FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Kalman Filter Approach to Movement Prediction in Automated Driving Scenarios

José João Pereira Oliveira

DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Rosaldo J. F. Rossetti, PhD

Second Supervisor: Eng.ª Filipa Ramos, MSc

June 28, 2019

A Kalman Filter Approach to Movement Prediction in Automated Driving Scenarios

José João Pereira Oliveira

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Ana Paula Rocha, PhD
External Examiner: João Emílio Almeida, PhD
Supervisor: Rosaldo J. F. Rossetti, PhD
June 28, 2019

Abstract

Autonomous vehicles are a dream almost as old as vehicles themselves. Achieving full vehicle autonomy could be one of the biggest daily routine changers for thousands of people around the world. Just take into account the amount of time that people lose to commute every day and how that time could be redirected to a panoply of other activities that go from leisure-related to work-related activities. It would also lead to an increase in the quality of life as well as a decrease in road accidents.

Following the topic of autonomous driving, object tracking is one of the stepping stones of self-driving vehicles. It can be characterized as the ability to estimate the trajectory of an object based on the motion it shows across a sequence of frames. Another part that has relevance in the context of autonomous driving is movement prediction which can be defined as the skill to forecast what positions will be occupied by a certain object in the future.

Given this, a solution is proposed in order to solve object tracking and movement prediction based on LiDAR scans and RGB images. The presented work consists of two parts, being the first one responsible for providing the 3D centers of the objects detected and the second one capable of tracking and predicting the movement of those objects. The explored pipeline leverages on the use of YOLO for 2D object detection followed by Frustum PointNet that extracts the corresponding 3D centers. Exploring the temporal evolution of the position of those centers, a tracker makes use of the Euclidean distance to associate them. Furthermore, a Kalman filter-based motion predictor is adapted to 3D space by assigning a filter to each cartesian coordinate axis.

In a context where there is no error induced in the detected centers received by the tracker, the presented solution shows very satisfactory results, in a dataset (part of KITTI Vision dataset) that is composed of 544 trajectories of cars. Those results translate into an overall precision of 91.39%, 42 ID switches and 41 trajectory fragmentations. Moreover, it correctly tracked, for more than 80% of its length, 87.45% of all the trajectories. The motion predictor delivers great results having an average error of 0.63 meters for the X-axis and of 1.03 meters for the Y-axis. Even in a scenario with 50% update failures, it showcases an average error close to the minimum value in ideal conditions, reaching an average error of 0.83 meters for X-axis and of 1.35 meters for the Y-axis.

Keywords: *Autonomous Driving, Object Tracking, Motion Prediction, Deep Learning, RGB image, LiDAR scan, Euclidean distance, Kalman filter*

Resumo

Os veículos autónomos são um sonho quase tão antigo como os carros por si só. Atingir autonomia completa para veículos seria um dos maiores modificadores da rotina diária de milhares de pessoas por todo o mundo. Basta ter em conta a quantidade de tempo perdido todos os dias em comutações e como esse tempo poderia ser redirecionado para uma panóplia de outras atividades que vão de atividades de lazer a atividades relacionadas com o trabalho. Levaria também levar a um aumento da qualidade de vida assim como um decréscimo no número de acidentes rodoviários.

Seguindo o tópico de condução autónoma, o rastreamento de objetos é uma das bases dos veículos autónomos e pode ser caracterizado como a capacidade de estimar a trajetória de um objeto baseada no seu movimento ao longo de uma sequência de momentos. Outra parte que tem relevância no contexto de condução autónoma é a previsão de movimento que pode ser definida como a habilidade de prever que posições serão ocupadas por um certo objeto no futuro.

Posto isto, uma solução é proposta de maneira a resolver o rastreamento de objectos e a previsão de movimento baseados em *scans* LiDAR e imagens RGB. O trabaho apresentado consiste em duas partes, sendo a primeira responsável por fornecer os centros 3D dos objetos detetados e a segunda capaz de rastrear e prever o movimento desses objetos. A *pipeline* explorada faz uso do YOLO para deteção 2D de objetos e de seguida do Frustum PointNet que extrai os correspondentes pontos 3D. A exploração da evolução temporal da posição desses centros, o rastreador faz uso da distância Euclidiana para os associar. Além disso um módulo que prevê o movimento baseado no Filtro de Kalman é adaptado ao espaço 3D atribuindo um filtro a cada eixo de coordenadas cartesianas.

Num contexto em que não há erro induzido nos centros detetados recebidos pelo rastreador, a solução apresentada mostra resultados muitos satisfatórios, num *dataset* (que faz parte do *dataset* "KITTI Vision") composto por 544 trajetórias de carros. que se traduzem numa precisão global de 91.39%, e apenas 42 trocas de ID e 41 fragmentações de trajetória, e ainda rastreou corretamente, por mais de 80% do seu comprimento total, 87.45% de todas as trajetórias. O módulo previsor de movimento fornece resultados promissores apenas mostrando um erro médio de 0.63 metros para o eixo dos X e de 1.03 metros para o eixo dos Y. Mesmo num cenário em que há 50% de erros nos *updates*, erro médio de 0.83 metros para o eixo dos X e de 1.35 metros para o eixo dos Y.

Palavras-Chave: *Condução Autónoma, Rastreamento de Objetos, Previsão de Movimento, Deep Learning, Imagem RGB, scan LiDAR, Distância Euclidiana, Filtro de Kalman*

Acknowledgements

Thank you... ...to Bosch, for giving my first taste of professional "waters" and believing in me to embrace this project. To the LiDAR team, for receiving me so well and for making wake up at 7:00 am every day for the last 5 months easier. To the Innovation Team for all the patience, help and all the non-sense questions answered. And a special one to of my supervisors, Filipa, for being the compass I didn't knew I needed, for all the willingness to help me and make me go that extra mile and for all the "pontos e bírgulas" attention calls.

...to my other supervisor, professor Rosaldo Rossetti, for always having time in his busy schedule for a weekly meeting and answering my messages and e-mails.

...to my friend Pedro Sá and my cousin Paulo Jorge, for always having time for an advice and a tech talk.

...to **random.org*, for being the best fleet I could ask for during this long trip, for welcoming a tall guy with a funny accent, for the random and creative verbiage that flooded the chat in the hours preceding big deliveries and most of all for being my family in Porto during these 5 years of "fair winds and following seas" and some waves that seemed too high to overcome.

...to *Dr. Fonte Show*, for making home always feel like home every time we had the chance to be together.

...to Porto and its people, for welcoming and receiving so well, five years ago, a 17 years old kid and making him feel one of them.

...to Krakow and the people I met there, for making those cold, low daylight and alcohol-rinsed months so extraordinary and remarkable.

Last but not least, to my family, for always instigating me to know more and always try to go further. To my cousin Vanessa for showing me Porto and always be there on those weekends that home seemed really far away. To my aunt Cristina for always having a roof available when I had to stay overnight in São Miguel. To my aunt Fátima, for the immense support since always. Lastly but firstly, to my mother and my father, for some of the things they set aside to make sure me and my siblings always had everything we needed, I will always owe you that, for inspiring me to be a better person everyday and for being my two anchors and my north star when the current seemed too strong and there was no land in sight.

José Oliveira

"The further you get away from yourself, the more challenging it is.
Not to be in your comfort zone is great fun."

Benedict Cumberbatch

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ACC	Adaptive Cruise Control
ALFD	Aggregated Local Flow Descriptor
ADAS	Advanced Driver-Assistance System
CNN	Convolutional Neural Network
GMM	Gaussian Mixture Model
IPT	Interest Point Trajectory
KF	Kalman Filter
LiDAR	Light Detection and Range
MLS	Mobile Laser Scanning
MRF	Markov Random Field
MOT	Multi-Object Tracking
NMOT	Near Multi-Object Tracking
R-CNN	Region Based Convolutional Neural Network

Chapter 1

Introduction

Object tracking is one of the crucial phases involved in autonomous driving. It can be defined as the ability to estimate the trajectory of an object based on the motion it shows across a sequence of frames. Another phase that is not considered as important as tracking but that can have an important role mainly regarding road safety, not only of the vehicle itself but also of people, objects and other vehicles around it, is movement prediction. That can be defined as the skill to forecast what positions will be occupied by a certain object in the future. It is on these two phases of autonomous driving that this dissertation will focus on.

On this chapter, an introduction to the environment in which this dissertation was developed is presented and the project itself is framed regarding the technology fields. In section 1.2, the motivation that drove the project is presented and the goal this dissertation aspires to achieve defined. It ends with section 1.3, where the structure of the document is presented and what to expect in each part of it is explained.

1.1 Context

This work was developed at Bosch Car Multimedia Portugal, SA, as part of their project of the object tracking and movement prediction techniques for autonomous and assisted driving systems, during a five months internship, in their facilities in Braga.

Bosch Car Multimedia Portugal, SA is a branch of the group Bosch that is in Portugal since 1911 and offers a panoply of products ranging from home appliances to industrial products, passing by mobility solutions like the ones developed by Bosch Car Multimedia.

Bosch Car Multimedia Portugal aims at providing intelligent solutions for mobility scenarios, those solutions include connected and automated mobility as well as power-train systems and electrified mobility.

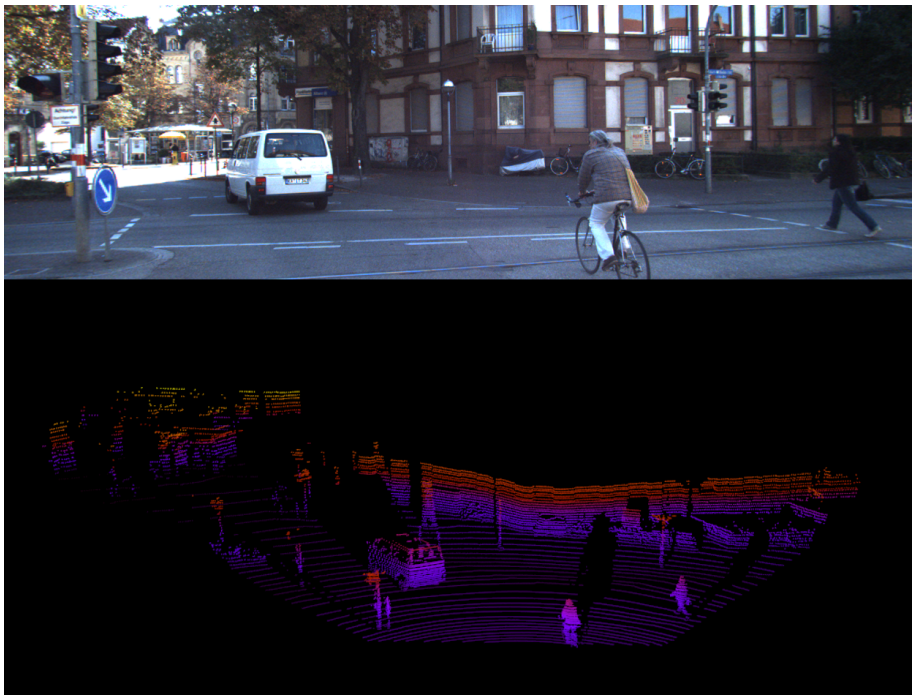


Figure 1.1: Image and LiDAR scan of a scene (image 0 from sequence 0 of KITTI dataset).

The technological areas this project belongs to includes autonomous driving, as the main area and as sub areas object tracking and motion prediction. Regarding data retrieval, the 2D data is obtain through RGB imagery and the 3D through LiDAR scanning (figure 1.1).

1.2 Motivation and Goals

The main problem that this work aims to solve is related to the long-lasting dream of achieving full-autonomous vehicles. Autonomous vehicles are a fantasy almost as old as vehicles themselves, imagined by scientists and engineers and dreamt by tech and cars enthusiasts. Just take a look at all the science fiction from the 60s to the 80s, and see TV shows like "Knight Rider" where one of the main characters was KITT, a self-conscious car, or "The Jetsons" and how they depict life in the future.

This desire of achieving full vehicle autonomy could be one of the biggest daily routine changers for thousands of people around the world, just take into account the amount of time that people lose to commute every day and how that time could be redirected to a panoply of other activities that go from leisure-related to work-related activities. It would also lead to an increase in the quality of life as well as a decrease on road accidents.

Toth [Tot09] predicted that in the 2010s the first operational autonomous vehicles would appear and he might have failed his prediction for just a couple of years. It is close to become reality, but for it to be accomplished the security and safety of the methods and tools used has to

be proven. This is one of the things related with autonomous vehicles that society is more apprehensive about, just take into account the repercussion in the media a self-driving car accident has compared with 20 normal car accidents. Nevertheless, much progress in this area, which can be tested in appropriate simulation environments prior to deployment [PR12], is advancing currently in a quite fast pace.

Taking all this into account there is a need to deal with all the road scenarios that may arise, from the most simple to the most complex. Scenarios which include the most varied situations, from the urban ones with a high pedestrian presence where at any moment a pedestrian can show up in the vehicle's route, to highway scenarios without many vehicles but that during night at any moment an animal can cross the road and in the future some even more complex scenarios like preparing for a kid to come on the street after a ball appears.

Although they differ in their generality, both scenarios have in common the fact that in both there is a need to control the objects around the vehicle. Controlling them involves tracking and predicting their movement. One the most important stages to deal with such scenarios comprises of two parts: first, identifying the objects surrounding the vehicle and second, tracking and predicting the features of the objects' movement (being that the most important ones are velocity and direction).

It's on the second part, tracking and predicting the movement, that this work will focus. The main objective is to use the information provided by an RGB camera and a LiDAR scan mounted on a vehicle to track in the best possible way the objects around, using a heuristic based on the minimum Euclidean distance, and at the same time predict their movement, using an implementation of the Kalman filter.

1.3 Document's Structure

The rest of this document is composed of four more chapters and four appendixes that are presented below.

Chapter 2 The current state of the art as well as related projects about identification, classification, and tracking in LiDAR scans are presented. Other relevant information is also presented, this information includes available MOT datasets, what metrics are used to evaluate MOT solutions and finally the benchmarks that combine these datasets and these metrics.

Chapter 3 The developed solution is described. Firstly, an overview of the pipeline is provided followed by an in-depth explanation of each module and how data is processed and flows in the different modules, from its raw format to the final center predictions.

Chapter 4 The experiments, done to evaluate and tune the performance of the solution developed, are presented and explained in this chapter and their results discussed. The goal of these

experiments is to find out the best possible values for the parameters of both the tracker and the predictor.

Chapter 5 A summary of all the work developed is provided with the purpose of informing the reader on what was produced. An analysis is performed on the proposed solution, identifying its strengths and weaknesses. Following the identified weaknesses, paths for improvement are purposed as well as work on other areas that can lead to different enhancements.

Chapter 2

Literature Review

In this chapter, an analysis of the state of the art methods and approaches related to Multi-Object Tracking is presented. They cover not only MOT algorithms but also algorithms of object detection given their relation of support to the MOT ones. The available MOT datasets, the benchmarks and the metrics used by the said benchmarks are also explored.

2.1 Object Detection

Object detection is the phase that precedes the object tracking. Without it, object tracking (both single and multiple variants) would not be possible so this section will provide an overview of some of the most recent and largely used approaches. The majority of solutions that present better results in this area appeared during this decade. The growing effort to have better results in computer vision is due to the big spectrum of applications it has, as well as the diverse areas it can be used in, naturally including Intelligent Transportation Systems [LRB09, LKR+16, NSR18]. The literature review of this part will analyze the following proposals for 3D detection: Mono3D from Chen *et al.* [CKZ+16]; 3DOP from Chen *et al.*[CKZ+15]; MV3D from Chen *et al.* [CMW+17]; F-PointNet from Qi *et al.* [QLW+18]; VeloFCN from Li [Li17] and RetinaNet from Lin *et al.* [LGG+18].

Before talking about the 3D methods it is important to do a quick overview of the methods that are seen as state-of-the-art for 2D detection. One of them is RetinaNet from Lin *et al.* [LGG+18] and it consists of simple a dense detector that aims to tackle the problem that they identified as being the main cause of one-stage detectors being less accurate than the two-stage ones. The problem detected is the extreme foreground-background class imbalance present in the training of dense detectors. They solved it by reshaping the cross-entropy loss so that it down-weights the loss assigned to examples that were correctly classified. According to them, *RetinaNet* matches the speed of one-stage detectors and has better accuracy than all existing state-of-the-art two-stage detectors. They evaluate it on the bounding box detection task of the COCO dataset [CUF18] and compare it with the current methods with the best results, both one-stage and two-stage. Another approach, more simple and easier to adapt, is YOLOv3 (You Only Look Once version 3) [RF18]

from Redmon and Farhadi. It predicts four coordinates for each bounding box, then predicts a score for each of them using logistic regression. It is also capable of classifying the object. For that, the multilabel classification is performed without softmax because it does not allow overlapping classes and multi-labeling is sometimes necessary (i.e. Person, Bicycle and Cyclist). Feature extraction is done using a concept similar to feature pyramid networks. They add a set of convolutional layers to their feature extractor, where the last of these layers predicts the bounding box, the score, and the class. Their feature extractor is a new network that is a mixed approach between the network used in YOLOv2 (also known as YOLO9000) [RF17], Darknet-19, and a residual network, resulting in a network with fifty-three convolutional layers that was named Darknet-53. This resulting network presents higher efficiency than ResNet-101 and ResNet-152 [HZRS16].

Now talking about 3D object detection, there are several approaches with state-of-the-art results. The ones considered more relevant are presented below.

3DOP from Chen *et al.*[CKZ+15] explores stereo imagery and contextual models related to autonomous driving to place suggestions on where objects can be. These suggestions are presented in the form of cuboids. The formulation of the problem consists of an inference in a Markov random field encoding ground plane, object size priors and a variety of depth informed features. Another solution presented by Chen is Mono3D [CKZ+16] performs the 3D detection of objects from monocular images. Firstly, it generates a set of object proposals specific for different classes, that are then passed through a normal CNN pipeline to increase the precision of the detections. They propose an approach, based on energy minimization, that generates the object candidates by assuming that they should be at ground level. After that, each candidate is ranked with criteria that include intuitive potentials encoding semantic segmentation, contextual information, size and location priors, and typical object shape. A more recent solution also from Chen is MV3D [CMW+17], and it consists of an object detection network that has as input RGB images and LiDAR point clouds. From this data, it predicts 3D bounding boxes. The point cloud is encoded into a compact multi-view representation. The network they present is split into two sub-networks, one that fuses the multi-view feature and one that generates 3D object proposals. A deep fusion scheme was also developed to combine region-wise features from multiple views and allows interaction between middle layers in different paths. A solution presented by Li is VeloFCN [Li17] and it consists of a fully convolutional network that extends the techniques used in 2D FCN based detection to the 3D field and applies it to point cloud data to detect and localize objects in the 3D data and represent them as bounding boxes.

F-PointNet from Qi *et al.* [QLW+18] directly works on point clouds in raw format by popping up RGB-D scans and weighting 2D object detectors and 3D deep learning object locators. It receives 2D detections from a 2D object detector and projects it into the scene's point-cloud extracting the point cloud frustum associated with each 2D object detection. The changes that they introduce that are seen as relevant to the increase in performance are the point cloud normalization, the regression loss formulation and corner loss.

2.2 Multi-Object Tracking State of the Art

In this section, some of the state of the art methods associated with the subject of Multi-Object Tracking will be presented. The majority of the methods that exist are aimed at 2D-RGB image sequences. This is related to the fact that the technologies needed to support these methods are more common and accessible than the ones needed for 3D solutions. Even though this has changed in recent years, due to the increase in the availability of tools aimed at 3D scanning (LiDAR), MOT solutions based on it are still taking is first steps.

Petrovskaya *et al.* [PT09] introduce an approach that estimates the movement of the vehicles tracked using a single Bayesian filter for each one of them. At the time a novelty it introduced was the fact that it did not need to do the segmentation and association steps before doing the filtering step because the tracking is done by direct range measurements instead of relying on features of the object — this also allows to handle occlusions in a natural way.

Another proposal based on a Bayesian approach is the proposal from Dewan *et al.* [DCTB16]. This method aims at detecting and tracking dynamic objects in city scenarios but instead of detecting changes in between scans they split the different objects in motion cues. The reason they do it is related to the requirement of a prior map or online mapping technique to be able to detect changes. They start by matching the respective points in consecutive scans and use the expected motion pattern (based on a Bayesian approach) to properly track the object given that it helps in a situation of partial or total occlusions.

A third solution that is also based on the Bayesian method is the one from Wang *et al.* [WPN15] that detects and tracks objects. It starts by modeling the objects with a set of strictly joined sample points around their limits. The positions of the objects are then initialized according to those boundaries and then updated according to raw laser measurements. This allows a free representation of the objects without depending on their class or shape. The tracking of the objects is then done following a Bayesian approach and implemented as an Extended Kalman Filter. In this approach, the movement and shape of the objects are represented as a joint state that includes the geometry of the static part of the world as well as the sensor's pose. A variant of the Joint Compatibility Branch and Bound [NT01] algorithm is used to deal with the uncertainty introduced by correlations between observations.

Shi *et al.* [SLH+14] present a solution that consists in modelling interactions "between neighbor targets by pair-wise motion context", and then encoding it into a global association optimization. This leads to a global non-convex maximization for which they propose an efficient and effective power iteration framework able to solve it. The framework presents two benefits for MOT, firstly it allows the combination of the global energy accumulated from individual trajectories and the between-trajectory interaction energy into a united optimization, that can be solved by the proposed power iteration algorithm; secondly, the framework is flexible to accommodate various types of pairwise context models. The paper proposes computationally efficient motion contexts to model the interaction between local associations and integrate the contexts into the framework. The framework suggested has three key factors to address the challenges of MOT: the

first being the between-trajectory interaction being treated in a global data association framework widely reducing the association ambiguity; the second being the encoding of the united energy term into a tensor approximation representation that can then be solved via the proposed power iteration solution; lastly the framework is flexible to the point where different context information can be used.

A different way the MOT problem was tackled is the one presented by Zhang *et al.* [ZWW+15] that proposes a novel detection based solution that connects detections to tracklets forming long trajectories, instead of what other approaches do currently. Most approaches split the data association into 2 different optimization problems: linking detections locally and tracklets globally. The approach presents a unified algorithm that automatically relearns the trajectory models based on the local and global information in order to find the joint optimal assignment. The approach consists in the beginning of each sliding window initializing the trajectory models with the local information from either the first frame or the previous sliding windows, using a Markov Random Field (MRF) model to deduce target IDs for all detections in the sliding window and then apply the loopy belief propagation to calculate the joint object function. The detections with the same label in consecutive frames are connected to create trustworthy tracklets, then the trajectory models are updated by them. The number of trajectory models can be reset to remove false models generated by false alarms and add new models for newly appearing targets. Alternatively, optimization of the trajectory models and for all targets and maximization of the conditional probability of the MRF model is done until the results converge. As the MRF model generates more complete target trajectories, the approach proposed is able to create more accurate trajectory models with more global information collected from a larger amount of frames, the more accurate the models the more accurate probabilistic computations (by the MRF) then better data association solutions. Another solution that also uses the Markov model us the one from Xiang *et al.* [XAS15] [XCLS17] that presents an approach that formulates the online MOT problem as a decision-making problem and solves it with a Markov Decision Process (MDPs) framework, with the lifetime of each object being modeled as a MDP. The advantages of this method are the fact that learning a policy in a MDP is equivalent to learning a similarity function for data association and but the policy learning is approached in a reinforcement learning fashion that takes advantage of both offline and online learning for data association. The proposed framework can also handle, in a more natural way, the birth/appearance and death/disappearance of objects as they are treated as state transitions in the MDP while at the same time it leverages existing online single object tracking methods.

Feng *et al.* [FHLA18] propose a vehicle-free point cloud framework to achieve better on-vehicle localization. In each laser scan, the vehicle points are detected, tracked and after, removed. At the same time, a 3D map is rebuilt by registering each scan on GPS based global coordinates. Instead of detecting the vehicles directly on the cloud point date it uses auxiliary RGB images through their YVDN (YOLOv2 Vehicle Detection Network). They track each object through a K-Frames forward-backward object tracking algorithm that links each detection with consecutive images.

Dimitrevski *et al.* [DVP19] presents a solution that aims at tackling multi-object tracking, in

the context of an autonomous vehicle, using 2D images as well as LiDAR scans. The approach presented has various steps that consist of three phases: one for preparation, one for object-detection and one for tracking. The pre-tracking has two steps, the first one consists in estimating the vehicles' orientation relative to the road based on the LiDAR data, and a second one where the image optical flows are computed through a third-party algorithm. The object detection step is done frame by frame and using only the RGB camera data. Afterwards a CNN is used for each camera frame and, through point-cloud segmentation, 3D non-ground plane regions in the LiDAR data are obtained after an association between the detections on the image plane and the detections on the point cloud is made . An association is made instead of a complete fusion because the calibration they have between the pair camera-LiDAR is not ideal. The matched 2D-3D pairs are then tracked with dual 2D-3D particle filters (under the prediction and update principle). Given that currently, the sensors that are used in this context are not accurate enough to give a proper correspondence between the 2D bounding boxes and LiDAR point-clouds, the solution proposed is prepared to deal with miscalibration of the sensors by letting the states' predictions (2D and 3D) move separately.

The three main contributions of this approach identified by Dimitrevski *et al.* are:

- A novel pedestrian motion model that when applied in a particle filter is able to track unpredictable behavior;
- A new approach to split data association where they perform the tracking separately in the 2D and 3D environment and the data association in a blended 2D-3D environment;
- An improved track management system, responsible for handling tracks and their interactions that relays on the track confidence score.

A combined solution that pretends to fill the gap between online and global algorithms is the one from Choi [Cho15] and He *et al.* [HZRS16] who propose two approaches that contributes to two of the main parts of the MOT problem: one of the approaches consists of an accurate affinity measure to link detections and another one involves an accurate and efficient algorithm for online near MOT. For the first one is used a new Aggregated Local Flow Descriptor (ALFD) that translates the relative motion pattern between two time-wise distant detections through long term interest point trajectories - IPTs. Balancing on the IPTs the ALFD is able to provide a hefty affinity measure that estimates how likely the matching detections are, disregarding the application scenario. For the online MOT, they formulate the tracking problem as a data-association between the detections and the targets for a given time frame, being computed repeatedly at each frame. This approach shows to be reliable and efficient through integration in the model of various cues like ALFD metrics, appearance similarity, long term track regularization, and target dynamics.

Scheidegger *et al.* [SBR+18] introduces a solution that aims at tackling the issue of not being possible to directly measure distances in 2D images. For this, they present a MOT algorithm that receives a 2D image and proposes tracks for the detected objects in a world coordinate system. They trained a deep neural network to detect objects and suggest distances, from the camera

(mounted on the vehicle) to the object, from a single image. A sequence of images is provided to a Poisson multi-Bernoulli mixture (PMBM) tracking filter that will associate the objects between frames and also filter object detections from the DNN. This combination of the PMBM filter and the trained DNN results in an algorithm that is capable of 3D tracking having only RGB frames as input.

Sharma *et al.* [SAMK18] suggests an approach based on a simple two-frame Hungarian association scheme that introduces geometry and novel object and pose costs to the MOT area in driving contexts. Their solution also takes advantage of the particular geometry of road scenarios, showing how it can help infer 3D tracking cues. Their approach uses RGB images from a single camera, and then arranges pairwise costs for each object's track based on various 3D cues like object motion, pose and shape. The costs are easy to implement and it's also possible to calculate them in real-time. They also complement themselves allowing to account imaginable errors in detection based tracking frameworks.

2.3 Kalman Filters

A Kalman filter, as described by Welch and Bishop [WB+95], is a recursive solution for the discrete-data linear filtering problem, presented in 1960 by R. E. Kalman on "A New Approach to Linear Filtering and Prediction Problems" [Kal60]. It is a set of mathematical equations that brings an efficient recursive computational application of the least-squares method. The main strength of this approach is the support of past, present and future estimations and the ability to do it without knowing the precise nature of the system being modeled.

The Kalman filter consists of an estimator for a system state at a given time and then receiving feedback in the form of noisy measurements of the state, these measures are then used to update the estimator. As so we can split the KF's equations into two subsets, prediction and correction. The prediction's one project the current state and the error covariance estimates into the next time step, obtaining *a priori* the estimates for that time step. The correction equations receive the measurements of the state (with associated noise) and update the previous estimate achieving a better estimate. It resembles the predictor-corrector algorithm.

The equations associated with the predictor (or time update) are presented below. In equation 2.1, A and B are matrices that describe the behavior of the system in function of time, $\hat{x}_{k+1}$ represents the next state estimate and in equation 2.2 P^{-}_{k+1} represents the covariance estimate for the next time step.

$$\hat{x}^{-}_{k+1} = A_k \hat{x}_k + B u_k \tag{2.1}$$

$$P^{-}_{k+1} = A_k P_k A^T_k + Q_k \tag{2.2}$$

The equations associated with the corrector (or measurement update) are presented below. In equation 2.3,K_k represents the computation of Kalman gain and is the first part of the corrector

phase of the algorithm. In equation 2.4, $\hat{x}_k$ represents the *a posteriori* state estimate generated by incorporating the received measurements and in equation 2.5, P_k represents the *a posteriori* covariance estimate.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.3}$$

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H_k \hat{x}_k^-) \tag{2.4}$$

$$P_k = (I - K_k H_k) P_K^- \tag{2.5}$$

After the execution of each predictor-corrector pair, the process is repeated for the next time frame where k+1 becomes k. The recursive nature of this filter is one of the appealing features that makes it a good option to be applied in this scenario compared to other filters like the Weiner filter [BH+92].

Some of the filter's parameters can be measured before the use of the filter itself and be tuned accordingly. Regarding R_x, the error covariance matrix, this tune is achievable as it is possible to get some measurements of the system before using the filter. Regarding Q_k, the noise associated with the system, it is less tunable as it depends on the reliability of the measurements. In case of these two parameters being constant, the estimation error covariance (P_k) and the Kalman gain (K_k) will stabilize quickly generating more precise estimates.

2.4 Datasets

Different datasets that cover various and different contexts are needed in order to test the solution. In this section, some of the most used and recognized datasets will be presented and what the context they best emulate will be explained.

One of the most used datasets to test MOT solution in scenarios involving pedestrians is PETS 2009 [FS09]. It has three main areas of evaluation: crowd count and density estimation; tracking of individuals in a crowd; and detection of separate flows and specific crowd events. All of the dataset scenarios are in a real-world environment, involved multiple actors and were recorded using numerous cameras. From the three areas it covers, the only that has more relevance for this work is the second one (tracking individuals in a crowd), the dataset that concerns this context is the dataset identified as S2, wherein the S0 is also important as it is the training dataset. These datasets are composed of multi-sensor sequences with different crowd scenarios and with growing scene complexity. The training dataset, S0, has 3 subsets of sequences that allow the developers to create three models: one for a background model (size - 1.8GB), one for a city center model with random flow by the crowd (size - 1.8GB), and one for a regular crowd flow model (size - 1.3GB). The dataset S2, that, as said previously, focus on people tracking and has 3 parts (presented with increasing difficulty): the first one emulates a scattered crowd walking, where the main task is to

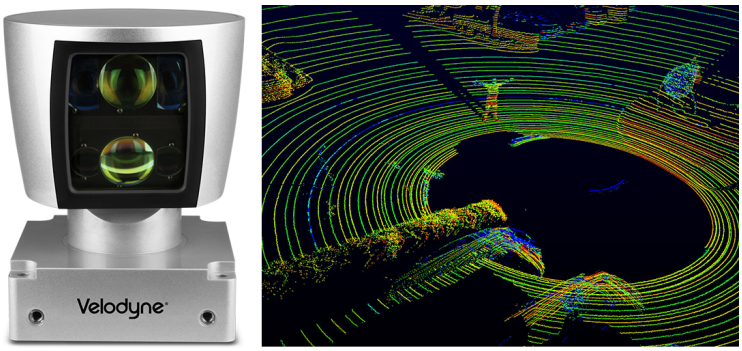


Figure 2.1: Velodyne scanner and scan example.

track all the pedestrians in the sequence, presenting for each of them a 2D (and optionally 3D) bounding box (size - 1.54 GB); the second one, has the task to follow two predefined pedestrians and present for each a 2D (and optionally 3D) bounding box in each frame (size - 484 MB); and the third one that has the same task as the second one but in a more crowded environment (size - 536 MB).

Another popular and more complete dataset for driving scenarios is the one used in "The KITTI Vision Benchmark Suite" [GLU12] [GLSU13]. "The KITTI Vision Benchmark Suite" [GLU12] is a project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago [aC19], it uses Annieway [dLBW11] (their autonomous driving platform) and has as a goal providing real-world computer vision benchmarks for areas like stereo, optical flow, visual odometry, 3D object detection, and 3D tracking being the last one the more relevant for this dissertation. The sensor used to collect the 3D was a 360^{o} Velodyne laser-scanner similar to the one on figure 2.1, the datasets were collected in and around the mid-sized city of Karlsruhe and aggregates the three main types of road scenarios (urban, rural and highway). The benchmark available for object tracking is composed of 21 training sequences and 29 test sequences and have 8 classes pre-defined and classified (but only 2 are evaluated - "Car" and "Pedestrian"), have up to 15 cars and 30 pedestrians per image. The data is available in various formats as you can see on table A.1 from appendix A and a file with the labels' structure presented in the table A.2 from appendix A.

2.5 Tracking Metrics

Nowadays, tracking multiple objects simultaneously is a growing and active research field given the different applications it can have. Because of that, many solutions have appeared based on different techniques and algorithms. With that, the need to evaluate and compare all of these approaches emerged. However, there is a lack of consensus in the community of what metrics should be taken into account to evaluate these methods.

The literature review made by Luo *et al.* [LZK14] identifies two subsets of metrics that should be taken into account when evaluating MOT algorithms, metrics for object detection and metrics

Table 2.1: Metrics used for MOT (adapted from [LZK14]).

Name	Description	Group	Rank	Ref.
Recall	Calculated by dividing the number of true positives by the sum of true and false positives	DA	High	[YHN11]
Precision	Calculated by dividing the number of true positives by the sum of true positives and false negatives	DA	High	[YHN11]
FAF	Average number of false alarms (false positives and false negatives) per frame	DA	Low	[YHN11]
MODA	Calculated using the cost functions of false alarms divided by the number of objects in the frame	DA	High	[KGS+09]
MODP	Average overlap between true positives and ground truth	DP	High	[KGS+09]
MOTA	Joins the false negatives, false positives and mismatch ratios in all frames	TA	High	[BS08]
MOTP	Overlap between the estimated positions and the real positions divided by the matches	TP	High	[BS08]
TDE	Distance between the real position and the tracking proposal	TP	Low	[KN10]
OSPA	Cardinality error and spatial distance between the real and the tracking results	TP	Low	[RVCV11]
MT	% of real trajectories that are covered by the tracker result for >80% of their length	TC	High	[LHN09]
ML	% of real trajectories that are covered by the tracker result for <20% of their length	TC	Low	[LHN09]
IDS	Number of times a single ground truth trajectory is disrupted	TA	Low	[LHN09]
FRAG	Number of times the identities were changed trajectory-wise	TA	Low	[LHN09]
PT	1 - MT - ML	TC	-	[LHN09]
FM	Number of times that a real trajectory is split in the tracking result	TC	Low	[LHN09]
RS	Ratio of tracks that are properly recovered from short occlusion	TR	High	[SJSRC10]
RL	Ratio of tracks that are properly recovered from long occlusion	TR	High	[SJSRC10]

for object tracking (some of the metrics that belong to this two subsets as well as their subgroups are briefly explained in table 2.1).

For object detection the metrics can be classified into two subsets:

- Accuracy (DA)
- Precision (DP)

For object tracking the metrics can be split into four groups:

- Accuracy (TA)

- Precision (TP)
- Completeness (TC)
- Robustness (TR)

From the metrics presented in table 2.1 the two presented by Bernardin and Stiefelhagen [BS08], MOTA and MOTP, and the four by Li *et al.* [LHN09] MT, ML, IDs and FRAG were the one chosen for the test-bed that will be developed.

The ones from Bernardin and Stiefelhagen were chosen given that they allow a clear and intuitive analysis of the tracker properties: how precise it is at estimating the object positions, how many they are and their configuration, and it's the ability to track them consistently over time. They aimed at metrics that would "allow to judge the tracker's precision in determining exact object locations and reflect its ability to consistently track object configuration and through time, that is, to correctly trace object trajectories". They also pretended to present a framework clear and understandable with as less free parameters as possible but general enough to work with different types of trackers (2D/3D and object-area/object-centroid based).

Hereupon the two metrics proposed are:

- MOTP - Multiple object tracking precision is calculated as shown in 2.6, where d_t^i (t being for the frame and i being for the object in t frame) represents the total amount of errors in position estimates for each associated object-hypothesis pairs across all frames of the sequence. Evaluates the skill of the tracker to calculate precise positions.

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{2.6}$$

- MOTA - Multiple object tracking accuracy can be either calculated as shown in 2.7 or as show in 2.12, where m_t is the total of misses (or false negatives), fp_t is the total of false positives and mme_t is the total of mismatches.

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{2.7}$$

It can also be calculated through the total error rate E_{tot} 2.11 2.12 that consists in the sum of three error ratios: ratio of misses $\overline{m}$ 2.8, ratio of false positives $\overline{fp}$ 2.9 and ratio of mismatches $\overline{mme}$ 2.10.

$$\overline{m} = \frac{\sum_t m_t}{\sum_t g_t} \tag{2.8}$$

$$\overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \tag{2.9}$$

$$\overline{mme} = \frac{\sum_t mme_t}{\sum_t g_t} \tag{2.10}$$

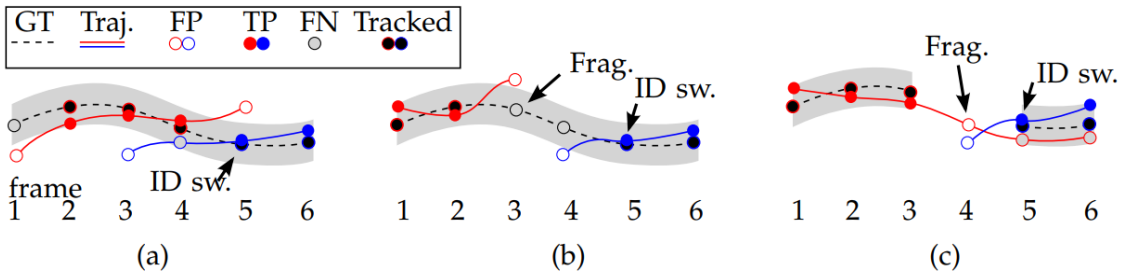


Figure 2.2: ID-switches and trajectories fragmentations [MLTR+16].

$$E_{tot} = \overline{m} + \overline{fp} + \overline{mme} \tag{2.11}$$

$$MOTA = 1 - E_{tot} \tag{2.12}$$

The four metrics proposed by Li *et al.* [LHN09] can be split into two pairs, MT/ML and FRAG/IDS, the first pair is associated with the percentage of the length of the tracks that are effectively covered by the tracker, MT (mostly tracked) represents the % of tracks which at least 80% of its length is covered by the tracker and ML (mostly lost) represents the % of tracks which more than 20% of its length is covered by the tracker; the second pair is associated with identity change (IDS) and fragmentation of the ground truth trajectory (FRAG). In figure 2.2, case (a) represents a case of an ID switch because the ground truth object (black) is switched from the red to the blue trajectory, in frame 4; case (b) is an example of a fragmentation when the ground truth that is tracked in frames 1 and 2 is interrupted in 3 and then re-tracked in 4 but as a new hypothesis so an ID switch is also counted and finally case (c) shows when a target re-identification is not handled correctly, the ground truth trajectory stops in frame 3, but is identified in frame 4 causing a fragmentation and then in frame 5 it is reassigned to a new trajectory leading to a ID switch.

2.6 Benchmarks

Benchmarking is relevant for context of this dissertation as it provides a way to measure the performance and evalate the delivered solution.

"The KITTI Vision Benchmark Suite" [GLSU13] is one of the most accepted and recognized benchmarks related to driving scenarios, an explanation on the origin of the benchmark is already provided in the section 2.4 when introducing the dataset with the same name. It evaluates the performance in the road scenarios present in the dataset present. The metrics (check table 2.1) used by this benchmark are the two introduced by Bernardin and Stiefelhagen in their paper [BS08], MOTP and MOTA, as well as the four presented by Li *et al.* [LHN09], MT, ML, IDS and FRAG, and finally runtime that is related with the ability of the algorithm to be used in real-time applications.

Another popular benchmark is is the "Multiple Object Tracking Challenge" [MLR+16] that uses the dataset PETS2009 [FS09] as well as its own dataset, it evaluates the different approaches using the same six metrics (check table 2.1) referred in the previous benchmark, MOTP, MOTA, MT, ML, IDS and FRAG, as well as number of false positives and false negatives, but for the test bed this will be ignored given the fact that they are part of the calculation of MOTA and MOTP, the frequency is also calculated and just as the runtime in "The KITTI Vision Benchmark Suite" is related with how suitable the solution is for real-time applications.

2.7 Summary

As seen there are many different methods to obtain 2D detections of objects given an RGB image, from all the presented ones YOLOv3 [RF18] was the one chosen as it is the one considered more suitable for this solution given the execution times, precision it ensures and simplicity of use. Regarding the 3D detection, the chosen solution was the Frustum PointNet[QLW+18] as it presented acceptable results and runtimes for what is pretended.

3D tracking is currently one of the most prolific areas regarding computer vision given all its uses as well as the growing processing power available in today computers. The proliferation and volatility of this area lead to a lack of standard or go-to approach so this was one of the reasons why this dissertation explores this field.

Chapter 3

Proposed Tracker and Predictor

This chapter presents the proposed solution to tackle the problem of tracking objects around a vehicle in a road scenario as well as predicting its movement. Section 3.1 presents an overview of the solution by introducing the main steps for the processing of each frame of the sequence as well as which modules handle those steps. In section 3.2, the modules are explained in-depth.

3.1 Pipeline Overview

The created solution addresses the problem of object tracking, treating it as a cyclic problem with a set of steps that must be performed in each iteration. An iteration corresponds to each frame of the sequence of images and point clouds that represent the vehicle's road scenario.

Figure 3.1 shows the modules that handle each step. The set of steps starts by processing the data received and ends with providing, for each object, a center for the current frame and a center prediction for the next frame.

The first step consists of preparing the input data (both an RGB image and a LiDAR scan of the surroundings of the vehicle) by transforming it from its raw format into a structured format — "Data Processor" module in fig. 3.1 and explained in-depth in sub-section 3.2.1. The next step detects the objects in the RGB image, outputting a bounding box to each and classifying it accordingly to the COCO dataset — "2D RGB Detector" module in fig. 3.1 which is explained in-depth in sub-section 3.2.2. After that, a point cloud is extracted and a 3D center proposed for each object based on the bounding box computed previously — "3D LiDAR Extractor" module in fig. 3.1 and explained in-depth in sub-section 3.2.3. The fourth step associates the new centers with the previous centers and creates a new sub-tracker for each new object — "Tracker" module

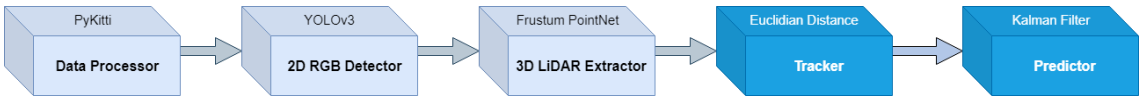


Figure 3.1: Main modules of the pipeline (light blue represent third-party-based modules and blue represents modules developed from scratch).

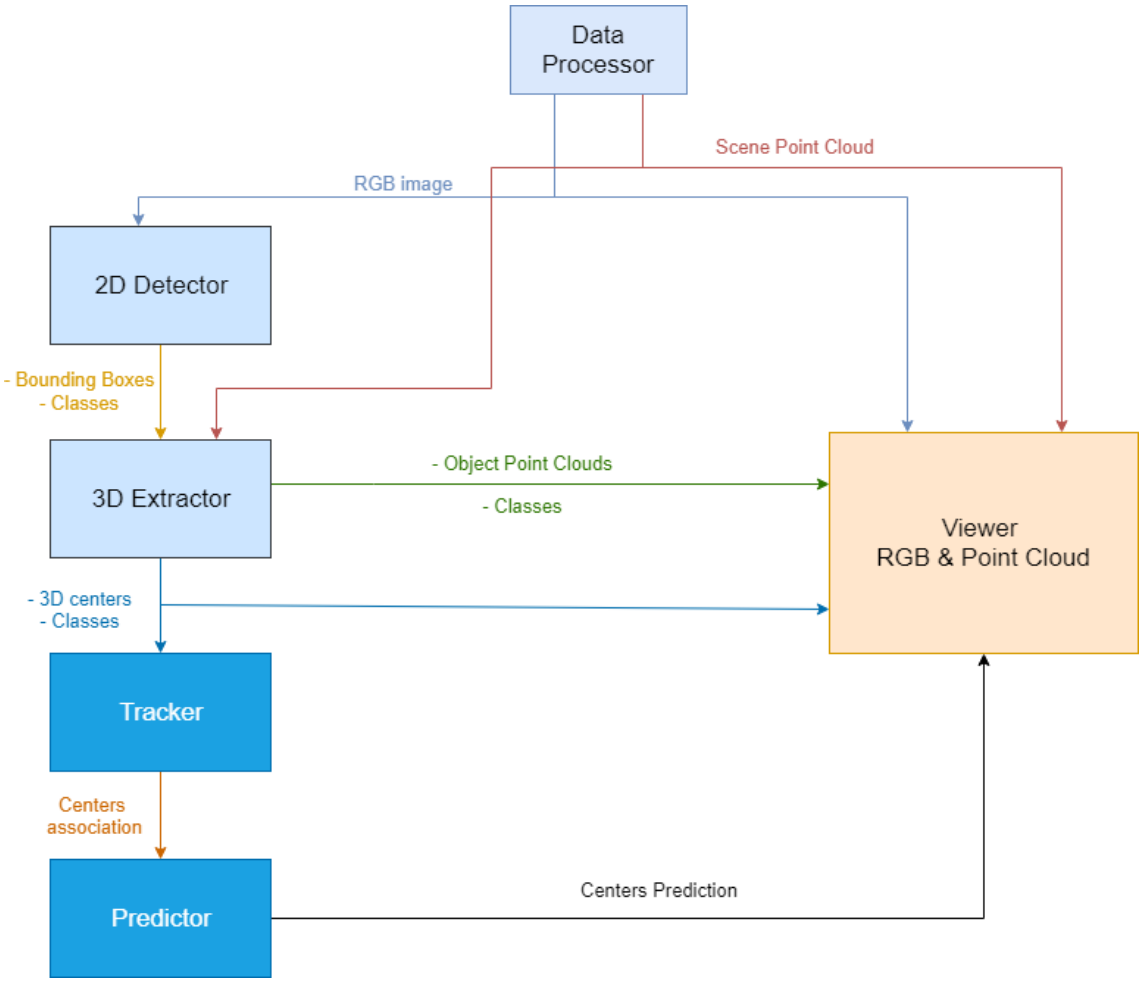


Figure 3.2: Dataflow between the different modules of the solution (light blue represent third-party-based modules whereas blue represents modules developed from scratch).

in fig. 3.1 and explained in-depth in sub-section 3.2.4.1. Finally, the previous predictions are corrected and centers for the next frame are predicted — "Predictor" module, also seen in fig. 3.1 and explained in-depth in sub-section 3.2.4.2.

3.2 Theoretical basis and implementation

In this section, a theoretical overview for each module is provided and the module's implementation is explained in-depth as well as the flow of data (see figure 3.2) between the different modules.

3.2.1 Data processor

The "Data Processor" module receives the data from the surroundings of the vehicle in both RGB and LiDAR raw format.

The RGB image is the output from a front-facing camera and the LiDAR scan is a 3D point cloud representation of the environment surrounding the vehicle. It is transformed into a 2D array with size H by W, being H the height and W the width of the image (for the KITTI dataset the height is 375 pixels and the width 1242 pixels). An element of the HxW array is a ternary representation of each color channel (red, green and blue) of the corresponding pixel.

The LiDAR scan is stored as floating point binaries: each point is stored in a quaternary format with the three first fields being the XYZ coordinates of the point and the fourth field being the reflectance value of the point (the reflectance field is only used by the Frustum PointNet method), and is translated to an array with dimension equal to the number of points stored in the binary file (each file in the KITTI dataset has around 120,000 points). This array is then cropped to only include the points that belong to the field-of-view of the RGB image.

3.2.2 2D object detector

The "2D Object Detector" performs the detections on the RGB image that was prepared by the module "Data Processor" (3.2.1). The detections are performed using the detector YOLOv3 [RF18] trained for the COCO dataset [CUF18]. A Python wrapper, that runs the C++ dll of YOLO, is used in order to reduce the running time from an average of 460ms (on a full Python implmentation) to an average of 80ms.

YOLO outputs a bounding box and a class for each detected object. The format of each bounding box is a quaternary array with the first field corresponding to the X coordinate of the top left corner of the bounding box, the second field to the Y coordinate of the top left corner, the third field to the X coordinate of the bottom right corner of the bounding box and the fourth field to the Y coordinate of the bottom right corner. The first class associated to each object is one of the following COCO classes: *person*, *bicycle*, *car*, *motorbike*, *bus*, *train* and *truck*. The remaining classes are ignored as they are not relevant to the context of this dissertation. For evaluation purposes, these classes are then translated to KITTI classes as shown in table 3.1. As COCO does not have a class *Cyclist*, which is required by KITTI, this module identifies the class every time a bounding box of the class *bicycle* overlaps a bounding box of the class *person* in more than 30% replacing the *bicycle* by *cyclist* in the classes and in the bounding boxes array deleting the entries associated with *person*.

The outputs (bounding boxes and classes) of this module are passed to the module "3D Object Point Cloud Extractor" (3.2.3) and to the 2D viewer sub-module 3.2.5.

3.2.3 3D object point cloud extractor

The 3D object point cloud extractor uses the Frustum-PointNet extractor [QLW+18]. It takes as an input an array of 2D bounding boxes (the output of the module "2D Object Detector" 3.2.2).

It outputs the point cloud it identified for each object as well as the predicted center for that object. The point clouds are used in the 3D viewer sub-module 3.2.5 to highlight the objects as

Table 3.1: Equivalence between COCO and KITTI classes.

COCO Class	KITTI Class
bicycle + person	Cyclist
person	Pedestrian
bicycle	Misc
car	Car
motorbike	Misc
bus	Van
train	Tram
truck	Truck

well as calculate the bounding boxes for them. The centers are used to update the tracker 3.2.4.1 which then updates the predictor module 3.2.4.2.

3.2.4 Tracker and predictor

These two modules are the main novelty introduced by this dissertation. Although they are two main modules on their own, they are highly dependent on each other.

3.2.4.1 Tracker

The tracker module is responsible for tracking the objects in-between frames, inside the area of interest. To perform the tracking it needs to be able to associate the centers from one frame (frame k) to the next one (frame k+1).

To complete this association it takes into account the Euclidean distance between the centers in frame k and in $k+1$ and the class associated with the center. The Euclidean distance, also known as the Pythagorean distance, is the distance in a straight line between points in a Cartesian coordinate space. For the context of this dissertation the space considered is the three dimensional and as so each center is represented by a ternary set of coordinates. To calculate the distance between the center of object A and the center of object B, and considering they have the following coordinates(x_A, y_A, z_A) and (x_B, y_B, z_B) respectively, we use equation 3.1.

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2} \tag{3.1}$$

The first step of this process is to calculate the distance between each center from frame k and all the centers detected in frame $k+1$ and storing it into a N x M matrix, considering there are N centers in frame k and M in frame $k+1$. After that, the smallest distance is chosen and the corresponding centers are associated as being the same object, as long as it is smaller than the maximum distance defined for associations when the object's class is the same. After this association, the variables associated with the object are updated as well as the predictor associated with it(thoroughly explained in section 3.2.4.2). After that, all the unassigned centers are registered in the tracker as a new object.

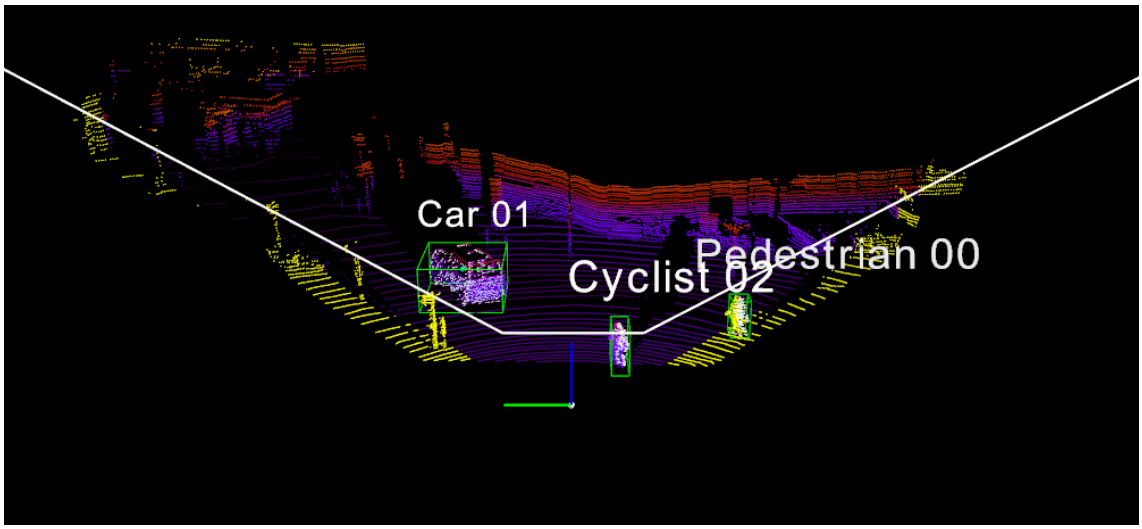


Figure 3.3: Scene view with the area of interest and the offset area highlighted with yellow (frame 0 of sequence 0 of the KITTI dataset [GLSU13]).

The object stops being tracked and is removed from the tracker whenever it is not detected for a certain consecutive number of frames (value of `self.maxDisappeared`). The other scenario where the object stops being tracked is when it stays consecutively outside the area of interest, without being detected in more than a certain number of frames (value of `self.maxOffArea`). The area of interest is the area inside the field of view except for an offset area near the border (highlighted with yellow in fig. 3.3).

To handle this, a class was created and its initialization can be seen in listing 3.1. What is stored in each variable is explained below:

- `self.nextObjectID`, represents the unique ID that is assigned to each tracked object and works as the key to access the corresponding data in the different ordered dictionaries of the class (`objects`, `disappeared`, `offArea`, `kalmanFilters`, `predCenters`, `classes` and `boxes_2d`).
- `self.objects` stores the center of each object associating it with the object ID.
- `self.disappeared` holds the number of consecutive frames the object was undetected, initialized with value zero and reset to zero every time the object is re-detected.
- `self.offArea` keeps the number of consecutive frames the object was outside the area of interest initialized with value zero and reset to zero every time the object returns to the area of interest.
- `self.kalmanFilters` is another crucial dictionary that keeps the Kalman Filter associated with each object and is updated each time the center in `objects` is updated. Each of the Kalman Filters stored in this dictionary are able to estimate the next position of the

```
class CentroidTracker():
    def __init__(self, maxDisappeared=50, seq_id=0, offArea = 5, maxDistance = 10):
        self.nextObjectID = 0
        self.objects = OrderedDict()
        self.disappeared = OrderedDict()
        self.offArea = OrderedDict()
        self.kalmanFilters = OrderedDict()
        self.predCenters = OrderedDict()
        self.classes = OrderedDict()
        self.boxes_2d = OrderedDict()
        self.areaDet = [0, 0, 0, 0, 0]
        self.maxDisappeared = maxDisappeared
        self.maxOffArea = offArea
        self.maxDistance = maxDistance
```

Listing 3.1: Tracker initialization

center associated with it. An in-depth explanation of how it works is provided in section 3.2.4.2.

- `self.predCenters` keeps the predicted centers in the next frame for each object tracked, being updated after `kalmanFilters`.
- `self.classes` saves the class associated with the corresponding object.
- `self.boxes_2d` is the dictionary that associates a 2D bounding box to each object. It is updated at the end of each frame where the object is detected.
- `self.areaDet` retains the properties of the area of interest. The first field is the offset, the second field is the minimum coordinate in the X-axis, the third field is the maximum coordinate in the X-axis, the fourth field is the negative angle and the fifth field is the positive angle. It is updated at the beginning of each frame.
- `self.maxDisappeared` and `self.maxOffArea` are the auxiliary values used to know when an object stops being tracked for being undetected or for being outside the area of interest respectively. The predefined values are 50 and 5.
- `self.maxDistance` is another auxiliary value that limits the distance between centers when the centers' association is done between frames.
- `self.wr_file` is the handler of the file that is used to store the status of the tracker during a sequence.

3.2.4.2 Predictor

The predictor module, as its name says, foresees where the objects that are being tracked will be in the next time interval.

It is based on the Kalman filter algorithm also known as linear quadratic estimation. The Kalman filter concentrates on the problem of trying to predict the next state of a certain discrete-time controlled system, like the one the solution of this dissertation covers.

The first part of the Kalman filter consists of projecting the state estimation from frame k to frame $k+1$ (equation 3.2) as well as the covariance estimation (equation 3.3).

$$\hat{x}_{k+1}^{-} = A_k \hat{x}_k + Bu_k + \varepsilon_x \tag{3.2}$$

$$P_{k+1}^{-} = A_k P_k A_k^T + Q_k \tag{3.3}$$

The matrix A relates the state in frame k to the state in frame $k+1$, while the matrix B associates the control input u to the state x, matrix P is the covariance matrix and Q_k the expected variance in that time step. For the context of this dissertation, this means applying the position and velocity, $x(t) = x_0 + +v_0 t + \frac{1}{2}at^2$ and $v(t) = v_0 + at$ to equation 3.2 that can be translated into the following system of matrices:

$$\hat{x}_{k+1} = \begin{bmatrix} Pos_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Pos_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{t^2}{2} \\ t \end{bmatrix} a_{k+1} + \varepsilon_x$$

With Pos representing the position of the object, v the velocity, a the acceleration, ε_x the noise associated and k the frame. The equation 3.2 is then defragmented into the following relations:

$$A = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \qquad \hat{x}_k = \begin{bmatrix} Pos_k \\ v_k \end{bmatrix} \qquad B = \begin{bmatrix} \frac{t^2}{2} \\ t \end{bmatrix} \qquad u_k = a_{k+1}$$

$$\varepsilon_x = E_x = \begin{bmatrix} \sigma_{Pos}^2 & \sigma_{Pos}\sigma_v \\ \sigma_v \sigma_{Pos} & \sigma_v^2 \end{bmatrix}$$

σ_{Pos} and σ_v stand for the standard deviation, or variance, of the position and the velocity respectively.

The second part is related with the the measurements received from the sensors and correction of the estimate based on those received measures providing an improved estimate.

$$K_k = P_k^{-} H_k^T \left(H_k P_k^{-} H_k^T + R_k\right)^{-1} \tag{3.4}$$

$$\hat{x}_k = \hat{x}_k^{-} + K(z_k - H_k \hat{x}_k^{-}) \tag{3.5}$$

$$P_k = (I - K_k H_k) P_K^{-} \tag{3.6}$$

The first step of this part of the Kalman filter cycle consists in computing the Kalman gain (equation 3.4), K. H selects the variable to have into account for the computation of the Kalman

```
def predict(self):
    self.sk = np.matmul(self.A,self.sk) + self.B * self.u
    self.P = np.matmul(np.matmul(self.A,self.P),self.A.T) + self.eX
```

Listing 3.2: Kalman Filter predictor function (implementation of equation 3.2 and 3.3)

gain and R_k represents the noise associated with the measurement. The higher the variance associated to the measurement the smaller the value of the Kalman gain will be as the measurement is less informative. Under the scope of this dissertation, the two variables will have the format shown bellow as the variable that matters is the position.

$$K = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad \text{and} \quad R_k = \sigma_x^2$$

After that, in equation 3.5 the final estimate is calculated and in equation 3.6 the covariance value is updated and prepared for the next iteration.

Given that it works on 3D data, the predictor module has a Kalman Filter assigned to each of the axes.

This module is able to predict (Python funtion used in 3.2) at the beginning of each frame where a certain object will be after a certain number of frames. However, the bigger the number of frames the less precise those predictions will be.

Ideally it is updated (Python funtion used in 3.3) at the end of each frame. As it is self-tuning, the more updates it receives, the more accurate it will be and the faster it will be properly tuned. In a scenario with irregular updates it still works but without optimal performance, for example when there is not an available measure of the center of the object in a certain frame.

3.2.5 Viewer

This module is used for data visualization as well as debugging. It has two sub-modules which are presented below.

2D viewer sub-module — This sub-module receives an RGB image and optionally an array with the bounding boxes and classes from the YOLO module and displays it in a window. The

```
def update(self, meas, meas_noise=0):
    K = np.matmul(self.P,self.C.T)
    aux = np.matmul(np.matmul(self.C,self.P),self.C.T) + meas_noise**2
    K = np.matmul(K,np.linalg.inv(aux))
    self.sk = self.sk + np.matmul(K,(meas - np.matmul(self.C,self.sk)))
    self.P = np.matmul(np.eye(2) - np.matmul(K,self.C), self.P)
```

Listing 3.3: Kalman Filter corrector function (implementation of equation 3.4 3.5 and 3.6)

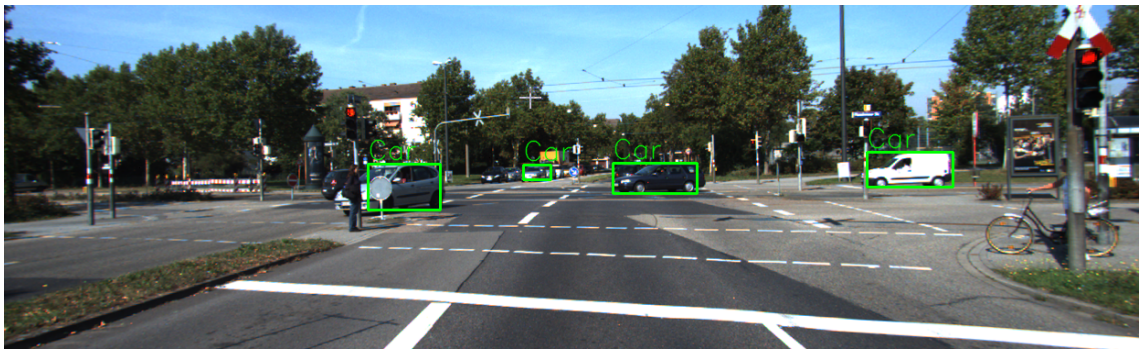


Figure 3.4: Example of 2D visualization of scene point cloud by the sub-module "2D viewer sub-module" (frame 124 of sequence 2 of the KITTI dataset [GLSU13]).

Python library *OpenCV* [Bra00] is used to display the image as well as drawing the bounding boxes and showing the classes associated with each object as seen in figure 3.4.

3D viewer sub-module — The input of this module is a scene point-cloud (that can be then trimmed to the field of view associated with the RGB image), a point cloud of the objects, the detected centers, the predicted centers, the bounding boxes, the object classes and the object IDs. It uses the Python library *mayavi* [RV11] that helps displaying the point cloud of the scene according to the field of view as seen in figure 3.5. The points in the area of interest are shown under the *gnuplot* color scale depending on its Z coordinate and the points in the field of view but outside the area of interest are shown as yellow. The limits of the area of interest are displayed by the white lines in the Z = 0 plane. The points that are assigned to each object are shown in white and surrounded by a green 3D bounding box. A label is also displayed over the point cloud of each object, which is composed by the class of the object and the object unique ID in the tracker (module 3.2.4.1).

3.3 Summary

This chapter shows the theory behind the implemented solution and how it is implemented. We can divide the algorithm into different parts: a core one and the accessory one. The core part can then be split into two different subparts on its own. Being one of those parts the one comprised by the third-party-based detection modules, the 2D detector and the 3D extractor, that are guided by the YOLO and the Frustum Point-Net approaches, respectively. The second part of the core is the novelty introduced by this dissertation and is composed of two sub-parts itself: the tracker module and the predictor module that are regulated by an association based on the Euclidean distance and by a Kalman filter, accordingly.

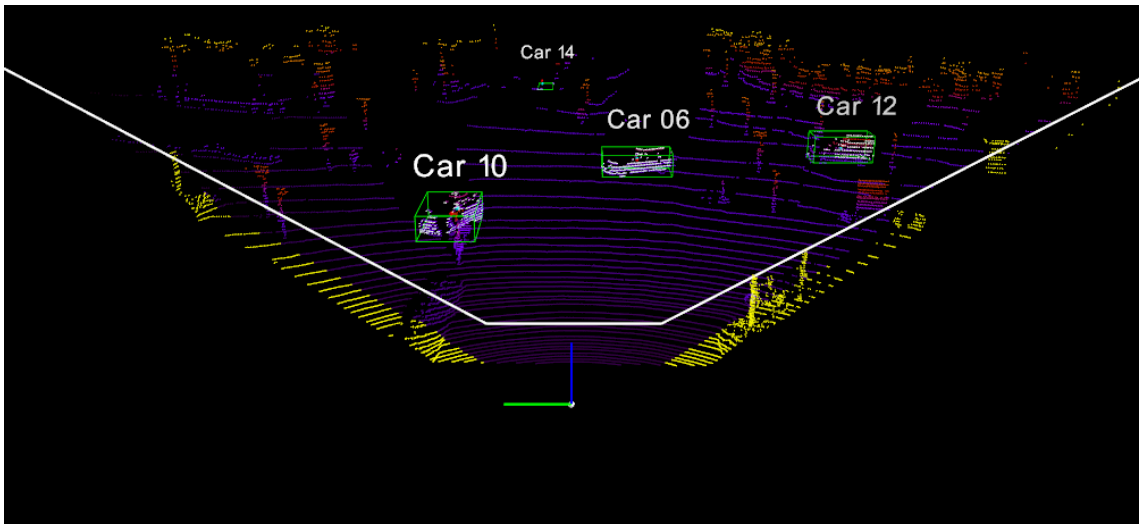


Figure 3.5: Example of 3D visualization of scene point cloud by the sub-module "3D viewer sub-module" (frame 124 of sequence 2 of the KITTI dataset [GLSU13]).

Chapter 4

Experiments and Result Discussion

The experiments done to evaluate and tune the performance of the solution developed are presented and explained in this chapter and their results are discussed. The goal of these experiments is to find out the best possible values for the parameters of both the tracker and the predictor.

The hardware specifications of the testing environment had as a CPU an Intel Xeon E3-1270 with 4 cores and 8 threads and as a GPU an Nvidia Quadro P2000 which is not an ideal GPU for neural networks. A better GPU, like the Nvidia Tesla V100, would lead to an increase in performance mainly concerning running time. The complete setup of the testing environment is shown in table 4.1. Each iteration took around 170 milliseconds what corresponds to around 5.8 frames per second. All the parameters related to distances are in meters.

4.1 Different values for maxDisappeared and offArea

Some experiments with different values for `maxDisappeared` and for `offArea` are presented in this subsection. As previously said, `maxDisappeared` limits the consecutive number of frames an object can be undetected until it stops being tracked and `offArea` limits the consecutive number of frames an object can be outside the area of interest until it stops being tracked. As these variables are crucial to the decision to stop tracking a certain object, it was seen as relevant to test them together, with different pairs of values for each.

Table 4.1: Hardware specifications of the testing environment.

Component	Specification	Add. Info.
CPU	Intel Xeon E3-1270 v6 @ 3.80GHz	8 cores
RAM	32,0 GB	2400MHz DDR4
System type	Windows 10	64-bit
GPU	Nvidia Quadro P2000	1024 CUDA cores
VRAM	5,0 GB	2400MHz GDDR5

Table 4.2: Experiments with offArea (offA) and maxDisappeared (maxD), both values are in frames.

offA	maxD	MOTA	MOTP	Precision	ID	Frags	MT	PT	ML
1	5	**35.77**%	75.64%	**78.75**%	570	1045	29.29%	52.72%	17.99%
1	10	33.49%	75.66%	75.29%	534	1038	**29.50**%	55.44%	15.06%
1	25	24.38%	75.84%	69.43%	540	1015	25.94%	56.07%	17.99%
1	50	18.77%	75.77%	66.01%	516	964	25.52%	55.44%	19.04%
5	5	29.34%	75.80%	72.72%	539	1026	29.08%	**56.69**%	**14.23**%
5	10	23.86%	75.72%	69.06%	587	1063	28.45%	56.28%	15.27%
5	25	15.75%	75.81%	64.26%	537	1007	25.94%	54.39%	19.67%
5	50	7.88%	75.94%	59.98%	570	992	24.90%	53.97%	21.13%
10	10	12.09%	75.89%	62.57%	567	1040	23.85%	55.44%	20.71%
10	25	2.40%	75.87%	57.42%	**501**	**914**	20.71%	51.67%	27.62%
10	50	0.00%	**76.04**%	53.41%	591	1026	23.22%	56.07%	20.71%
20	25	0.00%	**76.04**%	50.66%	550	957	19.46%	53.01%	28.24%
20	50	0.00%	76.02%	51.87%	509	922	19.24%	50.00%	30.75%

In table 4.2, the results are presented according to the metrics discussed in chapter 2. Column offA and maxD are the values of `offArea` and `maxDisappeared` respectively for the said experiment. The value of `offset` was 2 meters and of `maxDistance` was 10 meters.

From the experiment, one of the conclusions reached is that the lower the value of the pair `offArea-maxDisappeared` the better the overall precision of the tracker which goes according to expectations if we take into account that if an object keeps being tracked for too long after disappearing, it will increase the risk of being associated with a different object that gets close to the area in which the missing object was last detected. The pair of values that provided the best overall precision were 1 (one) meter for `offArea` and 5 (five) frames for `maxDisappeared`.

Another conclusion that can be extracted is that the best values of the pair that leads to the lower combined value of fragmentations and ID switches are 10 (ten) frames for `offArea` and 25 (twenty-five) frames for `maxDisappeared`. This is due to various factors. One of them is that when an object leaves the area of interest it can reappear. If the values of `offArea` are too low it can stop being tracked right away and if it returns into the scene it would be considered a new object causing an ID switch. On the opposite side of the last example is the case when the `offArea` value is too high, that leads to the case of when an object goes outside the area of interest and is tracked for too long. This excessive tracking can cause an object that is detected near the area where the missing object was last seen to be associated with the "ghost" object creating an ID switch and a fragmentation. Another factor related with `maxDisappeared` that can lead to an increase in fragmentations and ID switches is when `maxDisappeared` is too low and can stop tracking objects if they stop being detected for short periods of time. For example, for being occluded. On the other side, when the value of `maxDisappeared` is too high, just like `offArea`, it causes ID switches, for example, if a pedestrian gets in a building in front of the sensor and another pedestrian passes by the position where the first pedestrian was last seen, whilst still being tracked, it can switch the ID of the second pedestrian with the first one.

Table 4.3: Experiments with offset, value in meters.

offset	MOTA	MOTP	Precision	ID	Frags	MT	PT	ML
0.5	34.92%	**75.67%**	78.10%	561	1035	28.66%	53.14%	18.20%
1	35.42%	75.66%	78.40%	546	1028	**29.29%**	52.93%	17.78%
2	35.77%	75.63%	78.75%	570	1045	**29.29%**	52.72%	17.99%
4	36.70%	75.65%	79.48%	**535**	**1020**	29.08%	53.35%	17.57%
6	**36.97%**	75.62%	**79.70%**	540	1026	28.87%	**53.77%**	**17.36%**

4.2 Different values for offset

In this section, some experiments with different values for offset are presented. As previously said, offset creates a boundary area in the limit of the field-of-view (yellow area in fig. 1.1). These experiments allow us to analyze what should be the size of the said boundary area to obtain better tracking results. The value of `maxDistance` was 10 (ten) meters and of `offArea` was 5 (five) frames.

In table 4.3, the results are shown according to the metrics presented in chapter 2.

We can conclude that the higher the value of offset, the better the results are. This is due to the larger offset area making the counter of frames outside the area starting earlier, compared to its position relative to the limit of the field of view, leading to when the object effectively exits the scene, it takes less time to stop being tracked. This is similar to having a smaller value of `offArea`, as it was shown in section 4.1.

4.3 With maximum distance for the association between centers

Some experiments with different values for `maxDistance` are presented in this section. As previously said, `maxDistance` restrains the maximum distance between association of centers in consecutive frames. The value of the other important variables are 2 (two) meters for `offset`, 1 (one) frame for `offArea` and 5 (five) frames for `maxDisappeared`.

If `maxDistance` was too big or did not exist at all, it could lead to the center association being made from one side of the field of view to the other. For example, if a car exited the field of view on the right side of the sensor and on one of the next frames, where the car was still being tracked, another car entered the scene from the left of the sensor, the association between the two centers could happen. On the other hand, if the value is too small, it can prevent associations of centers if an object is moving faster, as the difference in the position from one frame to the next is higher in those cases.

In table 4.4 the results are showcased. From these results, we can conclude what was expected, that a value too small or too big, leads to worse results. However these differences are not as big as expected. This is related to the fact that the dataset does not have enough scenarios where the context and purpose of this experiment can be properly tested as it is an edge case.

Table 4.4: Experiments with maxDistance (mDist), values are in meters.

mDist	MOTA	MOTP	Precision	ID	Frags	MT	PT	ML
5	34.75%	75.67%	78.24%	**541**	**1002**	27.62%	**53.97%**	18.41%
10	**35.77%**	75.64%	**78.75%**	570	1045	29.29%	52.72%	17.99%
15	35.69%	**75.69%**	78.65%	554	1048	29.70%	52.93%	**17.36%**
30	35.68%	75.68%	78.54%	565	1046	**30.13%**	52.09%	17.78%

4.4 Ideal 2D detector and 3D extractor

In this subsection, some experiments to simulate the use of an ideal 2D detector and 3D extractor were carried out. An ideal detector means a detector that detects all the objects and corresponding centers in the scene without any kind of error associated. For that, the output of the 2D detector was replaced with the ground truth values for one experiment ("2D" in column "Exp." of table 4.5) and both the outputs from the 2D detector and the 3D extractor in the other experiment ("3D" in column "Exp." of table 4.5).

The values used in the critical variables, tested in the previous subsections, are 4 (four) meters, 10 (ten) meters, 5 (five) frames and 1 (one) frame for `offset`, `maxDistance`, `maxDisappeared` and `offArea` respectively.

In table 4.5, the results are showcased according to the metrics presented in chapter 2. Column "Value" represents the values of `maxDistance` in the experiment. The experiment represented as "OG" represents the original solution without the ideal 2D detector and 3D extractor.

The purpose of these experiments is to evaluate how much the errors from the third-party-based modules ("2D detector" and "3D extractor") propagate through the created solution and affect the final results.

The "2D" experiment shows a big increase in the overall precision of the results and the percentage of trajectories that are tracked for more than 80 (eighty) percent of their overall length but at the same time more than doubling the number of ID switches. This huge increase in ID switches is related with the fact that the "2D detector" does not detect all objects in a crowded scenario and now having all the objects properly identified the "3D extractor" has difficulties extracting and separating the centers of overlapped or really close objects. This eventually leads to objects in crowded areas having IDs switched a lot of times.

On experiment "3D" and after removing all the errors introduced by the referred modules we can evaluate the proposed solution on its own. An enormous decrease on the number of ID

Table 4.5: Experiments with ideal detector.

Exp.	MOTA	MOTP	Precision	ID	Frags	MT	PT	ML
OG	35.77%	75.64%	78.75%	570	1045	29.29%	52.72%	17.99%
2D	54.93%	95.70%	91.31%	1215	1471	65.90%	10.46%	23.64%
3D	**84.98%**	**95.95%**	**91.39%**	**42**	**41**	**87.45%**	**11.51%**	**1.05%**

Table 4.6: Trajectories from KITTI dataset isolated for experiments with the Kalman Filter.

Sequence	Track ID	Length (in frames)
0	0	154
0	1	154
1	86	87
2	3	180
3	1	122
4	2	314
5	31	297
8	8	390
8	13	262
8	14	70
9	66	612
10	0	294
11	0	373
14	15	51
15	2	373
17	6	145
17	7	145
18	2	264
18	3	285
19	88	89
20	122	255

switches and fragmentations can be seen as well as a considerable increase in the percentage of trajectories that are tracked for most than 80 (eighty) percent of its length.

Finally, we can conclude that object detection has a big influence on the efficiency of the tracking approach as the increment of 13 percent in the overall precision shows.

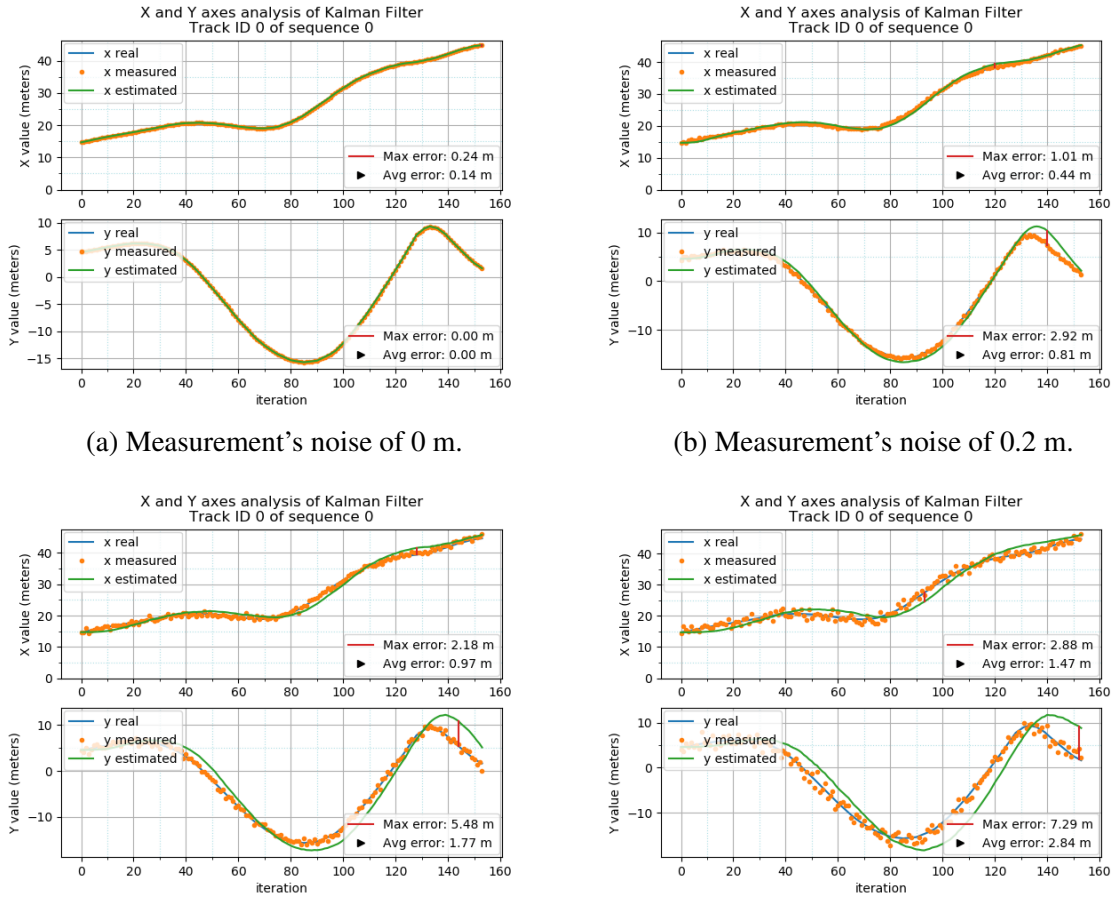
4.5 Different Kalman filter setups

The Kalman filter has different parameters that influence its performance. Those parameters include the time interval used to update/correct the filter, *dt*, the acceleration magnitude and the noise linked with it, *u* and *u_noise*, and finally the noise associated with the provided measurements, *meas_noise*.

The Z axis was not considered on these experiments as it represents vertical movement that is not relevant for the scope of this dissertation, as the tracked objects move on the ground plane.

For these experiments, some objects' trajectories (a total of twenty one) from the KITTI dataset were isolated in order to enable an easier and cleaner way to perform the experiments and achieve results representative of the performance of the KF in the different setups.

The object trajectories isolated are presented in table 4.6. These sequences were chosen for representing different object movements, road conditions and scenarios.



(a) Measurement's noise of 0 m.

(b) Measurement's noise of 0.2 m.

(c) Measurement's noise of 0.5 m.

(d) Measurement's noise of 1 m.

Figure 4.1: Kalman Filter's experiments with measurement's noise.

4.5.1 Noise associated with measurements (meas_noise)

On this section, experiments with various values of noise associated with the measurements used to update the KF are explained. The importance of these experiments is to evaluate how imprecise measurements can affect the estimates. For that, a random noise, bounded by the noise with which the filter was initialized, was added to the ground truth trajectory. The noise values used were 0, 0.2, 0.5 and 1, all in meters. The value used for dt was 100 milliseconds, for u 0.2 meters per square second and 0.5 for u_noise.

Graphics showing the behavior of the KF, in these experiments, can be examined in figure 4.1.

As figure 4.1.a shows, if the measurements provided to update the KF do not have any noise associated with them, the predictions are close to perfect, only having a maximum error of 0.24 meters and an average error of 0.13 meters, for the example shown. This case mimics a perfect scenario that does not reflect what happens in the implementation done in this dissertation because the obtained measures always have a minimum error associated with it.

A value of noise that approximates the estimates to a more realistic context is 0.5 meters (figure 4.1.c that leads to an average error of 0.97 meters for the X axis and of 1.77 meters on the Y axis,

with a maximum error of 2.18 meters and of 5.48 meters, respectively. This high value for the maximum error on the Y axis happens because of a change in the Y component of the direction of the movement, to which the KF takes some time to adjust.

4.5.2 Acceleration (u) and noise related with it (u_noise)

The acceleration is one of the values that conditions the estimates of the KF. Even without being known, a initial pair (acceleration-acceleration noise) must be chosen for the KF to work properly. This part presents the experiments done with different values for the said pair and tries to select the one which gives the most desirable results. The value used for *dt* was 100 milliseconds and a noise associated with the measurements(*meas_noise*) of 0.5 meter. The experiments used all the possible pairs for the following values for *u* -0.2, 0, 0.2 and 0.5 meters per square second and for *u_noise* 0, 0.5 and 1.

Graphics presenting the behavior of the KF in these experiments can be examined in figure 4.2 and 4.3. As it is possible to check, in figures 4.2.a, 4.2.b, 4.3.a and 4.3.b, a value of 0 for the noise associated with the acceleration, independently of the value of the acceleration, leads to an inflexible KF that does not adjust itself based on the updates provided — only the original value of the acceleration is taken into account to predict the next state. This leads to an average error of 26.51 meters and a maximum average error of 62.22 meters in the worst case — figure 4.3.b.

On the other side, higher noise values lead to a more flexible KF in which the measurements provided have a higher influence on the predictions and the tuning of the KF itself. However, values too high for this noise will lead to the measurements having a influence too big on the estimates.

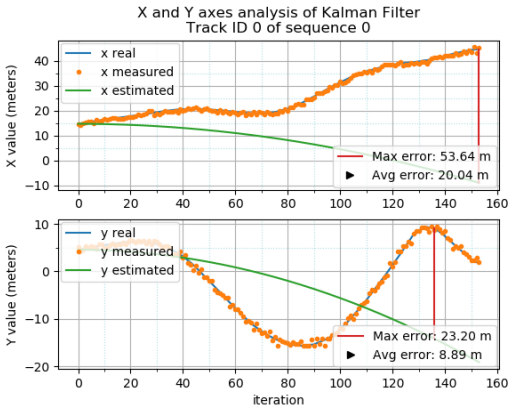
Having all this into account and in order to achieve a balanced solution, that represents the natural movement, the chosen value for the noise (*u_noise*) was 1 and for acceleration (*u*) 0.2 meters per square second — figure 4.3.e — that, in the example provided, had a maximum error of 1.68 meters for the X axis and of 3.37 meters for the Y axis, while the average error were 0.62 meters and 0.97 meters, respectively. Like previously said, this high value for the maximum error on the Y axis happens because of a change in the Y component of the direction of the movement.

4.5.3 Interval update (dt)

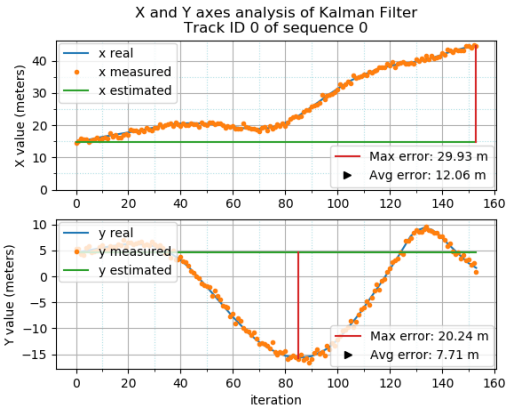
The refresh rate of the Velodyne scanner used in the KITTI dataset was set to 10 Hz (ten hertz) that corresponds to approximately an update each 100 ms (one hundred milliseconds). This leads to the minimum possible value of *dt* being 0.1 seconds. The other parameters of the KF have the following value: 0.2 meters per square second for acceleration (*u*); 1 for acceleration's noise (*u_noise*); 0.5 meters for noise associated with measurements (*meas_noise*).

The selected time intervals chosen for testing purposes were 100, 200 and 500 milliseconds corresponding to ten, five and two hertz respectively.

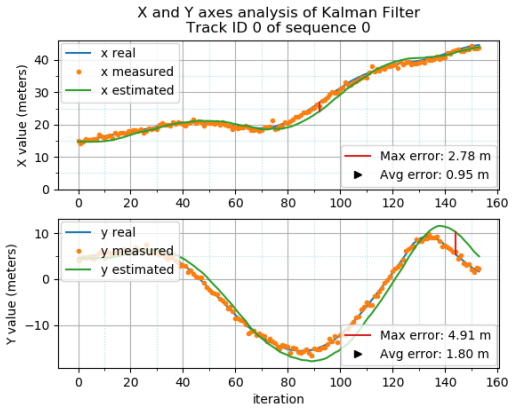
Graphics showing the behavior of the KF, in these experiments, can be examined in figure 4.4.



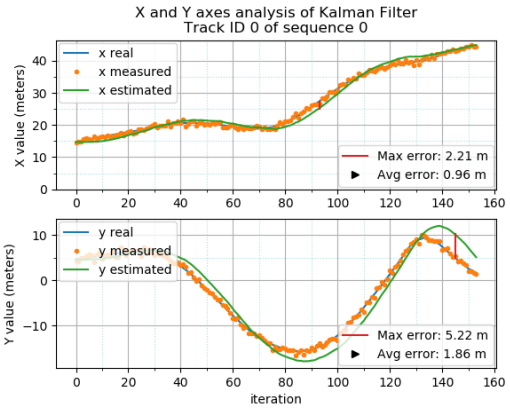
(a) Acceleration of -0.2 m/s^2 and noise of 0.



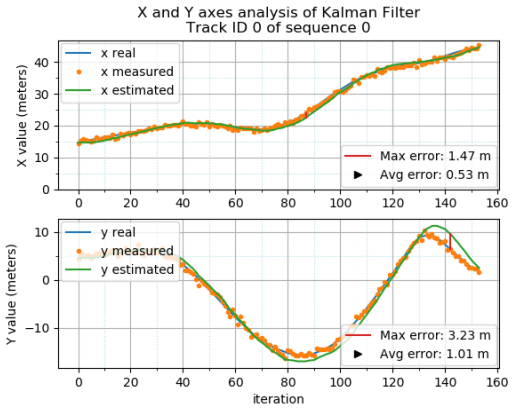
(b) Acceleration of 0 m/s^2 and noise of 0.



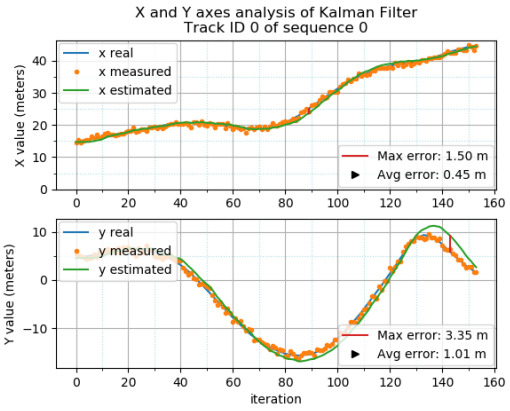
(c) Acceleration of -0.2 m/s^2 and noise of 0.5.



(d) Acceleration of 0 m/s^2 and noise of 0.5.

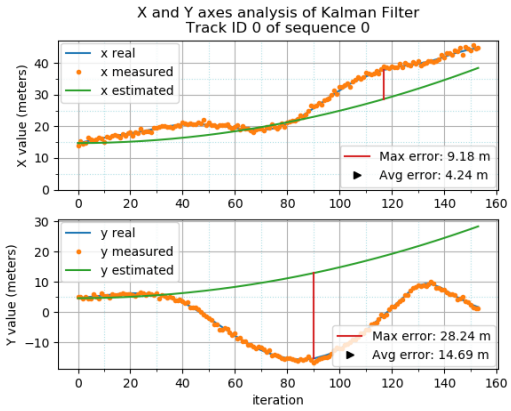


(e) Acceleration of -0.2 m/s^2 and noise of 1.

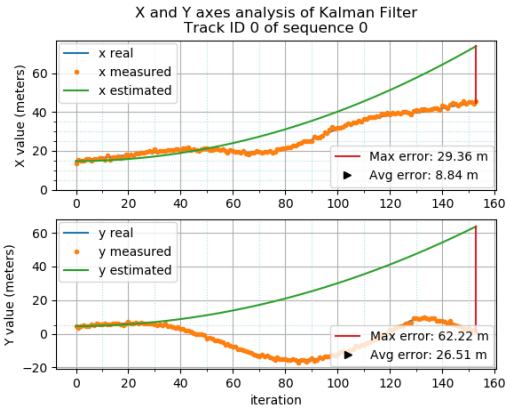


(f) Acceleration of 0 m/s^2 and noise of 1.

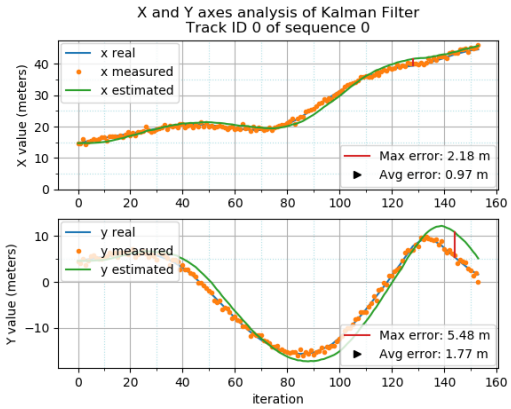
Figure 4.2: Kalman Filter's experiments with acceleration and respective noise (part 1).



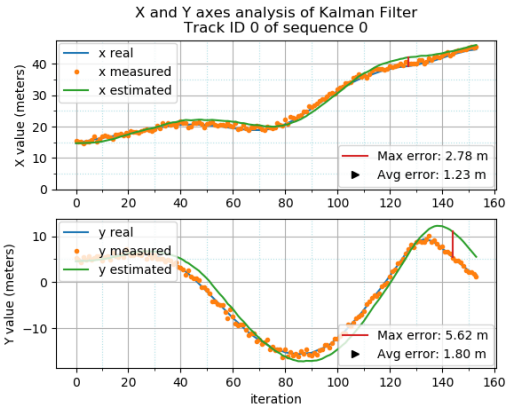
(a) Acceleration of 0.2 m/s^2 and noise of 0.



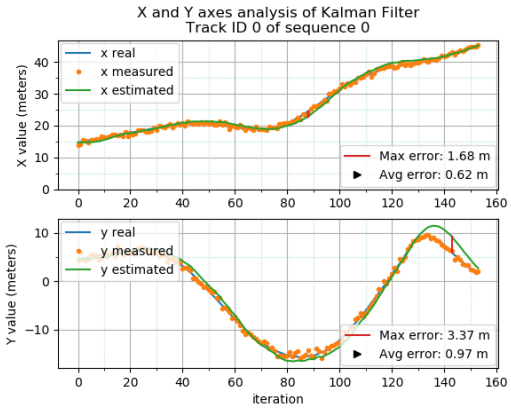
(b) Acceleration of 0.5 m/s^2 and noise of 0.



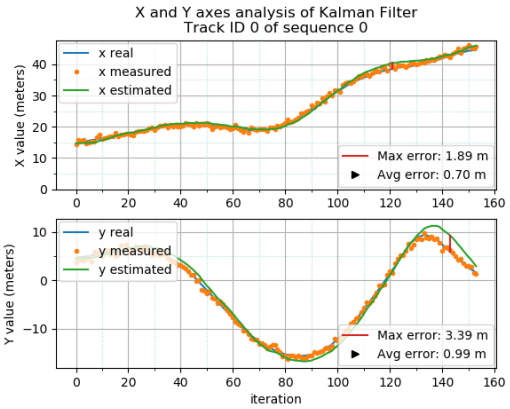
(c) Acceleration of 0.2 m/s^2 and noise of 0.5.



(d) Acceleration of 0.5 m/s^2 and noise of 0.5.

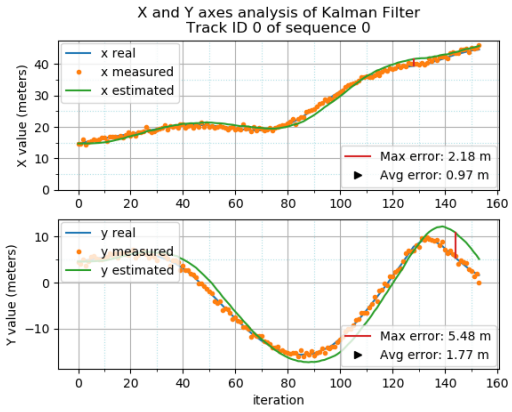


(e) Acceleration of 0.2 m/s^2 and noise of 1.

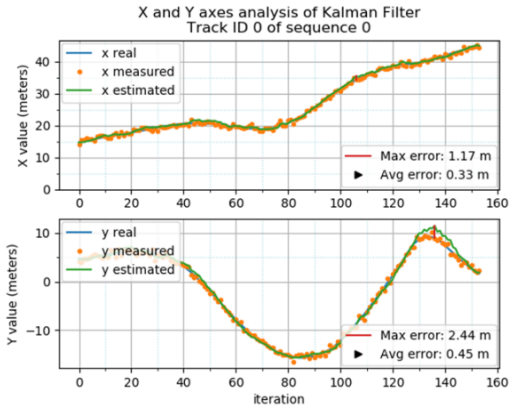


(f) Acceleration of 0.5 m/s^2 and noise of 1.

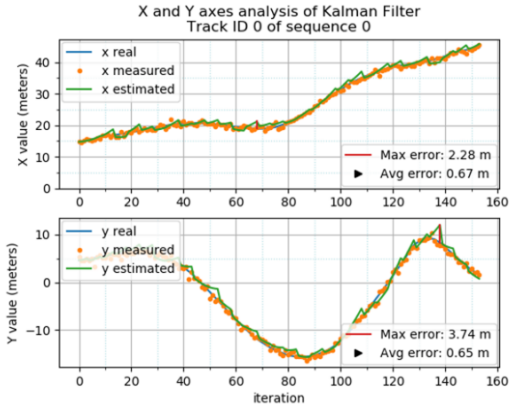
Figure 4.3: Kalman Filter's experiments with acceleration and respective noise (part 2).



(a) Refresh rate of 10 Hz



(b) Refresh rate of 5 Hz



(c) Refresh rate of 2 Hz

Figure 4.4: Kalman Filter's experiments with different refresh rates.

The lowest update rates tested, 100 and 200 milliseconds (figure 4.4.a and 4.4.b), showed the best performances. The 200 ms update interval, showed a slightly higher adaptability to changes in the direction of the movement. This is related to the higher influence the measures received have to the tuning of the KF in a lower update frequency scenario.

After a certain update rate, the measures are not enough to properly correct the KF, leading to an increase in the error through time, as figure 4.4.c shows for the X-axis. The conclusion achieved previously, that relates the direction change to the higher update rate also applies to the Y-axis.

Another scenario considered relevant to be tested is one where the updates are in irregular intervals, for reasons like not receiving a measure because of a sensor failure or receiving a corrupted measure, to simulate that an experiment where the update had a percentage of being successfully done associated. The used percentages were 100% (one hundred percent), 90 (ninety percent), 50 (fifty percent) and 25 (twenty-five percent). On this experiment, the expected update rate is 100 (one hundred) milliseconds.

Graphics displaying the behavior of the KF, in these experiments, can be examined in figure 4.5.

Table 4.7: Comparison of average and maximum errors for different percentages of successful updates.

Up. %	Avg. X ε	Dif.	Max. X ε	Dif.	Avg. Y ε	Dif.	Max. Y ε	Dif.
100%	0.63 m	——	1.74 m	——	1.03 m	——	3.81 m	——
90%	0.64 m	0.01 m	1.67 m	0.07 m	1.07 m	0.04 m	3.79 m	0.02 m
50%	0.83 m	0.20 m	2.34 m	0.60 m	1.35 m	0.32 m	6.21 m	2.40 m
25%	1.46 m	0.83 m	7.63 m	5.89 m	2.15 m	1.12 m	7.20 m	3.39 m

These experiments do not aim at choosing the best value for a certain parameter of the KF, they only have the goal to show the tolerance of the KF to measurement failures.

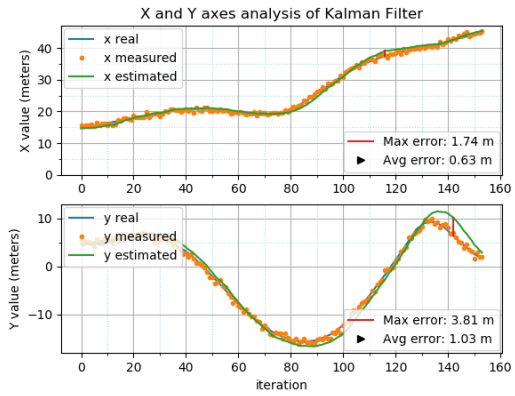
The implemented KF shows a great tolerance to errors, presenting minimal differences between success update rates of 100% (figure 4.5.a) and of 90% (figure 4.5.b). Even for a success rate of only 50% (figure 4.5.c), the results obtained can still be considered acceptable when compared to a success rate of 100% only showing a increment of around 0.30 meters in the average errors. For success rates of 25% (figure 4.5.d) the error values deteriorate immensely and the predictions show erratic behavior, with the errors more than doubling its values when compared to the 100% rate. The concrete values for the differences in the errors can be examined in table 4.7.

4.6 Summary

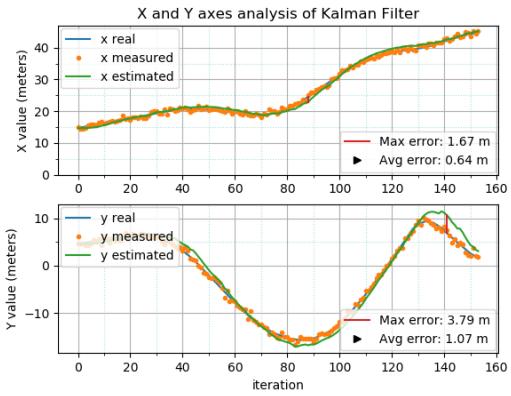
The conclusions reached through the experiments, done and explained in this chapter, are the definition of an ideal setup for both the tracker and the predictor module.

For the tracker, the parameters that can be tuned are *maxDisappeared*, $offArea$, $offset$ and *maxDistance*. The values were chosen to achieve a balanced configuration for the final setup of the tracker, and those values are: 5 frames for *maxDisappeared*; 1 frame for $offArea$; 4 meters for $offset$; and 10 meters for the *maxDistance*.

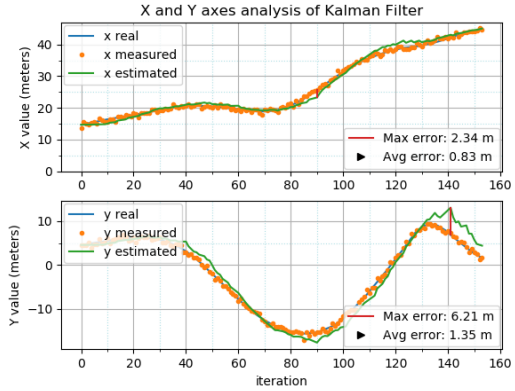
For the predictor, the parameters that can be adjusted are the acceleration (u), the noise associated with it (*u_noise*), the noise associated with the measurements *meas_noise* and the update interval (dt). The first two, u and *u_noise*, are the ones that should be regulated to achieve the best possible performance, as they represent an unknown variable, *meas_noise* and dt, should represent as close as possible the known parameters of the environment. The values assigned are, therefore: 0.5 meters for the measurement's noise; 0.1 seconds (100 milliseconds) for the update frequency; 0.2 meters per square second for the acceleration and 1 for the noise associated with it. Examples of the behavior of the Kalman filter configured based on these conclusions are displayed in appendix B.



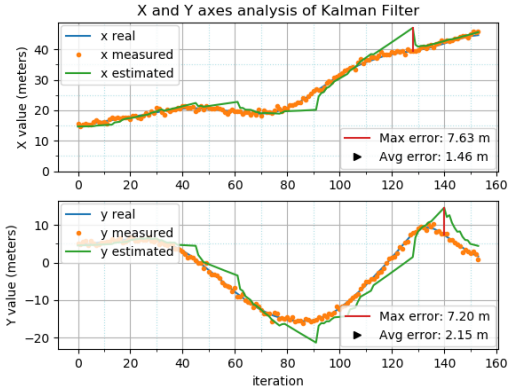
(a) 100% success rate



(b) 90% success rate



(c) 50% success rate



(d) 25% success rate

Figure 4.5: Kalman Filter's experiments with different update success rates.

Chapter 5

Conclusions

This chapter provides an overview of the achievements accomplished, constraints identified, how they can be dealt with and what type of projects can be followed to enhance the provided solution. The fulfillment of the objectives is therefore presented as a summary of the work developed during the elaboration of this dissertation.

5.1 Main Contributions and Findings

The global objective is a proposal of a possible pipeline for 3D object tracking and movement prediction. This pipeline combines the 2D information from an RGB camera with the 3D information of a LiDAR sensor in order to obtain the 3D centers and classes of the objects detected. This information is then used to associate the objects in consecutive frames, allowing to draw the trajectory of each one of them through time. The pipeline is also able to predict where each object will be in the future.

One of the main contributions of this dissertation is the application of the YOLO detector to road scenarios like the ones made available by KITTI and the ability to use a YOLO detector trained for a generic dataset to extrapolate classes of another dataset. For example, from the class *person* and *bicycle* of the COCO dataset, the pipeline is able to identify the class *Cyclist* from the KITTI dataset. This can be useful in scenarios where there are not enough resources to retrain YOLO for the needed environment. On the context of this dissertation that would be equivalent to retraining the model to the KITTI dataset. Another contribution of this project is the sub-pipeline between YOLO and Frustum-PointNet (adapted to Python 3) that allows a seamless flow of data between both components without the need to create auxiliary files.

The other main contributions are the predictor and the tracker themselves. The tracker uses the Euclidean distance for the association of the centers, an approach that is not much explored in the context of 3D laser data. The smooth and intertwined operation of the predictor and the tracker is also a novelty as both the modules work together during the cycle with a low demand

of computational power and in a short period of time (around 1.6 milliseconds). An extensive empirical evaluation of the proposed solution was also performed to understand its behaviour with different initial parameters and to configure it so that it can handle the scenarios it will be applied to with the best possible results.

5.2 Limitations of the Solution

Although the solution can track and predict objects at a good level it has some restraints in its modules that limit the overall performance. Some of the identified limitations in the first part of the pipeline, 2D detector and 3D extractor, are errors introduced that spreads into the succeeding modules. One of those includes the error introduced by a wrong initial classification of an object by the 2D detector. Such classification will compromise the association in future frames as it only happens if the class associated to the center in frame $k+1$ is the same as in k. Another error related with these modules, is the inability of the used YOLO to detect some crucial classes for road scenarios, for example cyclists, as it was trained for the COCO dataset — a generic object classification dataset. There is also time constraint linked with these two modules, that is the percentage of the total execution time that they occupy, around 80%.

Focusing on the modules that are the novelty of this dissertation. The main restraint caused by the tracker is how association is performed. It only takes into account the class and the distance in-between frames of the object, ignoring other characteristics of the object that can be useful like color or orientation.

Finally, limitations of the predictor are related with the nature of the simple Kalman filter used itself, as it does not handle acceleration variations in the most desirable way. Another limitation is associated with errors in the tracking part and how they affect the performance of the predictor. This predictor, as explained in section 3.2.4.2, is tuned online and as so depends on a correct tracking of objects to achieve the best predictions, so errors in tracking (ID switches and trajectories fragmentations) affect the predictor estimates.

5.3 Further Development and Future Work

Some further work that could be explored to tackle the limitations related to the tracker that have been exposed previously are explained below. One of the possible improvements would be to enhance the association heuristic by creating a more in-depth object identity that takes into account as much characteristics of the object as possible in order to differentiate them. An example of those characteristics includes the color histograms that represent the object, the orientation of its movement and the volume of its point cloud.

Taking into account the predictions of the KF to help with the association is also a scenario that can deserve some further investigation. A simple and straight-forward solution for the problem of not detecting classes like cyclist and motorcyclist would be training YOLO for a road scenario-specific context with the help of datasets like the KITTI itself.

Finally, approaching some of the flaws identified for the predictor, an approach based on an Extended Kalman filter instead of a simple one has the potential to deliver better results as it handles the acceleration related variations in a more suitable way. Also having a class-specific pre-tuned Kalman filter could bring interesting outcomes to the estimates.

Contextualization and identity of the object being followed are subjects that are giving their first steps nowadays and have the potential to revolutionize how tracking is done and other areas that depend on computer vision. However, in order to be able to take advantage of these topics, some major breakthroughs must be achieved not in the area of object tracking nor computer vision but in a broader area, the area of Artificial Intelligence itself as a whole. Those breakthroughs include giving a machine the ability to understand if a person is either running, walking, standing or sitting. Having this capability would allow the tracker to not only treat each case specifically but also to give an importance level to each one of them. A possible scenario that would require a higher importance level would be someone running towards the road. This can have some relevance in cases where some objects have to be chosen over others because of lack of computational power to process all the objects in the scene, for example. Another use case where contextual tracking could reveal itself as an invaluable tool would be when a ball is detected coming from one of the road sides which would signal the tracker to prepare itself to start tracking a person in the area where the ball was found.

This dissertation was a great opportunity to explore an area that is growing at an amazing pace. However, this pace brings some challenges with it. One of the biggest obstacles faced was related to the difficulty of leveraging methods from different domains to create an approach that works in a seamless and reliable way to detect and track objects and predict their movement. Despite having faced all of these hurdles, the proposed solution delivers more than satisfying results that contribute to this ever growing field, having made this a fruitful and rewarding experience.

References

[aC19] Toyota Technological Institute at Chicago. Toyota Technological Institute at Chicago. http://www.ttic.edu/, 2019. Accessed: 2019-02-06.

[BH+92] Robert Grover Brown, Patrick YC Hwang, et al. *Introduction to random signals and applied Kalman filtering*, volume 3. Wiley New York, 1992.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BS08] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008.

[Cho15] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE international conference on computer vision*, pages 3029–3037, 2015.

[CKZ+15] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[CKZ+16] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.

[CMW+17] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE Computer vision and pattern recognition (CVPR)*, volume 1, page 3, 2017.

[CUF18] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE, 2018.

[DCTB16] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Motion-based detection and tracking in 3d lidar scans. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4508–4513. IEEE, 2016.

[dLBW11] Universität des Landes Baden-Württemberg. Team AnnieWAY. https://www.mrt.kit.edu/annieway/, 2011. Accessed: 2019-02-06.

REFERENCES

[DVP19] Martin Dimitrievski, Peter Veelaert, and Wilfried Philips. Behavioral pedestrian tracking using a camera and lidar sensors on a moving vehicle. *Sensors*, 19:391, 2019.

[FHLA18] Mengdan Feng, Sixing Hu, Gimhee Lee, and Marcelo Ang. Towards precise vehicle-free point cloud mapping: An on-vehicle system with deep vehicle detection and tracking. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1288–1293. IEEE, 2018.

[FS09] James Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–6. IEEE, 2009.

[Gir09] Ross Girshick. *Object detection with heuristic coarse-to-fine search*. PhD thesis, University of Chicago, Chicago, 2009.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[KGS+09] Rangachar Kasturi, Dmitry Goldgof, Padmanabhan Soundararajan, Vasant Manohar, John Garofolo, Rachel Bowers, Matthew Boonstra, Valentina Korzhova, and Jing Zhang. Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):319–336, 2009.

[KN10] Louis Kratz and Ko Nishino. Tracking with local spatio-temporal motion patterns in extremely crowded scenes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 693–700. IEEE, 2010.

[LGG+18] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[LHN09] Yuan Li, Chang Huang, and Ram Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2953–2960. IEEE, 2009.

[Li17] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1513–1518. IEEE, 2017.

[LKR+16] G. Lira, Z. Kokkinogenis, R. J. F. Rossetti, D. C. Moura, and T. Rúbio. A computer-vision approach to traffic analysis over intersections. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 47–53, Nov 2016.

[LRB09] P. F. Q. Loureiro, R. J. F. Rossetti, and R. A. M. Braga. Video processing techniques for traffic information acquisition using uncontrolled video streams. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–7, Oct 2009.

[LZK14] Wenhan Luo, Xiaowei Zhao, and Tae-Kyun Kim. Multiple object tracking: A review. *CoRR*, abs/1409.7618, 2014.

[MLR+16] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[MLTR+16] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. March 2016.

[NSR18] J. Neto, D. Santos, and R. J. F. Rossetti. Computer-vision-based surveillance of intelligent transportation systems. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–5, June 2018.

[NT01] José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on robotics and automation*, 17(6):890–897, 2001.

[PR12] José L. F. Pereira and Rosaldo J. F. Rossetti. An integrated architecture for autonomous vehicles simulation. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 286–292, New York, NY, USA, 2012. ACM.

[PT09] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[QLW+18] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[RF17] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[RV11] P. Ramachandran and G. Varoquaux. Mayavi: 3D Visualization of Scientific Data. *Computing in Science & Engineering*, 13(2):40–51, 2011.

[RVCV11] Branko Ristic, Ba-Ngu Vo, Daniel Clark, and Ba-Tuong Vo. A metric for performance evaluation of multi-target tracking algorithms. *IEEE Transactions on Signal Processing*, 59(7):3452–3457, 2011.

[SAMK18] Sarthak Sharma, Junaid Ahmed Ansari, J Krishna Murthy, and K Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3508–3515. IEEE, 2018.

[SBR+18] Samuel Scheidegger, Joachim Benjaminsson, Emil Rosenberg, Amrit Krishnan, and Karl Granström. Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 433–440. IEEE, 2018.

[SJSRC10] Bi Song, Ting-Yueh Jeng, Elliot Staudt, and Amit K Roy-Chowdhury. A stochastic graph evolution framework for robust multi-target tracking. In *European Conference on Computer Vision*, pages 605–619. Springer, 2010.

[SLH+14] Xinchu Shi, Haibin Ling, Weiming Hu, Chunfeng Yuan, and Junliang Xing. Multi-target tracking with motion context in tensor power iteration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3518–3525, 2014.

[Tot09] Charles K Toth. R&d of mobile lidar mapping and future trends. In *Proc. ASPRS Annu. Conf*, pages 9–13, 2009.

[WB+95] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[WPN15] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. Model-free detection and tracking of dynamic objects with 2d lidar. *The International Journal of Robotics Research*, 34(7):1039–1063, 2015.

[XAS15] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *International Conference on Computer Vision (ICCV)*, pages 4705–4713, 2015.

[XCLS17] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[XLW+18] Hongyu Xu, Xutao Lv, Xiaoyu Wang, Zhou Ren, Navaneeth Bodla, and Rama Chellappa. Deep regionlets for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 798–814, 2018.

[YHN11] Bo Yang, Chang Huang, and Ram Nevatia. Learning affinities and dependencies for multi-target tracking using a crf model. In *CVPR 2011*, pages 1233–1240. IEEE, 2011.

[ZWW+15] Shun Zhang, Jinjun Wang, Zelun Wang, Yihong Gong, and Yuehu Liu. Multi-target tracking by learning local-to-global trajectory models. *Pattern Recognition*, 48(2):580–590, 2015.

Appendix A

Input/Output Files Structure

Table A.1: KITTI available data.

Type of Data	Format	Size (training - testing)
RGB Images	.png	6.07 GB - 8.64 GB
Stereo Images	.png	5.79 GB - 8.25 GB
Velodyne Point Clouds	.bin	20.3 GB - 20.0 GB
GPS/IMU	.txt	3.31 MB - 4.64 MB
Camera Calibration	.txt	33.1 KB - 45.7 KB
Training Labels	.txt	8.79 MB - n/a
L-SVM reference detections [Gir09]	.txt	317 MB - 449 MB
Regionlet reference detections [XLW+18]	.txt	89.1 MB - 126 MB

Table A.2: KITTI labels file structure.

Nr of Values	Name	Description
1	frame	Frame within the sequence where the object appears
1	track id	Unique tracking id of this object within this sequence
1	type	Describes the type of object: *Car*, *Van*, *Truck*, *Pedestrian*, *Person_sitting*, *Cyclist*, *Tram*, *Misc* or *DontCare*
1	truncated	Integer (0,1,2) indicating the level of truncation. Note that this is in contrast to the object detection benchmark where truncation is a float in [0,1].
1	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded 2 = largely occluded, 3 = unknown
1	alpha	Observation angle of object, ranging [-pi..pi]
4	bbox	2D bounding box of object in the image (0-based index): contains left, top, right, bottom pixel coordinates
3	dimensions	3D object dimensions: height, width, length (in meters)
3	location	3D object location x,y,z in camera coordinates (in meters)
1	location_y	Rotation ry around Y-axis in camera coordinates [-pi..pi]
1	score	Only for results: Float, indicating confidence in detection, needed for p/r curves, higher is better.

Table A.3: Output file structure for results and evaluation.

Nr of Values	Name	Description
1	frame	Frame within the sequence where the object appears
1	object id	Unique tracking id of this object within this sequence
1	class	Describes the type of object: *Car*, *Pedestrian*, *Cyclist*
1	truncated	As it could not be calculated is always set to 0 (not truncated)
1	occluded	As it could not be calculated is always set to 0 (fully visible)
1	alpha	As it could not be calculated is always set to 0 [-pi..pi]
4	bbox	As it could not be calculated all for points are set to -1
3	dimensions	3D object dimensions: height, width, length not calculated (in meters)
3	location	3D object location x,y,z in camera coordinates (in meters)
1	location_y	Rotation ry around Y-axis in camera coordinates [-pi..pi]
1	score	Only for results: Float, indicating confidence in detection, needed for p/r curves, higher is better.

Appendix B

Kalman Filter Experiments

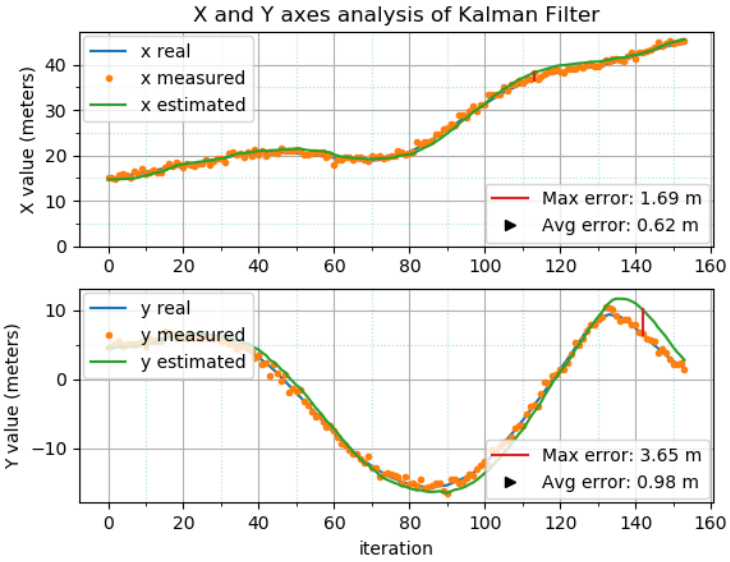


Figure B.1: Example 1 of the behavior of the KF with the chosen setup.

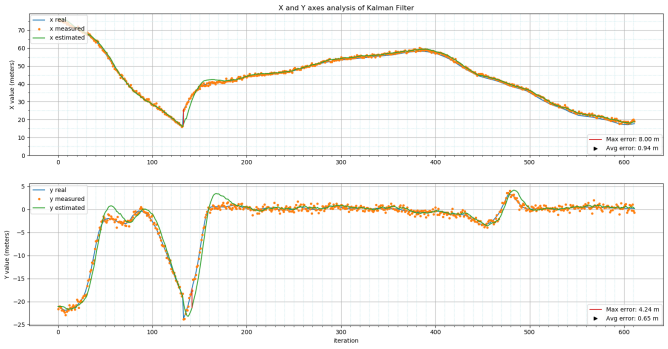


Figure B.2: Example 2 of the behavior of the KF with the chosen setup.

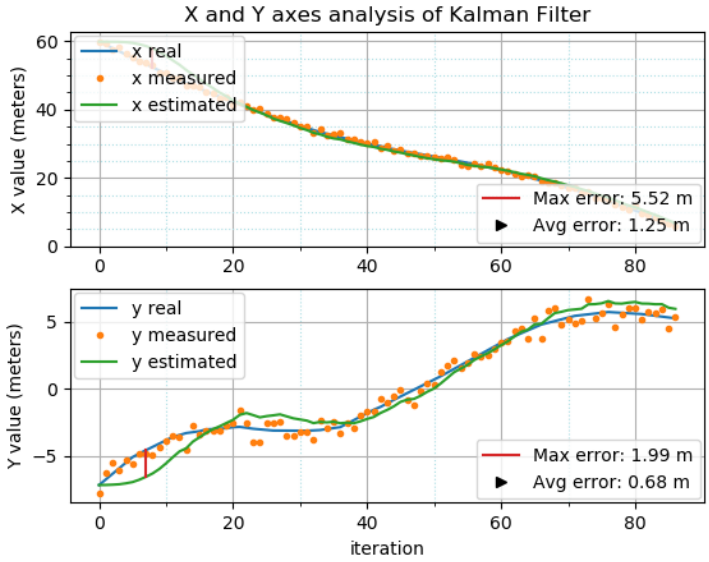


Figure B.3: Example 3 of the behavior of the KF with the chosen setup.

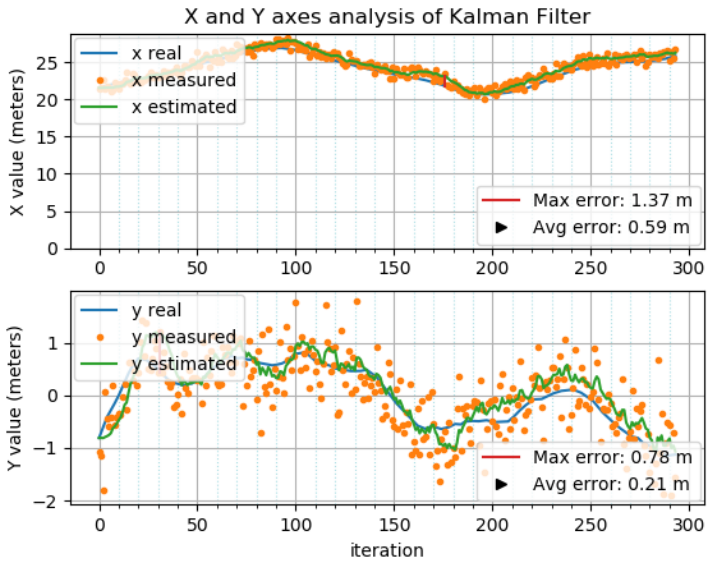


Figure B.4: Example 4 of the behavior of the KF with the chosen setup.

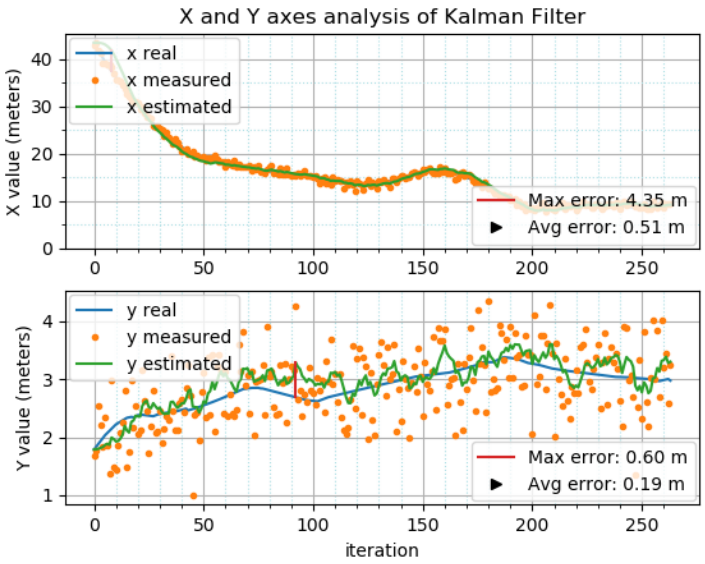


Figure B.5: Example 5 of the behavior of the KF with the chosen setup.