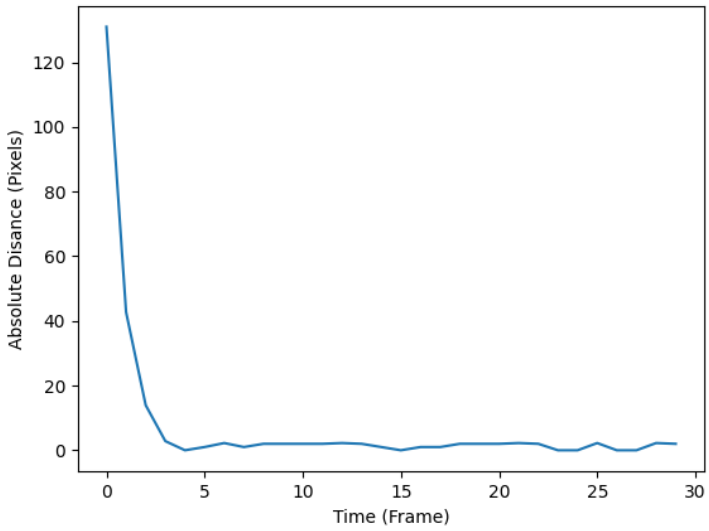

Figure 24: RMSE for SIF on Break 3

Figure 25: KF Video Tracker on Break 3

Figure 26: SIF Video Tracker on Break 3

Table 5: RMSE (Max Pixel Distance) for KF and SIF on different breaks

Break	Kalman Filter	Sliding Innovation Filter
1	2.8892	1.2950
2	2.9414	1.2304
3	3.0778	1.2030

5.4 Discussion

The results for the KF, EKF, SIF and ESIF were successful with respect to testing the filters in tandem with the object detection code. As the Kalman Filter was already implemented into the preexisting object detection framework, a few parameters were adjusted to fit the purpose of this report. The Kalman Filter results showed how well Estimation Theory techniques can improve the information gained through object detection when keeping track of separate identifications in the frame. The ability to not only predict the assumed direction of travel, but to separate different detections of the same class provides a significant amount of information that can be analyzed [11]. The other implemented filters needed to be created separately to be compatible with the preexisting framework. Most of the time spent implementing the other filters was identifying the proper dimensions of the data to be compatible with the filters. Creating the separate prediction update estimators were very simple due to their similarities to each other. The Sliding Innovation Filter was an example of this, especially due to its strong similarities to the preexisting KF [20] ,which is known to be very simple to implement and tune [32].

Implementation of the Extended Kalman Filter involved creating functions of f and h instead of implementing the system and measurement model equations. This led to

difficulties due to understanding the motion model of humans being tracked on screen. Assumptions for the system equation of motion were used to simplify the implementation of the filters. Due to these assumptions, the following RMSE values were calculated, which is shown in Table 3. As stated in Section XI. Results, the system of motion was assumed to be linear. This resulted in the linear filters, KF and SIF to outperform the nonlinear filters.

The Kalman Filter was able to correctly track the people in the provided video but seemed to fail at correctly tracking the stationary objects, such as the chair. This issue was present in the Extended Kalman Filter. The Extended Kalman Filter had the worst outcome when predicting the trajectory of the stationary objects but performed to a similar accuracy to the Kalman Filter for the moving objects. This resulted in an RMSE of 14.699 which is very close to the RMSE of the Kalman Filter. This is most likely due to the characteristics of the Extended Kalman Filter, as it is more accurate for nearly linear, nonlinear systems [32], [38].

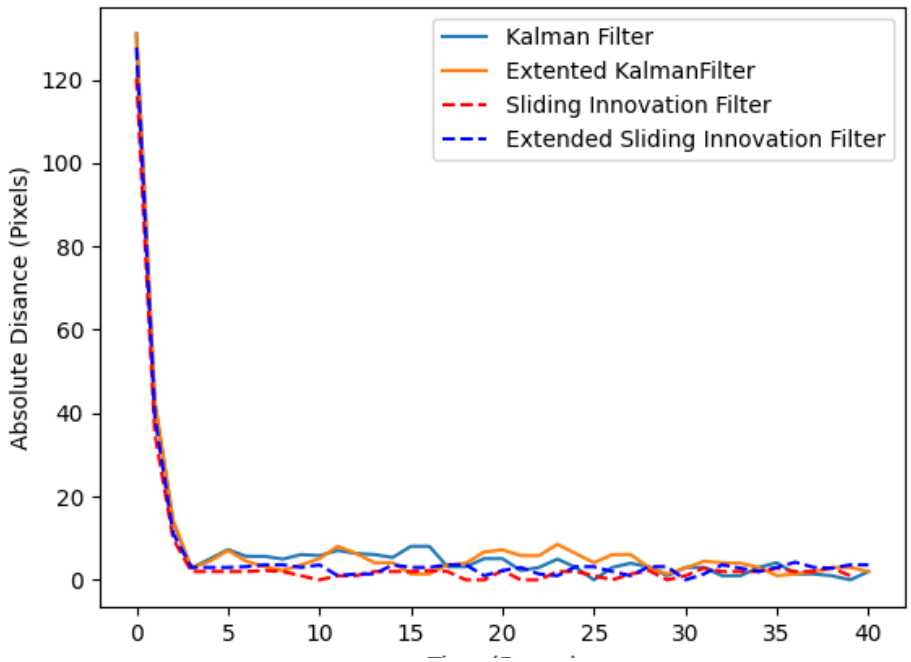

Figure 27: RMSE Graphs for all Filters

When directly comparing the resulting RMSE values for the SIF and ESIF, the clear winner for this specific test is the Sliding Innovation Filter. As the Sliding Innovation Filter was created on the basis of the Kalman Filter using the sliding boundary layer techniques from the Smooth Variable Sliding Function, it is intuitive that the SIF will greatly outperform the ESIF for linear systems. The interesting result from this experiment is that the Sliding Innovation Filter produced a very small RMSE even compared to the Kalman Filter. The implementation of the boundary layer allowed for the estimator to correctly predict the trajectory of the stationary objects as seen in Figure 13. This resulted in a much lower RMSE as the Kalman Filter failed to do this.

6 Conclusion

This thesis explored the possibilities of using Estimation Theory with innovative CNN for the purpose of detecting and tracking. The Kalman, Extended Kalman, Sliding Innovation and Extended Sliding Innovation are the five filters chosen for this research. It also served as the results of using the novel Sliding Innovation Filter as the most accurate and more dependable filter out of the existing ones. This experiment made a few things very clear. The first, is the nature of the model that is being detecting gravely effects that success of the type of filters used. For example, a linear model performs significantly better for linear filters than not, and vice-versa. However, the argument should be held that the point of this experiment was to track humans and people walking. This attempts to mimic real-world conditions and the unpredictability that a person has while walking. The EKF and ESIF all performed poorly due to the assumption that the equation of motion was assumed to be linear. The KF was able to perform as expected since it is the current standard. The KF was capable of producing accurate tracking lines for the moving objects in the frame. The newly introduced SIF performed the best. It was able to match the stability of the KF while also demonstrating its unique sliding boundary layer when creating the tracking lines. Further research should be done to incorporate the SIF for tracking purposes. Adjusting the delta sliding values for the filter can provide even better results depending on the tuning. The RMSE value was much lower, and the final output provided a much cleaner and finer image.

7 Future Work

The work presented here opened the possibilities of comparing and introducing brand new estimation filter to image detection neural networks to improve and benchmark tracking capabilities. As with any research, there is always more that can be done to further solidify the results presented. To begin, further work should be done into integrating the SIF into existing frameworks to compare how it compares with the KF. It is suggested to not completely replace the KF but instead provide a benchmark comparison between the capabilities of the two. Secondly, it can be suggested that having a framework that actually uses both filters can be beneficial for the overall results of video detection, as during disturbance cases the SIF may be able to outperform the KF. Since the SIF is still new and its footprint within the world of object detection is relatively new, it is suggested that further research is done for the tuning parameters, delta and observing if there is any bias for the algorithm on specific detected objects, such as people, cars or planes. It is recommended that future work can model different types of object detection and tracking as there is still a lot of unexplored potential when comparing the different core tracking filters between each. Additionally, this research compared each filter with the speed they required as well as a RMSE to calculate the error between each, but further exploration may be done to gain more metrics to compare between each filter.

REFERENCES

[1] C. Lexcellent, *Artificial Intelligence versus Human Intelligence*. Cham: Springer International Publishing, 2019.

[2] Statista, “Global AI software market size 2018-2025,” *Tractica*, 2020. https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/ (accessed Oct. 18, 2020).

[3] G. Tecuci, “Artificial intelligence,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 4, no. 2, pp. 168–180, Mar. 2012, doi: 10.1002/wics.200.

[4] K. Benke and G. Benke, “Artificial intelligence and big data in public health,” *International Journal of Environmental Research and Public Health*, vol. 15, no. 12. MDPI AG, Dec. 01, 2018, doi: 10.3390/ijerph15122796.

[5] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, “Neuroscience-Inspired Artificial Intelligence,” *Neuron*, vol. 95, no. 2. Cell Press, pp. 245–258, Jul. 19, 2017, doi: 10.1016/j.neuron.2017.06.011.

[6] D. E. O’Leary, “Artificial Intelligence and Big Data,” *IEEE Intell. Syst.*, vol. 28, no. 2, pp. 96–99, Mar. 2013, doi: 10.1109/MIS.2013.39.

[7] K.-H. Yu, A. L. Beam, and I. S. Kohane, “Artificial intelligence in healthcare,” *Nat. Biomed. Eng.*, vol. 2, no. 10, pp. 719–731, Oct. 2018, doi: 10.1038/s41551-018-0305-z.

[8] A. Bundy, “Preparing for the future of Artificial Intelligence,” *AI Soc.*, vol. 32, no. 2, pp. 285–287, May 2017, doi: 10.1007/s00146-016-0685-0.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[10] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003, doi: 10.1109/TPAMI.2003.1195991.

[11] S.-K. Weng, C.-M. Kuo, and S.-K. Tu, “Video object tracking using adaptive Kalman filter,” *J. Vis. Commun. Image Represent.*, vol. 17, no. 6, pp. 1190–1208, Dec. 2006, doi: 10.1016/j.jvcir.2006.03.004.

[12] Jong-Min Jeong, Tae-Sung Yoon, and Jin-Bae Park, “Kalman filter based multiple objects detection-tracking algorithm robust to occlusion,” in *2014 Proceedings of the SICE Annual Conference (SICE)*, Sep. 2014, pp. 941–946, doi: 10.1109/SICE.2014.6935235.

[13] M. X. Jiang, C. Deng, Z. G. Pan, L. F. Wang, and X. Sun, “Multiobject tracking in videos based on LSTM and deep reinforcement learning,” *Complexity*, vol. 2018, 2018, doi: 10.1155/2018/4695890.

[14] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, “What is the best multi-stage architecture for object recognition?,” in *2009 IEEE 12th International Conference on Computer Vision*, Sep. 2009, pp. 2146–2153, doi: 10.1109/ICCV.2009.5459469.

[15] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 253–256, doi: 10.1109/ISCAS.2010.5537907.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” 2016. Accessed: Oct. 18, 2020. [Online]. Available: http://pjreddie.com/yolo/.

[17] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” Apr. 2018, Accessed: Oct. 18, 2020. [Online]. Available: http://arxiv.org/abs/1804.02767.

[18] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, vol. 2017-September, pp. 3645–3649, doi: 10.1109/ICIP.2017.8296962.

[19] P. D. Wasserman and T. Schwartz, “Neural Networks, Part 2: What are they and is why is everybody so interested in them now?,” *IEEE Expert. Syst. their Appl.*, vol. 3, no. 1, pp. 10–15, 1988, doi: 10.1109/64.2091.

[20] S. Andrew Gadsden and M. Al-Shabi, “The Sliding Innovation Filter,” *IEEE Access*, vol. 8, pp. 96129–96138, 2020, doi: 10.1109/ACCESS.2020.2995345.

[21] P. A. Vela and I. J. Ndiour, “Estimation theory and tracking of deformable objects,” *Proc. IEEE Int. Symp. Comput. Control Syst. Des.*, pp. 1222–1233, 2010, doi: 10.1109/CACSD.2010.5612646.

[22] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” *Proc. - Int. Conf. Image Process. ICIP*, vol. 2016-Augus, pp. 3464–3468, 2016, doi: 10.1109/ICIP.2016.7533003.

[23] P. R. Gunjal, B. R. Gunjal, H. A. Shinde, S. M. Vanam, and S. S. Aher, “Moving Object Tracking Using Kalman Filter,” in *2018 International Conference On Advances in Communication and Computing Technology, ICACCT 2018*, 2018, pp. 544–547, doi: 10.1109/ICACCT.2018.8529402.

[24] J. Hu, H. Zhang, H. Huang, J. Feng, and G. Wang, “An improved Kalman filter algorithm base on quaternion correlation in object tracking,” *Proc. - 2012 9th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2012*, vol. 0, no. Fskd, pp. 1725–1729, 2012, doi: 10.1109/FSKD.2012.6234297.

[25] D. Jang, H. Choi, and G.-Y. Kim, “Kalman Filter incorporated Model Updating for Real-time Tracking,” *1996 IEEE TENCON - Digit. Signal Process. Appl.*, pp. 878–882, 1996, [Online]. Available: https://ieeexplore-ieee-org.subzero.lib.uoguelph.ca/stamp/stamp.jsp?tp=&arnumber=608463&tag=1.

[26] P. Jahanshahi, A. Masoud, and E. Moghadam, “Multi-view tracking using Kalman filter and graph cut,” Sep. 2015, doi: 10.1109/RIOS.2015.7270729.

[27] K. Granström, M. Baum, and S. Reuter, “Extended OBJECT TRACKING: Introduction, overview, and applications,” *J. Adv. Inf. Fusion*, vol. 12, no. 2, pp. 139–174, 2017, [Online]. Available: https://arxiv.org/pdf/1604.00970.pdf.

[28] J. M. Jeong, T. S. Yoon, and J. B. Park, “Kalman filter based multiple objects detection-tracking algorithm robust to occlusion,” in *Proceedings of the SICE Annual Conference*, Oct. 2014, pp. 941–946, doi: 10.1109/SICE.2014.6935235.

[29] M. S. Mohd Asaari, B. A. Rosdi, and S. A. Suandi, “Adaptive Kalman Filter Incorporated Eigenhand (AKFIE) for real-time hand tracking system,” *Multimed. Tools Appl.*, vol. 74, no. 21, pp. 9231–9257, 2015, doi: 10.1007/s11042-014-2078-z.

[30] K. Kondo, Y. Konishi, and H. Ishigaki, “Velocity estimation and moving object’s detection by using wavelet transform and parallel Kalman filter,” *Eur. Signal Process. Conf.*, vol. 2015-March, no. March, pp. 1–4, 2000, [Online]. Available: https://ieeexplore-ieee-org.subzero.lib.uoguelph.ca/stamp/stamp.jsp?tp=&arnumber=7075493.

[31] E. Kraft, “A quaternion-based unscented Kalman filter for orientation tracking,” in *Proceedings of the 6th International Conference on Information Fusion, FUSION 2003*, 2003, vol. 1, pp. 47–54, doi: 10.1109/ICIF.2003.177425.

[32] S. J. Julier and J. K. Uhlmann, “New extension of the Kalman filter to nonlinear systems,” in *Signal Processing, Sensor Fusion, and Target Recognition VI*, Jul. 1997, vol. 3068, p. 182, doi: 10.1117/12.280797.

[33] I. Ullah, Y. Shen, X. Su, C. Esposito, and C. Choi, “A Localization Based on Unscented Kalman Filter and Particle Filter Localization Algorithms,” *IEEE Access*, vol. 8, pp. 2233–2246, 2020, doi: 10.1109/ACCESS.2019.2961740.

[34] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *J. Fluids Eng. Trans. ASME*, vol. 82, no. 1, pp. 35–45, Mar. 1960, doi:

10.1115/1.3662552.

[35] S. H. Lee and K. S. Lim, “Indoor positioning method using BITON and linear Kalman filter,” *Soft Comput.*, vol. 22, no. 20, pp. 6741–6750, Oct. 2018, doi: 10.1007/s00500-018-3259-x.

[36] M. Bohner and N. Wintz, “The Kalman filter for linear systems on time scales,” *J. Math. Anal. Appl.*, vol. 406, no. 2, pp. 419–436, 2013, doi: 10.1016/j.jmaa.2013.04.075.

[37] M. Mangold, “Use of a Kalman filter to reconstruct particle size distributions from FBRM measurements,” *Chem. Eng. Sci.*, vol. 70, pp. 99–108, 2012, doi: 10.1016/j.ces.2011.05.052.

[38] H. Khazraj, F. Faria Da Silva, and C. L. Bak, “A performance comparison between extended Kalman Filter and unscented Kalman Filter in power system dynamic state estimation,” in *Proceedings - 2016 51st International Universities Power Engineering Conference, UPEC 2016*, Jul. 2016, vol. 2017-Janua, pp. 1–6, doi: 10.1109/UPEC.2016.8114125.

[39] S. van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: a structure for efficient numerical computation,” *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011.

[40] E. Loh, “Medicine and the rise of the robots: A qualitative review of recent advances of artificial intelligence in health,” *BMJ Leader*, vol. 2, no. 2. BMJ Publishing Group, pp. 59–63, Jun. 01, 2018, doi: 10.1136/leader-2018-000071.

[41] K. M. Alhawiti, “Advances in Artificial intelligence Using Speech Recognition,” *undefined*, 2015.

[42] A. J. Davison and D. W. Murray, “Simultaneous localization and map-building using active vision,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 865–880, Jul. 2002, doi: 10.1109/TPAMI.2002.1017615.

[43] F. Pedregosa FABIANPEDREGOSA *et al.*, “Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot,” 2011. Accessed: Oct. 18, 2020. [Online]. Available: http://scikit-learn.sourceforge.net.

[44] U. Paschen, C. Pitt, and J. Kietzmann, “Artificial intelligence: Building blocks and an innovation typology,” *Bus. Horiz.*, vol. 63, no. 2, pp. 147–155, Mar. 2020, doi: 10.1016/j.bushor.2019.10.004.

[45] X. Luo, S. Tong, Z. Fang, and Z. Qu, “Frontiers: Machines vs. humans: The impact of artificial intelligence chatbot disclosure on customer purchases,” *Mark.*

Sci., vol. 38, no. 6, pp. 937–947, Sep. 2019, doi: 10.1287/mksc.2019.1192.

[46] B. G. Buchanan, “A (Very) Brief History of Artificial Intelligence,” Dec. 2005. doi: 10.1609/AIMAG.V26I4.1848.

[47] J. Moolayil and J. Moolayil, “An Introduction to Deep Learning and Keras,” in *Learn Keras for Deep Neural Networks*, Apress, 2019, pp. 1–16.

[48] M. Abadi *et al.*, *This paper is included in the Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI ’16). TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning*. 2016.

[49] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” 2019.

[50] N. Ketkar and N. Ketkar, “Introduction to Keras,” in *Deep Learning with Python*, Apress, 2017, pp. 97–111.

[51] N. Ketkar and N. Ketkar, “Introduction to PyTorch,” in *Deep Learning with Python*, Apress, 2017, pp. 195–208.

[52] A. Ng, “What Artificial Intelligence Can and Can’t Do Right Now,” 2016. Accessed: Oct. 18, 2020. [Online]. Available: https://hbr.org/2016/11/what-artificial-intelligence-can-and-cant-...

[53] J. Manning *et al.*, “Machine-Learning Space Applications on SmallSat Platforms with TensorFlow,” *Small Satell. Conf.*, Aug. 2018, Accessed: Oct. 18, 2020. [Online]. Available: https://digitalcommons.usu.edu/smallsat/2018/all2018/458.

[54] C. P. Papageorgiou, M. Oren, and T. Poggio, “General framework for object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 1998, pp. 555–562, doi: 10.1109/iccv.1998.710772.

[55] C. Papageorgiou and T. Poggio, “Trainable system for object detection,” *Int. J. Comput. Vis.*, vol. 38, no. 1, pp. 15–33, Jun. 2000, doi: 10.1023/A:1008162616689.

[56] C. Szegedy, A. Toshev, and D. Erhan, “Deep Neural Networks for Object Detection,” 2013.

[57] D. M. Ramík, C. Sabourin, R. Moreno, and K. Madani, “A machine learning based intelligent vision system for autonomous object detection and recognition,” *Appl. Intell.*, vol. 40, no. 2, pp. 358–375, Mar. 2014, doi: 10.1007/s10489-013-0461-5.

[58] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense

Object Detection," Oct. 2017.

[59] Y. Wang, C. Wang, H. Zhang, Y. Dong, and S. Wei, "Automatic ship detection based on RetinaNet using multi-resolution Gaofen-3 imagery," *Remote Sens.*, vol. 11, no. 5, p. 531, 2019.

[60] M. Afif, R. Ayachi, Y. Said, E. Pissaloux, and M. Atri, "An evaluation of retinanet on indoor object detection for blind and visually impaired persons assistance navigation," *Neural Process. Lett.*, pp. 1–15, 2020.

[61] R. Tadeusiewicz and M. R. Ogiela, "The new concept in computer vision: automatic understanding of the images," in *International Conference on Artificial Intelligence and Soft Computing*, 2004, pp. 133–144.

[62] M. Dinham and G. Fang, "Autonomous weld seam identification and localisation using eye-in-hand stereo vision for robotic arc welding," *Robot. Comput. Integr. Manuf.*, vol. 29, no. 5, pp. 288–301, 2013.

[63] J. Fung and S. Mann, "Computer vision signal processing on graphics processing units," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004, vol. 5, pp. V--93.

[64] J. R. Parker, *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[65] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying scene breaks," in *Proceedings of the third ACM international conference on Multimedia*, 1995, pp. 189–200.

[66] D. Ribli, A. Horváth, Z. Unger, P. Pollner, and I. Csabai, "Detecting and classifying lesions in mammograms with deep learning," *Sci. Rep.*, vol. 8, no. 1, pp. 1–7, 2018.

[67] A. Barbu, Y. She, L. Ding, and G. Gramajo, "Feature selection with annealing for computer vision and big data learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 272–286, 2016.

[68] A. La Bella, P. Vaska, W. Zhao, and A. H. Goldan, "Convolutional Neural Network for Crystal Identification and Gamma Ray Localization in PET," *IEEE Trans. Radiat. Plasma Med. Sci.*, 2020.

[69] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3367–3375.

[70] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori, "A hierarchical

deep temporal model for group activity recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1971–1980.

[71] M. Babaee, D. T. Dinh, and G. Rigoll, "A deep convolutional neural network for video sequence background subtraction," *Pattern Recognit.*, vol. 76, pp. 635–649, 2018.

[72] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "Subsense: A universal change detection method with local adaptive sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, 2014.

[73] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "CDnet 2014: An expanded change detection benchmark dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 387–394.

[74] X. Wu, D. Sahoo, and S. C. H. Hoi, "Recent advances in deep learning for object detection," *Neurocomputing*, 2020.

[75] Y. Chen, C. Han, N. Wang, and Z. Zhang, "Revisiting feature alignment for one-stage object detection," *arXiv Prepr. arXiv1908.01570*, 2019.

[76] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[77] P. Soviany and R. T. Ionescu, "Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction," in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2018, pp. 209–214.

[78] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *2008 IEEE conference on computer vision and pattern recognition*, 2008, pp. 1–8.

[79] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv Prepr. arXiv2004.10934*, 2020.

[80] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*, 2016, pp. 525–542.

[81] L. Pang, H. Liu, Y. Chen, and J. Miao, "Real-time Concealed Object Detection from Passive Millimeter Wave Images Based on the YOLOv3 Algorithm," *Sensors*, vol. 20, no. 6, p. 1678, 2020.

[82] S. Geethapriya, N. Duraimurugan, and S. P. Chokkalingam, “Real-Time Object Detection with Yolo,” *Int. J. Eng. Adv. Technol.*, vol. 8, no. 3S, 2019.

[83] L. Zhao and S. Li, “Object Detection Algorithm Based on Improved YOLOv3,” *Electronics*, vol. 9, no. 3, p. 537, 2020.

[84] O. Emad, I. A. Yassine, and A. S. Fahmy, “Automatic localization of the left ventricle in cardiac MRI images using deep learning,” in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 683–686.

[85] J. Deng, Y. Lu, and V. C.-S. Lee, “Imaging-based crack detection on concrete surfaces using You Only Look Once network,” *Struct. Heal. Monit.*, p. 1475921720938486, 2020.

[86] H. Caesar, J. Uijlings, and V. Ferrari, “Coco-stuff: Thing and stuff classes in context,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1209–1218.

[87] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, “A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 27, no. 8, pp. 1861–1873, 2019.

[88] X. Zhang, W. Yang, X. Tang, and J. Liu, “A fast learning method for accurate and robust lane detection using two-stage feature extraction with YOLO v3,” *Sensors*, vol. 18, no. 12, p. 4308, 2018.

[89] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, “Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3,” in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, 2019, pp. 1–6.

[90] D. Barry, M. Shah, M. Keijsers, H. Khan, and B. Hopman, “xYOLO: A Model For Real-Time Object Detection In Humanoid Soccer On Low-End Hardware,” in *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2019, pp. 1–6.

[91] Z. Zhang and M. Sabuncu, “Generalized cross entropy loss for training deep neural networks with noisy labels,” in *Advances in neural information processing systems*, 2018, pp. 8778–8788.

[92] W. Liu, Y. Wen, Z. Yu, and M. Yang, “Large-margin softmax loss for convolutional neural networks.,” in *ICML*, 2016, vol. 2, no. 3, p. 7.

[93] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[94] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, “Densenet: Implementing efficient convnet descriptor pyramids,” *arXiv Prepr. arXiv1404.1869*, 2014.

[95] Y. Tian, G. Yang, Z. Wang, E. Li, and Z. Liang, “Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense,” *J. Sensors*, vol. 2019, 2019.

[96] R. Huang, J. Pedoeem, and C. Chen, “YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers,” in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2503–2510.

[97] W. He, Z. Huang, Z. Wei, C. Li, and B. Guo, “TF-YOLO: An improved incremental network for real-time object detection,” *Appl. Sci.*, vol. 9, no. 16, p. 3225, 2019.

[98] W. Fang, L. Wang, and P. Ren, “Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments,” *IEEE Access*, vol. 8, pp. 1935–1944, 2019.

[99] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” 2016. [Online]. Available: http://pjreddie.com/yolo9000/.

[100] J. Janai, F. Güney, A. Behl, A. Geiger, and others, “Computer vision for autonomous vehicles: Problems, datasets and state of the art,” *Found. Trends® in Comput. Graph. Vis.*, vol. 12, no. 1--3, pp. 1–308, 2020.

[101] “YOLO Creator Quits AI Research Citing Ethical Concerns.” https://analyticsindiamag.com/yolo-creator-joe-redmon-computer-vision-research-ethical-concern/ (accessed Oct. 18, 2020).

[102] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[103] D. Wu, S. Lv, M. Jiang, and H. Song, “Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments,” *Comput. Electron. Agric.*, vol. 178, p. 105742, 2020.

[104] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[105] J. Jeong, H. Park, and N. Kwak, “Enhancement of SSD by concatenating feature maps for object detection,” *arXiv Prepr. arXiv1705.09587*, 2017.

[106] Q. Zhu, H. Zheng, Y. Wang, Y. Cao, and S. Guo, “Study on the Evaluation

Method of Sound Phase Cloud Maps Based on an Improved YOLOv4 Algorithm," *Sensors*, vol. 20, no. 15, p. 4314, 2020.

[107] H. Deng, J. Cheng, T. Liu, B. Cheng, and Z. Sun, "Research on Iron Surface Crack Detection Algorithm Based on Improved YOLOv4 Network," in *Journal of Physics: Conference Series*, 2020, vol. 1631, no. 1, p. 12081.

[108] P. Mahto, P. Garg, P. Seth, and J. Panda, "Refining Yolov4 for Vehicle Detection," *Int. J. Adv. Res. Eng. Technol.*, vol. 11, no. 5, 2020.

[109] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European conference on computer vision*, 2016, pp. 354–370.

[110] X. Yi, Y. Song, and Y. Zhang, "Enhanced Darknet53 Combine MLFPN Based Real-Time Defect Detection in Steel Surface," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, 2020, pp. 303–314.

[111] C. Kumar, R. Punitha, and others, "YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2020, pp. 1316–1321.

[112] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[113] P. Singh and A. Manure, "Introduction to TensorFlow 2.0," in *Learn TensorFlow 2.0*, Springer, 2020, pp. 1–24.

[114] Ma-Dan, "Keras YoLoV4," *GitHub repository*. GitHub, 2020.

[115] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.," 2016.

[116] F. Outamazirt, L. Yan, F. Li, and A. Nemra, "Comparing between the performance of SVSF with EKF and NH for the autonomous airborne navigation problem," in *IEEE Aerospace Conference Proceedings*, Jun. 2016, vol. 2016-June, doi: 10.1109/AERO.2016.7500504.

[117] M. Ismail, M. Attari, S. Habibi, and S. Ziada, "Estimation theory and Neural Networks revisited: REKF and RSVSF as optimization techniques for Deep-Learning," *Neural Networks*, vol. 108, pp. 509–526, Dec. 2018, doi: 10.1016/j.neunet.2018.09.012.

[118] A. P. Sage and J. L. Melsa, "Estimation theory with applications to communications and control," 1971.

[119] X. Hou, Y. Wang, and L. P. Chau, “Vehicle tracking using deep SORT with low confidence track filtering,” Sep. 2019, doi: 10.1109/AVSS.2019.8909903.

[120] A. Vollant, G. Balarac, and C. Corre, “Subgrid-scale scalar flux modelling based on optimal estimation theory and machine-learning procedures,” *J. Turbul.*, vol. 18, no. 9, pp. 854–878, 2017.

[121] S. Mahfouz, F. Mourad-Chehade, P. Honeine, J. Farah, and H. Snoussi, “Target tracking using machine learning and Kalman filter in wireless sensor networks,” *IEEE Sens. J.*, vol. 14, no. 10, pp. 3715–3725, 2014.

[122] E. A. Wan and R. Van Der Merwe, “The unscented Kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, 2000, pp. 153–158.

[123] M. Simon *et al.*, “Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, p. 0.

[124] G. Kim *et al.*, “Object Tracking Method using Deep Learning and Kalman Filter,” *J. Broadcast Eng.*, vol. 24, no. 3, pp. 495–505, 2019.

[125] H. Fang, N. Tian, Y. Wang, M. Zhou, and M. A. Haile, “Nonlinear Bayesian estimation: From Kalman filtering to a broader horizon,” *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2. Institute of Electrical and Electronics Engineers Inc., pp. 401–417, Mar. 2018, doi: 10.1109/JAS.2017.7510808.

[126] S. Haykin, *Kalman filtering and neural networks*, vol. 47. John Wiley & Sons, 2004.

[127] W. Youn and S. Andrew Gadsden, “Combined quaternion-based error state kalman filtering and smooth variable structure filtering for robust attitude estimation,” *IEEE Access*, vol. 7, pp. 148989–149004, 2019, doi: 10.1109/ACCESS.2019.2946609.

[128] A. Cataford, S. A. Gadsden, and K. Turpie, “Air-LUSI: A robotic telescope design for lunar spectral measurements,” *Adv. Sp. Res.*, vol. 65, no. 10, pp. 2315–2323, May 2020, doi: 10.1016/j.asr.2020.02.014.

[129] M. Al-Shabi, K. S. Hatamleh, S. A. Gadsden, B. Soudan, and A. Elnady, “Robust nonlinear control and estimation of a PRRR robot system,” *Int. J. Robot. Autom.*, vol. 34, no. 6, pp. 632–644, Dec. 2019, doi: 10.2316/J.2019.206-0160.

[130] H. H. Afshari, S. A. Gadsden, and S. Habibi, “A nonlinear second-order filtering strategy for state estimation of uncertain systems,” *Signal Processing*, vol. 155,

pp. 182–192, Feb. 2019, doi: 10.1016/j.sigpro.2018.09.036.

[131] J. Goodman, J. Kim, A. S. Lee, S. A. Gadsden, and M. Al-Shabi, “A Variable Structure-Based Estimation Strategy Applied to an RRR Robot System,” *J. Robot. Netw. Artif. Life*, vol. 4, no. 2, p. 142, Sep. 2017, doi: 10.2991/jrnal.2017.4.2.8.

[132] R. Ahmed, M. El Sayed, S. A. Gadsden, J. Tjong, and S. Habibi, “Artificial neural network training utilizing the smooth variable structure filter estimation strategy,” *Neural Comput. Appl.*, vol. 27, no. 3, pp. 537–548, Apr. 2016, doi: 10.1007/s00521-015-1875-2.

[133] R. Ahmed, M. El Sayed, S. A. Gadsden, J. Tjong, and S. Habibi, “Automotive internal-combustion-engine fault detection and classification using artificial neural network techniques,” *IEEE Trans. Veh. Technol.*, vol. 64, no. 1, pp. 21–33, Jan. 2015, doi: 10.1109/TVT.2014.2317736.

[134] A. Gadsden, S. Habibi, D. Dunne, and T. Kirubarajan, “Nonlinear estimation techniques applied on target tracking problems,” *J. Dyn. Syst. Meas. Control. Trans. ASME*, vol. 134, no. 5, Sep. 2012, doi: 10.1115/1.4006374.

[135] S. A. Gadsden, M. Al-Shabi, I. Arasaratnam, and S. R. Habibi, “Combined cubature Kalman and smooth variable structure filtering: A robust nonlinear estimation strategy,” *Signal Processing*, vol. 96, no. PART B, pp. 290–299, Mar. 2014, doi: 10.1016/j.sigpro.2013.08.015.

[136] “mattzheng/keras-yolov3-KF-objectTracking: 以kears-yolov3做detector，以Kalman-Filter算法做tracker，进行多人物目标追踪.” https://github.com/mattzheng/keras-yolov3-KF-objectTracking (accessed Jan. 05, 2021).

[137] Panasonicsecurity, “4K camera example for Traffic Monitoring (Road) - YouTube,” Sep. 04, 2015. https://www.youtube.com/watch?v=jjlBnrzSGjc&list=PLcQZGj9lFR7y5WikozDSrdk6UCtAnM9mB&ab_channel=Panasonicsecurity (accessed Jan. 05, 2021).

[138] S.A. Gadsden, *Smooth Variable Structure Filtering: Theory and Applications*, PhD Thesis, McMaster University, 2011.

[139] H. Afshari, S. A. Gadsden, and S. R. Habibi, *Gaussian Filters for Parameter and State Estimation: A General Review and Recent Trends*, Signal Processing, Vol. 135, pages 218-238, 2017