12-2018

Object Detection, Classification, and Tracking for Autonomous Vehicle

Milan Aryal
Grand Valley State University

Object Detection, Classification, and Tracking for Autonomous Vehicle

Milan Aryal

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Engineering, Electrical and Computer Engineering

School of Engineering

December 2018

Acknowledgement

I would like to first thank my thesis advisor Dr. Nicholas Baine. This thesis would not have been completed without his guidance and support. His input and feedback have helped me to gain understanding and take the right approach when solving problems for this thesis. I have enjoyed time spent with him in our thesis meetings, his lecture class for automatic control system and modern control system and working under him as head GA for EGR 106/107 here at GVSU. I would also like to thank my committee members, Dr. Bruce Dunne and Dr. Samhita Rhodes for going through my work and providing me feedback.

I am grateful to Oxford University for open-sourcing the huge dataset collected from the RobotCar. This thesis was able to test algorithms using the datasets provided by Oxford University.

I would also like to thank my family and friends for always helping me and motivating me to be a better person.

Abstract

The detection and tracking of objects around an autonomous vehicle is essential to operate safely. This paper presents an algorithm to detect, classify, and track objects. All objects are classified as moving or stationary as well as by type (e.g. vehicle, pedestrian, or other). The proposed approach uses state of the art deep-learning network YOLO (You Only Look Once) combined with data from a laser scanner to detect and classify the objects and estimate the position of objects around the car. The Oriented FAST and Rotated BRIEF (ORB) feature descriptor is used to match the same object from one image frame to another. This information fused with measurements from a coupled GPS/INS using an Extended Kalman Filter. The resultant solution aids in the localization of the car itself and the objects within its environment so that it can safely navigate the roads autonomously. The algorithm has been developed and tested using the dataset collected by Oxford Robotcar. The Robotcar is equipped with cameras, LiDAR, GPS and INS collected data traversing a route through the crowded urban environment of central Oxford.

Table of Contents

LIST OF FIGURES

ABBREVIATIONS

ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
DATMO	Detection And Tracking of Moving Objects
EKF	Extended Kalman Filter
GPS	Global Positioning System
HOG	Histogram Oriented Gradient
INS	Inertial Navigation System
ORB	Oriented FAST and BRIEF
LiDAR	Light Detection And Ranging
R-CNN	Region with CNN features
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
YOLO	You Only Look Once

1 INTRODUCTION

1.1 Introduction

In recent years, autonomous/self-driving cars have drawn much interest as a topic of research for both academia and industry. For a car to be a truly autonomous, it must make sense of the environment through which it is driving. The autonomous car must be able to both localize itself in an environment and identify and keep track of objects (moving and stationary). The car gets information about the environment using exteroceptive sensors such LiDAR, cameras, inertial sensors, and GPS. The information from these sensors can be used together and fused to localize the car and track objects in its environment, allowing it to travel successfully from one point to another.

The process of path planning and autonomous vehicle guidance depends on three things: localization, mapping, and tracking objects. Localization is the process of identifying the position of the autonomous vehicle in the environment. Mapping includes being able to make the sense of the environment. Tracking of moving objects involves being able to identify the moving objects and track them during navigation. The processes of localization and mapping have been explored through the use of simultaneous localization and mapping (SLAM), which was initially proposed by Leonard and Durant-Whyte. [4] SLAM enables autonomous vehicles to simultaneously build the map of the unknown environment and localize itself in the unknown environment. The development of SLAM has been significant in the development of autonomous robots. Most of the research in SLAM assumes the environment to be static and considers the movement of objects as noise. The detection and tracking of moving objects (DATMO) is one of the most challenging

issues and is important for safety and essential for avoiding collisions. SLAM combined with DATMO helps solve this problem.

The combination of SLAM and DATMO has been approached in different ways. Most of the literature [5], [6], and [7] in the area of DATMO have solved the problem using a laser scanner as the main perception sensor of the vehicle. In these works, the laser data is used to detect the moving object. The problem with using only a laser-based sensor is that the observed shape of the object detected can change from one scan to another scan, making it hard to track the object. The object classified using the laser scanner usually depends upon the shape of the object. This can also lead to misclassification as the shape of one object might look like another, the result could be that a tree may be classified as a person. Consequently, using a laser scanner as the main or only perception sensor might not be right solution for tracking objects.

With advancements in the area of deep learning and incremental improvements in computing power, object detection using images outperforms other methods for the detection and classification of objects. An image is rich with information about the environment. An object detected using a camera fused with distance information from a laser scanner improves the performance of DATMO. In this paper, the problem of DATMO is explored with the use of a deep learning architecture for the detection and classification of the objects.

1.2 Purpose & Scope

The purpose of the thesis is to develop the algorithm to detect, classify, and track the objects for autonomous vehicles. The proposed algorithm is developed using information from a camera, laser scanners, and GPS/INS. The images from the camera is used to detect and classify the object. The laser scans give the distance and direction of the detected objects. The GPS/INS

give the information about the state of the autonomous vehicle. This information is all fused in an Extended Kalman Filter to track the objects and help the autonomous vehicle navigate safely.

1.3 Assumptions

In this thesis, any object detected, person, or any vehicles is represented by a point object. It is assumed that representing them as a point object would be sufficient to demonstrate the algorithm developed in this thesis. In this thesis, the detected objects are not modelled with the motion model but are assumed to have simple random walk model with expected variance.

1.4 Significance

The goal of this thesis is to identify the objects around the autonomous vehicle to aid in safe navigation of the autonomous vehicle. Information of the objects around the autonomous vehicle can be a great asset in avoiding collision. The detection, classification, and tracking of the objects will also help the autonomous vehicle to localize itself better in the environment.

1.5 Limitations

The algorithm developed in the thesis is only tested using one dataset. The algorithm was not implemented in a real system. Due to resource constraints, the object detection and classification on the images were preprocessed separately to decrease computational runtime over multiple simulations when the navigation of the autonomous car was performed.

2 PAPER

Detection, Classification, and Tracking of Objects for Autonomous Vehicles

Milan Aryal, Grand Valley State University
Nicholas Baine, Grand Valley State University

BIOGRAPHIES

Milan Aryal
Milan Aryal is a graduate student at Grand Valley State University pursuing his Master's in Electrical and Computer Engineering. His research interest includes deep learning, computer vision, self-driving cars, and control systems. Currently, he is completing his thesis in the area of self-driving cars.

Nicholas Baine
Nicholas Baine is an Assistant Professor in the School of Engineering at Grand Valley State University. At GVSU, he has been part of a group doing research on the safe repurposing of used lithium-ion battery packs. He also has ongoing research in navigation by utilizing signals of opportunity and sensor fusion. Prior to joining GVSU, Dr. Baine performed research in the area of navigation and integrity monitoring with the Air Force Research Laboratory's Sensors Directorate. While working there, he earned a PhD for work on integrity monitoring for vision based navigation systems.

ABSTRACT

The detection and tracking of objects around an autonomous vehicle is essential for operating safely. This paper presents an algorithm to detect, classify, and track objects. All objects are classified as moving or stationary as well as by type (e.g. vehicle, pedestrian, or other). The proposed approach uses state of the art deep-learning network YOLO (You Only Look Once) combined with data from a laser scanner to detect and classify the objects and estimate the position of objects around the car. The Oriented FAST and Rotated BRIEF (ORB) feature descriptor is used to match the same object from one image frame to another. This information fused with measurements from a coupled GPS/INS using an Extended Kalman Filter. The resultant solution aids in the localization of the car itself and the objects within its environment so that it can safely navigate the roads autonomously. The algorithm has been developed and tested using the dataset collected by Oxford Robotcar. The Robotcar is equipped with cameras, LiDAR, GPS and INS collected data traversing a route through the crowded urban environment of central Oxford.

INTRODUCTION

In recent years autonomous/self-driving cars have drawn much interest as a topic of research for both academia and industry. For a car to be a truly autonomous, it must make sense of the environment through which it is driving. The autonomous car must be able to both localize itself in an environment and identify and keep track of objects (moving and stationary). The car gets information about the environment using exteroceptive sensors such LiDAR, cameras, inertial sensors, and GPS. The information from these sensors can be used together and fused to localize the car and track objects in its environment, allowing it to travel successfully from one point to another.

The process of path planning and autonomous vehicle guidance depends on three things: localization, mapping, and tracking objects. Localization is the process of identifying the position of the autonomous vehicle in the environment. Mapping includes being able to make the sense of the environment. Tracking of moving object involves being able to identify the moving objects and track them during navigation. The process of localization and mapping have been explored through the use of Simultaneous localization and mapping (SLAM) which was initially proposed by Leonard and Durant-Whyte [4]. SLAM enables autonomous vehicles to simultaneously build the map of the unknown environment and localize itself in the unknown environment. The development of SLAM has been significant in the development of autonomous robots. Most of the research in SLAM suppose the environment to be static and consider the moving object as noise. The detection and tracking of moving objects (DATMO) is one of the most challenging issues. The detection of moving objects is important for safety and essential for avoiding collisions. SLAM combined with DATMO helps solve this problem.

The combination of SLAM and DATMO has been approached in different ways. Most of the literature [5], [6], [7] in the area of DATMO have solved the problem using a laser scanner as the main perception sensor of the vehicle. In these works, the laser data is used to detect the moving object. The problem with using only a laser-based sensor is that the observed shape of the object detected can change from one scan to another scan, making it hard to track the object. The object classified using the laser scanner usually depends upon the shape of the object. This can also lead to misclassification as the shape of one object might look like another, the result could be that a tree may be classified as a person. Consequently, using a laser scanner as the main or only perception sensor might not be right solution for tracking objects

With advancements in the area of deep learning and incremental improvements in computing power, object detection using images outperforms other methods for the detection and classification of objects. An image is rich with information about the environment. Object detected using a camera fused with distance information from a laser scanner improves the performance of DATMO. In this paper, the problem of DATMO is explored with the use of a deep learning architecture for the detection and classification of the objects.

To describe this work, this paper begins with a description of the dataset used in the Dataset section. In the System Overview section, the proposed algorithm is presented. This is followed by a description of object detection and by object matching. The results of this work are then presented and followed by a discussion.

DATASET

The algorithm developed on this paper and the results of the algorithm are all based on the datasets provided by Oxford University Robotcar. The Robotcar, an autonomous capable Nissan LEAF, traversed a route through central Oxford, UK twice a week on average over the period of May 2014 to December 2015 to collect data of different driving environments with camera, LiDAR, GPS and INS. The Robotcar was equipped with the following sensors and position of each sensor can be seen in figure 1 [3].

- Cameras:
    - 1 x Point Grey Bumblebee XB3 (BBX3-13S2C-38) trinocular stereo camera, 1280×960×3, 16Hz, 1/3” Sony ICX445 CCD, global shutter, 3.8mm lens, 66° HFoV, 12/24cm baseline
    - 3 x Point Grey Grasshopper2 (GS2-FW-14S5C-C) monocular camera, 1024×1024, 11.1Hz, 2/3” Sony ICX285 CCD, global shutter, 2.67mm fisheye lens (Sunex DSL315B-650-F2.3), 180° HFoV
- LIDAR:
    - 2 x SICK LMS-151 2D LIDAR, 270° FoV, 50Hz, 50m range, 0.5° resolution
    - 1 x SICK LD-MRS 3D LIDAR, 85° HFoV, 3.2° VFoV, 4 planes, 12.5Hz, 50m range, 0.125° resolution

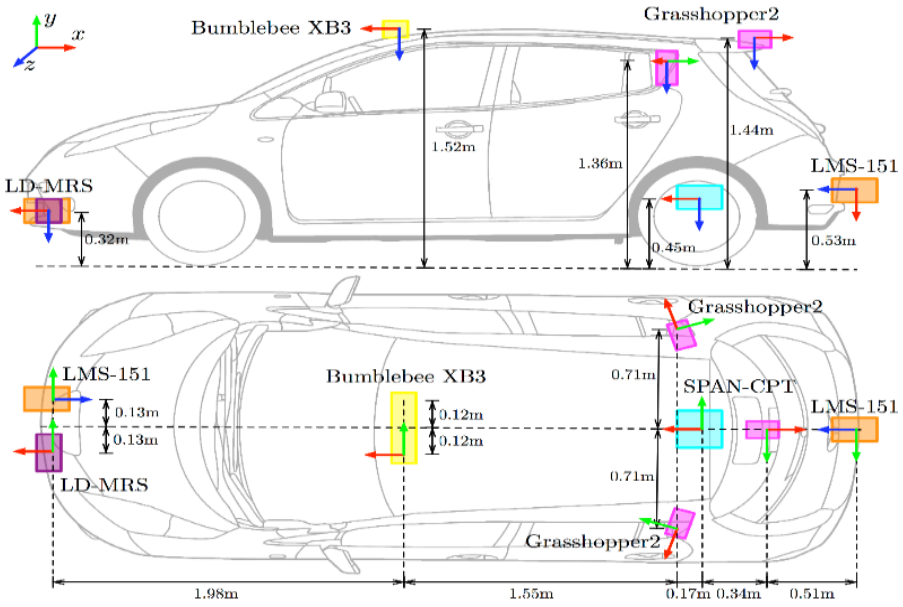

Figure 1:The image of the RobotCar with Sensors location

- GPS/INS:
  - 1 x NovAtel SPAN-CPT ALIGN inertial and GPS navigation system, 6 axis, 50Hz, GPS/GLONASS, dual antenna

As seen in Figure 1, sensors are in different places with the different coordinate frames. The information from each sensor must be transformed to the same coordinate system before processing and fusing in EKF. This work uses the Bumblebee XB3 sensor's location and orientation for the vehicle frame. Information in all other frames is transformed to this frame using homogenous transformation matrix.

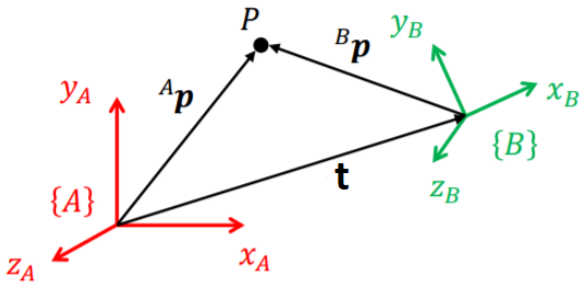

Figure 2: Transformation of point from one frame to another

The point **P** from frame **B** is to be transformed to the frame **A** as shown in the Figure 2. The origin of frame ***B*** is translated by vector **t= *[X, Y, Z]*** from frame **A**. Each *x, y* and *z* axis of the frame B is rotated by α, β, ψ angles compared with respect to respective axis in the frame **A**. The point **P** in frame **B** is translated to frame ***A*** by equation 1.

$$^A\mathbf{p} = \mathbf{H}\,^B\mathbf{p} \tag{1}$$

H is the homogenous transformation matrix given by equation 2.

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2}$$

The rotation matrix, **R** is calculated using equation 3.

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\beta & \cos\psi\sin\beta\sin\alpha - \cos\alpha\cos\psi & \cos\psi\sin\beta\cos\alpha + \sin\alpha\sin\psi \\ \cos\beta\sin\psi & \cos\alpha\cos\psi + \sin\psi\sin\beta\sin\alpha & \cos\alpha\sin\psi\sin\beta - \cos\psi\sin\alpha \\ -\sin\beta & \cos\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \tag{3}$$

The translation vector **t** is given by equation 4.

$$\mathbf{t} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{4}$$

SYSTEM OVERVIEW

In this paper, the information from the image from Bumblebee XB3 camera, SICK LD-MRS laser scanner and NovAtel SPAN-CPT INS and GPS system is used. The details about these sensors is discussed in previous section. Figure 3 shows the block diagram of the system that is used to study the problem of detecting objects around the RobotCar and track them.

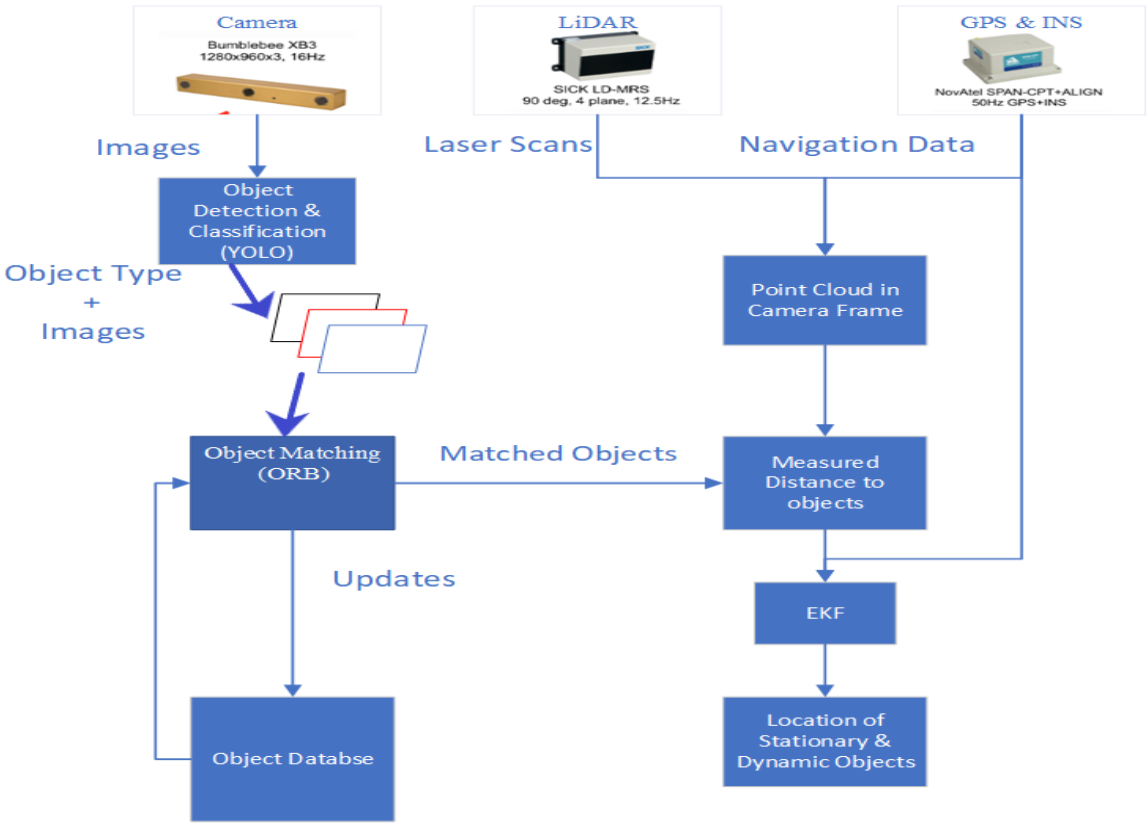

Figure 3: Block Diagram of Proposed System

The image obtained from the camera is used to detect the object and classify them. The objects in the image are detected using Convolutional Neural Networks (CNN). Once objects are detected they are stored in a database. Each detected object is matched with objects in the database to find association or added as a new object in the database. To manage the size of the database, objects that are no longer detected are deleted from the database.

With object detection in the image complete, the data from the laser scanner is projected onto the image. This allows for the measurement of the distance and direction of detected objects from the RobotCar. This information along with the state of RobotCar is combined with an Extended Kalman Filter (EKF). Both the state of the objects and the RobotCar are updated using EKF allowing for a combined localization and tracking of objects in the environment.

OBJECT DETECTION

To perform object detection, this work uses datasets that provide information of the environment through the LiDAR and camera. Using the information from these sensors, objects are detected, classified, and the distance and direction of the object relative to the Robotcar is measured. Usually object detection is achieved using a combination of feature-based modelling and appearance-based modelling. The image has more information that can be used to identify objects as compared to laser scan and allows both features based and appearance-based modelling. In this paper, the image is primarily used to detect objects and classify them, and the LiDAR is used to measure the location of the object relative to the vehicle.

The laser scan combined with the pose of the vehicle is used to create the 3D point cloud of the environment. This is then projected onto the image. The transformation of 3D coordinates obtained from LiDAR scan to 2D image pixels is done using pinhole camera model. The LiDAR (x, y, z) coordinates are transformed into pixels (u, v) in the image using equation 5.

$$u = \frac{f}{z}x + u_0$$
$$v = \frac{f}{z}y + v_0 \tag{5}$$

where f is the focal length of the camera and (u_0, v_0) is the optical center of the camera.

The Figure 4 shows sample image and the laser scans projected onto the image.

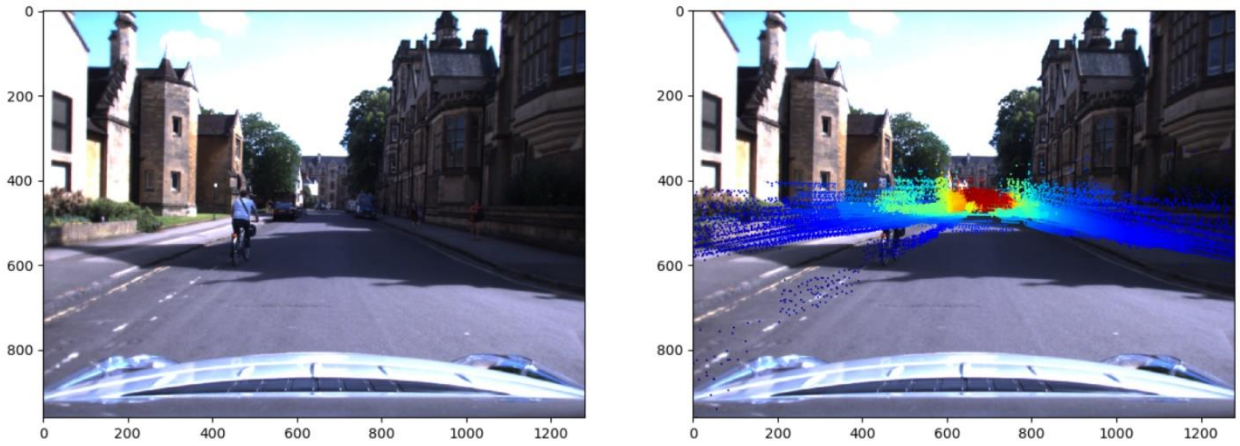

Figure 4: A Sample Image and Laser scan projected onto the Image

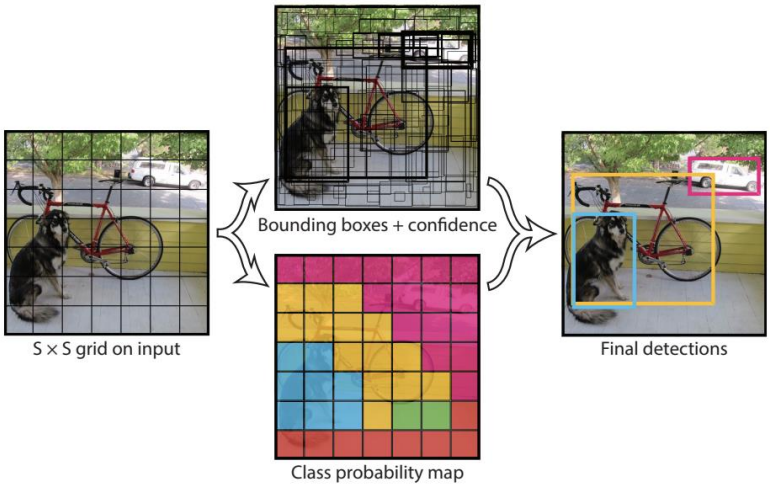

Figure 5 :Illustration of YOLO algorithm

A single image will have multiple objects that need to be identified. Detecting multiple objects in an image is a challenging task; however, using current methods, it is achievable with accuracy in real time. Object detection is one of the most researched topics in the area of Computer vision. The object detection usually starts with extracting features from the input image using different algorithm such as Haar [8] or HOG [9].These features are fed to the classifier to identify the object. With advancement in deep learning, there are many convolutional neural network (CNN) architectures that have outperformed the object detection method using a feature extractor. These CNNs attempt to imitate the working of human neurons. The CNN learns the feature in object during training process and can generalize the feature for the object and detect the object.

The R-CNN [10], Faster R-CNN [11], YOLO algorithms are all based on CNN and perform very well at multiple object detection. This work uses YOLO algorithm for the detection and classification of objects. The advantage of using YOLO is that it is orders of magnitude faster (45 frames per seconds) than other algorithms while maintaining a high accuracy. Other algorithms detect objects using region-proposal techniques or sliding-windows method and iterate over the image for many times. YOLO sees the entire image and only once hence the name. This makes YOLO faster compared to iterative methods. The entire image is fed through the CNN and detects multiple objects simultaneously.

YOLO works by taking an image and splitting it into an SxS grid. For each grid cell, it predicts bounding boxes and confidence for those boxes and class probability. Objects are located within the image where the bounding boxes have a class probability above a threshold value. For every object detected YOLO classifies the object, gives the confidence and the bounding box for the object. Figure 5 illustrates the working principle of the YOLO algorithm [1].

Each object detected has a bounding box, which localizes the object in the image. This information is helpful in finding the distance of the objects from the RobotCar. Once the objects are detected and classified and located in the image, the projected laser scans to the image can be isolated so that only laser bouncing back from the objects detected remain. Figure 6 shows the object detected by the YOLO algorithm and corresponding laser scan projected on the detected objects.

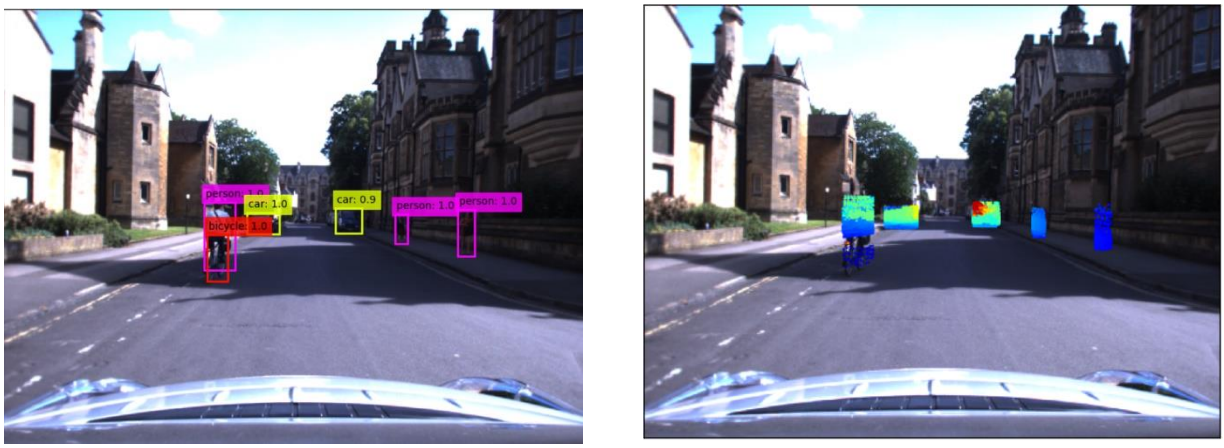

Figure 6: Object Detection by YOLO (left) and Laser Projection on detected objects (right)

The detected objects are represented as the point object for the purpose of the tracking. The centroid of the laser scan projected onto the object is taken as distance of the object from the RobotCar. Figure 7 shows the representation of the detected objects in figure 7 as point object and in vehicle frame.

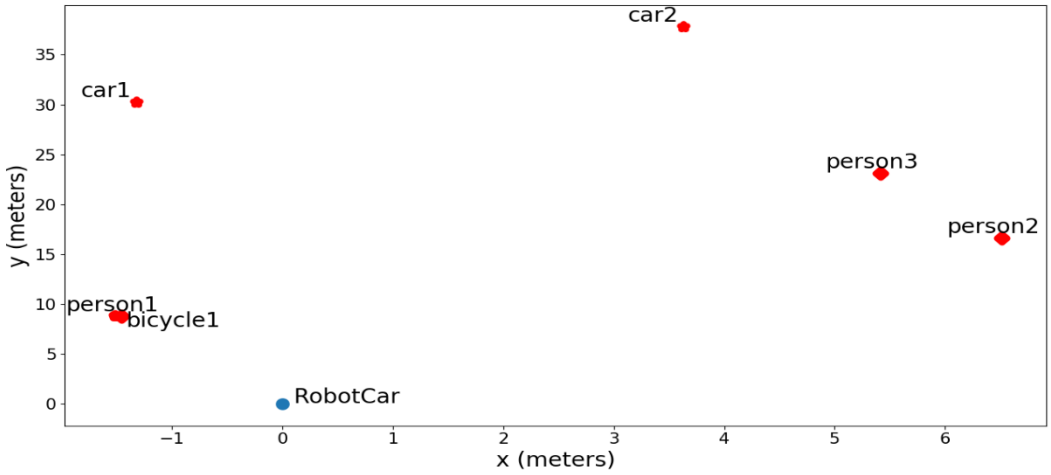

Figure 7: 2D Map of Object detected in Vehicle Frame

OBJECT MATCHING

Once the object is detected in one image, for successful tracking the object in one image must be associated with another image. This association is achieved by matching the features of the object in one image and matching that features with the object detected in the next image.

In this work, Oriented FAST and Rotated BRIEF (ORB) is used to match features of objects and to tracking match objects from one image to another. ORB is a very fast binary descriptor computationally-efficient replacement to SIFT [12] with similar matching performance. Using ORB keypoint from the training image and the query image are extracted, and they are matched. The match for each key point is found by using the Brute-Force Matcher function provided by OpenCV library [13]. The Brute-Force Matcher takes the descriptor of one feature in a training image and is matched with all features in query image using a distance calculation, and the one with the minimum distance is returned as matching. There is still a risk of a false match. The approach that best reduces this risk is to find second nearest neighbor and perform ratio of closest to second closest as described in [12]. All the matches with distance ratio between closest and second closest greater than 0.75 are discarded.

Figure 8:The Performance of the ORB on matching same object and different object

Figure 8 shows matching the key points from object detected in one image to objects detected in next frame. It can be observed that number of key points matched between same object in one image frame to another frame is very high compared to the match with a different object.

If the object detected in one frame cannot be matched with an object detected in the next image, the object is removed from the database. If an object is detected again after deletion, it is treated as new object and tracking resumes.

EXTENDED KALMAN FILTER

The information from the different sensors is fused together using Extended Kalman Filter (EKF) for the tracking of the objects around the autonomous car. Based on the information from INS, LiDAR and camera the following states are chosen for the filter.

$$\mathbf{x}_t = (\mathbf{R}, \mathbf{O})^T = \left(\underbrace{X, Y, \psi, v_x, v_y}_{RobotCar's\ state}, \underbrace{o_{1,x}, o_{1,y}}_{object\,1}, ..., \underbrace{o_{n.x}, o_{n,y}}_{object\,n} \right)^T$$

At any given time, state vector in the EKF consists of the state of the Robotcar, *R*, and the position of the, *n*, number of objects detected. The state of the Robotcar include the position of the vehicle (*X* and *Y*), yaw angle of the vehicle (ψ), the velocity (v_x and v_y). The velocity of the RobotCar is provided in the local frame. The local frame and the body frame are shown in figure 9.

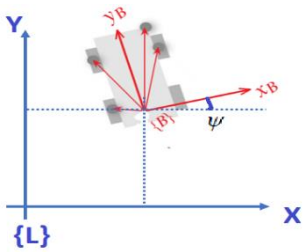

Figure 9:The Local Frame and the RobotCar body frame

The navigation of the car is done in UTM coordinates, Northing and Easting. The area navigated by the car is a small area around center of Oxford; hence, using local navigation coordinates gives accurate pose estimation of the RobotCar and the objects detected. The entire area falls under the 30U UTM zone, which is used for navigation.

The motion model for the Robotcar is modeled using equation 6.

$$\begin{aligned} X_t &= X_{t-1} + v_{x_{t-1}} \Delta t \\ Y_t &= Y_{t-1} + v_{y_{t-1}} \Delta t \end{aligned} \tag{6}$$

The velocity and the yaw are not defined by any motion model equation as the INS equipped in the RobotCar does not measures the acceleration or the yaw rate of the vehicle. These are updated using the motion noise. The rolling average for all the measurements provided by INS and GPS is calculated. After rolling average is calculated the variance and co-variance of these measurements is calculated as the motion noise, ***Q***, given by:

$$\mathbf{Q} = \begin{bmatrix} \sigma^2_{X_{rm}} & \sigma^2_{X_{rm}Y_{rm}} & \sigma^2_{X_{rm}\psi_{rm}} & \sigma^2_{X_{rm}v_{xrm}} & \sigma^2_{X_{rm}v_{yrm}} \\ \sigma^2_{Y_{rm}X_{rm}} & \sigma^2_{Y_{rm}} & \sigma^2_{Y_{rm}\psi_{rm}} & \sigma^2_{Y_{rm}v_{xrm}} & \sigma^2_{Y_{rm}v_{yrm}} \\ \sigma^2_{\psi_{rm}X_{rm}} & \sigma^2_{\psi_{rm}Y_{rm}} & \sigma^2_{\psi_{rm}} & \sigma^2_{\psi_{rm}v_{xrm}} & \sigma^2_{\psi_{rm}v_{yrm}} \\ \sigma^2_{v_{xrm}X_{rm}} & \sigma^2_{v_{xrm}Y_{rm}} & \sigma^2_{v_{xrm}\psi_{rm}} & \sigma^2_{v_{xrm}} & \sigma^2_{v_{xrm}v_{yrm}} \\ \sigma^2_{v_xyX_{rm}} & \sigma^2_{v_{yrm}Y_{rm}} & \sigma^2_{v_{yrm}\psi_{rm}} & \sigma^2_{v_{yrm}v_{xrm}} & \sigma^2_{v_{yrm}} \end{bmatrix} \tag{7}$$

The covariance matrix is setup in following ways:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P_{RR}} & \mathbf{P_{RO}} \\ \mathbf{P_{OR}} & \mathbf{P_{OO}} \end{bmatrix} = \begin{bmatrix} \mathbf{P_{RR}} & \mathbf{P_{RO_1}} & \cdots & \mathbf{P_{RO_n}} \\ \mathbf{P_{O_iR}} & \mathbf{P_{O_iO_1}} & \cdots & \mathbf{P_{O_iO_n}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P_{O_nR}} & \mathbf{P_{O_nO_1}} & \cdots & \mathbf{P_{O_nO_n}} \end{bmatrix} \tag{8}$$

where $\mathbf{P_{RR}}$ is the covariance matrix for the RobotCar, $\mathbf{P_{RO}}$ is the covariance matrix for the RobotCar and the landmarks, $\mathbf{P_{OR}}$ is the covariance matrix between the objects and the Robotcar, and $\mathbf{P_{OO}}$ is the covariance matrix between the objects being tracked.

The motion modelling for the state of RobotCar didn't have any non-linearity so computing Jacobian is not required for predict step of EKF. The predict step in EKF is done by equation 9.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{x}}_t = \mathbf{F}\mathbf{x}_{\mathbf{t-1}}$$

$$\hat{\mathbf{P}}_{\mathbf{t}} = \mathbf{F}\mathbf{P}_{\mathbf{t-1}}\mathbf{F}^{\mathbf{T}} + \mathbf{Q} \tag{9}$$

After the predict step, the EKF is updated based on the measurements obtained. First the state, x_t ,and state covariance matrix, $\boldsymbol{P_t}$, in EKF is updated based upon the measurements obtained from the GPS/INS and then based upon the measurements of the detected objects obtained from the LiDAR. The measurement for the RobotCar state, $\mathbf{z_t}$, at time *t* is directly obtained from the GPS and INS.

$$\mathbf{z_t} = \begin{bmatrix} X_t & Y_t & v_{x_t} & v_{y_t} & \psi_t \end{bmatrix}^{\mathrm{T}}$$

Once the object has been detected it is fused in the EKF. The measurements for the object position are given in rectangular coordinates in vehicle frame. The second stage of EKF update is done with the measurements for landmarks.
For all objects detected, the position of j^{th} object in vehicle frame from laser scanner at time *t* is given by,

$$\mathbf{z}^{\mathbf{j}}_{\mathbf{t}} = \begin{bmatrix} o_{x,t} \\ o_{y,t} \end{bmatrix}$$

If the j^{th} detected object is new object, then it is added to the state vector, $\boldsymbol{x_t}$, in navigation frame using equation 10.

$$\begin{bmatrix} O_{j,x} \\ O_{j,y} \end{bmatrix} = \begin{bmatrix} X_t \\ Y_t \end{bmatrix} + \begin{bmatrix} \cos(\psi_t) & -\sin(\psi_t) \\ \sin(\psi_t) & \cos(\psi_t) \end{bmatrix} \mathbf{z}^{\mathbf{j}}_{\mathbf{t}} \tag{10}$$

When an object is first detected, it is considered to be dynamic and is modelled with high process noise $\boldsymbol{Q_o}$. The motion noise **Q** is updated for each new object by,

$$\mathbf{Q}_{\mathrm{k+1}} = \begin{bmatrix} \mathbf{Q}_{\mathrm{k}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q_o} \end{bmatrix}$$

If the j^{th} detected object is already in the state vector, then the estimated distance of the object from the Robotcar in navigation frame is calculated by equation 11.

$$\boldsymbol{\Delta} = \begin{bmatrix} \Delta_{\mathrm{x}} \\ \Delta_{\mathrm{y}} \end{bmatrix} = \begin{bmatrix} \hat{O}_{j,x} - \hat{X} \\ \hat{O}_{j,y} - \hat{Y} \end{bmatrix} \tag{11}$$

where $(\hat{O}_{j,x}, \hat{O}_{j,,y})$ give the estimated position of the object in the navigation frame and $(\hat{X}, \hat{Y})$ give the estimated position of the RobotCar in navigation frame. The estimated distance of the object in navigation frame is converted to vehicle frame $\hat{\mathbf{z}}^{\mathbf{j}}_{\mathbf{t}}$ by equation 12.

$$\hat{\mathbf{z}}_t^j = \begin{bmatrix} \cos(\hat{\psi}) & \sin(\hat{\psi}) \\ -\sin(\hat{\psi}) & \cos(\hat{\psi}) \end{bmatrix} \boldsymbol{\Delta} = \mathbf{h}(\hat{\mathbf{x}}_t) \tag{12}$$

The difference in the measurements from the sensor and estimated measurements, $\mathbf{y}_j^t$ is given by equation 13.

$$\mathbf{y}_j^t = \mathbf{z}_t^j - \hat{\mathbf{z}}_t^j \tag{13}$$

The Jacobin for the j^{th} object is given by equation 14.

$$\mathbf{H}_t^j = \frac{\partial \mathrm{h}(\hat{\mathbf{x}}_t)}{\partial \hat{\mathbf{x}}_t} = \begin{bmatrix} -\cos(\hat{\psi}) & \sin(\hat{\psi}) & -\sin(\hat{\psi})\Delta_x - \cos(\hat{\psi})\Delta_y & \cos(\hat{\psi}) & \sin(\hat{\psi}) \\ -\sin(\hat{\psi}) & -\cos(\hat{\psi}) & \cos(\hat{\psi})\Delta_x - \sin(\hat{\psi})\Delta_y & \sin(\hat{\psi}) & \cos(\hat{\psi}) \end{bmatrix} \tag{14}$$

This Jacobian is only for one object with respect to estimated position and yaw angle of the RobotCar. The state vector for the EKF consists of the states for all objects and the vehicle, so this Jacobian has to be mapped to higher dimension which is done by matrix ***M***, as in equation 15.

$$\mathbf{M}_j = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \underbrace{0 \quad \cdots \quad 0}_{2j-2} & & & 0 & 1 & \underbrace{0 \quad \cdots \quad 0}_{2N-2j} & & \end{bmatrix}$$

$$\mathbf{H}_j^t = \mathbf{H}_j^t \mathbf{M}_j \tag{15}$$

The Kalman gain due to j^{th} object $\mathbf{K}_j^t$ is given by equation 16.

$$\mathbf{K}_j^t = \hat{\mathbf{P}}_t (\mathbf{H}_t^j)^T (\mathbf{H}_t^j \hat{\mathbf{P}}_t (\mathbf{H}_t^j)^T + \mathbf{R}_t)^{-1} \tag{16}$$

where $\mathbf{R}_t$ is the measurement noise.

The update in state $\mathbf{x}_t$ state covariance $\mathbf{P}_t$ due to j^{th} object is given by equation 17.

$$\begin{aligned} \hat{\mathbf{x}}_t &= \hat{\mathbf{x}}_t + \mathbf{K}_t^j \mathbf{y}_t^j \\ \hat{\mathbf{P}}_t &= \hat{\mathbf{P}}_t - \mathbf{K}_t^j \mathbf{H}_t^j \hat{\mathbf{P}}_t \end{aligned} \tag{17}$$

This process is repeated for all detected objects. After each iteration, the estimated position of the object is compared with the average past position of the objects in navigation frame. If the difference in current estimated position and the average past position of the objects is found to be crossing a threshold based on the object type, then the process noise associated with the object, $\boldsymbol{Q_o}$, is decreased to model the stationary object.

RESULTS

In this section the algorithm developed is tested using the Oxford RobotCar dataset to see the performance on tracking the objects and position estimation of the RobotCar. The exact ground truth data for the position of the vehicles are not provided so the exact accuracy cannot be calculated.

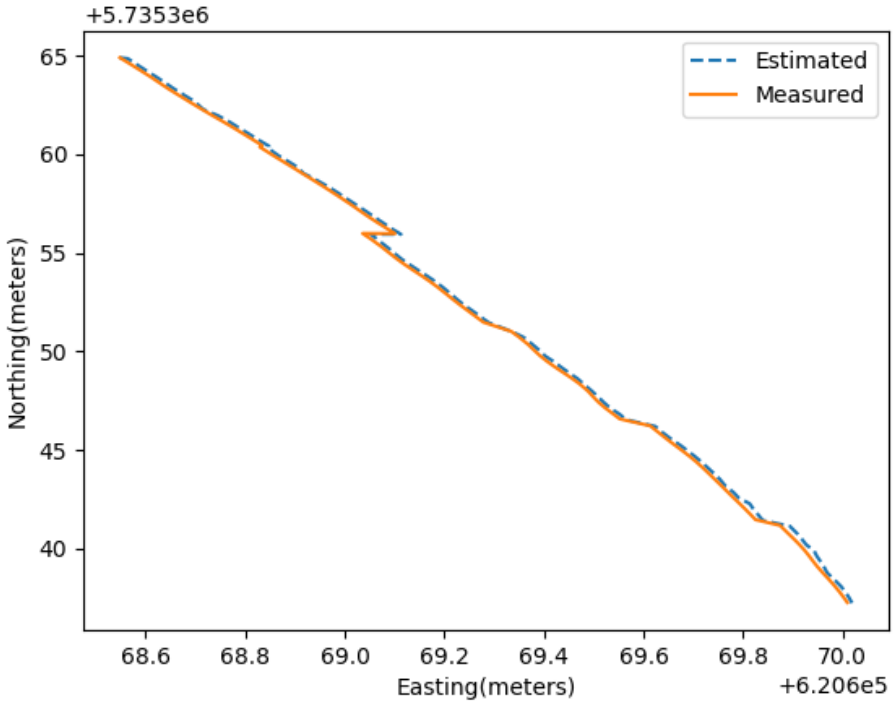

Figure 10: Estimation of postition of the Robotcar while tracking objects

The aim of the algorithm developed is to aid in localization of the RobotCar while tracking different objects. One Extended Kalman Filter is used to both estimate the position of the RobotCar and track the objects. The RobotCar must successfully be able to estimate its position and localize in the environment. Figure 10 shows the position of the RobotCar as estimated versus the position of the RobotCar measured by GPS. As ground truth for the position of the RobotCar is not present, the performance is compared with the measured values. The estimated position closely follows the measured position, with a slight smoothing relative to measured position.

The difference between the measured and estimated positions are displayed in Figure 11 and is broken down into northing and easting. It is calculated by taking the absolute difference between the estimated and measured position. From Figure 11, it can

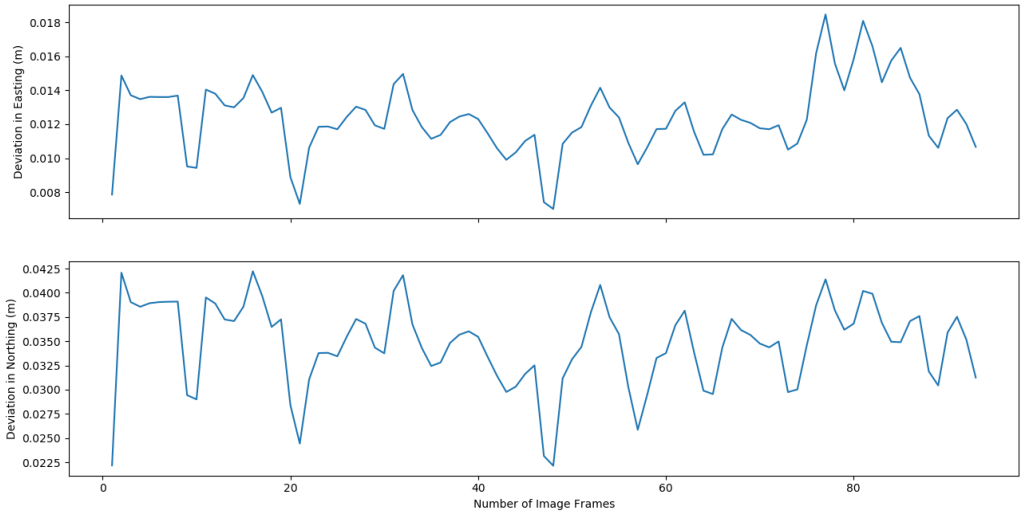

Figure 11: Deviation on the estimation for the position of the RobotCar

be observed that there is a translational deviation with peak of 0.018m in Easting and 0.0425m deviation in Northing Position. This shows that the RobotCar GPS measurements are not vastly different than the position estimation from the EKF, which is a good indication that the system is performing well, while tracking the objects around itself.

The performance of algorithm on tracking of the moving objects is shown in Figures 12 and 13. Figure 12 shows the image of a person riding a bicycle on the right and the tracking of both objects. The graph shows that position of the both objects (bike and the person on it in navigation frame. The separate plots are the position estimates of the person and bicycle based on the information from lidar scans. As seen in the image, it can be observed that the person is riding a bicycle, so tracking algorithm must estimate them to be together. The graph of the position estimates shows the person and the bicycle to be very similar and

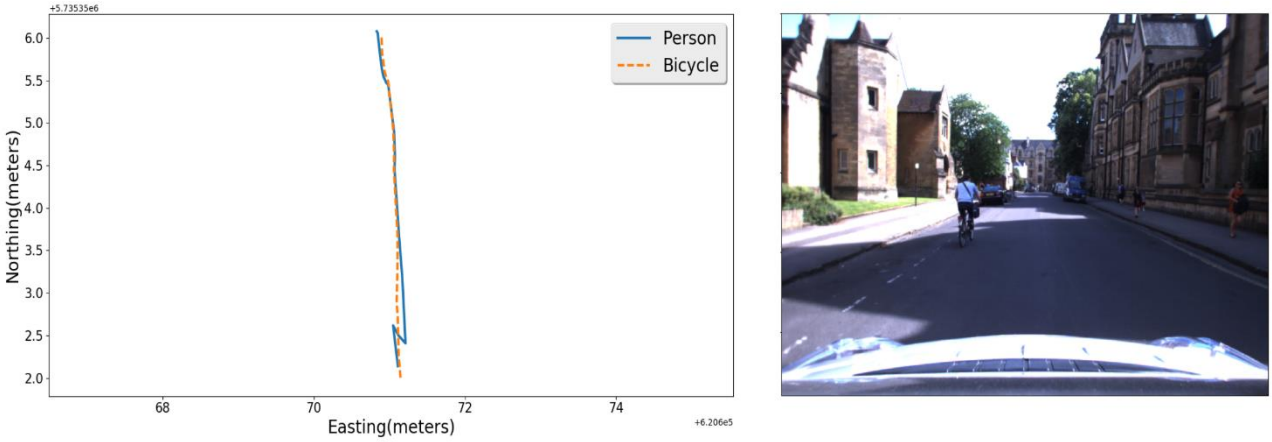

Figure 12: Tracking of Person Riding a bicycle

deviating by less than 20cm in the navigation frame. The minor differences in the estimates are due object not being modeled to their actual velocity and laser scanner might not pick both object in same scan. This shows the algorithm is able to track the different objects even if they are moving together.

Figure 13 shows the performance of the algorithm in tracking another moving car. Figure 13 depicts the estimated position of the car by the EKF and measured position of the car based on lidar scans in the navigation frame. The graph shows the estimated position compared with the measured position from the lidar scans. The estimate position follows the measured position. Additional performance could be obtained by adding velocity states for the target vehicle, but the work presented assumed a simple random walk model with an expected variance based on the average velocity of moving vehicles.

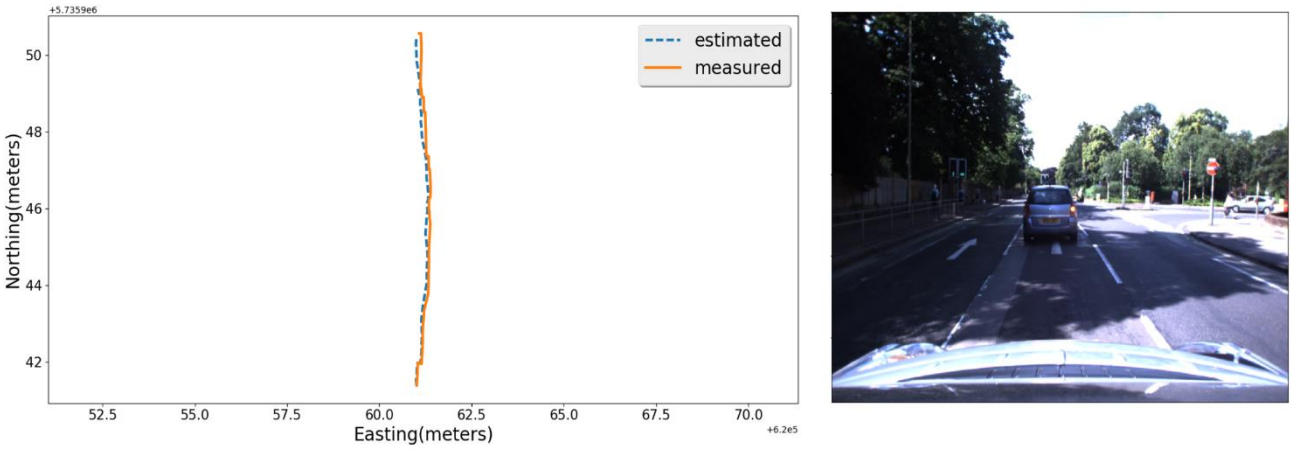

Figure 13: Tracking of moving car in Navigation frame

The algorithm developed in the paper tracks stationary object as well. Figure 14 shows Car_1 and Car_2 which are stationary being tracked simultaneously. The plot shows the estimated position of Car_1 and Car_2 when the RobotCar in navigating in navigation frame. Based on image frames the car are stationary and the tracking algorithm supports that as there is very minimal change in the position of both the cars. Car_1 was tracked for 25 image frames and Car_2 was tracked for 30 image frames.

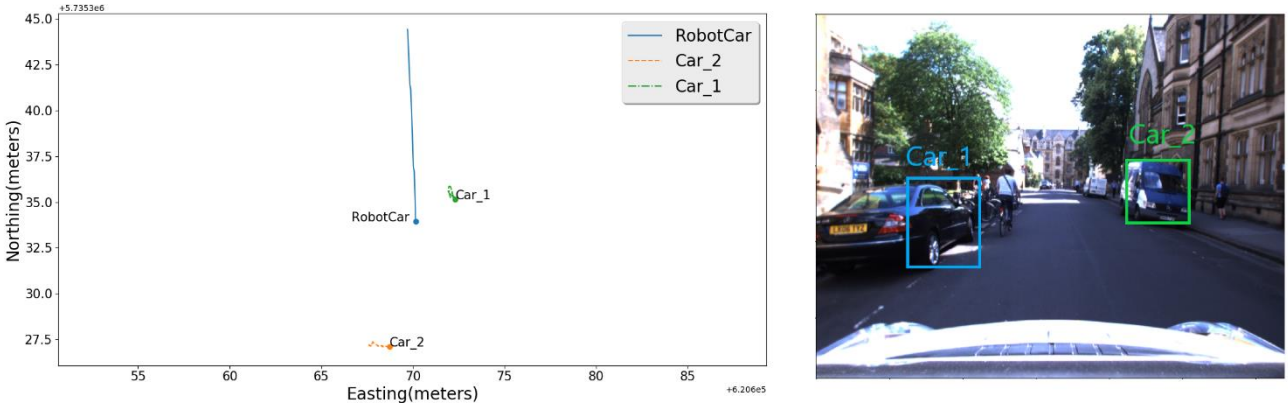

Figure 14: Tracking Stationary Cars

Figure 14 showed that tracking the stationary car there was some change in the position of the car. Figure 15 shows the estimated and measured position of the stationary Car1 in northing and easting with respect to number of image frame tracked. It can be observed that there is change in position of 0.5m in Eastings and 0.7m in Northings. The change in position is because of the measurement from the lidar scans. As the paper models object as a point object, during tracking the lidar scan return is not from the same point. When Car_1 is first detected the lidar scans returns from the bumper and that distance is associated with the object, as the Robotcar moves the lidar scan return is from the side of the car and that distance is associated with the object. As a result of this there is some change in object location.

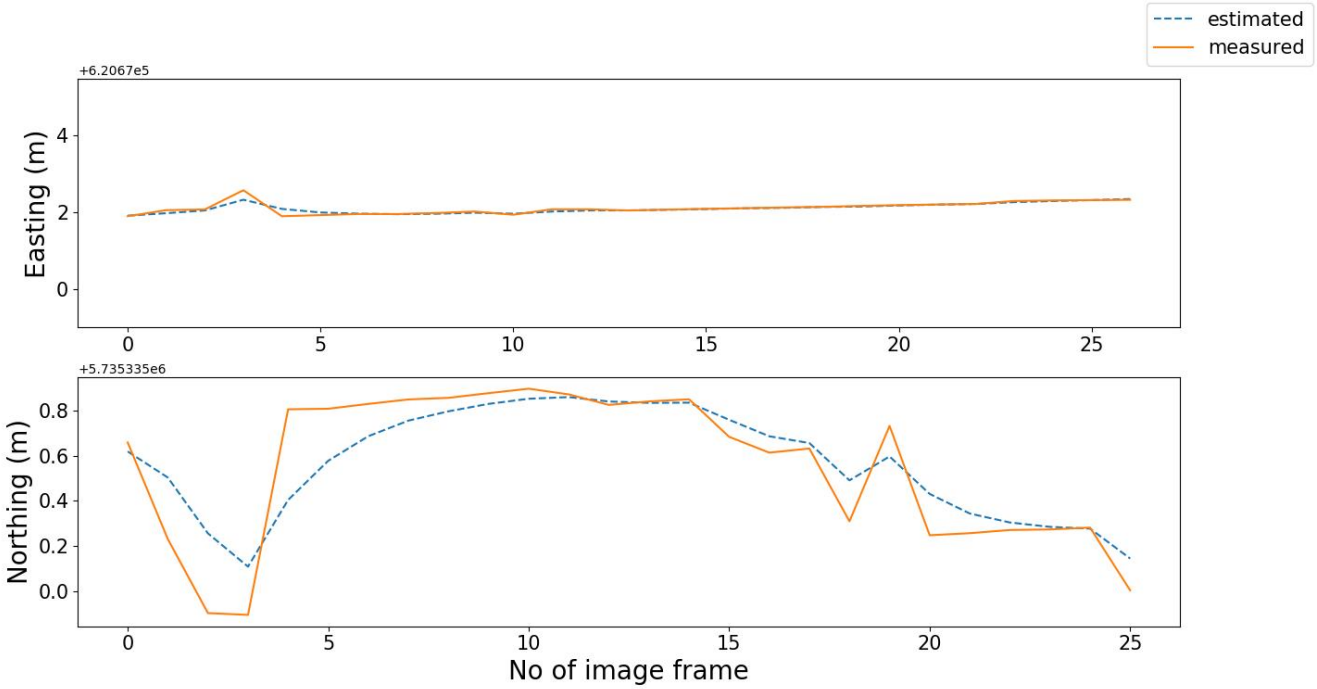

Figure 15: Tracking of Car_1 in Northing and Easting with respect to Number of Image frames

CONCLUSION AND FUTURE WORK

In this paper, an algorithm for object detection, classification, and tracking is presented. Tracking of moving objects for autonomous vehicle is important and is of great importance for the safety and collision avoidance. The data from cameras, LiDAR and INS is fused to achieve this goal. The paper presents how images can be of great source for the detection and classification of the objects around the vehicle. The paper uses the YOLO algorithm to detect and classify the object. This information is then fused with the information from the LiDAR and INS to successfully navigate the RobotCar with tracking and detection of objects.

There are number of ways the algorithm developed in this paper can be improved. The paper assumes the objects as the point object and tracks them. This has resulted in a less than optimal performance in tracking as the laser scan for the same object might be from different parts of the same object as objects are moving. Being able to model the motion of the object detected will be great extension to the work carried out in this paper, such as adding velocity states to moving vehicles. LiDAR in this paper is used to get the distance of the object detected. This can be also be achieved using the stereo camera. If such a sensor were added, it could provide additional information, improving the result.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015. [Online]. Available: http://arxiv.org/abs/1506.02640.

[2] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, Barcelona, 2011.

[3] W. Maddern, G. Pascoe, C. Linegar and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," 2017. [Online]. Available: http://robotcar-dataset.robots.ox.ac.uk/images/robotcar_ijrr.pdf.

[4] J. &. D.-W. H. Leonard, "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," in *IROS*, 1991.

[5] W. Chieh-Chih, C. Thorpe and Thrun.S, "Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicale in Crowded Urban Areas," in *2003 IEEE International Conference on Robotics and Automation*, Taipei, 2003.

[6] C. Mertz, L. E. Navarro-Serment, R. Maclachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel and M. &. T. C. Hebert, "Moving object detection with laser scanners," 2013.

[7] T.-D. Vu., " Vehicle Perception: Localization, Mapping with Detection, Classifcation and Tracking of Moving Objects.," Institut National Polytechnique de Grenoble - INPG, 2009.

[8] O. M. a. P. T. Papageorgiou C. P, "A general framework for object detection," pp. 555-562, 1998.

[9] D. N. a. T. B., "Histograms of oriented gradients for," in *Computer Vision and Pattern Recognition*, 2005.

[10] D. J. T. M. J. Girshick R., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[11] G. R, "Fast R-CNN," 2015.

[12] L. D.G, "Object recognition from local scale-invariant," in *Computer vision, 1999. The proceedings of the seventh IEEE Conference*, 1999.

[13] "Feature Matching," [Online]. Available: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html.

3 EXTENDED REVIEW OF LITERATURE AND EXTENDED METHODOLOGY AND EXTENDED RESULT

3.1 Extended Review of Literature

3.1.1 Sensors

The autonomous vehicle needs the information of the environment to make sense of it. The autonomous vehicle gets the information of the environment through the sensors. The main sensors used by the autonomous vehicle include a camera, LiDAR, and GPS/INS.

3.1.1.1 Camera

Camera is a device that senses the light and captures the images. The camera consists of image sensors and the lenses with certain focal length, f, to capture the images. A camera maps 3D world into 2D image. The mapping from 2D to 3D is done by perspective projection.

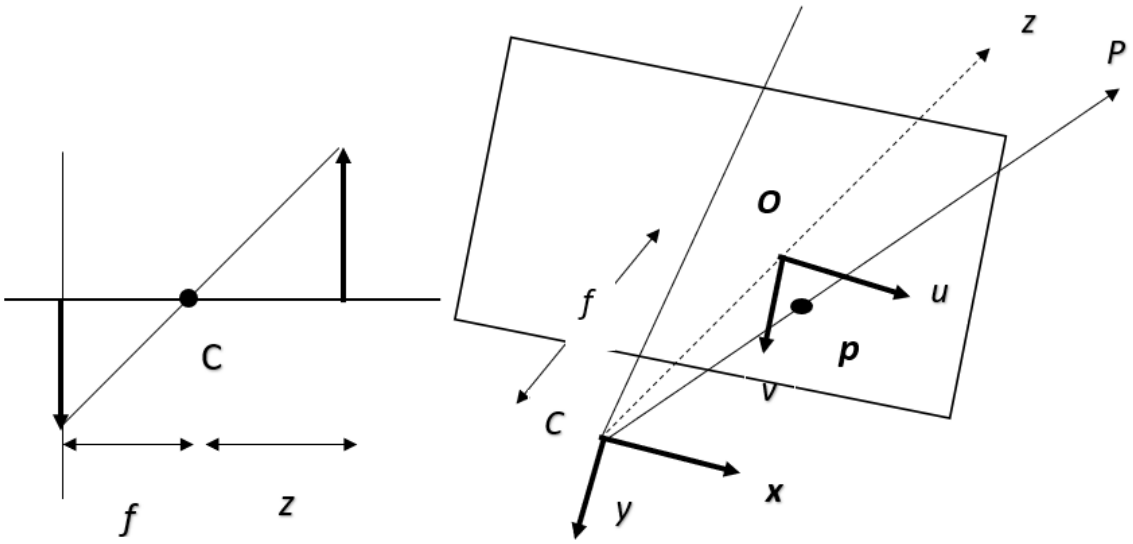

Figure 16: Pin hole model for Camera(left) Pin hole model with image plane (right) [20]

Figure 16 shows the simple model for pin hole camera and the perspective projection for the camera. Point *P* in the real world is to be projected onto point *p* in image plane. Let *(x, y, z)* be the

camera reference frame with origin *C* and z-axis coincident with optical axis and two-dimensional frame reference *(u, v)* for the image plane. If *P= (X, Y, Z)* and *p = (u, v)* then using similarity of triangles,

$$\frac{f}{z} = \frac{u}{x} = \frac{v}{y} \tag{3.1}$$

$$u = \frac{f}{z} x \tag{3.2}$$

$$v = \frac{f}{z} y \tag{3.3}$$

In this way, the 3D-point is mapped to pixels in the images.

The camera used in the thesis is a 16Hz trinocular stereo camera with 3.8mm focal length lens. The images obtained only from the center camera was used for the work in this thesis. The calibration of the camera for intrinsic and extrinsic parameters was done using the software development kit provided along with dataset [3].

3.1.1.2 LiDAR

LiDAR stands for Light Detection and Ranging and is a time of flight sensor used to measure ranges often in particular directions. It consists of a transmitter that illuminates a target with pulsed laser light and a receiver that can detect the bounced light from the object. The distance between the target can be calculated using the speed of light and the time of flight.

3.1.1.3 Inertial Navigation System (INS)

Inertial Navigation System is a device that uses gyroscopes and accelerometers to estimate the relative position, velocity, and the acceleration of a moving body. The outputs are estimates the position (x, y, z) and orientation (roll, pitch and yaw) of the object. The INS has three orthogonal accelerometers and three orthogonal gyroscopes. Gyroscopes measure the angular velocity of the sensor with respect to some reference frame. Taking the consideration of original orientation of the system and integrating the angular velocity the orientation of the object is determined. The accelerometers measure the

instantaneous acceleration of the object. The acceleration is transformed to navigation frame and integrated to obtain the velocity of the object.

3.1.1.4 Global Positioning System

The Global Positioning System is an RF space-based navigation system used to determine position, velocity, and time in a common reference system, anywhere on or near the Earth on a continuous basis. GPS always has 24 satellites operational and synchronized to send transmission signals at the same time. The GPS receiver can calculate a solution based on the signals from four or more satellites, based on the arrival time differences of each signal, which are represented as pseudoranges [14].

3.1.2 Position and Orientation

Points in rigid body can be defined if we attach a coordinate frame to the object. Pose (position and orientation) of the coordinate frame can then be defined with respect to a reference frame. The pose can be used to transform the point from the coordinate frame of the object to the reference frame.

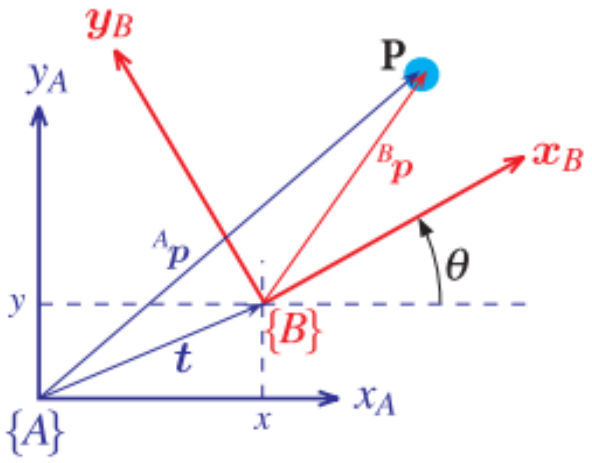

Figure 17: The point P representation in frames A and B [19]

Figure 17 shows the location of point **P** being described as ${}^A\mathbf{p}$ and ${}^B\mathbf{p}$ in frames {A} and {B} respectively. ${}^A\boldsymbol{\xi}_B$ describes the relative pose between frame {A} and {B} and is described by the

displacement ***t*** = *(x, y)* from the origin of the frame {A} and angle θ rotated counter clockwise. The location of point **P** in frame {B} can be described in frame {A} by equation (3.4),

$$ {}^{A}\mathbf{p} = {}^{A}_{B}\boldsymbol{\xi} \bullet {}^{B}\mathbf{p} \tag{3.4} $$

If the displacement *t* = (0,0) the origin of the both frames would be same and frame {B} is only rotated counter clock with respect to frame {A}. Then the point **P** can be represented by rotation matrix given by equation (3.5).

$$ {}^{A}_{B}\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{3.5} $$

Now accounting for the translation ***t*** = (x, y),

$$ {}^{A}\mathbf{p} = {}^{A}_{B}\mathbf{R}\,{}^{B}\mathbf{p} + \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.6} $$

$$ \begin{bmatrix} {}^{A}p_x \\ {}^{A}p_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} {}^{B}p_x \\ {}^{B}p_y \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.7} $$

$$ = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}p_x \\ {}^{B}p_y \\ 1 \end{bmatrix} \tag{3.8} $$

$$ = \begin{bmatrix} {}^{A}_{B}\mathbf{R} & \mathbf{t} \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} {}^{B}\mathbf{p} \tag{3.9} $$

$$ {}^{A}\mathbf{p} = {}^{A}_{B}\mathbf{H}\,{}^{B}\mathbf{p} \tag{3.10} $$

where ${}^{A}_{B}\mathbf{H}$ is a homogenous transformation matrix.

In similar way, orientation can be represented in 3D space, using three angles of rotation about each axis and the translation can be represented by a vector **t** = (x, y, z). The rotation can be considered as a sequence of rotation about different coordinate axes. The orthonormal rotation matrices for rotation of θ about the x, y, and z axis is given by equation 3.11, 3.12 and 3.13.

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{3.11}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.12}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

3.1.3 Convolutional Neural Networks

Convolutional Neural Network (CNN) is one type of artificial neural networks, which is designed to be trained with a set of known images and learn from them. An image is represented by a three-dimensional matrix, with height, width, and depth. Depth represents the three color channels Red, Green, and Blue. If regular artificial neural networks (ANNs) are used to train the image, the size of the input layer would be a large set of individual inputs representing each individual color channel of each pixel. For example, image with resolution 50x50x3 would have 7500 inputs making the neural network very large. CNN is designed so that it can take advantage of depth and model the image as a volume and train with it. A depiction of the difference between a regular artificial neural network and the CNN is shown in the Figure 18 [15].

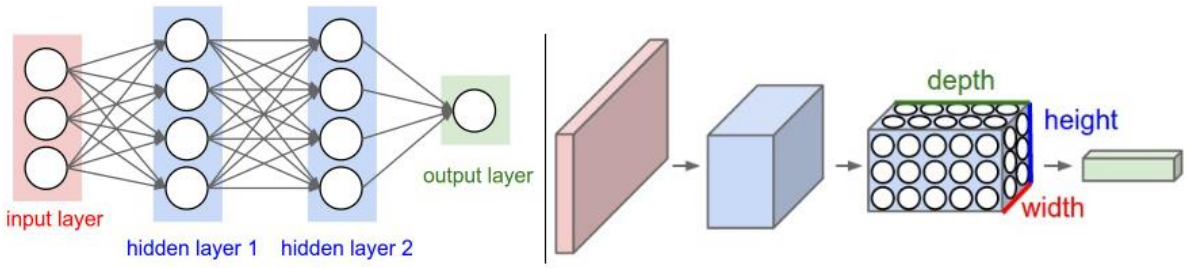

Figure 18: Regular Neural network (left) and CNN (right) [15]

Each layer of CNN transforms the input volume to an output volume. CNNs have the weights and biases and then outputs through a non-linear activation. While CNNs are trained with input images, these parameters are updated, resulting in the learning from the inputs. The learned parameters can then be used to detect the learned pattern in images. The deeper the layer, the more abstract the pattern can be. First layers of a CNN might learn about edges and lines in the images while a deeper layer in the CNN would learn about faces or certain shapes. CNNs are used to detect objects in the images and image captioning.

In this thesis, the object detection and classification are done using an algorithm called YOLO, which uses a CNN architecture.

3.1.3.1 You Only Look Once (YOLO)

You only look once is an algorithm based on CNN used to detect, classify, and localize objects in an image. Figure 19 shows working of YOLO. When objects are to be detected in an image, first it is resized into the size of the input layer of the YOLO architecture, then the image is run through the CNN and the output is the bounding box for each detected object with classification and probability score.

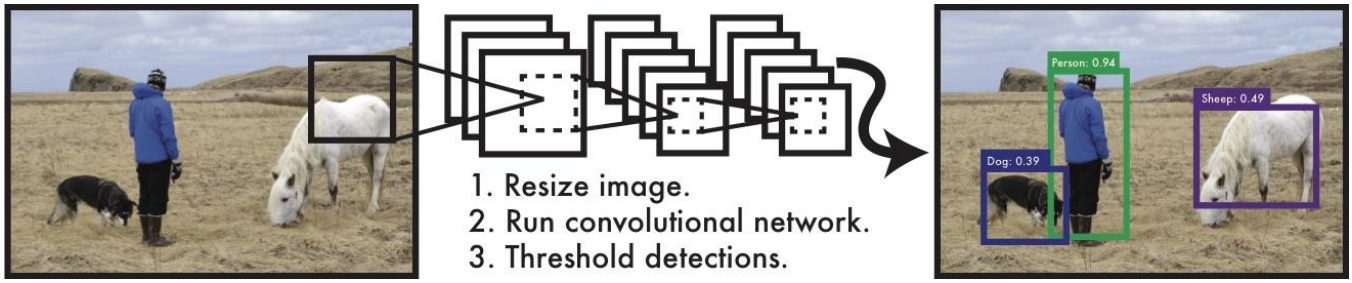

Figure 19: YOLO detecting, classifying and localizing objects in an image

YOLO can see an entire image at once and simultaneously predicts the multiple bounding boxes and class probabilities for each box, making it faster compared to other region-based method or sliding window methods such as DPM, R-CNN, Fast R-CNN, and Faster R-CNN. Figure 20 shows the performance of each of these method speed and accuracy on Nvidia Titan X GPU and the same test dataset Pascal 2007 [16]. The Faster R-CNN is accurate by 4 percent compared to YOLO, but YOLO's

speed is far more superior. This makes YOLO suitable for using in real-time systems such as autonomous vehicles.

	Pascal 2007	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	69.0	45 FPS	22 ms/img

Figure 20: Performance of different object detection Algorithm

The working of YOLO is illustrated in the Figure 21. The figure is split into 13x13 grid. For each grid square, YOLO predicts 5 bounding boxes, with object probability, and class probability. The probability score for the bounding box and the class probability is combined into one final score that tells probability that this bounding box contains a specific type of object. For a 13x13 grid, there will be 845 bounding boxes predicted by the YOLO, but only the bounding boxes crossing a predefined threshold is kept. In this thesis, the bounding boxes with a probability greater than 50% are kept for the object.

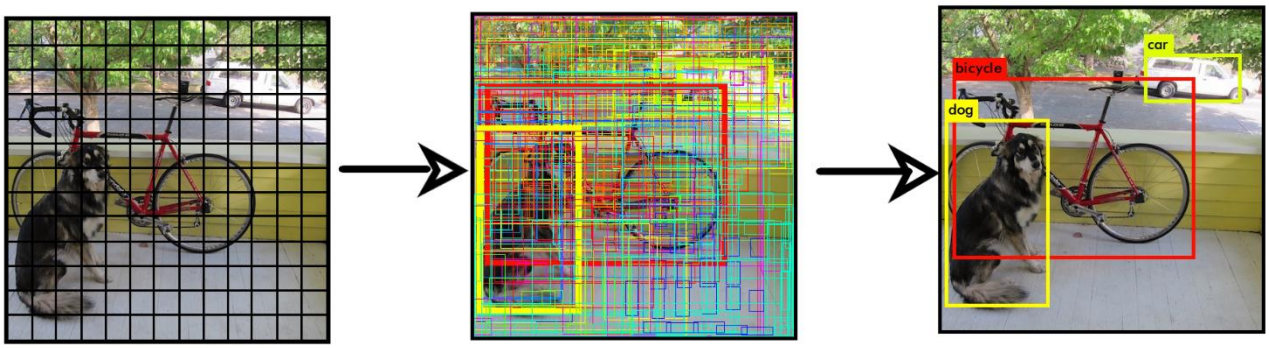

Figure 21: Working of YOLO

In this thesis, YOLO is used to detect six classes of objects. The classes include person, bicycle, motor bike, car, truck, and bus. The pretrained YOLO network obtained from [17] was used to detect and classify objects in the thesis.

3.1.4 ORB (Oriented FAST and Rotated BRIEF)

ORB is a very fast algorithm that creates a feature vector from the detected keypoints. ORB is invariant to rotations and changes in illumination and noise [2]. The first step in ORB is used to find the keypoints in an image, which is done by using the FAST algorithm. FAST stands for Features from Accelerated Segments Test, which finds keypoints by looking at changes in intensity around a pixel in an image. These keypoints give us information about the edges in an image. After the keypoints are calculated the top *N* points are selected using Harris corner measure. ORB uses BRIEF to create binary descriptors from the keypoints. BRIEF starts by smoothing a given image with a Gaussian Kernel in order to prevent the descriptor from being too sensitive to high frequency noise.

3.1.5 Extended Kalman Filter

Bayesian filtering is the process of estimating states over time, using incoming measurements and a mathematical process model. A system is represented as a probabilistic state-space model where the noise is additive and Gaussian. The Kalman filter is a special case of Bayesian filter where the model is linear and gaussian. The Extended Kalman filter (EKF) is a non-linear version of the Kalman filter. The EKF is obtained using a linear approximation of a non-linear system.

The dynamics of the nonlinear system is given by equation 3.14 and 3.15,

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{3.14}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{3.15}$$

where $\boldsymbol{x}_k$ is the state at time *k*, and $\boldsymbol{z}_k$ is the measurement for the system at time *k*, and $\boldsymbol{u}_k$ is the input to the system. The dynamics of the system is modeled by $f(\bullet)$ and $h(\bullet)$ is measurement model for the system. $\boldsymbol{w}_k$ is the process noise with zero mean and covariance $\mathbf{Q}_k$

$$\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \tag{3.16}$$

$\boldsymbol{v}_k$ is motion noise with zero mean and covariance R_k,

$$\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k) \tag{3.17}$$

In EKF, the state of the system at any time *k* is assumed as a Gaussian approximation with, mean $\bar{\mathbf{x}}_k$ and the covariance $\mathbf{P}_k$.

$$\mathbf{x}_k \sim \mathcal{N}(\bar{\mathbf{x}}_k, \mathbf{P}_k) \tag{3.18}$$

The non-linearity in EKF approximated as linear system near a point using Taylor series and is modeled using Jacobian matrix.

The state estimation of the system is done in two stage. First the predict stage and the update stage. The prediction state of EKF is given in following equations. $\hat{\mathbf{x}}_k$ is predicted estimate of the state based on prior information, $\hat{\mathbf{P}}_k$ is the predicted estimate of the covariance matrix.

$$\hat{\mathbf{x}}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{3.19}$$

$$\hat{\mathbf{P}}_k = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{Q}_{k-1} \tag{3.20}$$

where F_k is the Jacobian matrix given by equation (3.20)

$$\mathbf{F}_k = \frac{\delta f}{\delta x}|_{\hat{x}_{k-1}, u_k} \tag{3.21}$$

The update state of EKF is done by following equations,

$$\mathbf{S}_k = \mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R}_k \tag{3.22}$$

$$\mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^T \mathbf{S}_k^{-1} \tag{3.23}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k)) \tag{3.24}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\hat{\mathbf{P}}_k \tag{3.25}$$

where H_k is the Jacobian matrix given by equation (3.26)

$$\mathbf{H}_k = \frac{\delta \mathbf{h}}{\delta x}|_{\hat{x}_{k|k-1}} \tag{3.26}$$

where $\mathbf{K}_k$ is the Kalman gain, $\boldsymbol{S}_k$ is the innovation covariance matrix. The Kalman gain tells us trust in the measurements. If the sensor gives bad measurements, the Kalman gain will be low and high if the sensors give reliable measurements. The innovation covariance matrix $\boldsymbol{S}_k$ is the covariance matrix of the measurements and tells about the errors in measurements.

3.2 Extended Result

The result in the manuscript section is extended in this section. In the paper, the algorithm developed in the thesis was applied to a road with normal conditions. In the extended results, the algorithm is applied to two additional road conditions: night conditions and snowy conditions.

3.2.1 Results with Snowy Road Condition

The performance of the camera greatly depends on the lighting condition and contrast. The lighting condition during the snowy weather is more diffuse than during normal conditions. Since the

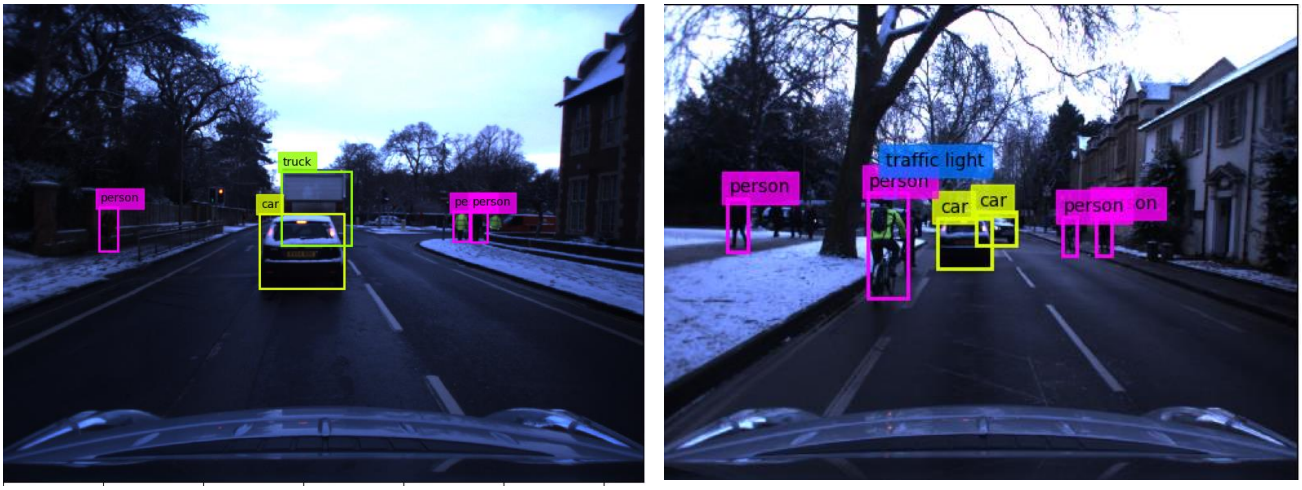

Figure 22: Object Detection by YOLO in snowy conditions

lighting conditions are different, the object on the images might not be clear or result in a match. Some of the images captured by the RobotCar during the snow along with the result of the object detection by the YOLO algorithm is shown in Figure 22.

Figure 22 shows that even though the car is covered by snow and the lighting condition is different from normal sunny day, the YOLO detects most of the objects around the RobotCar. It misses a few pedestrians that are far from the RobotCar, but it does not miss any that are nearby. From this it is possible to use YOLO to detect object even in snowy conditions.

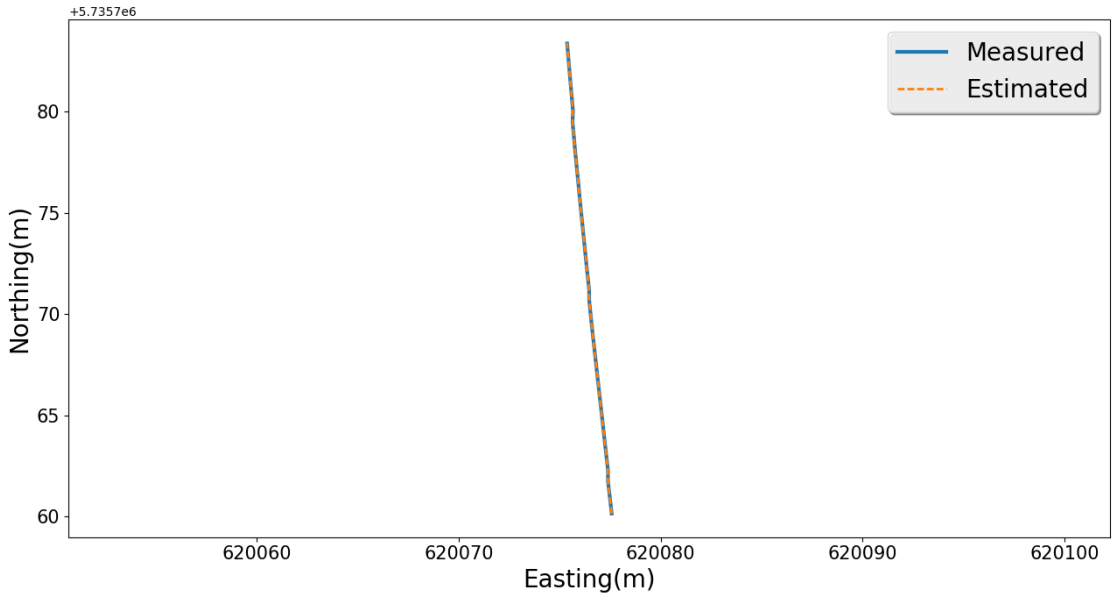

Figure 23: Navigation of the RobotCar in the Snow

The navigation of the RobotCar is presented in the Figure 23. We can see that the position estimated by the EKF closely follows the position of the RobotCar obtained from the GPS. The absolute deviation in the Northing and Easting is presented in the Figure 24. It is observed that the deviation in Northing is about 0.025m and in Easting it is 0.005m.

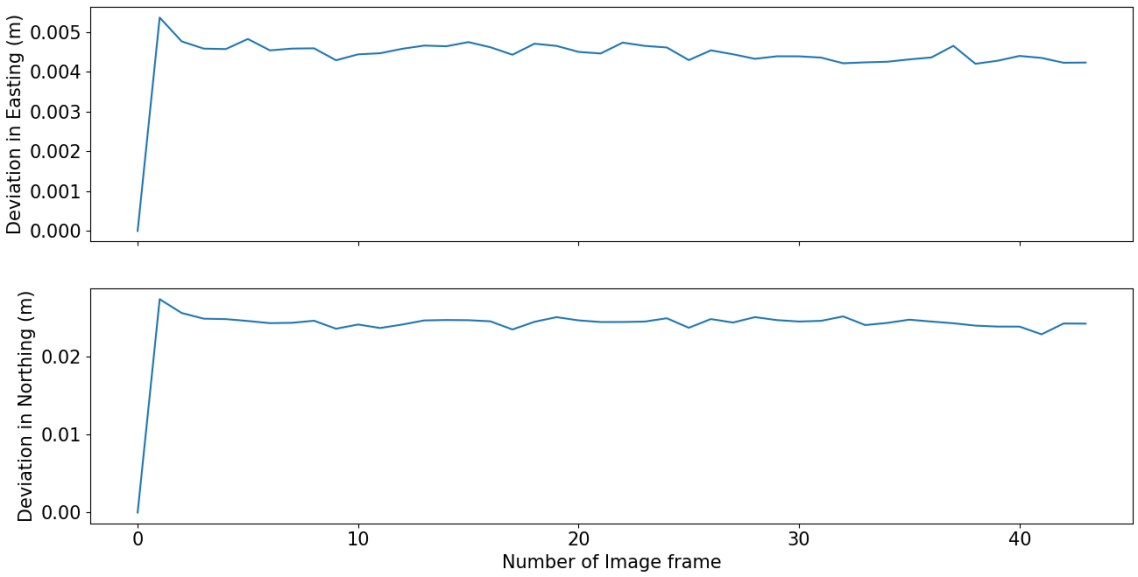

Figure 24: Deviation in Easting and Northing

The tracking of the moving objects during snowy is presented in Figure 25. The image in Figure 20 consists of three objects, two persons and a bus travelling in a opposite direction. Persons detected in the image were tracked for 10 frames and the bus was tracked for 37 frames by the algorithm developed in this thesis. This demonstrates that even in low light and snow, the feature matching algorithm developed in the thesis can match objects for multiple frames.

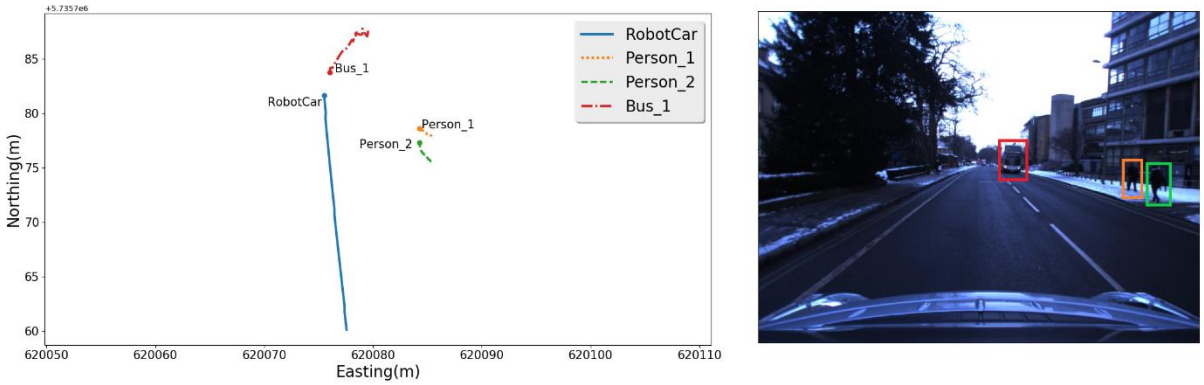

Figure 25: Tracking of Multiple Objects in Snowy Condition

Figure 26 shows the tracking of a person when the car is stopped at a red light. The car is stationary while the person is walking across the street as shown in the figure. This demonstrates the algorithm's ability to track the object even when at a stop.

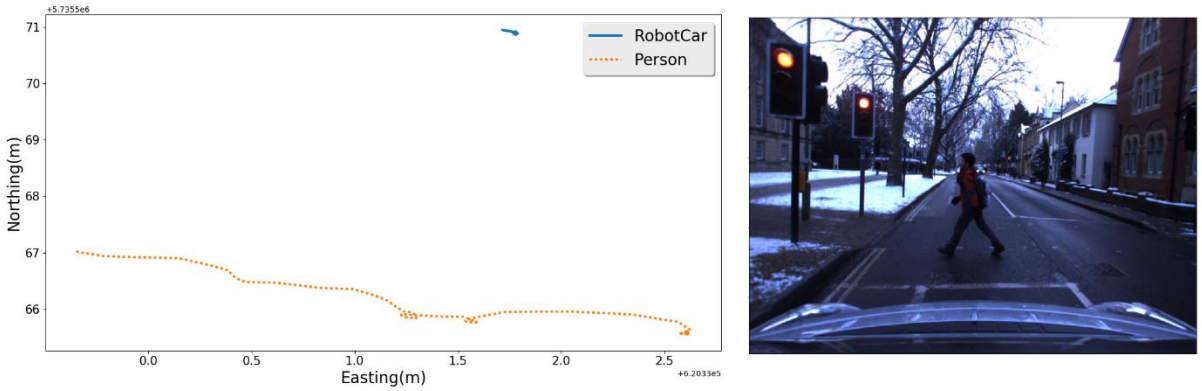

Figure 26: Tracking of Person when RobotCar is stopped in Navigation Frame

3.2.2 Results During the Night Time

Cameras are not considered to be good sensors for capturing the images at night time because of low light. The performance of the YOLO at night time for object detection is presented in Figure 27. This figure shows the object detected by YOLO in night time. The image in left shows how YOLO can detect two buses and a person. However, it missed the bicycle that the person is riding. In the image on

Figure 27: Object detection at Night Time by YOLO

the right, it can detect the bus and person in the same image but misses the bus that is quite far from the RobotCar that was previously detected in the left image. The result is still promising given that the bus detected in the right image is blurry. YOLO is trained to detect objects in such conditions. The performance of YOLO for object detection and classification can be considered good even in low light conditions, as it is able to detect the objects that are nearby the car that needs to be tracked.

Now the tracking of object during the night time is presented. In this case, a bus riding side by side the RobotCar is tracked. The graph of the tracking in navigation frame is presented below in the Figure 28, which demonstrates that, even during the night time, the tracking of object is possible using the algorithm developed in the thesis. Figure 28 shows person riding the bike, which is detected by YOLO and is tracked. Unfortunately, in every image frame, it is tracked as a new object. This is the result of the object's small size relative to the resolution of the image. In addition, the fact that it is dark contributes to the lack of keypoints to match it from one image to another.

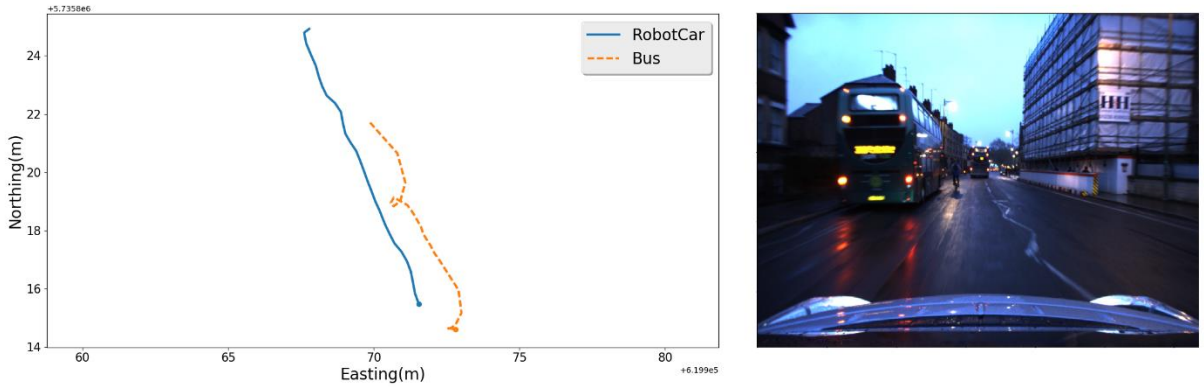

Figure 28: Tracking of Bus moving in same direction in navigation frame

3.3 Extended Conclusion and Future work

In the thesis, an algorithm for detection, classification and tracking of objects was developed. The algorithm developed was tested in three different road conditions: normal sunny day, night time, and

during snowy weather. The objects were detected and classified using the YOLO. The performance of YOLO in different weather condition is presented in the result section. YOLO performs well under all three test conditions, including at night time and during snowy condition. In the results section, it was observed that YOLO missed a few objects during the two more adverse conditions but did not miss objects that were nearby the autonomous car and of importance for navigating safely. Images contain different amounts of information about the environment based on the lighting conditions and this affects the performance of YOLO on object detection and classification.

Once the objects are detected and classified by YOLO, they are tracked using EKF given that the same object from one frame to another is matched using ORB. Using EKF, the state of the Robotcar was estimated, aiding in the localization of the RobotCar as well as tracking the position of objects detected in the navigation frame. The algorithm developed in the paper was able to track the multiple moving objects simultaneously and objects that are stationary. This was seen in the example where the algorithm tracked a pedestrian crossing the road when the RobotCar was stopped at a traffic light.

There are a number of ways the work developed in the thesis can be extended. The performance of the camera varies in different lighting conditions, and this affects the algorithm developed in this thesis. Using a camera as the main perception sensor for object detection may not be ideal for all weather condition. The performance of LiDAR remains undegraded at night time, so information from the laser scans can be combined with the images for the detection of the objects. Also matching objects from one from to another frame is performed using images only, this can be improved by adding motion modelling to the objects.

The algorithm developed in the thesis is also limited by only tracking moving and stationary objects and does not focus on the mapping the environment around the Robotcar. Being able to map the

environment using the information from camera and LiDAR would help the RobotCar to localize in the environment and navigate safely even in places with poor GPS reception.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2015. [Online]. Available: http://arxiv.org/abs/1506.02640.

[2] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, Barcelona, 2011.

[3] W. Maddern, G. Pascoe, C. Linegar and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," 2017. [Online]. Available: http://robotcar-dataset.robots.ox.ac.uk/images/robotcar_ijrr.pdf.

[4] J. &. D.-W. H. Leonard, "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," in *IROS*, 1991.

[5] W. Chieh-Chih, C. Thorpe and Thrun.S, "Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicale in Crowded Urban Areas," in *2003 IEEE International Conference on Robotics and Automation*, Taipei, 2003.

[6] C. Mertz, L. E. Navarro-Serment, R. Maclachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel and M. &. T. C. Hebert, "Moving object detection with laser scanners," 2013.

[7] T.-D. Vu., " Vehicle Perception: Localization, Mapping with Detection, Classifcation and Tracking of Moving Objects.," Institut National Polytechnique de Grenoble - INPG, 2009.

[8] O. M. a. P. T. Papageorgiou C. P, "A general framework for object detection," pp. 555-562, 1998.

[9] D. N. a. T. B., "Histograms of oriented gradients for," in *Computer Vision and Pattern Recognition*, 2005.

[10] D. J. T. M. J. Girshick R., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[11] G. R, "Fast R-CNN," 2015.

[12] L. D.G, "Object recognition from local scale-invariant," in *Computer vision, 1999. The proceedings of the seventh IEEE Conference*, 1999.

[13] "Feature Matching," [Online]. Available: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html.

[14] H. L. &. J. C. B. Hofmann-Wellenhof, GPS Theory and Practice, Springer, 2001.

[15] "Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[16] "The PASCAL Visual Object Classes Challenge 2007," [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/.

[17] J. Redmond, "YOLO: Real-Time Object Detection," [Online]. Available: https://pjreddie.com/darknet/yolo/.

[18] "Convolutional Neural Networks for visual Recognition," [Online]. Available: http://cs231n.github.io/convolutional-networks/.

[19] P. Corke, Robotics, vision and control fundamental algorithms in MATLAB, Springer, 2011.

[20] I. R. N. &. D. S. R. Siegwart, Introduction to Autonomous Mobile Robots, 2011.