BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING
MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING

YOLOV3 AND KALMAN FILTER BASED STAR DETECTING ALGORITHM FOR CUBESATS' STAR TRACKER SENSORS

BY

KEVSER YILMAZ

MASTER OF SCIENCE THESIS

ANKARA - 2022

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING

YOLOV3 AND KALMAN FILTER BASED STAR DETECTING ALGORITHM FOR CUBESATS' STAR TRACKER SENSORS

BY

KEVSER YILMAZ

MASTER OF SCIENCE THESIS

ADVISOR

PROF. DR. SEDAT NAZLIBİLEK

ANKARA- 2022

BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by Kevser YILMAZ, for the program of Electrical-Electronics Engineering with thesis, has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in the Institute of Science and Engineering by the following committee.

Date of Thesis Defense: … / … / ……..

Thesis Title: Yolov3 and Kalman Filter Based Star Detecting Algorithm for Cubesats' Star Tracker Sensors

Examining Committee Members	**Signature**
Prof. Dr. Sedat NAZLIBILEK (Advisor)	……..
Assoc. Prof. Dr. Selda GÜNEY (Member)	……..
Prof. Dr. Necmi ALTIN (Member)	……..
……...……	……..
……...……	……..
……..……	……..

APPROVAL

Prof. Dr. Faruk ELALDI

Director, Institute of Science and Engineer

Date: … / … / ……..

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLER ENSTİTÜSÜ

YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: … / … / 20…

Öğrencinin Adı, Soyadı : Kevser YILMAZ

Öğrencinin Numarası : 21220283

Anabilim Dalı : Elektrik Elektronik Mühendisliği Ana Bilim Dalı

Programı : Elektrik Elektronik Mühendisliği

Danışmanın Unvanı/Adı, Soyadı : Prof. Dr. Sedat NAZLIBILEK

Tez Başlığı : Yolov3 and Kalman Filter Based Star Detecting Algorithm for Cubesats' Star Tracker Sensors

Yukarıda başlığı belirtilen Yüksek Lisans/Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam ..... sayfalık kısmına ilişkin, ….. / … / 20 ..… tarihinde şahsım/tez danışmanım tarafından …… adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % …….'dır. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:…….

ONAY

Tarih: … / … / 20…

Öğrenci Danışmanı Unvan, Adı, Soyadı, İmza:

……..

To my lovely daughter Bilge Ece

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor, Prof. Dr. Sedat Nazlıbilek for his guidance and aid during all the stages of this thesis. He always encouraged and inspired me.

I am grateful to my mother, Asiye Duran, for always supporting me and her endless love. I would like to thank my sisters Emine Duran and Esma Duran for their support and motivation.

I would like to my special thanks to my one and only husband Osman Zeki Yılmaz and my dear lovely daughter Bilge Ece Yılmaz for being my source of support and motivation.

ÖZET

Kevser YILMAZ

KÜP UYDULARDA YILDIZ İZLEYİCİ SENSÖRLER İÇİN YOLOV3 VE KALMAN FİLTRESİ TABANLI YILDIZ KONUM TESPİT ALGORİTMASI

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik Elektronik Mühendisliği Anabilim Dalı

2022

CubeSat'larda tutum tahmininin yüksek doğruluğu, sistemin güvenilirliği ve CubeSat'ın görevlerini eksiksiz yerine getirmesi için çok önemlidir. Yıldız takip kameraları, uydularda yüksek doğrulukta tutum tahmini sağlamak için güvenilir seçeneklerdir. Tutum tahmininde hassas ölçümler yapabilmesi için yıldızların konum bilgisi kritik konulardan biridir. Literatürde çeşitli yıldız algılama ve konum algılama algoritmaları kullanılmaktadır. Özellikle makine öğrenmesinin gelişmesiyle birlikte geleneksel yöntemlerin yanı sıra son teknoloji algoritmalar da geliştirilmeye başlanmıştır. Literatürde makine öğrenimi kullanan yıldız tespit algoritmalarında yıldız tespitinin doğruluğu, geleneksel yöntemlere göre daha düşük kalmaktadır. Bu nedenle, bu tez, son teknoloji bir yöntem olarak yıldız tespiti için YOLOv3 derin öğrenme ağını ve yıldızların konumlarını izlemek ve olası hatalardan kaçınarak doğruluğu daha da artırmak için geleneksel Kalman filtresini kullanır. Gerçek zamanlı nesne algılama algoritmaları içerisinde yüksek hassasiyet ve hızlı çalışma yapısına sahip olan YOLOv3 algoritması, yıldız takip kameralarından gelen görüntüleri işleyerek, yıldız algılama ve konum bilgisi için konum tahmininin hassasiyetini neredeyse yüzde 96 oranında tek başına tamamlamaktadır. Ayrıca küp uydular için yıldız izleyici ile geliştirilen tutum tahmin algoritmasında YOLOv3 ağından gelen bilgiler bilinen en güvenilir tahmin yöntemlerinden biri olan Kalman filtresi ile filtrelenmekte ve doğruluk oranı neredeyse yüzde 99'a ulaşmaktadır.

ANAHTAR KELİMELER: Küp uydu, Yıldız izleyici sensör, YOLOv3, Kalman Filtresi, Gerçek zamanlı nesne takibi, derin öğrenme

ABSTRACT

Kevser YILMAZ

YOLOV3 AND KALMAN FILTER BASED STAR DETECTING ALGORITHM FOR CUBESATS' STAR TRACKER SENSORS

Başkent University Institute of Science

Department of Electrical and Electronics Engineering

2022

High accuracy of the attitude estimation in CubeSats is crucial for the reliability of the system and the complete fulfillment of CubeSat's missions. Star tracker cameras are reliable options for providing highly accurate attitude estimation on the satellites. The position information of the stars is one of the critical issues for high accuracy. Various star detection and position detection algorithms are used. Especially with the development of machine learning, in addition to traditional methods, state-of-the-art algorithms have begun to be developed. The accuracy of star detection in star detection algorithms using machine learning in the literature remains lower than in traditional methods. Therefore, this thesis uses the YOLOv3 deep learning network for star detection as a state of art method, and the traditional Kalman filter to track stars' positions and further increase accuracy by avoiding potential errors. The YOLOv3 algorithm, which has high sensitivity and fast working structure within the real-time object detection algorithms, processes the images from the star tracker cameras, increasing the sensitivity of the attitude estimation for star detection and position information by almost 99 percent. Moreover, in the attitude estimation algorithm developed with the star tracker for the cube satellites, the information coming from the YOLO network is filtered with the Kalman filter, which is one of the most reliable fractionation methods known, and the accuracy rate reaches almost 99 percent.

KEY WORDS: CubeSat, Star tracker, YOLOv3, Kalman Filter, Real-time object detection. Deep learning

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

K_k	Kalman Gain
Cal Poly	California Polytechnic State University
GPU	Graphics processing unit
R	Covariance Matrix
P-POD	Poly Picosatellite Orbital Deployer
LEO	Low Earth orbit
1U	1 Unit
2U	2 Unit
3U	3 Unit
COTS	COTS (commercial off-the-shelf)
H	Measurement Matrix
ADS	Attitude Determination System
ADCSs	Attitude Detection and Control Systems
NASA	National Aeronautics and Space Administration
Q	Covariance Matrix
INS	Inertial Navigation System
GPS	Global Positioning System
FOV	Field Of View
mAP	Mean Average Precision
Pascal VOC	Pascal Visual Object Classes
CNN	Convolutional Neural Network
RCNN Region based CNN	Region based CNN
ϕ	Transition Matrix
YOLO	You Look Only Once
YOLOv3	You Look Only Once Version 3
IOU	Intersection over union
v_k	Measurement Noise
σ	Sigmoid Function
ω_k	Process Noise

1. INTRODUCTION

CubeSat was first developed by Stanford University and California Polytechnic State University (Cal Poly). Previously, it used only in space exploration for university students due to its low cost and relatively easy design compared to other satellites. However, with the developing space technology, CubeSats started to be used widely by companies that wanted to do space-related research.

Like every spacecraft, CubeSats must be able to determine its position in space as accurately as possible in order to fully fulfill its mission. For this reason, attitude prediction is a very critical issue. Star tracker sensors are the most accurate option for CubeSats to provide the highest precision results. With the developing technology, these sensors give very high sensitivity results. However, the hardware design of these sensors is almost frozen. It is possible to increase the sensitivity rate with star detection algorithms.

The purpose of this thesis is to develop a star detection algorithm using state-of-art YOLOv3 and Kalman filter, different from traditional star detection algorithms. The YOLOv3 network, which has been widely used in real-time applications in recent years and gives very fast and high precision results, has been used in star detection. In addition, it is aimed to increase the sensitivity by using the traditional Kalman filter. Shooting star images on a single axis were created artificially and trained with the YOLOv3 network. The training was carried out using the transfer-learning method. The trained YOLOv3 network detects stars and finds the position in the form of pixels. The position information consist x, and y axis values of the image. The star positions found, on the other hand, have been smoothed with the Kalman Filter, one of the most robust estimation methods known, and a success rate of almost 99 percent has been achieved. The algorithm's codes are developed Google Colab environment. Pro extension is used for the training process because of needing for high GPU. The algorithm is developed using Python language. Figure 1.1 summarizes this process of the algorithm.

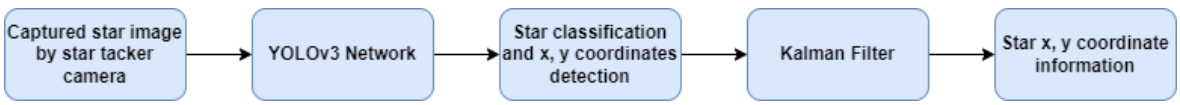


Figure 1.1. General Block Diagram of the Algorithm.

CubeSat can find its own attitude information using the reference point by following the star. Figure 1.2 shows that a star detection for attitude control using by reference point. The aim of this thesis is to provide the necessary estimation for attitude control according to the distance from the reference point by following the stars.

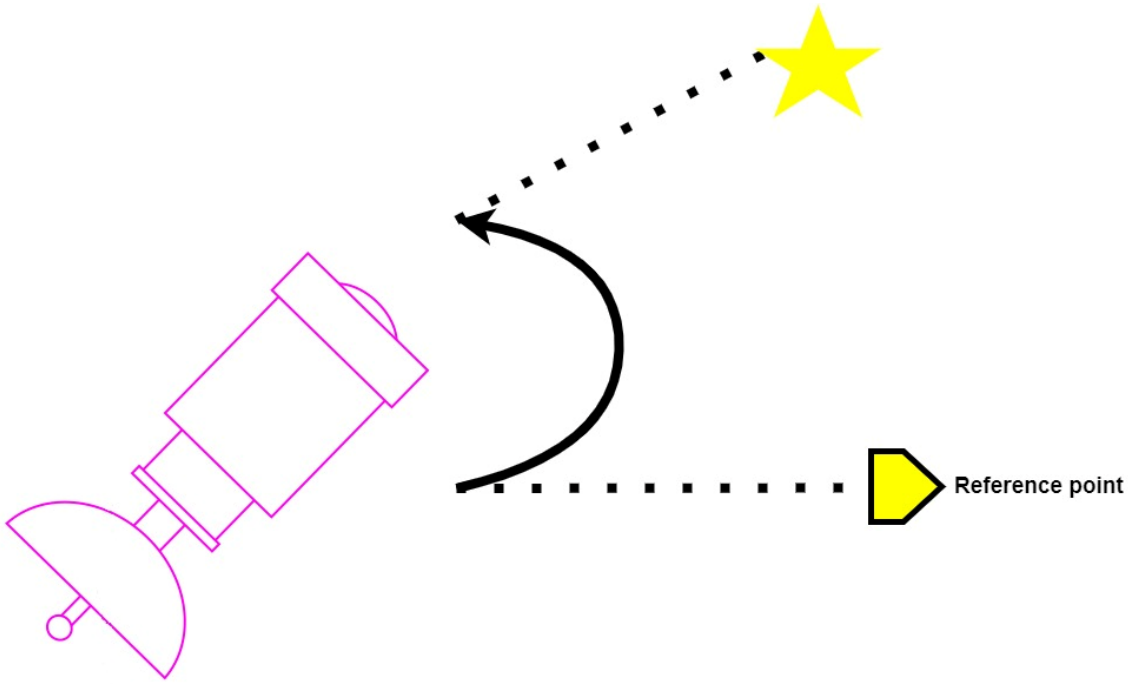


Figure 1.2. Star Detection.

The thesis consists of 5 main chapters, the first chapter includes a literature survey for star trackers and CubeSats. In the second part, the star detection and coordinate finding algorithm with YOLOv3 are explained. In the third chapter, smoothing star coordinates with the classical Kalman filter is explained. In the fourth chapter, the analysis and evaluation of the results are completed. In the fifth chapter, suggestions are made about how the proposed algorithm can be applied and improved for the conclusion and future research.

1.1. CubeSats

CubeSat is a spacecraft with design specifications developed by Stanford University and California Polytechnic State University (Cal Poly). These specifications were firstly developed in 1999. The CubeSat standards were developed for a " Poly Picosatellite Orbital Deployer " or " P-POD " launcher developed by Cal Poly. Figure 1.1.1 shows the P-POD launcher. These specifications are design standards, production processes, and system testing. The proposed CubeSat standards are for LEO and the CubeSat's main purposes are

increasing space research and easy access to space carrying small payloads. A unit (1U) CubeSat is a maximum of 1.333 kg and 10cm for every side of the cube. This is a standard 1U CubeSat. Figure 1.1.2 shows 1U CubeSat CP1 and 3U CubeSat CP10. The purpose of standardized CubeSats is appropriating for the standardized launcher. Using a standard launcher reduces risks and saves money. Because launchers are overpriced, and also debugged from possible bugs as it has been tested before.[1], [2], [3]

Figure 1.1.1. Poly-Picosatellite Orbital Deployer (P-POD) [4].

Figure 1.1.2. 1U and 3U CubeSats [5].

CubeSats becomes more popular day by day due to its advantages such as low-cost platforms, easy integration, small payloads, and timesaving in design. As mentioned in the paragraph above, standardizing CubeSats, and standardizing the launchers used in CubeSats is very helpful in saving time during the design phase. While Cube Satellites were mostly developed by university students, nowadays, they have been widely used by researchers and companies who want to conduct space exploration, due to the above-mentioned advantages. The spread of space-related studies on many different subjects, and the developments in camera and image processing have led to an increased interest in CubeSats. In addition,

developments in recent years such as the wide variety of COTS (commercial off-the-shelf) products, growing accessibility, small area coverage, and affordable prices have contributed greatly to the increase in CubeSat applications. With the increasing application areas, some subsystems become critical for CubeSats to perform their missions correctly.[1]

Like any spacecraft, a CubeSat must be able to determine its current position in space to accurately perform its mission. The subsystem that realizes this is the attitude determination system, known as ADS for short. Attitude estimation algorithms play a key role for the CubeSats. Because attitude control directly depends on the attitude estimation. The more accurate the attitude estimation, the more successful the attitude control will be, which will make perfect the purpose that the satellite will directly serve. The most basic and important factor that determines the accuracy of attitude determination is sensor selection. In order to make our proposed attitude determination algorithm more sensitive, star trackers, one of the most sensitive ADS sensors known, are used. [2]

1.2. CubeSat Attitude Determination Devices

Attitude detection and control systems (ADCSs) are one of the most important subsystems for any spacecraft to properly achieve its missions. ADCS consists of two main parts. The first part is ADCS components for CubeSat compatible and second part is attitude determination and control algorithms. ADCS sensors have been miniaturized by the dimensions of the CubeSats. These sensors were photodiode, magnetorquer reaction wheels, magnetometers, magnetometers, sun sensors, horizon sensors, star trackers. Sun and horizon sensors will not be enough to increase attitude estimation accuracy for CubeSats. Because these sensors depend only on the sun and horizon vectors. A CubeSat's ADCS requires the use of a star tracker to operate with high precision. [6]

1.3. Star Tracker

Finding a location has been a curious subject for humanity for many years. Since it was discovered that it is possible to find positions and determine directions with celestial bodies, and stars have been followed. The first sextant by John Bird in 1757 could measure the Earth's horizon and the angles of celestial bodies, and in 1437 it could be designed by Hipparchus as the first star to contain 992 stars. The latest technologies, a more advanced star option developed from observations from the sky. One of the most popular star catalogues Tycho-2 contains 2.5 million brightest stars in the sky. The first star sensor was

developed by NASA for use on Mars missions. With the development of star catalogs and star sensors, the use of this sensor in satellites and spacecraft has become widespread and reliability has increased.[13]-[19]

In recent years, the autonomous capabilities and other capabilities of star tracking cameras have improved significantly. Figure 1.3.1 represents a star tracker which is CubeStar. Many spacecraft use star trackers to increase attitude determination accuracy. They are designed for:[7] [8] [9] [10] [11]

- no support needed for attitude angle and ratio estimation
- working in various mission circumstances
- finding fast and autonomously solutions to the lost-in-space problem
- to provide angular velocity information to the system in difficult conditions where estimating an orientation for spacecraft is not possible

Basically, the star tracker works as an electronic camera connected to the computer. Stars are found and identified by processing the perceived image of the sky. After detecting stars from the star tracker cameras, an estimation of the satellite's attitude according to the star catalog. Star trackers have made notable improvements lately. First-generation star trackers detected only a few bright stars and send to the spacecraft's computer the data of focal plane coordinates. Since this coordinate data is not related to the inertia space, direct attitude information could not be obtained. Improvements on star tracker sensors, allow to obtain direct attitude information. Hardware parts' improvements of star trackers have reached a certain point today. Increasing the sensitivity of star trackers is now possible with star detection algorithms. Thanks to, the developments in space-qualified microcomputers in recent years, information about stars has begun to be stored. This has led to the development of a wide variety of algorithms. In order to increase the sensitivity and reliability of the new generation star trackers, it is necessary to develop algorithms instead of hardware. Therefore, the algorithm we proposed in this thesis can be used in new generation star tracker cameras with speed and accuracy advantages. [9],[10],[11]

Figure 1.3.1. CubeSat Star Tracker Sensor, CubeStar. [12]

Over time, many studies have been done on the integration of inertial navigation system (INS), GPS and star sensor and different techniques have been tried to be developed. One of these studies is the integration of INS and star sensor in the article published by Ali and Fang in 2004. Later, in the article published by Ali and Fang in 2006, INS and star tracker with Kalman Filter is discussed. This article is important in terms of using Kalman Filter and speed for error compensation. [20]-[24] In 2008, Xu and Fang suggested the integration of INS and CNS using neural networks. [25] Figure 1.3.2 represents a star tracker simulated image captured from Stellarium. Figure1.3.3 represents the summarized general parts of a autonomous star tracker.

Figure 1.3.2. Star Tracker simulated Image from Stellarium.

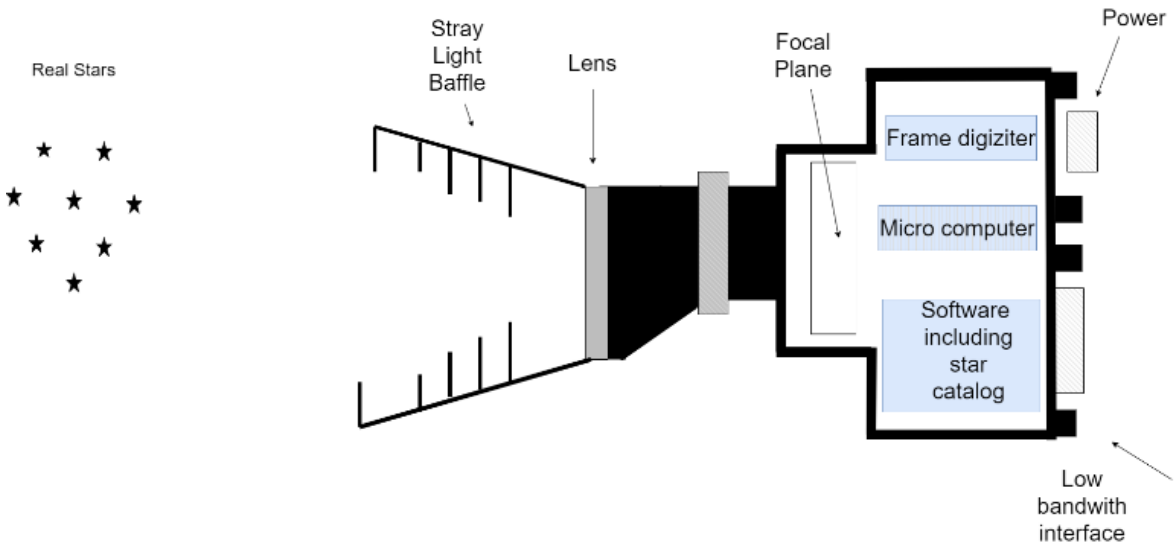


Figure 1.3. 3. Sketch of a general autonomous star tracker [11].

Star tracker's performance is affected by the following factors: [26]

- Field of view (FOV)
- Star detection threshold
- Starlight detecting
- Algorithm of star identification
- Star catalog
- Calibration

Firstly, the star image captured by the star tracker device (CCD or CMOS) the centroids are detected according to the illumination intensity of stars. The centroiding algorithm's purpose is detecting positions of the stars in the image from the star tracker device. After the centroiding, the process continues with star identification. This step is called the star matching algorithm. The centroiding algorithms are discussed the following subtitle.[27]

1.4.Literature Survey About Star Detection Algorithms

Star tracker algorithms' first step is detecting stars and their locations on the image. This step is vital for attitude estimation and control. To identify the stars correctly, the star

detection algorithm must be able to obtain high sensitivity and real-time system synchronization speed.

There are two different traditional star detection methods. These traditional methods are the image moment analysis method and point spread function method. In the image moment analysis method, there is a threshold value determined by the brightness. All objects above this threshold value are detected as stars. However, non-star objects such as planets can be detected as stars. Therefore, this method is not a robust method. Point spread function method detects the stars above the threshold. This threshold value is obtained according to background noise level's standard deviation. This method allows to true star detection when compared with image moment analysis. On the other hand, point spread function method cannot detect all stars on the image.

There are several deep learning-based star detection and identification algorithms. Neural networks were first used in star detection in 1989. After that, an algorithm with fuzzy logic and neural networks to identify stars was proposed by Hong in 2000. It was proposed by Jiang in 2012 by combining neural networks with the triangle and grid algorithm. In 2019, this algorithm, called RPNet by Xu, Jiang and Liu, is based on representation learning. In 2020, Rijlaarsdam proposed another star recognition neural network algorithm. Another algorithm proposed in 2021 is the one-dimensional CNN based star identification algorithm by Bendong Wang and his friends. [28]-[33]

All the methods are developed that traditional methods and deep learning-based star detection algorithms have some deficiencies. In traditional star detection algorithms, image moment analysis method detects too inaccurate stars and point spread function cannot detect all stars. In addition, deep learning-based algorithms recommended so far are not optimal options in real-time systems due to speed or large memory occupancy. This thesis studies, the YOLOv3 and Kalman Filter star detection algorithm detects all stars no more or less stars. Besides this algorithm offers high accuracy and fast execution. The fact that YOLOv3 is very successful in detecting small objects, and the Kalman filter is the most robust filter known, has made this study successful.

2. DETECTION BY STAR TRACKER USING YOLOV3

In this chapter, deep learning object detection algorithms, YOLOv3 architecture, and proposed YOLOv3 star detection algorithm are explained. As a mentioned previous chapter, with the increasing popularity of CubeSats, attitude determination has become one of the most important issues. Star trackers are a reliable option for CubeSat attitude determination with their high accuracy in attitude determination. Detection of stars is one of the most important factors affecting the accuracy of attitude estimation. In respect thereof, the object detection and recognition issues become important.

Figure 2.1 represents block diagram of the algorithm. First step of the algorithm is preparing star dataset. In this study, dataset created artificially for shooting star on a single axis. After this step train dataset, validation dataset and test datasets were divided according to Pareto principle. Second step is annotation these datasets according to Pascal VOC format.

One of the most important steps is selecting the most accurate model. Training models are obtained using transfer learning approach. Transfer learning approach is increase accuracy and saves development time. The training model is decided by mAP and loss values. After training model images have applied this model. Star classification and location information are obtained from the YOLOv3 model.

The main second part of the algorithm is Kalman Filtering. This part's aim is smoothing the results of the YOLOv3 network. In this algorithm star's velocity is assumed constant velocity.

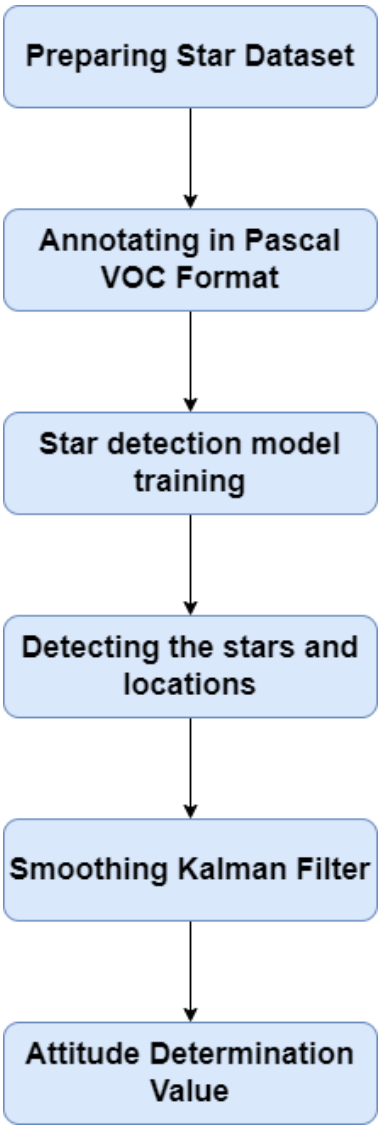


Figure 2.1. Block Diagram of Algorithm.

2.1.Deep Learning Object Detection Algorithms

In the last decade, real-time object detection algorithms including artificial neural networks have made enormous progress. Performing object detection in machine learning, localizing, and recognizing are the focus issues in order to be able to distinguish objects. Object detection methods can be surveyed under the headings of machine learning-based and deep learning-based approaches. Deep learning-based approaches realize end-to-end object detection through convolutional neural networks. [34]

Object detection algorithms can basically divide into two main classes. These are one-stage object detection algorithms and two-stage object detection algorithms. When we compare these algorithms, the main difference between them is the generation of region

proposal. In single-stage object detection algorithms, there is no need for a region proposal. The coordinate and accuracy information of the object can be obtained directly. Thus, single-stage object detection algorithms come to the fore with their high extraction speed. Two-stage object detection algorithms create region proposal before classification and positioning processes. The impressive feature of two-stage object recognition algorithms is that they can achieve high accuracy in location determination and recognition.[34]

The most popular CNN-based real-time image processing algorithms are R-CNN algorithm and R-CNN variants, YOLO algorithm and YOLO variants. R-CNN methods are based on two-stage detection and YOLO methods are based on one-stage detection. R-CNN does not work on the entire image. The image is divided into regions and then classification is performed. YOLO methods have no need region proposal. Hence YOLO methods are faster than R-CNN methods. Faster R-CNN is an algorithm that has emerged by replacing the region search used in R-CNN with the object detection algorithm. [35][36][37][38]

Bilel Benjdira and his friends, in their research comparing YOLOV3 and faster-RCNN, found that both algorithms outperform high accuracy, but YOLOV3 outperforms more sensitive results. In addition, YOLOv3 has faster results in the processing time. [39]

After the literature survey, it was decided to use the YOLOV3 network in this study due to the advantages of high precision performance, success of small object detection and fast performance of YOLOV3.

2.2. YOLOv3 Architecture

In 2016, Joseph Redmon presented a different approach with the YOLO algorithm. YOLOv3 is one of the state-of-art and sensitive algorithms of these deep learning techniques. Unlike CNN algorithms developed with regional approaches, YOLO (You Only Look Once) algorithm passes the image once through a fully convolutional neural network (FCNN). This approach makes the YOLO algorithm real-time and faster than others.[40]

In 2018, Joseph Redmon and Ali Farhadi presented YOLO version 3. Yolov3's approach for the object detection is different from other versions. It accepts the object detection process is a regression case. YOLOv3 use just one feed-forward CNN. YOLOv3 use just one feed-forward CNN. It directly estimates box offsets and probabilities. Thus, it achieves all the processes required for end-to-end detection in a single network. [41]

YOLOv3's detection processes are summarized:

- Dividing the input image according to S x S grid cell:

  All grid cells are responsible for detecting that an object's center falls off its own cell.

- Predicting B bounding boxes and scores of confidences for those boxes.

- Obtaining with confidence scores for object detecting:

  Confidence is defined in 2.1

$$Confidence_j^i = Pr_{i,j}(Object) * IOU_{pred}^{truth} \quad (2.1)$$

  Pr_{object}= Function of the object

  i = ith grid cell

  j = jth bounding box

  IOU = Intersection over union

  Truth = Ground truth

  Pred = Predicted box

The confidence score should be zero when the cell does not contain any object. The confidence score should be one when the cell contains an object.

- YOLOv3's method uses to calculate a part of loss function, truth objectness scores and binary cross-entropy predicted objectness scores. It is expressed as equation 2.3.

$$E_1 = \sum_{i=0}^{S^2} \sum_{j=0}^{B} w_{ij}^{obj} [\hat{C}_i^j log(C_j^i) - (1 - \hat{C}_i^j) log(1 - C_i^j)] \quad (2.2)$$

  S^2 : Grid cells numbers

  B= Bounding box numbers

$\hat{C}_i^j$ = Predicted objectness score

C_i^j = Truth objectness score

- Detecting and localization of an object:

Bounding box position and dimension calculations are related to following parameters which are t_x, t_y, t_w, t_h . Width, height and x and y coordinate of bounding box calculations as shown in Figure 2.2.1. The box's x and y coordinates are b_x, b_y respectively. The box's width and height values are b_w, b_h respectively.

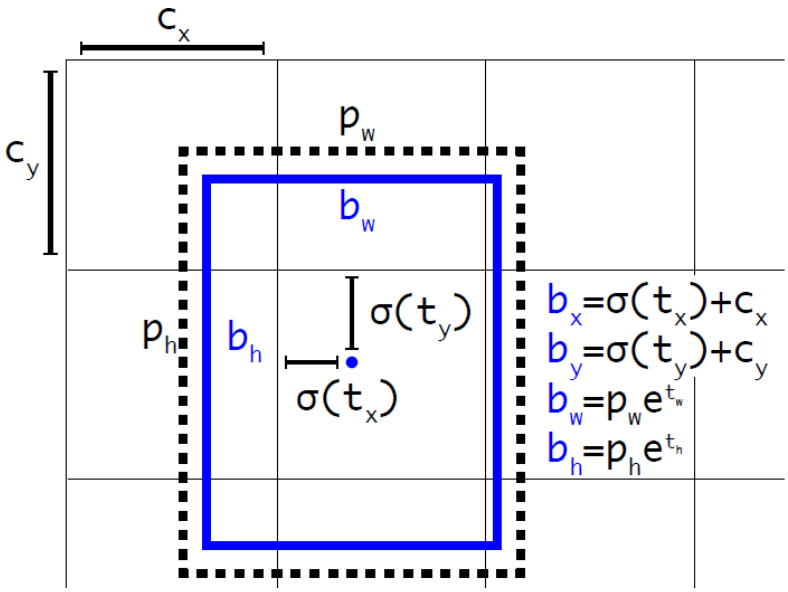


Figure 2.2.1. YOLOv3 Bounding Box Calculations.

- The ground truth box's parameters are g_w, g_h, g_x,g_y and truth values can be expressed as follows:

$$\sigma(\hat{t}_x) = g_x - c_x \tag{2.3}$$

$$\sigma(\hat{t}_y) = g_y - c_y \tag{2.4}$$

$$\hat{t}_w = log(g_w/p_w) \tag{2.5}$$

$$\hat{t}_h = log(g_h/p_h) \tag{2.6}$$

- YOLOv3 method uses sum of the square error loss for coordinate estimation.

$$E_2 = \sum_{i=0}^{S^2}\sum_{j=0}^{B} w_{ij}^{obj}[\,(\sigma(t_x)_i^j - \sigma(\hat{t}_x)_i^j)^2 + \sigma(t_y)_i^j - \sigma(\hat{t}_y)_i^j)^2] + \sum_{i=0}^{S^2}\sum_{j=0}^{B} w_{ij}^{obj}[\,(\sigma(t_w)_i^j - \sigma(\hat{t}_w)_i^j)^2 + \sigma(t_h)_i^j - \sigma(\hat{t}_h)_i^j)^2] \quad (2.7)$$

YOLOv3 uses 53 layered Dark net-53 architecture which is presented in Table 2.2.1. This network was created with the hybrid approach of the Darknet-19 network and the new network. Darknet-53 uses sequential 3x3 and 1x1 convolution layers, but there are some shortcut links. Darknet-53 is enormous, larger, efficient, and faster when compared with other backbones. One of the improvements in the use of the feature pyramid networks.

YOLOv3 is more effective because of:

- Faster than previous real time object detection algorithms
- High accurate estimations from a single network

YOLOv3 is different from the other versions. These differences are:

- Using binary cross entropy

  Instead of the mean square error, YOLOv3 uses binary cross entropy

- Bounding box prediction

  YOLOv3 uses diverse bounding box prediction. It uses objectness score and threshold.

- Pyramid networks

  One of the improvements is the use of feature pyramid networks. This improvement is caused to the change for tensor dimensions.

- Darknet-53

  3x3 and 1x1 CNN layers are used. Darknet 53 allows to obtain a faster and more accurate architecture.

To sum up, YOLOv3 is a good option for an orientation (attitude) determination algorithm with star trackers with both speed and accurate.

Table 2.2.1. Darknet-53 Network Architecture.

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

2.3. Star Detection and Localizing Algorithm Model for CubeSats

The proposed algorithm is a star tracker attitude determination model for the CubeSats by artificial intelligence and robustness. This algorithm is developed by using the Google Colab environment. Google Colab's Pro extension used to access to GPU easily for the training process and, code was developed codes by using Python language. The algorithm is implemented with single axis positioning. The algorithm basically consists of two main parts. These are:

- First Part: Star detection and star localizing algorithm with YOLOv3
- Second Part: Tracking / Smoothing algorithm with the Kalman filter.

This subtitle focus on the first part of algorithm which is star detection and star localizing step. Figure 2.3.1 represents the architecture of the algorithm.

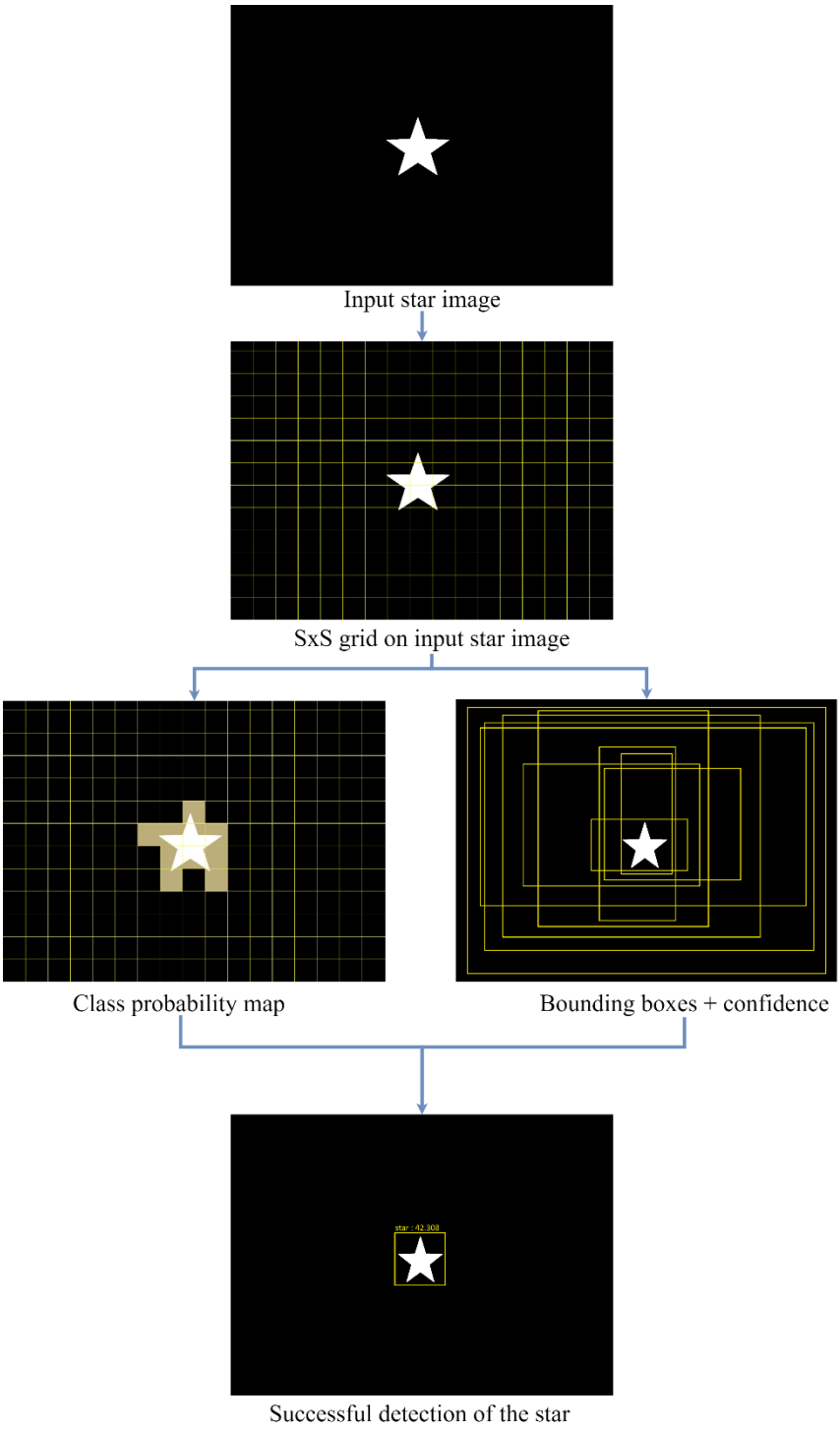


Figure 2.3.1. Star detection with YOLOv3 architecture.

Figure 2.3.2 represents the YOLOv3 block diagram of the algorithm. Star dataset was divided three parts. These are training dataset, validation dataset, and test dataset. After the splitting datasets, stars are labelling on these images by using LabelImg tool. Then, YOLOv3 pre-training approach give train models. After, obtaining the training model, using test

images star detection and localization results obtained. In the subtitles of this section, it is aimed to explain these steps of the first part of the algorithm.

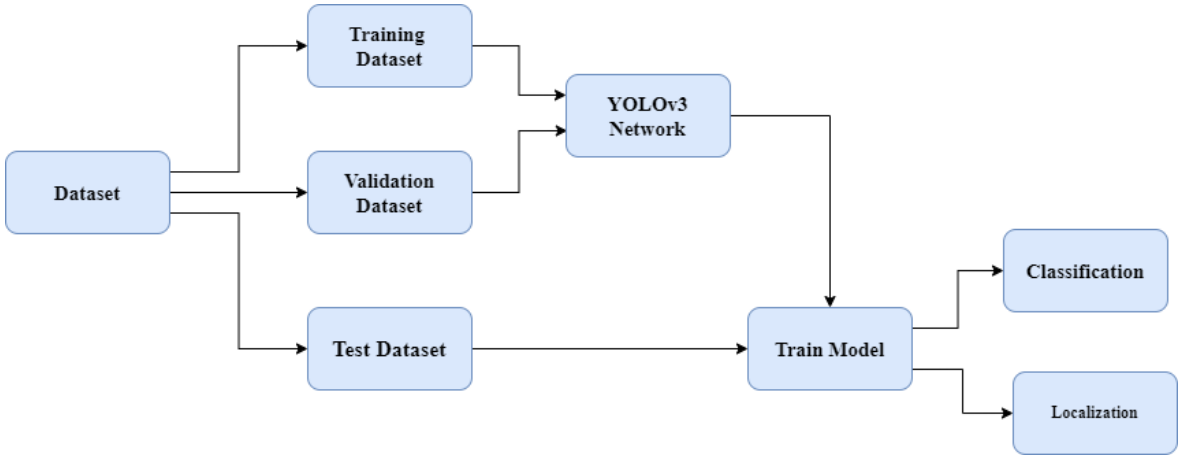
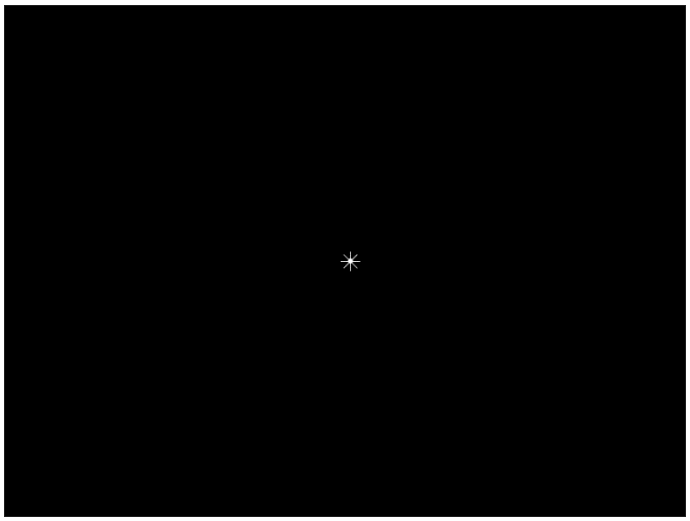


Figure 2.3.2. YOLOv3 Star Classification and Localization.

2.3.1. Preparing Star Dataset

Preparing a dataset is the most important step because of the effect on the training model. While preparing or collecting the dataset, all images must have the same extension (for example, all of them should be .jpg, .jpeg, .png) so that the training model can be created properly. All images were saved with .png extension. In this training process, simulated images were used. This dataset artificially consists of equally spaced shooting star images between 0 and +875 pixels. The dataset consists of a total of 501 images (between -255 and +255) and shooting star images at 1.75-pixel intervals (calculated by rounding as there will be no decimal pixels). First, middle, and final images are presented in Table 2.3.1.1. Images have a star, and this star is moving on single axis. And also, Figure 2.3.1.1 represents the shooting star in a single axis.

Table 2.3.1.1. Moving a star from dataset.

1st image	
256th image	
501th image	

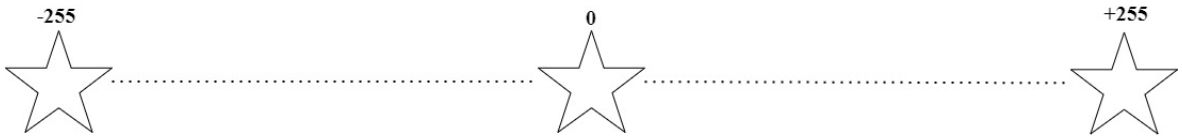


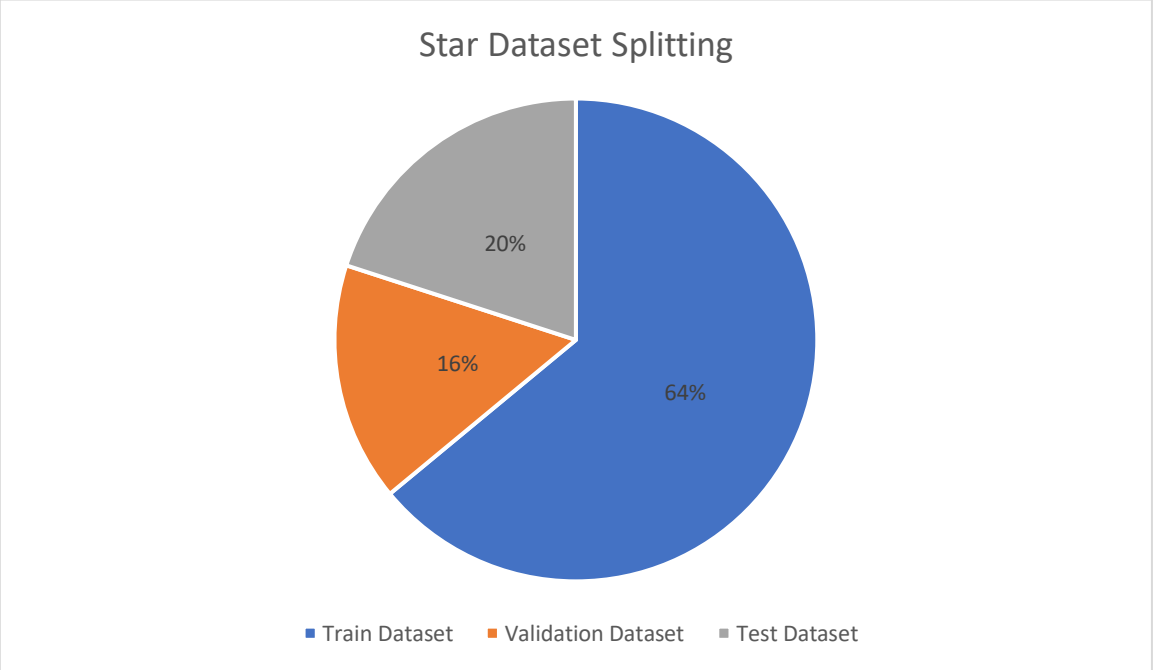
Figure 2.3.1.1. Moving on a single line star.

After the dataset is collected, the next step is to divide the dataset into train, validation, and test datasets. While preparing validation and train datasets, the Pareto principle was taken as a reference. This principle is known as the 80/20 rule. According to this principle, 80 percent of the data set should be used for training and 20 percent for test. For validation set divide again training set, now 80 percent of them train and rest of them validation. In this model, 500 images are used for training. These images are separated train and validation. 80% (400 images) of the images are used for training and 20% (100 images) of the images are used for test. After this, again splitting 401 images, 321 images train, 80 images for validation part are selected from the dataset randomly. The training and validation numbers and percentages of the images used are shown in Table 2.3.1.2 and Table 2.3.1.3 respectively. [42]

Table 2.3.1.2. Dataset separation using by Pareto Principle.

Number of Total Images	Training Set	Validation Set	Test Set
501	321	80	100

Table 2.3.1.3. Star Dataset Splitting Ratios.



The three main steps performed in this section are as follows. The following steps summarize the preparation dataset process:

- Collecting dataset with the same file extension
- Collecting/creating enough images

Separating train, validation, and test from the dataset

2.3.2. Annotating in Pascal VOC Format

Pascal VOC and Microsoft COCO annotating formats are the most popular formats. In this model, images are annotated using Pascal VOC format because the libraries (eg. imageAI) used in this algorithm and YOLOv3 algorithm are compatible with the Pascal VOC format. After the preparing dataset, the star object was annotated with a bounding box for each image.

The LabelImg which is popular for annotation tool was used to add annotations according to Pascal VOC format. LabelImg allows for tagging object bounding boxes on images and it supports formats of YOLO and CreateML. Figure 2.3.2.1 represents the LabelImg tool labelling star image with box. After tagging objects of the images XML and image extension files are generated. These files are created both train and validation part.

When completing this step, dataset is ready for the create training models. Datasets were created as follows:

Train dataset:

>>Star_1.png, Star_5.png, Star_15.png…etc. (images)

>> Star_1.xml, Star_5. xml, Star_15. xml …etc. (annotations)

Validation dataset:

>>Star_2.png, Star_7.png, Star_10.png…etc. (images)

>> Star_2.xml, Star_7. xml, Star_10. xml …etc. (annotations)

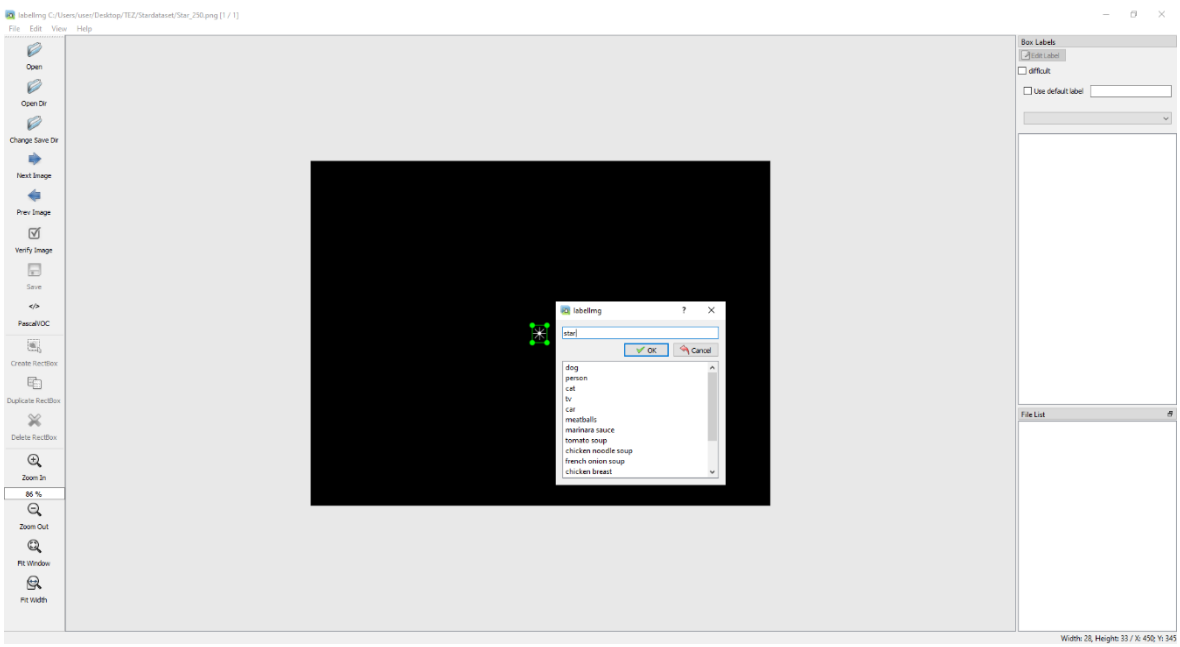


Figure 2.3.2.1. LabelImg Star Annotating.

2.3.3. Star Detection Model Training

Transfer learning is a widely used process where models trained for a purpose on large datasets can be reused for training and testing on small datasets for some purpose. The transfer learning (pre-trained model using) method is widely used because of the advantages it provides in computer vision. The most important of these advantages is not having to obtain a new model starting from scratch with the model trained on a large dataset that has

been trained before. Thanks to this advantage, the performance of the algorithm directly increases. Also saving time is another important benefit from this advantage supplied. While developing this algorithm, the transfer learning method was used. In this algorithm ImageAI is used, and this library allows to this method. The YOLOv3 pre-trained model is embedded in the code and there is no need to rearrange parameters such as weights. Therefore, this method improves accuracy and saves algorithm development time. [43][44][45]

While model training used parameters as follows:

- Batch_Size: 4
- Num_experiments: 100 (number of iterating)
- Object_array: “Star”
- Pretrained_model: Importing pretrained model

2.3.4. Detecting Stars and Their Locations

After the training models (which are obtained from the training process), evaluation the appropriate model is selected with respect to the mean average precision (mAP) and loss values. Among the models produced in the previous step, the model with a high mAP value and low loss value is usually the model that will get the most accurate result. However, since it is possible to see a low loss value in cases such as overfitting, the resulting models should be examined very carefully. The highest mAP value was equal to 0.95, and this model used for the test images.

When the test images are sent to the model selected according to the mAP and loss values, the trained model could obtain star detection and the x, y coordinate information of the stars on the image. This coordinate information is in pixel format. All test images’ stars and their coordinate information were successfully obtained.

3. KALMAN FILTER

R.E. Kalman's famous paper was published in 1960, that is describing self-iterative or recursive solutions to the problem of discrete-data linear filtering. Since its publication, the Kalman filter has become a crucial research and application for autonomous and assisted navigation fields. In this research, developed algorithm's Kalman Filtering part assists to obtain more accurate results for star coordinate information. This chapter aimed to the Kalman Filter basics and smoothing with Kalman Filter.[46] [47]

The Kalman filter provides a mathematical approach to the discrete data linear filtering problem using the least-squares method. The Kalman filter, which has many applications, is very successful in predicting the response of the system for the past, present and future, even if the characteristics of the system are not known precisely. In real life, since the measured data from any sensor does not reflect the real value because of the noise or other unwanted situations, it is aimed to approximate the measured value to the real value by using the Kalman filter. In short, the Kalman filter is an iterative set of computations with feedback that estimates metrics.

The application area of the Kalman filter is quite wide, some of them are as follows.

- Tracking objects
- Navigation
- Sensor fusion
- Econometrics

3.1. The Discrete Kalman Filter Algorithm

The Kalman filter predicts a process using a feedback control form. The filter predicts a process state within a given time frame and then continues to feed on noisy data from the sensors. Therefore, Kalman filter equations can be examined into two parts as follows:

- Time update equations
- Measurement update equations

The time update equations forward the covariance estimates error and current state to obtain an a priori estimate from time k to k+1 time steps. Thus, it tries to catch the most

accurate prediction possible. For clarity, we can also call measurement update equations corrective equations. Measurement update equations provide feedback. Incorporates the new measurement into priori estimate to obtain a more accurate posteriori estimate. For clarity, we can also call the Time update equations as predictive equations. In fact, the final estimation algorithm is very similar to the estimator and corrective algorithm used for solving a numerical problem, as shown in Figure 3.1.1.

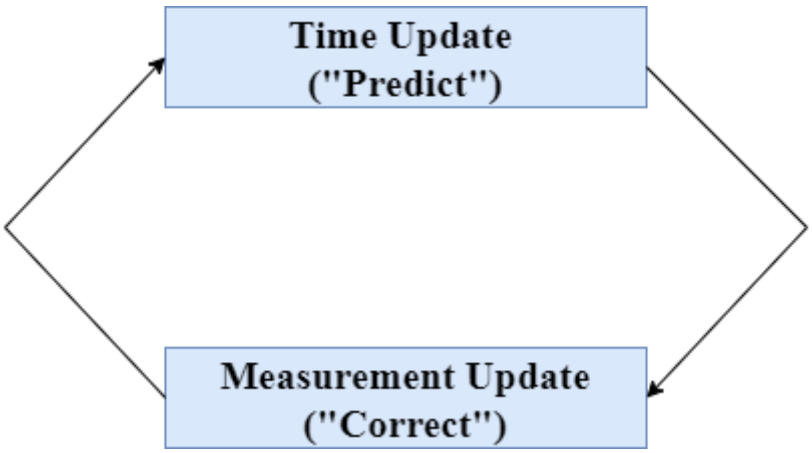


Figure 3.1.1. Continuing Discrete Kalman Filter Cycle.

The time update predicts the current state estimate. The measurement update specifies the predicted estimate with an actual measurement at that time.

Equations specific to time and measurement updates:

Table 3.1. 1. Time Update (Prediction).

Project the state ahead

$$\hat{x}_{k+1}^{-} = A_k \hat{x}_k + B u_k \quad (4.1)$$

Project the error covariance ahead

$$P_{k+1}^{-} = A_k P_k A^T + Q_k \quad (4.2)$$

Table 3.1.1 reflects; from k time steps to k+1 steps time update the state and covariance estimates. A_k, B are from the following equation 4.3:

Linear stochastic difference equation at time k+1 and k:

$$x_{k+1} = A_k x_k + B u_k + w_k \quad (4.3)$$

Q_k is comes from the following equation 4.4:

ω_k represents the process noise and v_k represents the measurement noise.

$$p(\omega) \sim N(0, Q) \tag{4.4}$$

$$p(v) \sim N(0, R) \tag{4.5}$$

Table 3.1. 2. Measurement Update (Correction).

Compute the Kalman gain

$$K_k = P_k^- H^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{4.6}$$

Update the error covariance

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H_k \hat{x}_k^-) \tag{4.7}$$

Update the error covariance

$$P_k = (I - K_k H_k) P_k^- \tag{4.8}$$

The first step, when measurement update is to calculate the Kalman gain, K_k. The second step is to real measure the process to get the z_k specified in the equation 4.9 below, followed by constructing a state estimation of posteriori by including the measurement as in (4.7). Finally, to find posteriori covariance estimate use equation 4.8 from Table 3.1.2.

$$\dot{z_k} = H_k x_k + v_k \tag{4.9}$$

After each update cycle time and measurement, the process of the Kalman filter continues to repeat with previous a posteriori estimates to reflect or predict new a priori estimates. The Kalman filter allows for easier use in practice. Because it works on the past data and the current data instead of working on all the data. Figure 3.1.2 below represent a predict and correct flowchart.

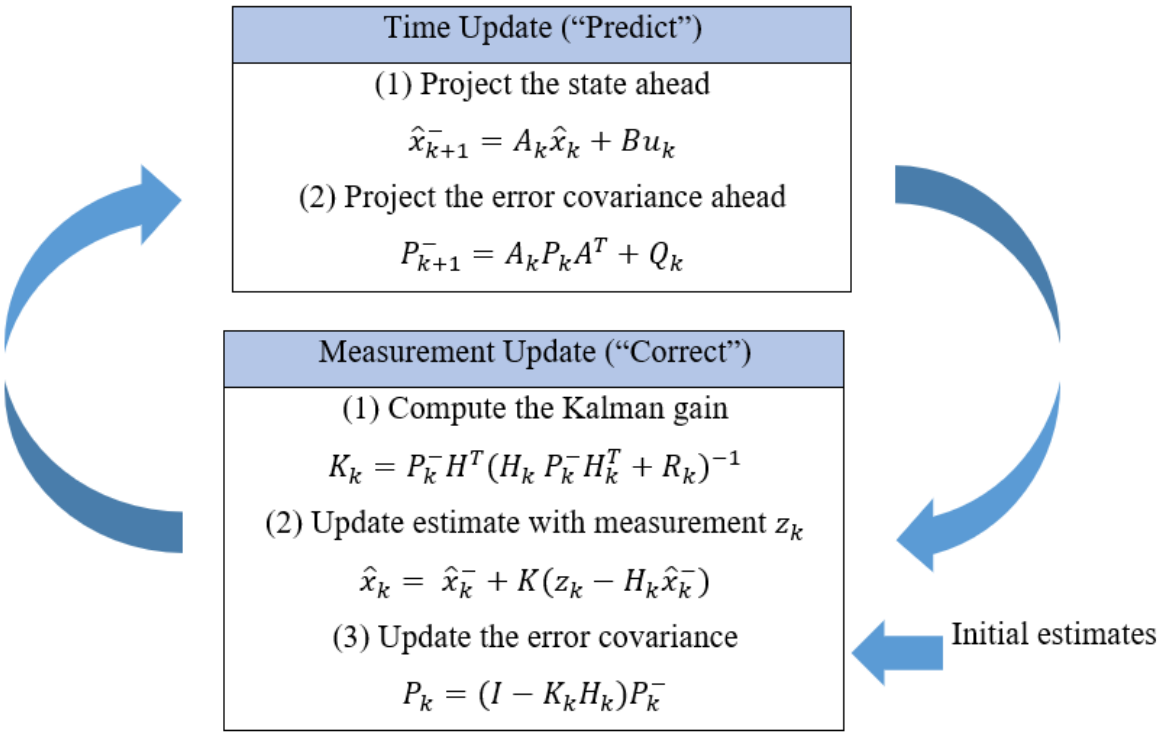


Figure 3.1.2. Predict and correct flowchart

3.2.Position Estimation Process of Kalman Filter

When detecting the position of a moving object, the Kalman filter combines variables such as position and velocity parameters to make an estimate of the state. In this section, Kalman filter integration is explained for constant velocity (i.e., CV model) tracking model for a moving object. The method described in this section is used for star position detection. [48]

True state vector for one dimensional tracking:

$$x_t = (x_t\ v_t)^T \tag{4.9}$$

x_t represents the true position of the tracking object and v_t represents the true velocity of the tracking object. The following equation describes the position according to CV model:

$$x_{tk} = \phi x_{tk-1} + \omega_k \tag{4.10}$$

x_{tk} represents the true state or position at time kT with sampling interval T. ω_k represents the process noise with covariance matrix Q. ϕ is expressed below transition matrix from kT to (k+1)T;

$$\phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad (4.11)$$

The Kalman filter generates the target state prediction based on the dynamic model described above.

Measurement model expressed as below:

$$z_k = \mathbf{H}x_{tk} + v_k \quad (4.12)$$

Where z_k represents the measurement vector and H represents the measurement matrix, v_k represents the noise vector which comes from the measurements with R covariance matrix.

Position only measurement systems (POM) measure only position information of the target. Generally, POM structure is used in moving object tracking algorithms. (e.g. radar, laser and sonar sensors) According to POM, measurement matrix **H** and covariance matrix **R** are expressed as

$$\mathbf{H} = (1, 0) \quad (4.13)$$

$$\mathbf{R} = (B_x) \quad (4.14)$$

B_x represents the variance of errors of position measurement.

Position velocity measured systems (PVM) measure both position and velocity at the same time. (e.g. pulse Doppler radar) PVM models matrices expressed as:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (4.15)$$

$$\mathbf{R} = \begin{bmatrix} B_x & 0 \\ 0 & B_v \end{bmatrix} \quad (4.16)$$

B_v represents the variance value of the errors of velocity measurements.

Kalman filter tracking calculate the prediction and estimation as follows:

$$\tilde{x}_k = \phi \hat{x}_{k-1} \quad (4.16)$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}} = \tilde{\boldsymbol{x}}_{\boldsymbol{k}} + \mathbf{K}_{\boldsymbol{k}}(\boldsymbol{z}_{\boldsymbol{k}} - \mathbf{H}\tilde{\boldsymbol{x}}_{\boldsymbol{k}}) \quad (4.17)$$

"~" denotes the prediction and "^" denotes the estimation. Also K_k, denotes the gain of Kalman filter. Aim of the kalman gain is minimizing error in the Kalman's estimations of position and velocity. Calculation equation of Kalman gain K_k:

$$K_k = \tilde{P}_k \mathbf{H}^T (\mathbf{H} P_k \mathbf{H}^T + R) \quad (4.18)$$

Covariance matrix of errors P_k is:

$$\tilde{P}_k = \phi \hat{P}_{k-1} \phi^T + Q \quad (4.19)$$

$$\hat{P}_k = \tilde{P}_k - K_k \mathbf{H} \tilde{P}_k \quad (4.20)$$

3.3. Smoothing Kalman Filter

As mentioned in the previous sections, the Kalman Filter is widely used to minimize uncertainty. The Kalman filter part of the algorithm was also developed in Google Colab in Python. In this part, the star x, y coordinate information, which is the output of the YOLOv3 CNN network, is applied as an input to the filter. The Kalman Filter corrected the error caused by the trained model, smoothed the results, and gave the final coordinate information for the stars. The matrix information used while developing the Kalman filter algorithm is given below. We need to estimate the x and y star coordinate information at time k.

- State matrix

  Following equation 4.8 represents the state matrix.

$$X = [x \quad x_{velocity} \quad y \quad y_{velocity}] \quad (4.21)$$

  In the state matrix elements, x and y are position information, $x_{velocity}$ and $y_{velocity}$ are velocities of those directions.

- Measurement matrix

  The measurement matrix is shown in the equation 4.9

$$Z = [x \quad y] \quad (4.22)$$

- **H** matrix

$$z = \boldsymbol{H}x \Rightarrow H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.22)$$

- "k" represents the current samples and "k-1" represents the previous samples. System dynamics as follows:

$$x_k = x_{k-1} + \Delta t \, x_{velocity_{k-1}} \quad (4.23)$$

$$x_{velocity_k} = x_{velocity_{k-1}} \quad (4.24)$$

$$y_k = y_{k-1} + \Delta t \, y_{velocity_{k-1}} \quad (4.25)$$

$$y_{velocity_k} = y_{velocity_{k-1}} \quad (4.26)$$

If we write the above 4 matrices as F matrix in the form of the transition matrix, which is the system dynamics.

$$x_k = Fx_{k-1} \tag{4.27}$$

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.28}$$

4. RESULTS

The star detection and coordinate estimation algorithm consists of two parts which are star detection and coordinate estimation using YOLOv3, and smoothing coordinate information using traditional KF. All codes were developed Google Colab Pro environment by using Python language.

Firstly, images are created artificially and labeled. Train, validation, and test datasets are split and .xml and .png files are classified. These images are trained using the transfer learning method. After the method is applied star dataset, star detected boxes with images are obtained as represented Figures 4.1, 4.2, 4.3. Also, the network gives the star x, and y coordinate information in pixel format.

Figure 4.1. Star detection with YOLOv3 Network.

Figure 4.2. Star detection with YOLOv3 Network.

Figure 4.3. Star detection with YOLOv3 Network.

Also, multiple star images tested and trained network is %100 percent successful. These images are represented below in Figure 4.4.

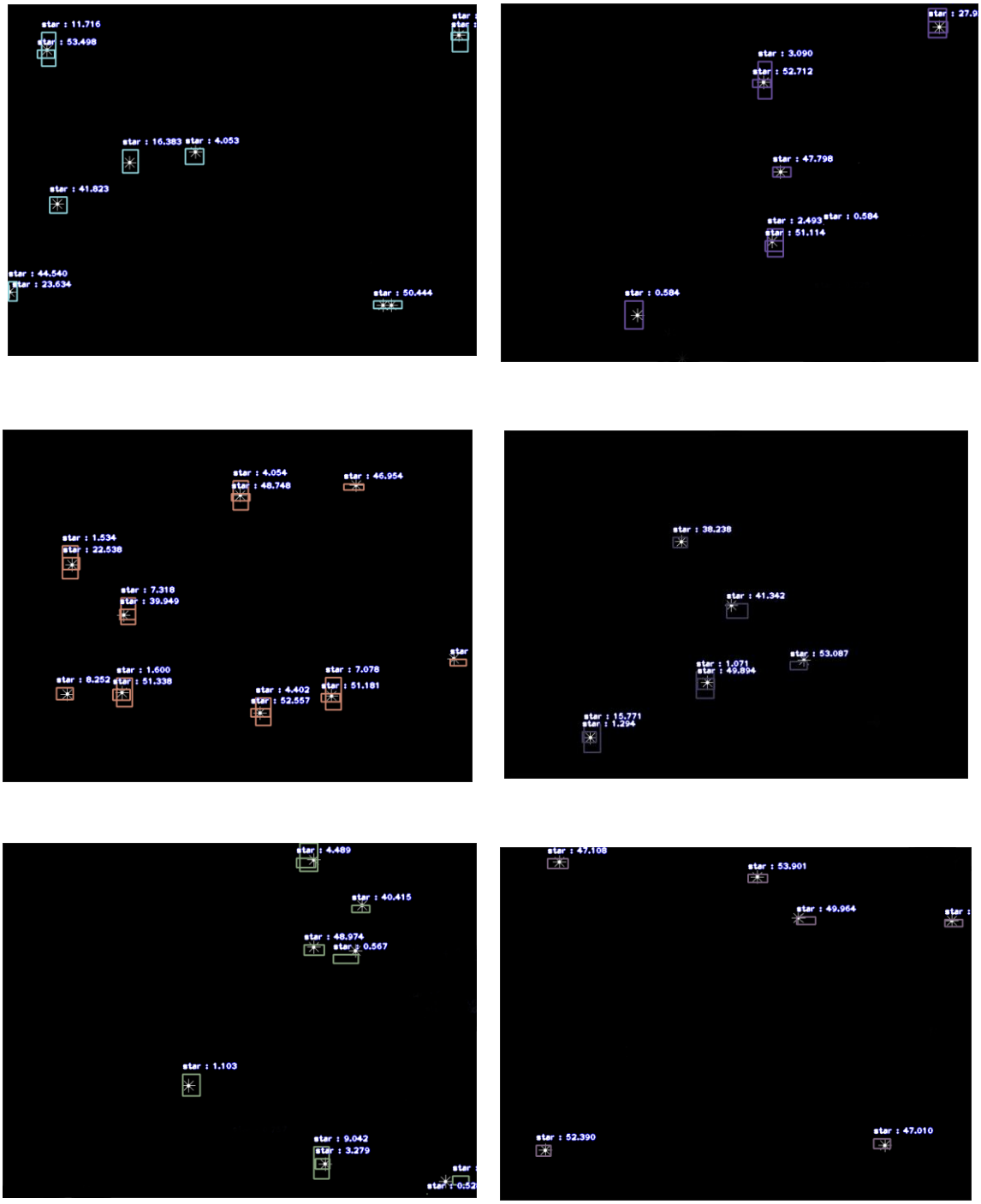


Figure 4.4. Multiple star detection outputs of YOLOv3 Network.

The algorithm's second part is smoothing. The star coordinate information, which is the output of the YOLOv3 network, is smoother with the Kalman Filter. Kalman Filter creates estimation by using previous and current star coordinate information. The Kalman Filter increases the YOLOv3 output to 99.2 percent, which is the least accurate at 82.5 percent.

Coordinate values x, y, which are the output of the YOLOv3 network of the star in the input image, can be detected with approximately 96% accuracy. Up to 99% accuracy was obtained when Kalman Filter has applied to the output of the YOLOv3 network coordinate values.

The x and y coordinates information, which are the measured values, the YOLOv3 network output values and the Kalman filter output values, are shown in Figure 4.5 and Figure 4.6. These graphs obtained 52 x, y coordinate information from star images. By applying the Kalman filter, the erroneous measurement of the YOLOv3 network, which is the peak of the graph, was smoothed out, bringing it closer to the true value from 82.5 percent to 95 percent.

But if the covariance increased 10 times this peak low accurate value 82.5 percent becomes 99.2 percent. This means 10 times covariance for calculations improve the smoothing. The low-sensitivity measurement referred to as this peak was found only once in 52 images. Thus, the measurement of the YOLOv3 network with the lowest accuracy was determined as 82 percent. However, as mentioned above, the Kalman filter increased the accuracy, and this coordinate information was made accurate up to 98 percent.

Kalman Filter 1 named filter is implemented with just one times covariance. Kalman Filter 2 named filter is implemented with 10 times covariance. Following figures 4.5 and 4.6 shows the difference between x-axis values from Kalman Filter 1 and Kalman Filter 2. Clearly these figures represent the Kalman Filter 2 much smoother than Kalman Filter 1. Because the measurement values are less reliable for the filter. Same as x-axis values, y-axis values also much smoother when applied Kalman Filter 2 which represent from Figure 4.7 and Figure 4.8.

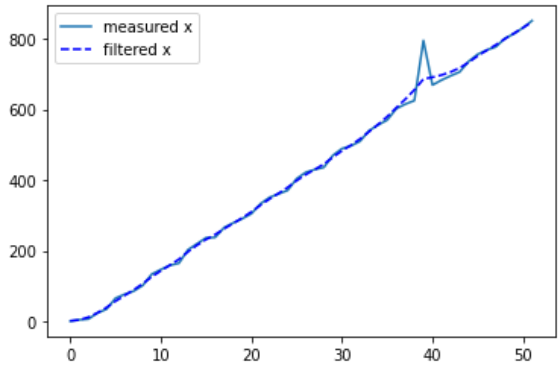


Figure 4.5. Kalman Filter 1 x-axis output.

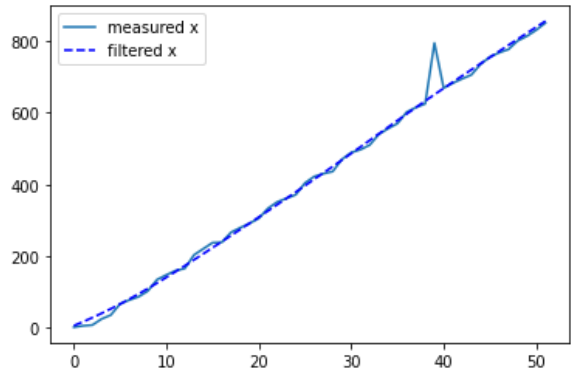


Figure 4.6. Kalman Filter 2 x-axis output.

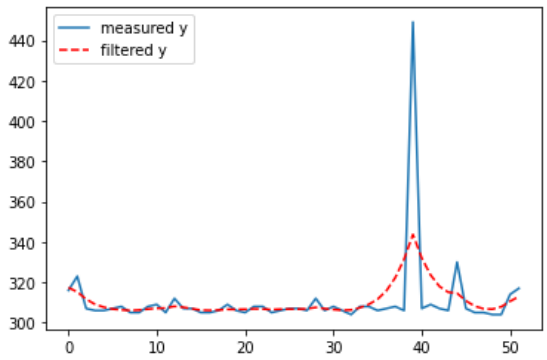


Figure 4.7. Kalman Filter 1 y-axis output.

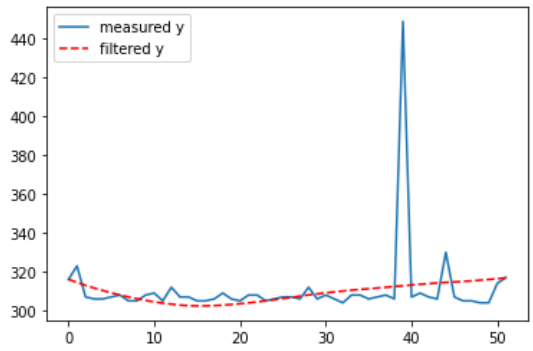


Figure 4.8. Kalman Filter 2 y-axis output.

To represent these results more clearly, Figure 4.9 and Figure 4.10 shows all true, measured, and filtered values. YOLOv3 network obtains a highly accurate detection algorithm for these small pixels for the star dataset. Kalman Filter 1 smooths the YOLOv3's obtained results. Kalman Filter 2 improves the smoothing and filtered values are almost the same as the true values. The true value represents the stars' true location values, measured x and y values are YOLOv3 network coordinate information. Filtered x and y values are smoothed values from the Kalman Filter 1 and Kalman Filter 2.

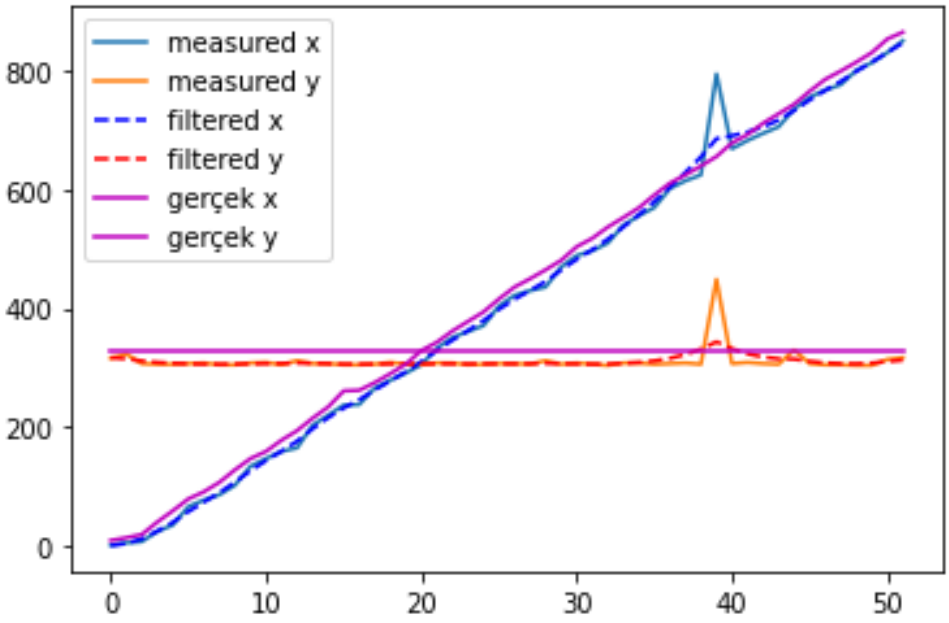


Figure 4.9. Star x, y Coordinate Values Smoother Kalman Filter 1.

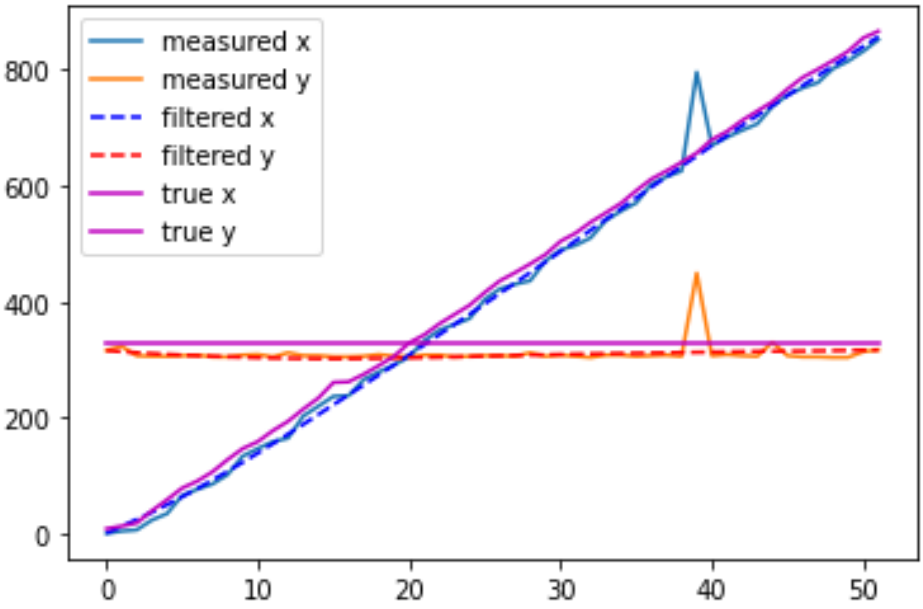


Figure 4.10. Star x, y Coordinate Values Higher Covariance Kalman Filter 2.

Figure 4.11 represents the YOLOv3 network x- axis error graph and Table 4.1 represent the error values and accuracy values.

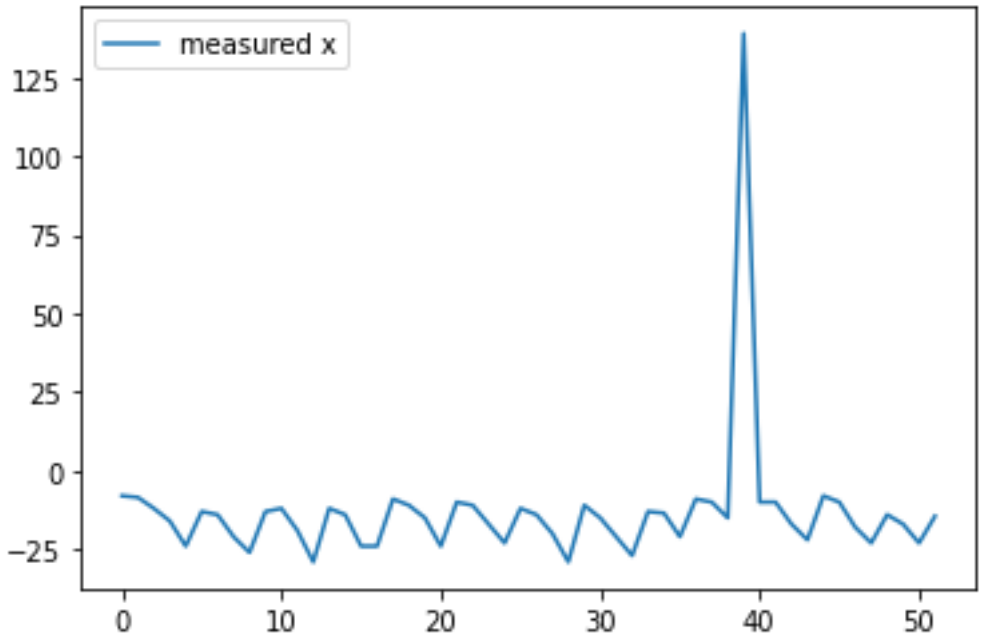


Figure 4.11. YOLOv3 network x- axis error graph.

Table 4.1. YOLOv3 network x-axis estimation accuracy analysis.

Real Star Position	CNN estimation	Error	Accuracy
656	795	139	% 82.5
679	669	-10	%98.5
693	683	-10	%98.5
712	695	-17	%97.6
728	706	-22	%96.7
865.5	831	-34.5	%96
854	851	-3	%99.6

Figure 4.12 represents the Kalman Filter 1 network x- axis error graph and Table 4.2 represent the error values and accuracy values.

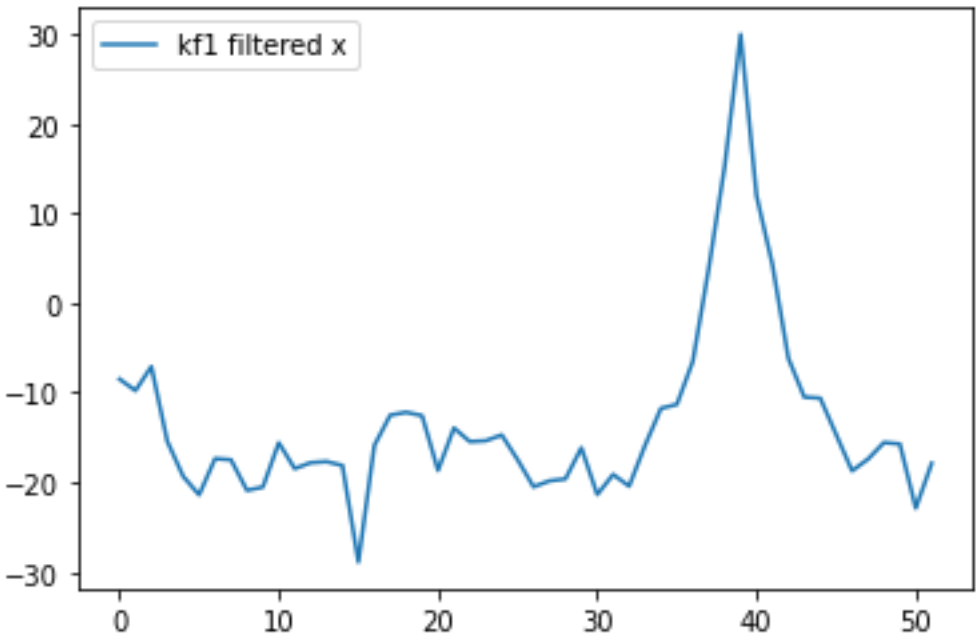


Figure 4.12. Kalman Filter 1 x- axis error graph.

Table 4.2. Kalman Filter 1 x-axis estimation accuracy analysis.

Real Star Position	KF1 estimation	Error	Accuracy
656	685.9	29.9	%95
679	690.9	11.9	%98.2
693	697.2	4.2	%99.3
712	705.8	-6.2	%99.1
728	717.7	-10.3	%98.5
865.5	847.6	-17.9	%97.9
854	831	-23	%97.3

Figure 4.13 represents the Kalman Filter 2 network x- axis error graph and Table 4.3 represent the error values and accuracy values.

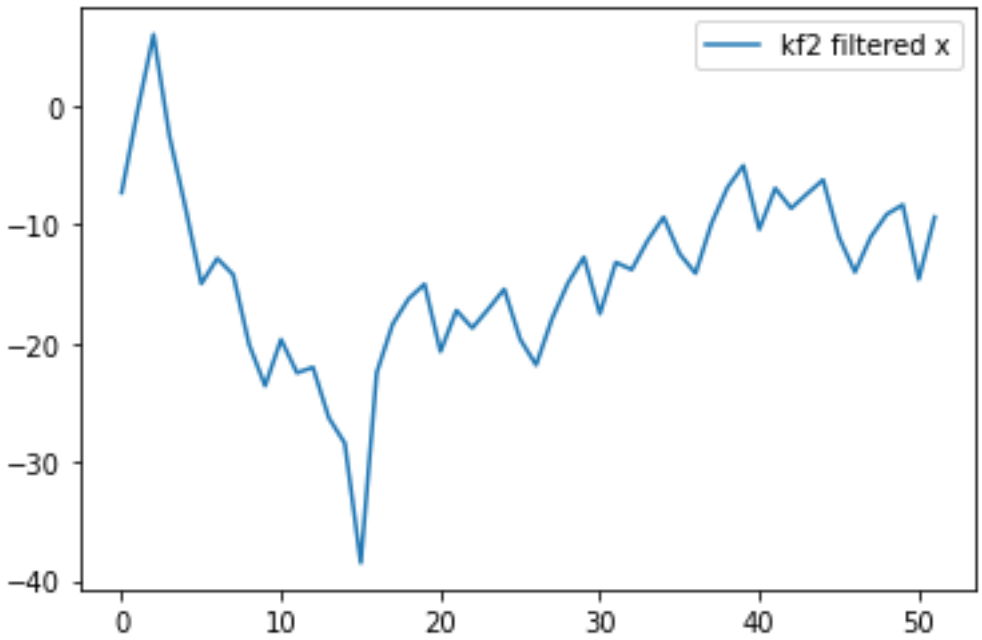


Figure 4.13. Kalman Filter 2 x- axis error graph.

Table 4.3. Kalman Filter 2 x-axis estimation accuracy analysis.

Real Star Position	KF2 estimation	Error	Accuracy
656	650.91	-6.91	%99.2
679	668.5	-10.5	%98.4
693	686	-7	%98.9
712	703.3	-8.7	%98.7
728	720.5	-7.5	%98.9
865.5	839.3	-26.2	%96
854	856	2	%99.7

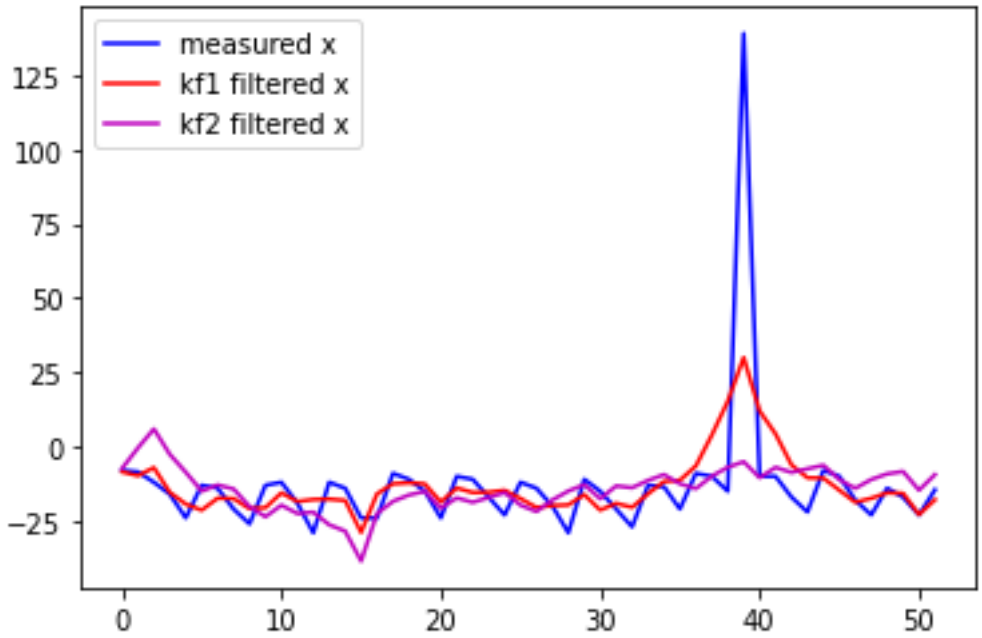


Figure 4.14. Total error graph for x-axis.

Figure 4.17 represents the YOLOv3, Kalman Filter 1, and Kalman Filter 2 error values. Clearly the graph show that Kalman filter minimize the errors. And Kalman Filter 2 achieves the best results. Especially, when the error peak, Kalman Filter 2 minimizes the error considerably.

5. CONCLUSION AND FUTURE WORK

In this thesis, star tracking cameras, which are the most reliable attitude prediction sensors for CubeSats, are discussed. Star detection and star positioning algorithm has been developed for star trackers. Unlike the algorithms in the literature, state-of-art YOLOv3 and traditional Kalman filter were applied together. The purpose of the algorithm is to detect a shooting star on a single axis and find its position in pixels with high accuracy.

First, images of shooting stars in a single axis were obtained artificially. Then, train, validation, and test datasets were created to train these images. Pareto principle was used when separating these datasets. The dataset is 501 in total. According to the Pareto principle, 321 trains, 80 validation, and 100 test images were used. These images are labeled according to Pascal VOC annotation method. The labeled images were trained using the transfer learning method in the YOLOv3 network. Among the trained models, the model with the highest mAP value and the smallest loss value was used. With this model, the star classification and the position of the star in the image were determined in the images from the star tracker. Applying the obtained x, and y coordinate information to the classical Kalman filter, it is aimed to achieve higher accuracy of the YOLOv3 output.

The YOLOv3 network output has approximately 96 percent accuracy. With the Kalman filter, this rate reached 99 percent when we increased the covariance value 10 times. The Kalman Filter applied in the algorithm accepted this system as a linear system and tracked it assuming that the star was moving at a constant velocity.

In future studies, this proposed YOLOv3 and Kalman filter combined star recognition algorithm can be used in attitude determination and control applications with real star tracker images. The Kalman filter part of the algorithm was applied for a linear system. This part can be replaced with non-linear filters.

6. REFERENCES

[1] M. Swartwout, ‘The First One Hundred CubeSats: A Statistical Look’, Journal of Small Satellites, vol. 2, no. 2, 2013, pp. 213–233

[2] C. R. McBryde and E. G. Lightsey, "A star tracker design for CubeSats," 2012 IEEE Aerospace Conference, 2012, pp. 1-14, doi: 10.1109/AERO.2012.6187242.

[3] D. Brasoveanu, J. Hashmall, D. Baker, ‘Spacecraft attitude determination accuracy from mission experience’, 11.1994

[4] A. Chin, R. Coelho, R. Nugent, R. Munakata, and J. Puig-Suari, “CubeSat: the pico-satellite standard for research and education,” in AIAA Space 2008 Conference & Exposition, p. 7734, 2008.

[5] NASA CubeSat Launch Initiative, “CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers,” October 2017.

[6] X. Xia et al., "NanoSats/CubeSats ADCS survey," 2017 29th Chinese Control And Decision Conference (CCDC), 2017, pp. 5151-5158, doi: 10.1109/CCDC.2017.7979410.

[7] G. Rufino, D. Accardo, M. Grassi, G. Fasano, A. Renga, and U. Tancredi, ‘Real-Time Hardware-in-the-Loop Tests of Star Tracker Algorithms’, International Journal of Aerospace Engineering, 2013.

[8] A. M. Rad, J. H. Nobari and A. A. Nikkhah, "Optimal attitude and position determination by integration of INS, star tracker, and horizon sensor," in IEEE Aerospace and Electronic Systems Magazine, vol. 29, no. 4, pp. 20-33, April 2014, doi: 10.1109/MAES.2014.130093.

[9] Huffman, K. M. Designing star trackers to meet microsatellite requirements. M. S. thesis, Massachusetts Institute of Technology, 2006.

[10] Samaan, M. A. Toward faster and more accurate star sensors using recursive centroiding and star identification. Ph.D. dissertation, Texas A&M University, 2003.

[11] C. C. Liebe, A. R. Eisenman, B. R. Hancock, J. A. Mellstrom, J. M. Ratliff, G. M. Swift, J. W. Alexander, M. Wadsworth, and W. J. Walker, "Star Tracker design considerations for the Europa Orbiter Mission," 1999 IEEE Aerospace Conference. Proceedings (Cat. No.99TH8403), 1999.

[12] Cubespace, "CubeStar" https://www.cubespace.co.za/products/adcs-components/cubestar/ Accessed: 16.05.2022

[13] Umland, H. A short guide to celestial navigation. 2004.

[14] Bowditch, N. American Practical Navigator, An Epitome of Navigation. Bethesda, MD: National Imagery and Mapping Agency, 2002.

[15] Thomson, D. B. Introduction to geodetic astronomy. Department of Geodesy and Geomatics Engineering, Univ. of New Brunswick, Canada, Technical Report No. 49, 1981.

[16] Thurmond, R. A history of star catalogues. http://www. rickthurmond.com/History of star catalogues.pdf, 2003.

[17] Høg, E., 'The Tycho-2 catalogue of the 2.5 million brightest stars', Astronomy and Astrophysics, vol. 355, pp. L27–L30, 2000.

[18] Huffman, K. M. Designing star trackers to meet microsatellite requirements. M. S. thesis, Massachusetts Institute of Technology, 2006.

[19] Samaan, M. A. Toward faster and more accurate star sensors using recursive centroiding and star identification. Ph.D. dissertation, Texas A&M University, 2003.

[20] Nemra, A., and Aouf, N. Robust INS/GPS sensor fusion for UAV localization using SDRE nonlinear filtering. IEEE Sensors Journal, Vol. 10, 4 (Apr. 2010), 789–798.

[21] Gao, S., Zhong, Y., Zhang, X., and Shirinzadeh, B. Multi-sensor optimal data fusion for INS/GPS/SAR integrated navigation system. Journal of Aerospace Science and Technology, Vol. 13 (2009), 232–237.

[22] Ali, J., and Fang, J. C. In-flight alignment of inertial navigation system by celestial observation technique. In Proceedings of the International Symposium on Inertial Navigation Technology and Intelligent Traffic, Nanjing, China, Oct. 2004.

[23] Ali, J., and Fang, J. C. SINS/ANS integration for augmented performance navigation solution using unscented Kalman filtering. Journal of Aerospace Science and Technology, Vol 10, 3 (2006), 233–238.

[24] Ali, J., and Fang, J. C. Realization of an autonomous integrated suite of strap down astro-inertial navigation systems using unscented particle filtering. Journal of Computers and Mathematics.

[25] Xu, F., and Fang, J. C. Velocity and position error compensation using strapdown inertial navigation system/celestial navigation system integration based on ensemble neural network. Journal of Aerospace Science and Technology, Vol. 12, 4 (2008), pp. 302–307.

[26] C. C. Liebe, “Accuracy performance of star trackers - A tutorial,” IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 2, pp. 587–599, Apr. 2002, doi: 10.1109/TAES.2002.1008988.

[27] J. Yang, Y. Liu, F. Han, Q. Feng, and Q. Pan, “Pico-satellite attitude determination using a star tracker with compressive sensing,” in Chinese Control Conference, CCC, Sep. 2015, vol. 2015-September, pp. 5331–5336. doi: 10.1109/ChiCC.2015.7260472

[28] S. Sargın Güçlü, “Investigation of the star tracker algorithms and kalman filter integration,” M.S. - Master of Science, Middle East Technical University, 2019.

[29] P. Alveda, “Neural network star pattern recognition of spacecraft attitude determination and control,” Advances in Neural Information Processing System, pp. 213–322, 1989.

[30] J. Hong, J. A. Dickerson, ‘Neural-Network-Based Autonomous Star Identification Algorithm’, Journal of Guidance, Control, and Dynamics, vol. 23, no. 4, pp. 728–735, 2000.

[31] D. Rijlaarsdam, H. Yous, J. Byrne, D. Oddenino, G. Furano, D. Moloney, ‘Efficient Star Identification Using a Neural Network’, Sensors, vol. 20, no. 13, 2020.

[32] L. Xu, J. Jiang and L. Liu, "RPNet: A Representation Learning-Based Star Identification Algorithm," in IEEE Access, vol. 7, pp. 92193-92202, 2019, doi: 10.1109/ACCESS.2019.2927684.

[33] B. Wang, H. Wang, and Z. Jin, ‘An Efficient and Robust Star Identification Algorithm Based on Neural Networks’, Sensors, vol. 21, no. 22, 2021.

[34] A. Lohia, K. Kadam, R. Joshi, D. Bongale, ‘Bibliometric Analysis of One-stage and Two-stage Object Detection’, 02 2021.

[35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587, June 2014.

[36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, June 2016.

[37] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, pp. 1137–1149, June 2017.

[38] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” CoRR, vol. abs/1804.02767, 2018.

[39] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar and K. Ouni, "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3," 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS), 2019.

[40] L. Ouyang and H. Wang, “Vehicle target detection in complex scenes based on yolov3 algorithm,” IOP Conference Series: Materials Science and Engineering, vol. 569, p. 052018, 2019.

[41] L. Zhao, and S. Li, ‘Object Detection Algorithm Based on Improved YOLOv3’, Electronics, vol. 9, no. 3, 2020.

[42] P. Kumari and S. K.r., ‘Periocular Biometrics for Non-ideal Images Using Deep Convolutional Neural Networks’, 02 2020, pp. 143–151.

[43] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” Neural Computation, vol. 29, no. 9. MIT Press Journals, pp. 2352–2449, Sep. 01, 2017. doi: 10.1162/NECO_a_00990.

[44] M. Olafenwa, “ ImageAI ”, github.com, https://github.com/OlafenwaMoses/ImageAI, (Accessed: May. 15,2022)

[45] M. Olafenwa, “ Train Object Detection AI with 6 lines of code ”, medium.com, https://medium.com/deepquestai/train-object-detection-ai-with-6-lines-of-code-6d087063f6ff (Accessed: May. 15, 2022)

[46] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Transaction of the ASME—Journal of Basic Engineering, pp. 35-45,1960.

[47] G. Welch, G. Bishop, ‘An Introduction to the Kalman Filter’, Proc. Siggraph Course, vol. 8, 01.2006.

[48] K. Saho ‘Kalman Filter for Moving Object Tracking: Performance Analysis and Filter Design’, Kalman Filters - Theory for Advanced Applications, G. L. Serra, Brazil February 21 2018, pp. 233-248 2018 [online]. Available: https://www.intechopen.com/chapters/57673 doi: 10.5772/intechopen.68249.