LANE TRACKING USING DEPENDENT EXTENDED TARGET MODELS

LANE TRACKING USING DEPENDENT EXTENDED TARGET MODELS

BY

BEHZAD AKBARI, Computer Engineering PhD Candidate

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2020) McMaster University

(Electrical and Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Lane Tracking using Dependent Extended Target Models
AUTHOR:	Behzad Akbari Computer Engineering PhD Candidate, McMaster University, Hamilton, Canada
SUPERVISOR:	Prof. T. Kirubarajan
NUMBER OF PAGES:	xvi, 143

Abstract

Detection of multiple-lane markings (lane-line) on road surfaces is an essential aspect of autonomous vehicles. Although several approaches have been proposed to detect lanes, detecting multiple lane-lines consistently, particularly across a stream of video frames and under varying lighting conditions is still a challenging problem. Since the road's markings are designed to be smooth and parallel, lane-line sampled features tend to be spatially and temporally correlated inside and between frames. In this thesis, we develop novel methods to model these spatial and temporal dependencies in the form of the target tracking problem. In fact, instead of resorting to the conventional method of processing each frame to detect lanes only in the space domain, we treat the overall problem as a Multiple Extended Target Tracking (METT) problem.

In the first step, we modeled lane-lines as multiple "independent" extended targets and developed a spline mathematical model for the shape of the targets. We showed that expanding the estimations across the time domain could improve the result of estimation. We identify a set of control points for each spline, which will track over time. To overcome the clutter problem, we developed an integrated probabilistic data association filter (IPDAF) as our basis, and formulated a METT algorithm to track multiple splines corresponding to each lane-line.

In the second part of our work, we investigated the coupling between multiple

extended targets. We considered the non-parametric case and modeled target dependency using the Multi-Output Gaussian Process. We showed that considering dependency between extended targets could improve shape estimation results. We exploit the dependency between extended targets by proposing a novel recursive approach called the Multi-Output Spatio-Temporal Gaussian Process Kalman Filter (MO-STGP-KF). We used MO-STGP-KF to estimate and track multiple dependent lane markings that are possibly degraded or obscured by traffic. Our method tested for tracking multiple lane-lines but can be employed to track multiple dependent star convex rigid-shape targets by using the measurement model in the radial space.

In the third section, we developed a Spatio-Temporal Joint Probabilistic Data Association Filter (ST-JPDAF). In multiple extended target tracking problems with clutter, sometimes extended targets share measurements: for example, in lane-line detection, when two-lane markings pass or merge together. In single-point target tracking, this problem can be solved using the famous Joint Probabilistic Data Association (JPDA) filter. In the single-point case, even when measurements are dependent, we can stack them in the coupled form of JPDA. In this last chapter, we expanded JPDA for tracking multiple dependent extended targets using an approach called ST-JPDAF. We managed dependency of measurements in space (inside a frame) and time (between frames) using different kernel functions, which can be learned using the training data. This extension can be used to track the shape and dynamic of dependent extended targets within clutter when targets share measurements.

The performance of the proposed methods in all three chapters are quantified on real data scenarios and their results are compared against well-known model-based, semi-supervised, and fully-supervised methods. The proposed methods offer very

promising results.

To my family

Acknowledgements

I would like to use this opportunity to express my great appreciation to my supervisor, Prof. T. Kirubarajan, for supporting me financially and for all the academical guidance and suggestions through the journey of this research. Working with him was one of my most profound honours. Also, I want to thank and express my sincere gratitude to my committee members, Prof. I. Bruce and Prof. Jeremic, for all of their great and helpful suggestions during my degree journey. I appreciate all the time that they have spent on my supervisory meetings and reading my thesis. I give my Special thanks to Cheryl Gies, for her continuous administrative support, Christina Dalima for the careful review of my thesis.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviations

ADAS	Advanced Driver Assistance Systems
ACC	Adaptive Cruise Control
CNN	Convolutional Neural Network
CD	Collision Detection and Avoidance
CPN	Convolutional Patch Network
ET	Extended Target
EKF	Extended Kalman Filter
FP	False-Positive
FN	False-Negative
GTT	Group Target Tracking
GP	Gaussian Process

GTT Group Target Tracking

GP-KF Gaussian Process – Kalman Filter

GP-STKF Gaussian Process Spatio-Temporal – Kalman Filter

HT Hough Transform

IPM Inverse Perspective Mapping

IPDA Integrated Probabilistic Data Association

JPDA Joint Probabilistic Data Association

JPDACF Joint Probabilistic Data Association Coupled Filter

K-JPDACF Kernel-based Joint Probability Data Association Coupled Filter

ST-JPDAF Spatio-Temporal Joint Probabilistic Data Association Filter

LDW Lane Departure Warning

LKA Lane Keep Assist

LCM Lane Change Merge

LSD Line Segment Detection

METT Multiple Extended Target Tracking

MO-STGP-KF

Multi-Output Spatio-Temporal Gaussian Process Kalman Filter

MOGP Multi-output Gaussian Process

MAP	Maximum A Posteriori
MMSE	Minimum Mean Square Error
MDTT	Multiple Detection Target Tracking
PDA	Probabilistic Data Association
PDF	Probability Density Function
PSD	Power Spectral Density
RANSAC	RANdom SAmple Consensus
ROI	Region of Interest
RM	Random Matrices
RTS	Rauch-Tung-Strieble
RMSE	Root Mean Square Errors
STGP	Spatio-Temporal Gaussiann Process
SOS	Sum of Separable
SDE	Stochastic Differential Equation
SCNN	Spatial Convolutional NN
VP	Vanishing Point
YOLO	You-Only-Look-Once

Declaration of Academic Achievement

This research presents analytical and computational work carried out solely by Behzad Akbari, herein referred to as "the author", with advice and guidance provided by the academic supervisor Prof. T. Kirubarajan. The other co-authors helped with editing and reviewing the papers. Information that is presented from outside sources, which has been used towards analysis or discussion, has been cited when appropriate. All other materials are the sole work of the author.

Chapter 1

Introduction

1.1 Lane Detection and Tracking: A Brief Review

Unlike common object detection tasks that only require an approximate bounding region (rectangle), lane detection programs require precise prediction of curves. In the Lane Detection application, after feature extraction, we generally cluster and fit measurements to a model to estimate degraded parts and parts that are not visible. Many studies have focused on detecting lane-lines using parametric straight line, polynomial, or spline curve models. The first are parametric in the sense that the parameters of the system need to be defined in the program design stage. To estimate the position of the lane markings in the current time, some of the methods only use the last frame, while some use the stream of previous frames. In this thesis, we especially focus on algorithms that both use the spatial and temporal domain together and use the stream of frames.

1.1.1 Traditional Lane Detection Algorithms

Most traditional methods of lane-line estimation only use the current frame to predict the position of lane-lines. In fact, in these methods, we consider all information limited to the single last frame, methods that are sometimes called spatial methods. The procedure of a spatial method usually starts with feature extraction, clustering features and finally filling in the missing parts using a mathematical model ([51, 66, 3, 68, 78, 93, 19]).

For feature extraction, most of these approaches predominantly rely on information such as color, color cues, and edge-specific details. Color cues exploit the color contrast information between the lane markings and roads. However, the conditions have to be favorable for the differences, in contrast, to be realized by algorithms. Conditions such as illumination, backlights, shadows, night lights, and weather conditions, such as rain and snow, can significantly affect the performance of color cue-based algorithms. Although some preprocessing can improve detection ([66]), consistently differentiating lane boundaries from other artifacts, such as shadows and vehicles, is a challenge. Inverse Perspective Mapping (IPM) is one preprocessing approach that helps lane detection methods to distinguish lane boundaries from other artifacts in the science. The central idea behind IPM is to remove the perspective distortion of lines that are parallel in the real world ([15, 3, 10]). The main limitation is that the transformation is very sensitive to obstacles on the road, such as vehicles, and to terrain conditions ([63]).

Some works use line segments, the result of Hough transforms, or line segment detection (LSD) to decrease the dimension of measurements as an alternative to using

pixels as measurements. Because most of the road lane is designed to be parallel, corresponding extracted line segments should pass through a vanishing point in camera perspective. This property can be exploited to eliminate and filter out line segments that do not constitute lanes ([89, 50, 85]).

RANdom SAmple Consensus (RANSAC) or other parametric regression models have also been used for clustering features ([3] [64]). In Parametric methods the number of parameters stay fixed with respect to the size of the data, in non-parametric models, the number of parameters grows with the number of data points and can make the better estimation.

With recent improvements in the performance of CPUs and GPUs, fully supervised methods like Convolutional Neural Network (CNN) and deep learning have also been introduced for the detection and classification of lane markings in real time. In lane-line tracking, the lack of distinctive features tends to make the algorithms confused by other objects with similar local appearance. In some works, NN is used to remove clutter. For example, the CPN network can be used to detect road surfaces, and the Region of Interest (ROI) and You-Only-Look-Once (YOLO) network can detect and remove cars or other patterns on the road ([60]). Many approaches have been introduced to overcome the problem of the lack of distinctive features in lane markings. In [68], Spatial Convolutional NN (SCNN) is used to extract spatial correlation between rows and columns. These relationships help in cases of semantic objects that have dependent parts, like lane markings. Another method that can be used is a two-step learning approach. For example, [86] use a real-time network called "Lanenet" that is based on a two-stage Deep NN for lane detection. One network is used to detect lane marking edges, while the other is for semantic purposes like

grouping and clustering lane markings. Due to the variety of situations on roads, training a network for lane detection is a substantial, time-consuming process. Most of the time, a network that works well with one data set does not work as well with others.

The viability of most of these approaches is dependent on the preprocessing of images. As such, the robustness of these approaches varies depending on a number of conditions, and the quality of results is subject to the techniques applied to the preprocessing stage.

In this thesis, working within the spatial domain to detect the features belonging to each target, we introduced two Bayesian clustering methods to group features belonging to each lane line. In the first method, we used the line-segments result of the Hough Transform and apply the Maximum A Posterior estimator to use prior information, the curvature of lanes, to cluster line segments belonging to lane-lines. In two subsequent works, we take advantage of dependency and the addition of more semantic to measurements, introducing a kernel-based joint probability association couple filter, K-JPADACF, to group pixel features belonging to each lane-line.

1.1.2 Extended Target Models for Lane Tracking

The second category of lane detection methods uses the temporal domain in addition to spatial information. It would be possible to use previous information as a prior using Bayesian recursive techniques and then use a tracking algorithm like a Kalman filter to expand estimations to the time domain. A number of approaches have been proposed in the literature for tracking a single lane or multiple lanes, such as ([54, 4, 92, 94, 46, 85, 19]).

In most of these cases, the traditional parametric methods like line, polynomial, or spline fitting have been used as a measurement model combined with a dynamic motion model like constant velocity or random walk for scattering points (extends). In these cases, the parameters of the shape typically need to be defined at the program design stage (parametric); they are highly sensitive to prior information and not flexible for modeling dependency. Since the lanes are parallel and the road's shape is designed to be smooth enough for safe driving, extracted features of lane markings tend to be spatially and temporally correlated together between frames. Since the shape of lane-lines change over time, it is possible to track the change of the lane-lines using Extended Target (ET) models.

Extended Target Tracking (ETT) is the process of estimating both the evolution of the shape of the target and the kinematic, using a sequence of noisy measurements ([34]). Sometimes it is considered similar to Group Target Tracking (GTT), when states of closely grouped single-point targets moving with similar dynamics are estimated.

Extended Target Tracking is used in many modern applications, including robot navigation, tracking of people and cars, chemical and biological reaction estimation, epidemics, and pollution ([76]). ETT is also considered an integral part of many Advanced Driver Assistance Systems (ADAS) and autonomous or self-driving cars in urban traffic ([52]).

Extended Target Tracking is sometimes called Multiple Detection Target Tracking (MDTT). MDTT requires nonlinear estimation methods that can address clutter and data association ([6], [36]). Typical extended targets can model simple shapes like circles, ellipsoids, or rectangles ([48], [33]). The Random Matrices (RM) technique

was the first Bayesian method for tracking extended targets ([48]). The formulation of the RM technique assumes that the extended target is ellipsoidal. However, in some practical applications, this assumption doesn't hold. To describe and estimate the non-ellipsoidal shape of an extended target, several techniques, such as the multi-ellipsoid RM model ([53]) and the random hyper-surface model ([9]) have been proposed.

B-spline curves models have also drawn some attention in recent years in the field of ET tracking and have shown promising results in modeling the extent of ET objects ([43, 45, 24]).

A Gaussian process (GP) is another excellent non-parametric Bayesian modeling method that has been used recently to track extended targets ([82, 35, 90, 1]). In GP, the process noise is considered Gaussian, and the ET object shape is assumed to be star-convex; these can be represented with the help of a radial basis function by means of the GP approach.

In real-time applications, the kernel-based GP regression is not feasible because it is a batch process and needs the inversion of a covariance matrix with cubic complexity with respect to the number of inputs. In this case, a recursive GP regression model can be used ([75]). In [82], GP was used for the first time to model extended targets, by applying a recursive GP regression called GP-KF to estimate the shape and kinematics of extended targets in linear time. The evolution of the shape was considered to be Markovian with a forgetting factor for the temporal dynamic, similar to the Huber method ([41]).

In tracking extended targets when the shape of the object is changing over time, the spatial part of the object is correlated in both the space and the time domains. In

this case, considering the evolving shape only in the spacial domain will compromise performance. In [1], STGP was used for the first time to track extended targets, using a variant called STGP-KF. They used the idea proposed in [74], in which the power spectral density function of the STGP covariance function was factorized in order to find the transfer function. [1] used this idea to track rigid objects in radial space.

To exploit the correlation between outputs, we can model the interference of targets using the Multi-output Gaussian Process (MOGP) method ([14]). In MOGP, one single input (an index point) has a set of correlated outputs. [2] demonstrate that the overall covariance function could be parameterized as a combination of the output similarity's matrix and the input's spacial kernel. In fact, instead of viewing each output separately, this model corresponds to a scenario where only one Gaussian process exists, and each task simply shares this latent process with various weights plus additive white noise ([72]).

1.2 Theme and Objectives of Dissertation

The chapters in this dissertation are focused on new lane tracking algorithms using extended target tracking methods under challenging conditions. The general focus of the the thesis is as follows:

1. Developing a Bayesian clustering algorithm, based on a maximum a posteriori (MAP) estimator, to cluster line segments into different groups

2. Developing a new METT model based on the IPDA filter and augmented with an intensity feature for tracking Catmull splines

3. Introducing a new Kernel-based Joint Probabilistic Data Association Coupled

Filter (K-JPDACF) to cluster dependent features belonging to lane-lines in linear time.

4. Implementing a dependency model based on MOGP for training lane-lines and initializing the filter

5. Introducing a new recursive MO-STGP-KF for tracking multiple dependent extended targets

6. Introducing a ST-JPDA filter to track independent or dependent multiple extended targets that share measurements with each other

1.3 Publications

This thesis is the result of my five years of research, summarized in three publications:

1. Expanding Lane Detection Methods into the Time Domain Using an IPDA Filter with Intensity Feature

   Behzad Akbari, Jeyarajan Thiyagalingamy, and Thia Kirubarajan

   Submitted in MDPI sensors journal

2. Lane Tracking Using Dependent Extended Target Models and Multi-Output Spatio-temporal Gaussian Processes

   Behzad Akbari and Thia Kirubarajan

   To be submitted in IEEE Transaction journal

3. A Spatio-Temporal Joint Probabilistic Data Association Filter (ST-JPDAF) for Tracking Extended Targets

Behzad Akbari and Thia Kirubarajan

To be submitted in MDPI sensors journal

1.3.1 First Paper

In the first paper, we propose a tracking algorithm for multiple lane-lines across a stream of frames. Instead of resorting to the conventional method of processing each frame to detect lanes in the space domain alone, we treat the overall problem as a Multi Extended Target Tracking (METT) problem, across both the space and the time domains. We use the intensity of lane-lines as an augmented feature to correctly cluster multiple lane-lines. By representing these lane-lines as splines, we then identify a set of control points. These become states of our extended target, which need to be tracked over time across a stream of frames.

By representing an integrated probabilistic data association filter (IPDAF) as our basis, we formulate a METT algorithm to track multiple splines corresponding to each lane-line.

1.3.2 Second Paper

In the second paper, we propose a novel approach called the Multi-Output Spatio-Temporal Gaussian Process Kalman Filter (MO-STGP-KF) for estimating and tracking multiple correlated lane markings that are possibly degraded or covered with traffic. For feature extraction and clustering of lane markings, we propose a new Kernel-based Joint Probabilistic Data Association Coupled Filter (K-JPDACF) to cluster multiple lane-lines with linear computational complexity. For training and initialization of the filter, a model based on MOGP is proposed.

1.3.3 Third Paper

In the last paper, we develop a Spatio-Temporal Joint Probabilistic Data Association Filter (ST-JPDAF). We manage dependency of measurements in space (inside a frame) and time (between frames) using different kernel functions. The kernel functions are learned using the training data. This method can be used to track the shape and dynamic of non-parametric dependent extended targets with share measurements in clutter.

1.3.4 Relationship Between Three Algorithms

These three papers are different approaches for tracking multiple extended targets implemented in the lane marking detection application where:

1. Targets are independent with spline shape (parametric model), and the clutter exist(first paper).

2. Targets are continuous functions (non parametric model) , clutter exist and the shape or dynamic of targets can be dependent (second paper).

3. Targets are continuous functions (non parametric), shape or dynamic can be dependent, clutter exist and they can share measurements (third paper).

Chapter 2

Expanding Lane Detection Methods into Time Domain Using IPDA Filter with Intensity Feature

2.1 Introduction

Advanced Driving Assistance Systems (ADAS) are no longer an optional luxury component in modern vehicles ([12]). Instead, they are becoming a core component in modern vehicles, especially with the migration toward autonomous vehicles. ADAS covers a number of varying functionalities, such as lane departure warning (LDW), lane keep assist (LKA), lane change merge (LCM), adaptive cruise control (ACC), collision detection and avoidance (CD), and night vision and blind-spot detection, to mention a few ([44, 87, 28, 12, 78, 23, 11, 27, 73, 15, 21, 58]). The overall functionality of the ADAS is underpinned by a machine vision component, whose ability to understand the surroundings, particularly the ability to extract a group of related

lane markings (lane-line) and markings in roads, decides the overall performance of the ADAS. With ADAS becoming a core component, it is essential that potential errors arising out of the machine vision component be as low as possible. However, correctly and consistently extracting the lane markings, at all times and across a range of weather conditions, is not trivial. In addition to this, varying lane marking standards, obscure lane markings, splitting and merging of lanes, and the shadows of vehicles and objects exacerbate this problem even more ([46, 57, 91, 54]). We show several such examples in Figure 2.1.

Figure 2.1: Example cases where extracting lane markings is challenging (adopted from [3]).

Road markings can be extracted using image-based sensors like monocular or stereo vision cameras or using LIDAR sensors. Among these, the use of monocular cameras is a cost-effective approach, albeit carrying the disadvantage of lacking depth information. Stereo vision cameras can, however, make it possible to infer the depth information and hence reconstruct three-dimensional scenarios for improved functionality, such as collision detection ([87]). LIDAR sensors exploit the fact that road markings are painted using retro-reflective paints. These extracted markings can then be used to extract the lane markings. However, like stereo vision cameras, LIDAR sensors are far more expensive than monocular cameras. As such, seeking a trade-off between performance, reliability, and cost is essential. Treating cost-effectiveness as the primary objective, we assume that the lane detection is performed on images obtained from a monocular camera system.

The literature on lane detection and tracking is considerable, with a variety of techniques that cover various application domains, including LDW, LKA, LCM, and CD. Some of these perform lane marking detection (for example [22, 32]) and track them, while the rest perform only the detection (for example [58, 28, 66]). In particular, we focus on techniques that rely solely on images or videos obtained from monocular vision cameras for lane marking detection followed by tracking. For instance, vision-based lane detection has been used for LDW in [46, 10, 28, 15, 21]. These approaches predominantly rely on information such as color, color cues, and edge-specific details. Color cues exploit the color contrast information between the lane markings and roads. However, the conditions have to be favorable for the differences, in contrast, to be realized by algorithms. Conditions such as illumination, back lights, shadows, night lights, and weather conditions such as rain and snow

can affect the performance of color cue-based algorithms significantly. One approach to overcome these limitations is to use the Hough transform along with color cues ([51]). However, the Hough transform works well when the potential candidate lines are straight and visible enough. Although some preprocessing can improve detection ([66]), consistently differentiating lane boundaries from other artifacts, such as shadows and vehicles, is a challenge.

Inverse Perspective Mapping (IPM) is another approach to determine the lane boundaries in LDW systems. The central idea behind IPM is to remove the perspective distortion of lines that are parallel in the real world ([15, 3, 10]). To do this, images are transformed from a camera view to a bird's eye view using camera parameters. During the transformation, the aspect ratios are retained, so that gaps or widths between lane boundaries are transformed appropriately. As such, the lane boundaries are still detectable in the transformed space. However, there are several downsides to this approach. Primarily, IPM is often used with fixed camera calibration parameters, and this may not always be optimal owing to the surface conditions ([88]). Although these issues can reasonably be overcome by resorting to various techniques, such as calibration and adequate compute power systems ([88, 55, 71]), the main limitation is that the transformation is very sensitive to obstacles on the road, such as vehicles, and to terrain conditions ([63]).

Identification and clustering of lane boundaries is a fundamental requirement for any lane-line detection algorithm, and the proposed approach is no exception. Many papers have been published to date on clustering and grouping features in lane-line detection. Most of these were based on the Hough transform, RANdom SAmple Consensus (RANSAC), or other parametric regression models ([3], [64]). In real-world

situations, where clutter exists and lanes-lines are merging or splinting, clustering becomes more challenging.

If we define a road lane as a pair of parallel lane-lines, corresponding extracted line segments should pass through a vanishing point. This property can be exploited to eliminate and filter out the line segments that do not constitute lanes ([89, 50, 85]). A number of approaches have been proposed in the literature for tracking a single lane, such as ([54, 4, 92, 94, 46, 85, 19]). In [54] color, gradient, and line clustering information are used to improve the extraction of lane markings. In [4], multilevel image processing and a tracking framework are proposed for a monocular camera-based system. As such, it heavily relies on preprocessing of frames. Our approach also uses splines, but our tracking approach is significantly different from the one in [4]. In [92] and [94], techniques for personalized lane-change manoeuvring are discussed. They use driver-specific behaviors, collected as part of the system, to improve the results. Although this can improve the results, such approaches are difficult to implement practically. Unlike common object detection tasks that only require bounding boxes, lane detection requires precise prediction of curves, so after detection of lane markings, we need clustering and fitting to find lane-lines. In [19], lane tracking is simplified by forming a mid-line of a single lane using B-splines. Although this approach may be useful over a short distance, conditions such as diverging lanes or missing lane markings will render the approach susceptible to bad approximations of mid-lines. This can easily lead to suboptimal results.

The viability of most of these approaches is underpinned by preprocessing of images. As such, the robustness of the approaches varies depending on a number of conditions, and thus the quality of the results is subject to preprocessing techniques.

In [26], an approach for lane boundary detection based on Random Finite Sets and PHD filter is proposed as a multitarget tracking problem.

One viable approach to lane detection is to model the lanes as functions and apply curve-fitting on points detected by the edge detection algorithms ([93]). This approach, in comparison to the techniques outlined above, is less susceptible to varying conditions such as shadows. However, modeling complex lane features such as diverging or merging lanes as functions is a complex process.

Fully supervised methods like Convolutional Neural Network (CNN) and deep learning have also been used recently for detection and classification of lane markings. In lane-line tracking, the lack of distinctive features tends to make the algorithms confused by other objects with similar local appearance. In some works, NN is used to remove clutter. For example, a CPN network can be used to detect road surfaces and a Region of Interest (ROI) and YOLO network to detect and remove other patterns on the road or cars ([60]). Many approaches have been introduced to overcome the problem of the lack of distinctive features for lane markings. In [68], Spatial Convolutional NN (SCNN) is used to extract spatial correlation between rows and columns. These relationships help with semantic objects that have dependent parts, like lane markings. The other method that can be used is to have two steps for learning. In [86], a real-time network called "Lanenet" that is based on a two-stage Deep NN for lane detection is employed. One network detects lane marking edges,while the other is for semantic purposes, like grouping and clustering lane markings. Due to the variety of situations on the roads, training a network for lane detection is a substantial, time-consuming process. Most of the time, the network that works well with one data set does not work with others.

In this paper, we propose a novel method for lane detection in which the overall problem is treated as a METT problem. Our method also can be augmented to other lane detection methods to add the time domain and predict the position of the lane when there are not enough observations in a single frame. To this end, we first model each lane-line as a spline, defined by a finite set of control points or stats. By treating these splines (and thus the control points) as an extended target whose motions are defined during frame transitions, we develop a multitarget tracking problem with an appropriate motion model. Although we rely on some preprocessing algorithms to identify and extract lane markings and line segments, the overarching approach for continuous detection of lanes across a number of frames is significantly different from those found in the literature.

We utilize the probabilistic Hough transformation ([47]) to perform an initial extraction of lane markings. This is then followed by a series of algorithms before treating the lane-lines as extended targets. The first algorithm in the pipeline performs an initial clustering and grouping of line segments, then outputs the Hough transform into different clusters. We then devise a multitarget tracking algorithm based on a motion model that assumes that the transitions of splines across frames take place at a constant rate. In doing this, we make the following key contributions:

1. we develop a Bayesian clustering algorithm, based on a maximum a posteriori (MAP) estimator ([6]), to group different line segments into different groups; and

2. we develop a METT model based on the IPDA filter augmented with an intensity feature.

The remainder of this paper is organized as follows: In Section 3.2 we provide

a background into a number of aspects that this work builds upon, including the Catmull-Rom Spline, MAP estimator, and IPDA filter. In Section 2.3, we formulate the overall problem and discuss our approach for solving each of the subproblems. This is followed by Section 2.4, where we discuss a set of preprocessing steps on the input images prior to using our framework of methods. The process of selecting and estimating control points to describe the splines, and two of our key algorithms for this purpose are discussed in Section 2.5. We then describe the techniques employed to track the splines using the IPDA filter in Section 2.6. The results of our evaluations are then presented in Section 4.4 and we discuss conclusions in Section 4.5.

2.2 Background

2.2.1 Catmull-Rom Spline

A curve C in space $\mathbf{R}^n$ can be piece-wise defined by a set of polynomials, which are often referred to as splines ([84]). The splines are defined using a set of fixed points (known as control points), $P = \{p_0, \ldots, p_m\}$, which also lie in the same space as C. The specialty of the Catmull-Rom spline ([20, 49, 17]) is that the control points lie on the spline itself and for each polynomial $m = 4$. In other words, the Catmull-Rom spline requires four control points in each section to describe any curve. Given that we extract only the lane markings (and not the surroundings), when mapping lanes to splines, it is essential to have the control points on the curves themselves. As such, the Catmull-Rom spline is an ideal choice here.

The Catmull-Rom spline is part of the cubic interpolating spline family ([84]), where the gradient or tangent at each point of the spline is calculated using the

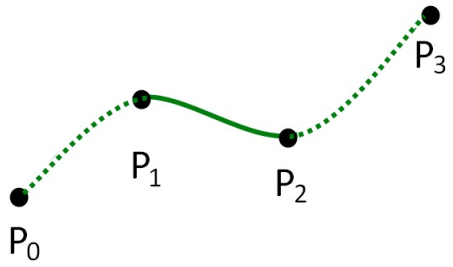


Figure 2.2: Catmull spline between two control points p_1 and p_2.

previous and next points on the spline. Let p_k be the k-th point on the spline. The Catmull-Rom splines have uniform parameter spacing between two consecutive points, and this spacing is assumed to be t. The geometry matrix for a single Catmull-Rom segment s, such as one shown in Figure 2.2, can be expressed as

$$p(s) = TXP, \tag{2.2.1}$$

where

$$T = \begin{bmatrix} 1, t, t^2, t^3 \end{bmatrix}$$

$$X = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\lambda & 0 & \lambda & 0 \\ 2\lambda & (\lambda - 3) & (3 - 2\lambda) & -\lambda \\ -\lambda & (2 - \lambda) & (\lambda - 2) & \lambda \end{bmatrix}$$

and

$$P = \begin{bmatrix} p_{k-2} \\ p_{k-1} \\ p_k \\ p_{k+1}. \end{bmatrix}$$

The parameter λ denotes the tension of the spline, determining the sharpness of the bends of the spline at the interpolated control points, with $\lambda = 1$ giving full freedom for very sharp bends to $\lambda = 0$, which offers no bends at all. In most cases, it is not unusual to set $\lambda = 0.5$.

2.2.2 Maximum a Posteriori (MAP) Estimator

Maximum a Posteriori (MAP) estimation ([6]) is a form of a approximate posterior inference model, where the aim is to estimate some parameter θ in the setting of a posteriori distribution $p(\theta|z)$, and a prior probability distribution $p(\theta)$, where z is the observation or measurement.

More specifically, the problem is such that there is a need to derive a single estimate of an unknown parameter θ that maximizes the posteriori probability distribution, i.e.,

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(\theta|z). \tag{2.2.2}$$

With Bayes' rule, this can be expressed as

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} p(z|\theta)p(\theta), \tag{2.2.3}$$

where $\hat{\theta}_{MAP}$ is the estimated value of the parameter θ. The likelihood distribution $p(z|\theta)$ is often derived from the observed data. It is possible to repeatedly apply the MAP estimator to estimate more than one parameter. In our case, we use the MAP estimator to group all the line segments belonging to a single lane-line, and to estimate the best control points that would best fit the spline. This will be discussed in detail in the next section.

2.2.3 IPDA Filter

The principle behind the IPDA filter ([62]) is to integrate the track initiation and data association operations into a single algorithm. This is achieved by treating track existence as a probabilistic event. In a situation where the number of targets, the number of measurements, and the locations of the targets are unknown, as discussed in Section 3.2, tracks are initialized during each time step, relying on the received set of measurements. Such a process inevitably initiates a number of false tracks along with the true tracks. Although these false tracks may be terminated over a number of time steps, differentiating between true and false tracks requires a track quality measure. This is achieved by modeling the existence of the target being followed by track τ using a discrete random variable $\epsilon_k^\tau(j)$. With this, the event $\epsilon_k^\tau(j) = 1$ denotes the existence of the target τ (the event that track τ is true), while the event $\epsilon_k^\tau(j) = 0$ denotes the absence of the target τ (the event that track τ is false), at time step k using the measurement j.

Let m_k and Z_k be the number of detections or measurements and the total sets of measurements at time step k, respectively. Let $Z^k = Z_k Z^{k-1}$ be the set of sets of measurements up to and including time step k. Each track τ at time k is described

by three components: state estimate, an error covariance of the state estimate, and a probability that the track is indeed the result of the target trajectory estimation. Furthermore, let P_D and P_G be the probabilities of detection, denoting the presence of a measurement whenever the target exists and probability that a measurement falling within the gating region, respectively. With these, the conditional probabilities that the measurement j at time step k originated from a target being tracked by track τ, provided that the track exists, are expressed as

$$\beta_{k,0}^{j} = \frac{(1 - P_D P_G)}{C_{j=0}} \tag{2.2.4}$$

$$\beta_{k,\mathbf{z}^k}^{j} = \frac{L_j e_j}{C_{j=1,\ldots,m_k}}, \tag{2.2.5}$$

where

$$C = (1 - P_D P_G) + \sum_{j=1}^{m_k} L_j e_j$$

and updated value equal to the conditional mean of

$$\begin{aligned} \hat{x}(k|k) &= \mathbf{E}(x(k)|z^k) \\ &= \sum_{j=0}^{m_k} \hat{x_j}(k|k)\beta_j(k). \end{aligned} \tag{2.2.6}$$

2.3 Problem Formulation and Our Approach

2.3.1 Problem Formulation

Given the background and context from Sections 4.1 and 3.2, the overall problem of lane identification across a sequence of frames can be reformulated as follows:

1. Finding pseudo measurements (control points): For a set of lane markings on frame k, cluster line segments and identify a set of control points that would uniquely describe each of the lane markings; and

2. METT: By considering control points in each of the lane-lines as pseudo measurements, the METT algorithm helps with the extraction and identification of lane markings on the frames yet to be seen.

To facilitate the process of deriving an overall approach and suitable algorithms, we use i as the index for the control points $i \in 0, 1, \ldots, N$, j as the lane index, and k as the frame index. For instance, the parameter $x_{i,j,k}$ denotes the i-th control point for the j-th lane on the k-th frame. The notations used in this manuscript are given in Table 3.1.

2.3.2 Our Approach

To address the overall problem outlined in Section 2.3.1, we decompose the problem into a number of sub-problems, each of which handles a specific aspect of the overall lane detection problem across frames. The overall agenda is to form an automated processing pipeline, where each stage of the pipeline is underpinned by one or more algorithms. This processing pipeline is shown in Figure 2.3.

Table 2.1: Symbols adopted in this manuscript.

Symbol	Description
$x_{i,k,j}$	Single control point in the line segment j
$\psi_s(j)$	Line segment j in the section s
$Z_{k,j}$	Pseudo measurements or control points for a spline
R	Covariance of the measurement noise
Q	Covariance of the process noise
τ	Track index
F	State transition matrix
H	Measurement matrix
P_G	Gating probability
$e_{k,j}$	Event on the jth spline at the kth frame
$\Psi_{k,s}$	Validated set of measurements for section s

Each of these stages is discussed in the following sections.

2.4 Preprocessing

The key aspects of the preprocessing stage include edge detection, probabilistic Hough transform, and extraction of ROI. We also use noise filtering before each of these stages to minimize the impact of noise amplification in the process.

2.4.1 Edge Detection

Essentially, edge detection in images is based on the convolution of a predetermined kernel with an image ([31]). In our case, each frame forms an image. However, this basic approach for edge detection, which is a gradient finding exercise, picks up the gradients of the noise along with the lane markings. Although basic noise filtering, such as averaging or median filtering, can minimize these effects, they do not guard the edge detection against these artifacts. For this reason, we used the Canny edge

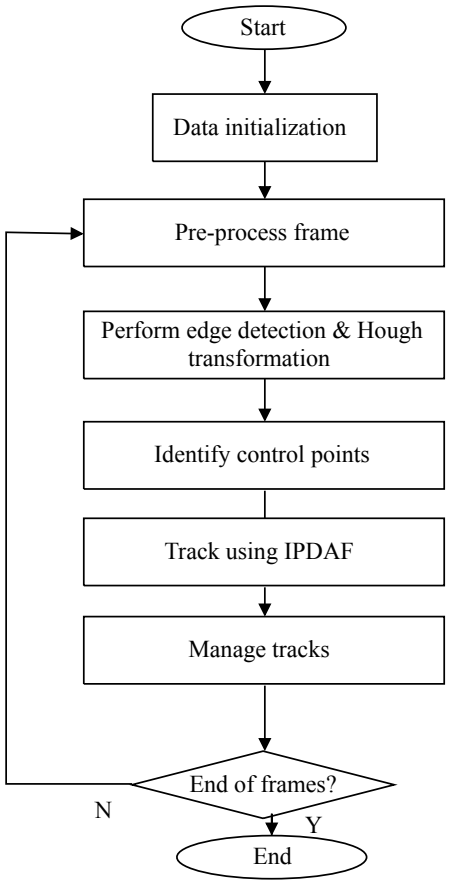


Figure 2.3: The automated processing pipeline of our approach.

detection ([31]), which incorporates Gaussian filtering as a precursor to the gradient calculation step. More specifically, we used two 3×3 kernels, namely, a Gaussian kernel H and an edge detection kernel K. For each input frame $\mathcal{F}_{in}$, we calculated the output frame $\mathcal{F}_{out}$, as

$$\mathcal{F}_{\text{out}} = K * (H * \mathcal{F}'_{\text{in}}), \tag{2.4.1}$$

where $\mathcal{F}'_{in}$ denotes the noise filtered version of $\mathcal{F}_{in}$, and the $*$ operator denotes the convolution operation. We have also prefixed the values of H (by fixing the variance).

2.4.2 Probabilistic Hough Transform

Considering each curve as a limited number of line segments, let us decrease the dimension of inputs by using line segment measurements instead of pixels. Many methods have been used to detect line segments in an image; one of these, the Hough transform, is a fast voting algorithm for this purpose. Given a straight line $y = mx+c$, it has an equivalent polar representation $\rho = x\cos(\theta) + y\sin(\theta)$, where ρ and θ represent the normal distance of the line from the origin, and the angle that normal subtends with the x axis, respectively. With this, a point (x_i, y_i) in Cartesian space (for $i = 1 \dots M$) becomes a sinusoid in the polar (θ, ρ) space. In the (θ, ρ) space, sinusoids that intersect at any single point (θ_i, ρ_i) represent the same straight line in the Cartesian space defined by the properties (θ_i, ρ_i). That is, two points in the image plane that belong to the same line can determine a point (ρ_i, θ_i) in the polar space, given by the intersection of two sinusoids.

We illustrate this in Figure 2.4. The number of sinusoids that pass through a point in the transformed space represents the weight of the line. Easier implementation of this is due to [25], where the (θ, ρ) space is discretized into a set of grid points and represented as an accumulator matrix, whose entries represent the number of intersections at that point. The entries are updated as each point is mapped to the transformed space and when intersections are found. This matrix is then exhaustively searched for a maximum (and other decreasing maxima) to find straight lines. As such, the original Hough transformation process is very computationally intensive.

In our case, we adopted the probabilistic version of the Hough transform ([29]), where only a subset of the edge points $m < M$ are selected through the random sampling process, particularly when updating the accumulator matrix. With this

approach, we reduce the amount of computation without considering all possible measurements.

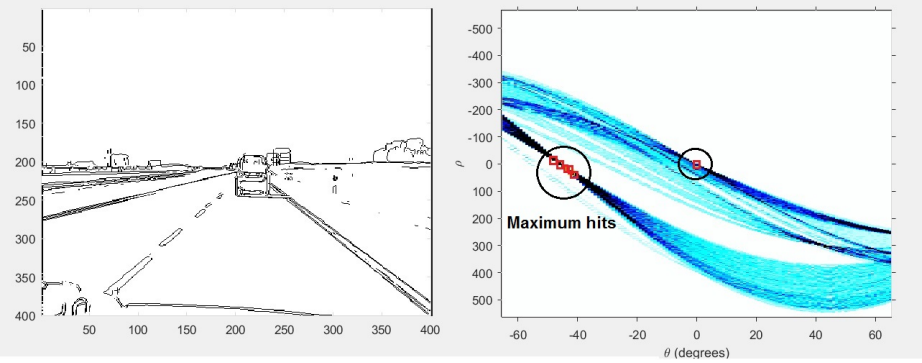


Figure 2.4: Transformation of a preprocessed image to the Hough space.

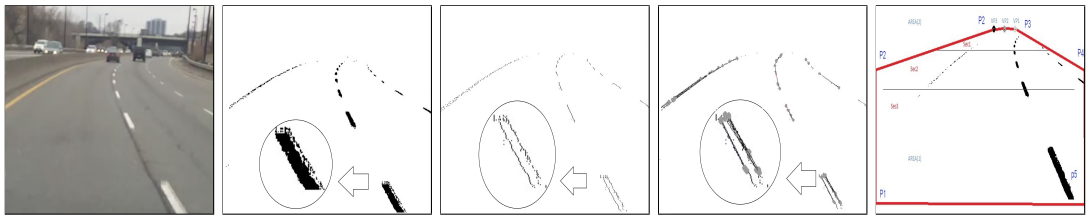
2.4.3 Extraction of Regions of Interest

Although the probabilistic Hough transform can filter out unnecessary edges and lead to straight lines, the extracted straight lines do not have to represent only the lanes. In fact, the extracted straight lines can be anything, including lanes, edges of the vehicles, lampposts, and buildings. An easier approach to filter out irrelevant components is to use the vanishing points. Each pair of lines ia a lane should have a vanishing point.

Vanishing points can be extracted by embedding a new process after the probabilistic Hough transform process outlined above. Sinusoids that pass through all of the maxima points in Hough space should correspond to the vanishing point in the image plane. In particular, we extract vanishing points for each partition of the image. We then use these vanishing points to eliminate irrelevant straight line segments in the image and to form regions of interest. In addition to this, the area outside the

vanishing line has no information that can aid lane boundary tracking and can be removed.

In Figure 2.5, we show the outputs of different stages of the preprocessing.



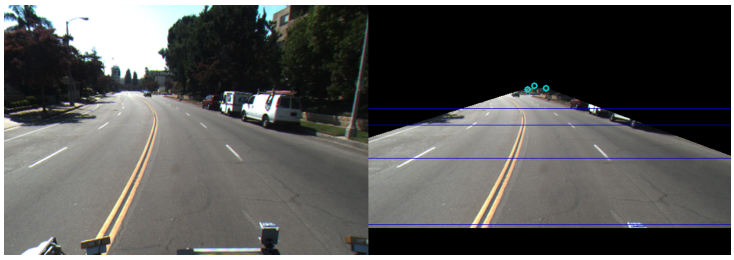
(a) (b) (c) (d) (e)

Figure 2.5: An example of preprocessing of image frames. (a) Raw image of the frame; (b) After edge and color segmentation; (c) Noise filtering and thinning; (d) Probabilistic Hough transform result; (e) Extracting VPs and regions of interest.

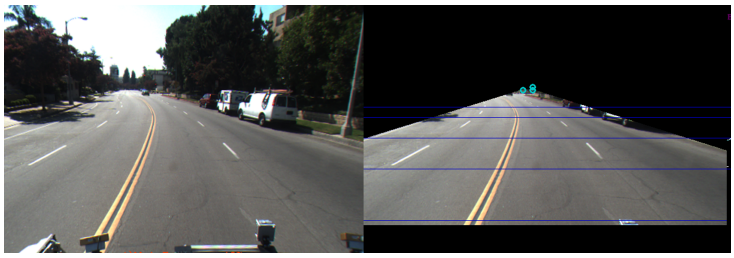
2.5 Identification of Pseudo Measurements or Control Points

2.5.1 Frame Partitioning and Measurement Likelihood

Once the preprocessing is over, the next stage of the pipeline extracts the pseudo measurements. Although we intend to identify a set of control points to model the lane-lines, the process is much simpler if the splines are small in size and straight in shape. However, the extracted lane markings are seldom straight. One approach to address this issue is to partition each frame into n horizontal tiles, each with an experimentally determined height, so the lanes on each partition are near straight. Figure 2.6 shows the same image partitioned in two different ways: for two different values of n (namely, $n = 3$ and $n = 4$), and different partition heights.



(a) Partitioning for $n = 3$



(b) Partitioning for $n = 4$

Figure 2.6: Two different examples of partitioning outputs (right) for the same input image (left). (a) $n = 3$ and $h_1 = \frac{1}{7}H$, $h_2 = \frac{2}{7}H$ and $h_3 = \frac{4}{7}H$; (b) $n = 4$ and $h_1 = \frac{1}{11}H$, $h_2 = \frac{2}{11}H$, $h_3 = \frac{3}{11}H$, and $h_4 = \frac{5}{11}H$.

However, considering the perspective characteristics of the camera and the distance of lanes from the camera, it is beneficial to have the heights of the partitions in increasing order toward the bottom of the frame. We experimentally determined that the extracted information is maximized for $n = 3$, such that $h_1 = \frac{1}{7}H$, $h_2 = \frac{2}{7}H$, and $h_3 = \frac{4}{7}H$, where H is the overall height of the region of interest (ROI). We use this configuration with the values of n and h_i $(i = 1, 2, 3)$ throughout the study conducted in this paper.

2.5.2 Intensity and Likelihood Ratio of a Line Segment

For each of the partitions, we extract the line segments. However, the extraction process, akin to the edge in most detection techniques, produces a number of broken,

small, non-continuous, and irrelevant line segments. As such, one of the key challenges following the extraction process is to distinguish the lane markings from background noise and clutter. To render a more robust and high-fidelity approach toward clutter and noise management, we augment the extractions with underlying intensity values. More specifically, we define the number of edge points that lie in an extended line segment s ($s = 1, \ldots, n$) as the intensity. The intensity can be extended to pseudo-measurements belonging to a curve. The intensity of an expanded line segment is represented as a likelihood ratio, which is defined below.

Let $p_0(f_j)$ be the probability density function (PDF) of the noise only, and $p_1(f_j)$ be the target-originated line segment detections before thresholding. Furthermore, let D_0 and D_1 be the scale parameters for false alarms and clutter, and target, respectively. These scale parameters are dependent on the minimum number of points used in the Hough transform. The noise only and target-originated measurement density functions are

$$p_0(f_j) = \frac{f_j}{D_0^2} e^{\left(\frac{-f_j^2}{2D_0^2}\right)} \tag{2.5.1}$$

$$p_1(f_j) = \frac{f_j}{D_1^2} e^{\left(\frac{-f_j^2}{2D_1^2}\right)}, \tag{2.5.2}$$

where $f_j \geq 0$ is the intensity of the candidate measurements j. Furthermore, let $\gamma = \gamma_{det}$ be the threshold to declare a detection. The probabilities of detection (P_D) and false alarm (P_{FA}) can be computed as follows:

$$\begin{aligned} P_D &= \int_{\gamma}^{\infty} p_1(f_j)df_j \\ &= e^{\frac{-\gamma^2}{2D_1^2}} \end{aligned} \tag{2.5.3}$$

$$\begin{aligned} P_{FA} &= \int_{\gamma}^{\infty} p_0(f_j)df_j \\ &= e^{\frac{-\gamma^2}{2D_0^2}} \end{aligned} \tag{2.5.4}$$

Although the probability of detection, P_D, can be increased by lowering γ, it will increase P_{FA}. Hence the choice of γ cannot be arbitrary. With these, the corresponding probability density functions after thresholding become

$$\begin{aligned} p_0^{\gamma}(f_j) &= \frac{1}{P_{FA}} p_0(f_j) \\ &= \frac{f_j}{P_{FA}D_0^2} e^{(\frac{-f_j^2}{2D_0^2})} \end{aligned} \tag{2.5.5}$$

$$\begin{aligned} p_1^{\gamma}(f_j) &= \frac{1}{P_D} p_1(f_j) \\ &= \frac{f_j}{P_D D_1^2} e^{(\frac{-f_j^2}{2D_1^2})}. \end{aligned} \tag{2.5.6}$$

where $p_0^{\gamma}(f_j)$ and $p_1^{\gamma}(f_j)$ are the probability density functions of the validated measurement ψ_j (for $j = 1, \ldots, m$) that are due to noise only and originate from the target, respectively.

Considering (2.5.5)–(2.5.6), the line segment intensity likelihood ratio e_j, which is the likelihood ratio of measurement (line segments) ψ_j with intensity of f_j edge pixels originating from the target rather than clutter, can be defined as

$$\begin{aligned} e_j(k) &= \frac{p_1^{\gamma}(f_j)}{p_0^{\gamma}(f_j)} \\ &= \frac{P_{FA} D_0^2}{P_D D_1^2} e^{f_j^2 \left(\frac{D_1^2 - D_0^2}{2 D_0^2 D_1^2}\right)}. \end{aligned} \tag{2.5.7}$$

2.5.3 Bayesian Clustering and Pseudo Measurements

Let $Z_{k,j}^{\tau}$ be the control points for track τ and lane-line j in frame k. More specifically,

$$Z_{k,j}^{\tau} = [x_{1,j,k}^{\tau}, x_{2,j,k}^{\tau}, x_{3,j,k}^{\tau}, x_{4,j,k}^{\tau}]. \tag{2.5.8}$$

Furthermore, let $\psi_s^k(j)$ denote the line segment in section s, at time step k, for lane j. Each such measurement has a position and the intensity as a likelihood ratio, $e_j(k)$.

Although we expect the pseudo-measurements to almost model the lane-line, in reality, a number of factors make this a challenging process. Some examples include missed detection, the non-deterministic nature of the preprocessing, and noisy measurements due to clutter. Therefore, it is essential to model these imperfections as part of the process.

To simplify analysis and derivation, we assume that measurements that originate from targets at a particular sampling instant are received by the sensor only once, with the probability of detection P_D. The measurement equation can be written as

follows:

$$\psi(j) = x + w(j), \tag{2.5.9}$$

where $j = 1, \ldots, m$ shows the number of validated measurements, $w(j)$ is the measurement noise, and $x = [x_1, x_2]^\top$ is the latent value that we are aiming to estimate in the presence of measurement noise.

We also assume that measurement noise is independent and zero-mean Gaussian distributed, with covariance R. In our case, various preprocessing stages, such as thinning and the Hough transform, contribute toward R. Thus, $w(j) \sim \mathcal{N}(0, R)$, where

$$R = \begin{bmatrix} \sigma_{11}^2 & 0 \\ 0 & \sigma_{12}^2 \end{bmatrix}. \tag{2.5.10}$$

Because of the condition of the road and perspective effect of the camera lens for values of σ_{11}^2 and σ_{12}^2, we would expect more deviation in the bottom part that is closer to the camera than the top, and we also assume measurements ψ to be normally distributed around x with covariance R while prior probabilities $p(x)$ are normally distributed around the predicted measurement $\bar{x}$ with a covariance Q. Thus $\psi \sim \mathcal{N}(x, R)$ and $p(x) \sim \mathcal{N}(\bar{x}, Q)$, where

$$Q = \begin{bmatrix} \sigma_{01}^2 & 0 \\ 0 & \sigma_{02}^2 \end{bmatrix}. \tag{2.5.11}$$

Again, similar to R, the perspective effect of the camera influences the values of σ_{01}^2 and σ_{02}^2, skewing them toward the bottom part of the frame. Furthermore, the

covariance Q is often linked to the curvature κ of the road. Assuming the maximum standard curvature of highways as a constant parameter, the posterior measurement density would be

$$p(x|\Psi) \triangleq \frac{1}{c}(p(\Psi|x)p(x)), \tag{2.5.12}$$

where Ψ is a set for validated measurements. Since the measurement and prior noises are assumed to be Gaussian, for a single measurement, (e.g., $m = 1$), $\psi(1) = \psi_1$ can be expressed as:

$$\begin{aligned} p(x|\psi_1) &\triangleq \frac{1}{c}(p(\psi 1|x)p(x)) \\ &= \frac{1}{c}\mathcal{N}(\psi 1; x, R)\mathcal{N}(x; \bar{x}, Q) \\ &= \frac{1}{c}\mathcal{N}(x; \xi(\psi_1), \mathcal{R}), \end{aligned} \tag{2.5.13}$$

where

$$\xi(\psi_1) = \frac{Q}{R+Q}\bar{x} + \frac{R}{R+Q}\psi_1$$

and

$$\mathcal{R} = \frac{RQ}{R+Q}.$$

For a Gaussian distribution, the mean is the optimal maximization value $\hat{x}$. Hence,

$$\hat{x} = \bar{x} + \frac{R}{R+Q}(\psi_1 - \bar{x}). \tag{2.5.14}$$

For $m > 1$, the optimal maximized value can be derived using the total probability and combined residuals, as follows:

$$\hat{x} = \bar{x} + \frac{R}{R+Q} \sum_{j=1}^{m} \beta_j (\psi(j) - \bar{x}), \tag{2.5.15}$$

where β_j is association probability, which we define as

$$\beta_j = \begin{cases} \frac{\mathcal{L}_j e_j}{1 - P_D P_G + \sum_{j=1}^{m_k} \mathcal{L}_j e_j}, \forall j = 1, \ldots, m_k \\ \frac{1 - P_D P_G}{1 - P_D P_G + \sum_{j=1}^{m_k} \mathcal{L}_j e_j}, j = 0, \end{cases} \tag{2.5.16}$$

where P_D and P_G are probabilities of detection and gating, respectively, m_k is the number of validated detections at time k, e_j is the intensity of the extended line segments as a likelihood ratio, and $\mathcal{L}_j$ is the probability density function for the correct measurement without the intensity feature (see Appendix A for derivations). The intensity feature is defined as

$$\mathcal{L}_j = \frac{1}{P_G} \mathcal{N}(\psi_j, \bar{x}, S),$$

where $S = R + Q$ and $\bar{x}$ is the prior information.

Ideally, each partition will have a sufficient number of full measurements $\psi^k(j)$ so that a spline can be fitted over those measurements. However, in reality, this is seldom the case. The associated challenges are dealt with using an algorithm that estimates the control points based on the available set of measurements. In particular, we use this Bayesian approach to find the optimal control points. These aspects are handled by two algorithms, Algorithm 1 and Algorithm 2, which are outlined and discussed

in detailed below.

Algorithm 1 Control Points Estimator

▷ Input: $\kappa, N, \mathbf{\Psi}$
▷ Output: A
▷ Section variables : P_i, R_i, S_i, Ψ_i
▷ κ : Vector of curvature
▷ Ψ_i : Sets of all extracted lines in partition i
▷ N : Number of partitions
▷ P_i : Prior noise covariance
▷ R_i : Measurement noise covariance
▷ S_i : Set of partitions indexes eliminating i
for i=1; i<N; i++ **do**
    for $l \in \Psi_i$ **do**
        for $S_i \in \{1..N\} - \{i\}$ **do**
            Initialize(P_{S_i}, R_{S_i})
            $\bar{x}_{S_i,l} \leftarrow$Predict($\kappa, l$)
            $\hat{x}_{S_i,l} \leftarrow$Update($\bar{x}_l, \psi_{S_i}, R_{S_i}, P_{S_i}$)
        end for
        $A \leftarrow \begin{bmatrix} A & \hat{x}_l \end{bmatrix}$
    end for
    A=RemoveSimilarCurves(A)
end for

Algorithm 1 handles each partition separately, but by extending the line segments into the next partition wherever needed. For a given partition s, it estimates the control points for each line, $\mathbf{x}_{i,s}$, using the curvature κ. Then the overall set of lines L is used to estimate the control points for that partition using the MAP estimator (see Algorithm 2). These control points are accumulated into A as a list.

Algorithm 2 combines both the data association and the posteriori PDF to adjust the estimated control points. In particular, it uses the IPDA-specific target-to-track association probabilities (covering both the track existence and non-existence) β_0 and β_j for finding the control points based on candidate control points $\bar{x}$ and measurements Ψ. More specifically,

Algorithm 2 Bayesian Update

▷ Inputs: $\bar{x}, \psi, R, P$
▷ Output: $\hat{x}$
▷ Section variables : P, R
▷ $\bar{x}$: Priors
▷ ψ : Measurements
▷ P : Prior noise covariance
▷ R : Measurement noise covariance
$\psi_{validated} \leftarrow$ `gate(`$\bar{x}, \psi, R, P$`)`
$m \leftarrow |\psi_{validated}|$
for `j=0; j<m; j++` **do**
    $r_j \leftarrow \psi(j) - \bar{\mathbf{x}}$
    $\beta_0 \leftarrow \frac{(1-P_D P_G)}{C}$
    $\beta_j \leftarrow \frac{\mathcal{L}_j e_j}{C}$
end for
$\hat{x} = \bar{x} + \frac{R}{R+Q} \sum_{j=1}^{m} \beta_j r_j$

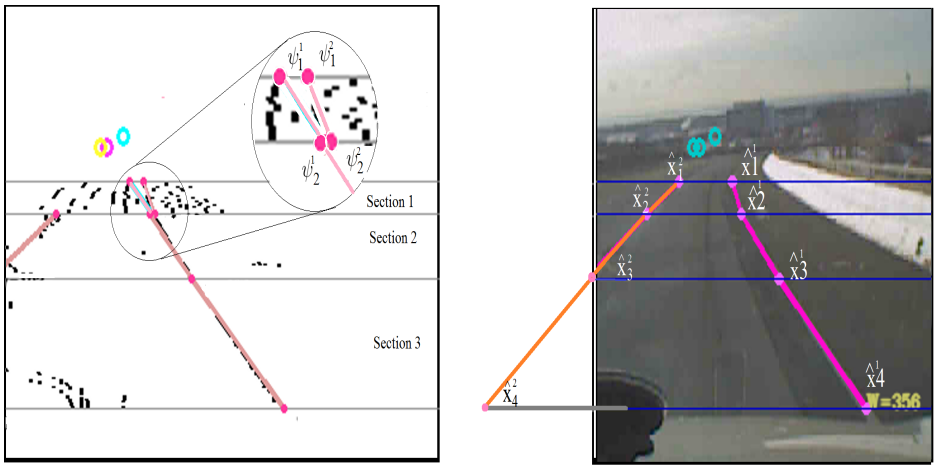
$$\begin{aligned} \hat{x} &= \underset{x}{\text{argmax}}\ p(x|\mathbf{\Psi}) \\ &= \underset{x}{\text{argmax}}[p(\mathbf{\Psi}|x)p(x)] \end{aligned} \tag{2.5.17}$$

We show a sample outcome of these algorithms in Figure 2.7. We first show two end-point measurements (ψ_1^1, ψ_2^1) and (ψ_1^2, ψ_2^2) (Figure 2.7a). These points are then corrected using the above algorithms to output corrected control points $\hat{x_1^1}$ and $\hat{x_2^1}$ (Figure 2.7b).

2.6 Multilane Tracking Using IPDA

2.6.1 Preliminaries

As stated above, we assume that control points are moving at constant velocity, and thus our dynamic model is a constant velocity model. With this, our state vector for



(a) (b)

Figure 2.7: An example of pseudo-measurement detection and clustering line segments. (a) Control point estimation based on measurements; (b) Pseudo-measurements or the result of clustering.

tracking τ in frame k becomes

$$x_k^\tau = \begin{bmatrix} x_{1,k}^\tau \\ x_{2,k}^\tau \\ x_{3,k}^\tau \\ x_{4,k}^\tau \\ \dot{x}_{1,k}^\tau \\ \dot{x}_{2,k}^\tau \\ \dot{x}_{3,k}^\tau \\ \dot{x}_{4,k}^\tau \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k^\tau \\ \dot{\mathbf{x}}_k^\tau \end{bmatrix} \tag{2.6.1}$$

and the intensity feature f augmented (pseudo) measurement vector $Z_{k,j}^{\tau,f}$ is

$$\begin{aligned} Z_k^{\tau,f} &= \begin{bmatrix} x_{1,k}^{\tau} \\ x_{2,k}^{\tau} \\ x_{3,k}^{\tau} \\ x_{4,k}^{\tau} \\ f_k^{\tau} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{x}_k^{\tau} \\ \mathbf{f}_k^{\tau} \end{bmatrix} \end{aligned} \tag{2.6.2}$$

With these, the state evolution and measurement updates for frame (time) index k become

$$x_k^{\tau} = F x_{k-1}^{\tau} + G\nu_{k-1}^{\tau} + \mathcal{G}u_{k-1}^{\tau} \tag{2.6.3}$$

and

$$Z_k^{\tau,f} = H x_k^{\tau} + \omega_k^{\tau}, \tag{2.6.4}$$

where F, G, and H are state transition, control input, and observation matrices, respectively, and ν and ω are measurement and process noises, respectively. In our

case, we preset F and G to

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6.5}$$

$$G = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 & 0 & 0 \\ 0 & \frac{\Delta t^2}{2} & 0 & 0 \\ 0 & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 0 & 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 & 0 & 0 \\ 0 & \Delta t & 0 & 0 \\ 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & \Delta t \end{bmatrix} \tag{2.6.6}$$

To minimize the variability of the outcomes of the lane extraction process, or its dependencies on the highly variable surrounding contexts, the selection process of the control points has to be adaptive enough to account for context information. For instance, when comparing the lane extraction process on roads with varying terrain conditions. as opposed to highways, it is desirable for the process noise to be more amenable to high variance on the spatial distribution of control points.

One approach to bringing adaptiveness into this process is to introduce adaptive process and measurement noises. Furthermore, due to partitioning of frames and the perspective effects of the camera, control points on the bottom of most partitions are closer to the camera than the control points on the top-most partition of a frame. For this reason, despite the constant velocity model, the rates of the spatial variations of control points are different. Considering different noise variance associated with the acceleration of control points will address this issue. As such, the diagonal form of the process noise ν_k can be expressed as

$$q = \begin{bmatrix} \ddot{x}^{\tau}_{1,k} & 0 & 0 & 0 \\ 0 & \ddot{x}^{\tau}_{2,k} & 0 & 0 \\ 0 & 0 & \ddot{x}^{\tau}_{3,k} & 0 \\ 0 & 0 & 0 & \ddot{x}^{\tau}_{4,k} \end{bmatrix}, \tag{2.6.7}$$

where $\ddot{x}^{\tau}_{i,k}$ for $(i = 1, 2, 3, 4)$ is the maximum acceleration of each control point. The process noise covariance Q can be expressed as

$$Q = GqG^T, \tag{2.6.8}$$

where

$$\text{diag}_0(Q) = \begin{bmatrix} \ddot{x}_1 \frac{\Delta t^4}{4} \\ \ddot{x}_2 \frac{\Delta t^4}{4} \\ \ddot{x}_3 \frac{\Delta t^4}{4} \\ \ddot{x}_4 \frac{\Delta t^4}{4} \\ \ddot{x}_1 \Delta t^2 \\ \ddot{x}_2 \Delta t^2 \\ \ddot{x}_3 \Delta t^2 \\ \ddot{x}_4 \Delta t^2 \end{bmatrix}^T \tag{2.6.9}$$

$$\text{diag}_{-4}(Q) = \begin{bmatrix} \ddot{x}_1 \frac{\Delta t^3}{2} \\ \ddot{x}_2 \frac{\Delta t^3}{2} \\ \ddot{x}_3 \frac{\Delta t^3}{2} \\ \ddot{x}_4 \frac{\Delta t^3}{2} \end{bmatrix}^T \tag{2.6.10}$$

and

$$\text{diag}_{+4}(Q) = \begin{bmatrix} \ddot{x}_1 \frac{\Delta t^3}{2} \\ \ddot{x}_2 \frac{\Delta t^3}{2} \\ \ddot{x}_3 \frac{\Delta t^3}{2} \\ \ddot{x}_4 \frac{\Delta t^3}{2} \end{bmatrix}, \tag{2.6.11}$$

where $diag_i(Q)$ is the i-th sub-diagonal of Q, with $i = 0$ denoting the main diagonal, and $-i$ and $+i$ denoting i-th sub-diagonals below and above the main diagonal,

respectively. Similarly, the measurement noise covariance R is

$$R = \begin{bmatrix} \sigma_{1x}^2 & 0 & 0 & 0 \\ 0 & \sigma_{2x}^2 & 0 & 0 \\ 0 & 0 & \sigma_{3x}^2 & 0 \\ 0 & 0 & 0 & \sigma_{4x}^2 \end{bmatrix}. \tag{2.6.12}$$

2.6.2 Multilane Tracking Using IPDAF

As stated in Section 3.2, an IPDA filter offers augmented information about track maintenance apart from state estimation of tracks. Instead of assuming the existence of targets as a hard-wired probability, the IPDA filter offers the choice to incorporate the track quality measure into the tracking process.

In the context of the PDA algorithm, for each validated measurement, the association probability (for each track) is calculated. To this end, the association probability $\beta_i^\tau(k)$ accounts for the probability of associating a measurement i to track τ_k, feature intensity of $f_i^\tau(k)$, and the likelihood ratio of associating a line with a feature measurement $e_i^\tau(k)$. These probabilistic pieces of information are used to associate new measurements to the targets. Given a linear dynamic model and an IPDAF based on [8], the state and measurement equations become

$$\hat{x}(k|k) = E[x(k)|Z^k] = \sum_{i=0}^{m_k} \hat{x}_i(k|k)\beta_i(k), \tag{2.6.13}$$

where $\hat{x}_i(k|k)$ is the updated state, conditioned on the event that the i-th validated measurement is correct, and $\beta_i(k)$ is the probability of associating a measurement i with a feature value of $f_i(k)$ to track k. The association probability for a set of m_k

gated or validated measurements with features $f_i(k)$ can be expressed as

$$\beta_i^\tau(k) = P\{\epsilon_i(k)|Z_k^\tau, f^\tau(k), m_k\}, \tag{2.6.14}$$

where $\epsilon_i(k)$ is the event described in Appendix A. In our case, we assume that each detected lane boundary measurement i has a feature of intensity $f_i(k)$. With reference to (A.0.13), the feature likelihood $e_i(k)$ can incorporated into the PDA algorithm as follows ([56]):

$$\begin{aligned} \beta_i(k) &= P\{\epsilon_i(k)|Z^k, m_k\} \\ &= \begin{cases} \frac{\mathcal{L}_i(k)e_i(k)}{1-P_DP_G+\sum_{i=1}^{m(k)}\mathcal{L}_j(k)e_i(k)}, \forall i \neq 0 \\ \frac{1-P_DP_G}{1-P_DP_G+\sum_{i=1}^{m(k)}\mathcal{L}_j(k)e_i(k)}, i = 0, \end{cases} \end{aligned} \tag{2.6.15}$$

where $i = \{0, 1, \ldots, m(k)\}$.

The overall IPDAF algorithm here embodies a traditional PDAF algorithm with special initialization and termination steps. This is outlined in Algorithm 4. The algorithm assumes a composite data type that is capable of capturing the evolution of the states over a period of time. As such, we use the `dot` notation to extract these properties. For instance, in our algorithm we use Γ as a variable that has all the information of all the tracks being considered in the problem. Various properties, such as *track type* and *track ID* (a unique identifier for each track), are extracted using the stated dot notation. Furthermore, the algorithm assumes the presence of the following auxiliary functions:

- `getTrackType()`: which returns the type of the track i, as temporary, retired,

Algorithm 3 IPDATracker.

▷ Inputs: Γ, Ψ
▷ Output: $\Gamma(Updated)$
▷ Section Variables : $\Lambda, \alpha, \alpha_T, \Omega$
▷ $\bar{x}$: Priors
▷ Ψ : Measurements
▷ Γ : Composite data structure for tracks
▷ Λ : A copy of Γ
▷ Ω : A set containing associated tracks
▷ α : A temporary variable
▷ α_T : A temporary set of tracks
$\Lambda = \Gamma.Tracks$
$\Omega \leftarrow \emptyset$
if $\Lambda.size() > 0$ **then**
    for `i=0; i<`$size(\Gamma)$`; i++` **do**
        $\hat{x}^i_{k|k-1} \leftarrow$ Eqn (2.6.18)
        $P^i_{k|k-1} \leftarrow$ Eqn (2.6.19)
        $W^i_k \leftarrow$ Eqn (2.6.27)
        $\hat{z}_{k|k-1} \leftarrow$ Eqn (2.6.20)
        $S_k \leftarrow$ Eqn (2.6.21)
        $V(k, \gamma) \leftarrow$ Eqn (2.6.23)
        if $V.size() > 0$ **then**
            for $j = 0$ **to** V.size() **do**
                $L^i_k \leftarrow$ Eqn (2.6.24)
                $e^i_k \leftarrow$ Eqn (2.6.25)
            end for
            $\beta^i_k \leftarrow$ Eqn (2.6.15)
            $\hat{x}^i_{k|k} \leftarrow$ Eqn (2.6.26)
            $P^i_{k|k} \leftarrow$ Eqn (2.6.28)
        else
            $\Lambda^i.x_k \leftarrow \hat{x}^i_k$
            $\Lambda^i.P_k \leftarrow P^i_k$
            $\Omega \leftarrow \Omega \cup 0$
            $\alpha \leftarrow$ `getTrackType`(Ω, Λ^i)
            $\Lambda^i.ID \leftarrow$ `getTrackID`(α, Λ^i)
            $\Lambda^i.type \leftarrow \alpha$
        end if
        $\Lambda^i.x_k \leftarrow \hat{x}^i_{k|k}$
        $\Lambda^i.P_k \leftarrow P^i_{k|k}$
        $\Omega \leftarrow \Omega \cup 1$
        $\alpha \leftarrow$ `getTrackType`(Ω, Λ^i)
        $\Lambda^i.ID \leftarrow$ `getTrackID`(α, Λ^i)
        $\Lambda^i.type \leftarrow \alpha$
    end for
    $\Gamma.NonAsscociated \leftarrow \Psi - \Omega$ //
    $\alpha_T \leftarrow$ `mergeTracks`(Λ)
    $\Gamma.Tracks \leftarrow$ `cleanTracks`(α_T)
else
    $\Gamma.NonAsscociated \leftarrow \Psi$
end if
return Γ

or active;

- `getTrackID()`: which returns the unique identifier for the track;
- `mergeTracks()`: which fuses the supplied set of tracks; and
- `cleanTracks()`: removes dead tracks from the list.

The underlying aspects of multilane tracking follow the principles of multitarget tracking as in [8], and are discussed in the following subsections.

Track Initialization

The track initialization process (for each track) can rely on one or two seed points. In the one-point initialization method, position can be initialized from a single observation with a zero velocity vector. Due to [61],

$$\text{diag}(P(0|0))^T = \begin{bmatrix} \sigma_{1x}^2 \\ \sigma_{2x}^2 \\ \sigma_{3x}^2 \\ \sigma_{4x}^2 \\ (\frac{V_{max}}{2})^2 \\ (\frac{V_{max}}{2})^2 \\ (\frac{V_{max}}{2})^2 \\ (\frac{V_{max}}{2})^2 \end{bmatrix} \tag{2.6.16}$$

and

$$\hat{x}(0|0)^T = \begin{bmatrix} x_{1_z} \\ x_{2_z} \\ x_{3_z} \\ x_{4_z} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.6.17}$$

This initialization allows the standard gating to be used during the following time step.

Measurement Prediction

For each track, the state vector, the measurements, and the state covariance matrices are predicted ahead as in the standard Kalman filtering. i.e.,

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} \tag{2.6.18}$$

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F'_{k-1} + Q_{k-1} \tag{2.6.19}$$

$$\hat{z}_{k|k-1} = H_k\hat{x}_{k|k-1} \tag{2.6.20}$$

$$S_k = H_kP_{k|k-1}H'_k + R_k \tag{2.6.21}$$

with the assumption of Gaussian posterior for $p(x)$ as

$$p[x_{k-1}|Z^{k-1}] = \mathcal{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P_{k-1|k-1}). \tag{2.6.22}$$

Measurement Gating

For each track, a validation gate is set up around the predicted measurement to select the candidate measurements for the data association. The size of the validation gate is correlated to the innovation covariance and the measurement noise. As per (2.6.22), at most one of the validated measurements can be assigned to the target. The measurements outside the gate are assumed to be false alarms or measurements belonging to other targets. The validation region is the elliptical shape as follows:

$$V(k, \gamma) = \{z : [z - \hat{z}_{k|k-1}]' S_k^{-1} [z - \hat{z}_{k|k-1}] \leq \gamma\}, \tag{2.6.23}$$

where n_z is the dimension of the measurement vector representing the degrees of freedom, and γ is the gating threshold. Here, the gating threshold γ is a chi-square distribution, parameterized by the probability of gating P_G, and by the degrees of freedom n_z [6].

Data Association

An incoming measurement at time index k, $z_i(k)$ with feature $f_i(k)$ is associated to track τ, based on the association probability given by 2.6.15:

$$\beta_i(k) = \begin{cases} \frac{\mathcal{L}_i(k)e_i(k)}{1-P_D P_G+\sum_{i=1}^{m(k)}\mathcal{L}_j(k)e_i(k)}, \forall i \neq 0 \\ \frac{1-P_D P_G}{1-P_D P_G+\sum_{i=1}^{m(k)}\mathcal{L}_j(k)e_i(k)}, i = 0 \end{cases}$$

Here, there are two likelihood ratios of interest: $L_i(k)$ and $e_i(k)$. The former is the likelihood ratio of the incoming measurement $z_i(k)$. This is defined as

$$\mathcal{L}_i(k) = \frac{\mathcal{N}[z_i(k); \hat{z}(k|k-1), S(k)]P_D}{\lambda}, \tag{2.6.24}$$

where λ is the uniform density of the location of false measurements. The second parameter of interest, $e_i(k)$, is the likelihood ratio of measurement z_i^τ with feature f_i^τ of the track τ. This is defined as

$$e_i(k) = \frac{p_1^\tau(f_i)}{p_0^\tau(f_i)}. \tag{2.6.25}$$

Both of these measurements, $z_i(k)$ and z_i^τ, are expected to originate from the target and not from the clutter.

State Update

The state, gain, and covariance update equations of the PDAF are

$$\hat{x}(k|k) = \hat{x}(k|k-1) + W(k)\mathcal{V}(k) \tag{2.6.26}$$

$$W(k) = P(k|k-1)H(k)'S(k)^{-1} \tag{2.6.27}$$

$$P(k|k) = \beta_0(k)P(k|k-1) + (1-\beta_0(k))P^c(k|k) + \tilde{P}(k), \tag{2.6.28}$$

where $W(k)$ is the Kalman gain, and $\mathcal{V}(k)$ is the combined innovation defined by

$$\mathcal{V}(k) = \sum_{i=1}^{m(k)} \beta_i(k)\mathcal{V}_i(k), \tag{2.6.29}$$

where m_k is the number of measurements inside the gating region, and $\beta_0(k)$ is the probability of all measurements being incorrect at time index k. With no information on which of the m_k measurements are correct or incorrect, the correct updated covariance can be expressed as

$$P^c(k|k) = P(k|k-1) - W(k)S(k)W(k)' \tag{2.6.30}$$

$$\tilde{P}(k) = W(k)\Big\{\sum_{i=1}^{m(k)} \beta_i(k)\nu_i(k)\nu_i(k)' - \nu(k)\nu(k)'\Big\}W(k)'. \tag{2.6.31}$$

Track Management

During the course of tracking, several tracks are maintained in parallel, and their states are continuously updated upon receiving measurements. A track can be in three different states: tentative, confirmed, and terminated. The potential state transitions of tracks are illustrated in Figure 2.8.

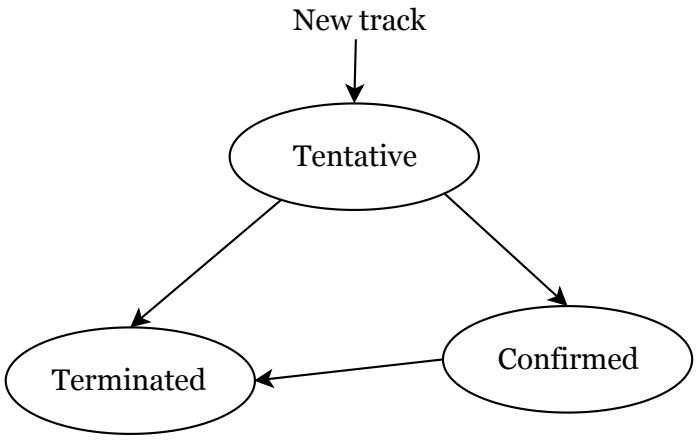


Figure 2.8: Track management state diagram.

During the initialization phase, every unassociated measurement will form a tentative track. However, upon following detections or measurements and gating operations, the tracks will begin to form and their status will be updated as confirmed. However, if no further measurements to track associations are possible—a possibility where no detections are observed from the target responsible for the track—the corresponding track is terminated. In the case where $P_D < 1$, it is essential to check the quality of measurement-to-track association before engaging in a track status update. One of the approaches for assessing the quality of measurement-to-track association is the goodness of fitting. The goodness of fitting, often represented by the log-likelihood function of the track, can be expressed as a recursive function as follows ([6]):

$$\lambda(k) = \lambda(k-1) + \mathcal{V}(k)' S(k)^{-1} \mathcal{V}(k), \tag{2.6.32}$$

where $\mathcal{V}(k)$ is the innovation matrix, and $S(k)$ is the covariance of the innovation matrix $\mathcal{V}(k)$. The last term in (2.6.32) has a chi-square density with n_z degrees of

freedom, where n_z is the dimensionality of the measurement vector. As the innovations are independent, the log-likelihood function at time k is chi-squared distributed with kn_z degrees of freedom. This is actually a measure of the goodness of fit to the assumed target model. Thus, the test function for keeping (or terminating) a track can be expressed as

$$\lambda^k \leq \lambda^k_{max} \tag{2.6.33}$$

$$\lambda^k_{max} = X^2_{kn_z}(1-\alpha). \tag{2.6.34}$$

This threshold follows from the chi-square for kn_z degrees of freedom, where the tail probability α is the probability that a true track will be rejected. In our case, this threshold is around 0.01.

However, the actual state transitions are performed through more rigorous checks. In our case, we maintain a number of (hidden) measurement and association counters for each track. These counters are used to assess how active the measurement-track association is. For instance, a track with a tentative state will be promoted to a confirmed state only if a proportion $r_{t\to c}$ of the recent measurements are associated together with the track in question; otherwise it is terminated. Similarly, a track's confirmed state will be retained only if a certain proportion $r_{c\to c}$ of the last set of measurements are associated with the track; otherwise it is terminated. In the case of $P_D < 1$, a track is kept alive if it satisfies the following two requirements:

- There have been m detections out of the last n sampling times; and
- The measurement-to-track fit is acceptable.

It is worth noting that traffic conditions vary throughout the day, and hence the speed of the vehicles. This poses a challenge in that not much variation is observed in slow-moving conditions between frames while the sampling is active. This requires either adaptive sampling or adaptive ratios of $r_{t\to c}$ and $r_{c\to c}$. However, we do not use these techniques in our case. Note that in using IPDA to track multiple lanes, we make the implicit assumption that the lanes do not intersect.

2.7 Experiments and Evaluations

The proposed algorithms were implemented using the OpenCV library and `C++` on a system with Intel i7 CPU, clocked at 2.9 GHz with 16 GB RAM. We evaluate our method with two group of algorithms, model-based and supervised methods.

2.7.1 Comparison with Model-Based Algorithms

In this section we test the precision of our estimation compared to two methods from ([3]) and ([42]). The algorithms were tested using the Caltech Lane dataset ([3]). Caltech Lane dataset is a standard real data benchmark in the lane detection applications. The dataset has four video clips taken at different times of the day around the urban areas of Pasadena in California. Each of these video clips has a resolution of 640×480 pixels, and includes various lighting and illumination conditions, writing along with lane markings (Clip#1), sun glint and different pavement types (Clip#2), shadows and crosswalks (Clip#3), and congested settings (Clip#4). As such, they are sufficiently representative of various challenging conditions in tracking lane markings. In total, $1,224$ frames and $4,399$ lane boundaries were processed. The details

of these video clips are given in Table 2.2.

Table 2.2: Caltech dataset used in the evaluation

Clip ID	Clip Name	No. of. Frames	No. of. Lane Boundaries
1	cordoval1	250	975
2	cordoval2	406	1,131
3	washington1	336	1,329
4	washington2	232	964

During the evaluation, we computed the true and false positive rates (TPR and FPR, respectively), where the TPR is the ratio of the number of detected lane boundaries to the number of target lane boundaries and the FPR is the ratio of the number of false positives to the number of target lane boundaries. The frames were processed at the rate of seven frames per second, similar to that of other methods in the literature ([3], [42]). In addition to TPR and FPR metrics, we also included another metric, false positives per frame (FP/Frame or FPF), which is an average of false positives across all frames. It would be equally valid to consider the true positives per frame rate. We compare the performance of the proposed algorithm using the three metrics, TPR, FPR, and FP/Frame, against two competitive methods, those of [3] and [42]. The results are shown in Table 2.3 and Figure 2.9.

Noting that higher TPR, lower FPR, and lower FP/Frame values are desirable, we highlight the best (boldface) and second best (underlined) results in Table 2.3, and the overall results in Figure 2.9. To ease the analysis, we also present the performance difference between the proposed and the two concerned algorithms in Figure 2.10. We show the differences so that the benefits can be assessed directly. That is, a positive difference means that the proposed algorithm outperforms a given algorithm. This is

Table 2.3: Comparison of Our Approach with Other Lane Detection Algorithms

	Method in [3]			Method in [42]			Proposed		
	TPR	FPR	FP/Frame	TPR	FPR	FPF/Frame	TPR	FPR	FP/Frame
Clip#1	0.823	0.099	0.384	0.892	0.125	0.488	0.899	0.093	0.405
Clip#2	0.839	0.224	0.672	0.865	0.209	0.628	0.870	0.166	0.535
Clip#3	0.934	0.148	0.542	0.850	0.111	0.408	0.937	0.107	0.455
Clip#4	0.890	0.102	0.418	0.898	0.063	0.259	0.974	0.099	0.424
Overall	0.871	0.148	0.529	0.874	0.131	0.469	**0.920**	**0.116**	**0.454**

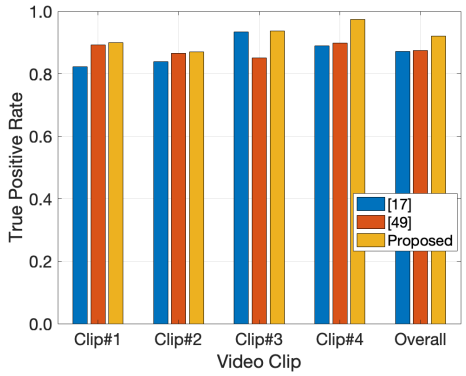
true both for TPR and FPR metrics.

A number of observations can be made about this performance comparison.

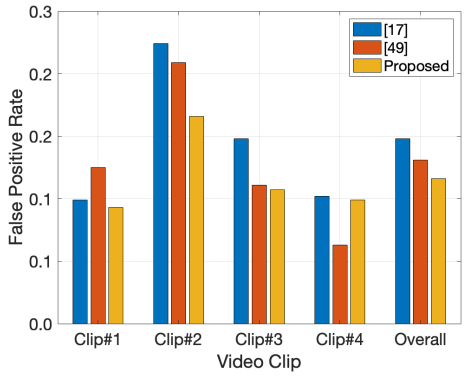
- The TPR performance of the proposed algorithm is consistently higher than those of the other two algorithms throughout all video clips. The performance of the proposed algorithm over Clip#3 is significantly higher than that of the method in [42];
- The FPR performance of the proposed algorithm is better than that of the method in [3] in all cases;
- The FPR performance of the proposed algorithm performs better than that of the method in [42] except in Clip#4. One potential reason for the suboptimal performance for this clip can be attributed to the difficulties in association in congested settings; and
- The FP/Frame performance is mixed across the cases.

Overall, the proposed approach outperforms the other two methods across all cases. The overall performance differences are 5% 2%, and 2% [1]for TPR, FPR, and

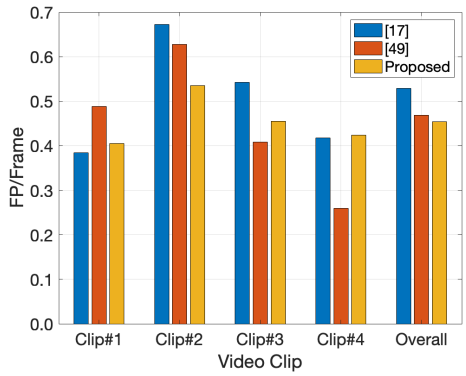
[1]For measurement standard deviation about 1.2, confidence level %95 and precision 0.05 and 3 control points for each pseudo measurement overall 4399 lane markings test is statistically significance



(a) TPR Performance



(b) FPR Performance



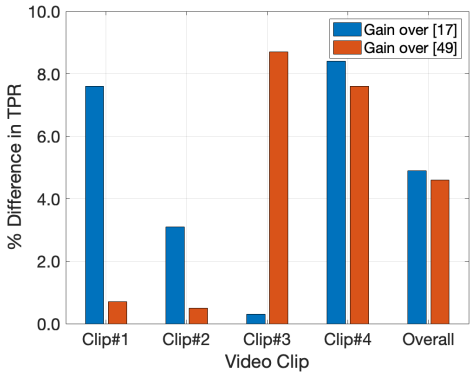
(c) FPF Performance

Figure 2.9: Performance of different lane detection algorithms.

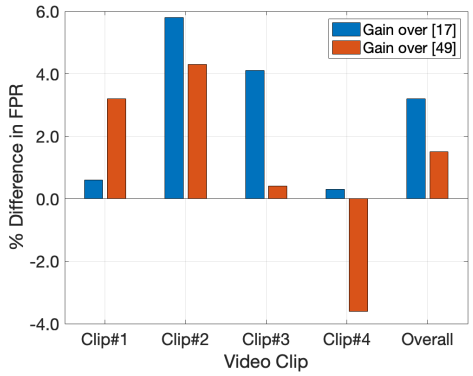
FPF cases, respectively, when compared against the second best version, the method in [42]. To ensure the operation, we manually analyzed the datasets to label all visible and invisible lane markings as target lane markings.

2.7.2 Comparison with Fully Supervised Methods

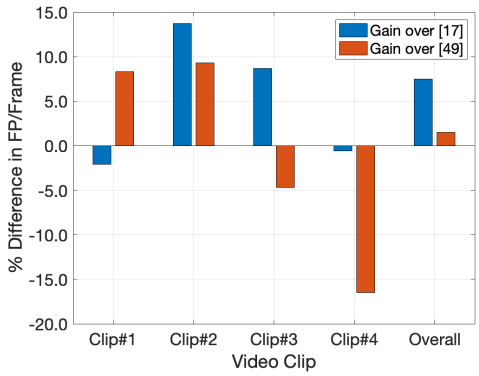
Fully supervised methods have recently been used for detecting lane markings in the environments with the presence of clutter as well. Pan and his team (2017) used the Spatial Convolutional NN (SCNN) to extract spatial correlation between



(a) Difference in TPR Performance.



(b) Difference in FPR Performance.



(c) Difference in FPF Performance.

Figure 2.10: Performance difference of the proposed algorithm across three metrics.

rows and columns. This structure enables message passing between pixels across the rows and columns in a layer. These relationships help with learning semantic objects with dependency between parts ([68]). They evaluated their method using the CULane and TuSimple datasets ([68], [18]), achieving outstanding results detecting the lane markings with clutter, especially when the lane-lines are straight. They used probability maps (probmaps) of lane markings derived from SCNN, and after verification, filled them with a cubic spline, which made the final prediction. The SCNN is effective when the lanes are straight and there are suficient feature cues, but in a situation with a lack of distinctive features, it misses clustering and yields

incorrect detection, as shown in Figure 3.10. This occurs mostly with curved, merging, or splitting lane-lines, when the symmetric structure is missed. In contrast, in our proposed method, Bayesian clustering of line segments combined with an IPDA filter enable the use of the stream of frames to make a proper prediction. Notice that our method can be attached to any supervised lane detection algorithm to remove false alarms and add more semantics to its result, using the time domain information.

We evaluated our method with the SCNN method ([68]) on the TuSimple dataset ([81]), using the method's trained model weights. The TuSimple dataset has about 7,000 one-second-long video clips of 20 frames each. Pan and his team prepared the ground truth result for the last frame (Frame 20), including height (h-sample) and width values corresponding to lane-lines.

The TuSimple dataset includes most challenging situations, from curves and shadows to splitting and merging highway lanes. We tested our algorithm against the SCNN algorithm on simple straight and curved lane-lines only. We tested on five different videos overall, using 100 frames each for straight and curved lanes [2]. Our methods yielded significant improvements over the SCNN method in accuracy, false positive, and false negative rates in both curved and straight situations. We used the same accuracy formula as the TuSimple benchmark, which is:

$$\text{accuracy} = \frac{\sum_{\text{clip}} C_{\text{clip}}}{\sum_{\text{clip}} S_{\text{clip}}}, \tag{2.7.1}$$

where C_{clip} is the number of correct points in the last frame of the clip, and S_{clip} is the number of requested points in the last frame of the clip. If the difference between the width of ground truth and prediction is less than a threshold, the predicted point

[2]For measurement standard deviation about 1.2, confidence level %95 and precision 0.05 and 45 degree of freedom, overall 200 frames and 600 lane markings test is statistically significance

is a correct one. We evaluated the values of all heights in h-sample.

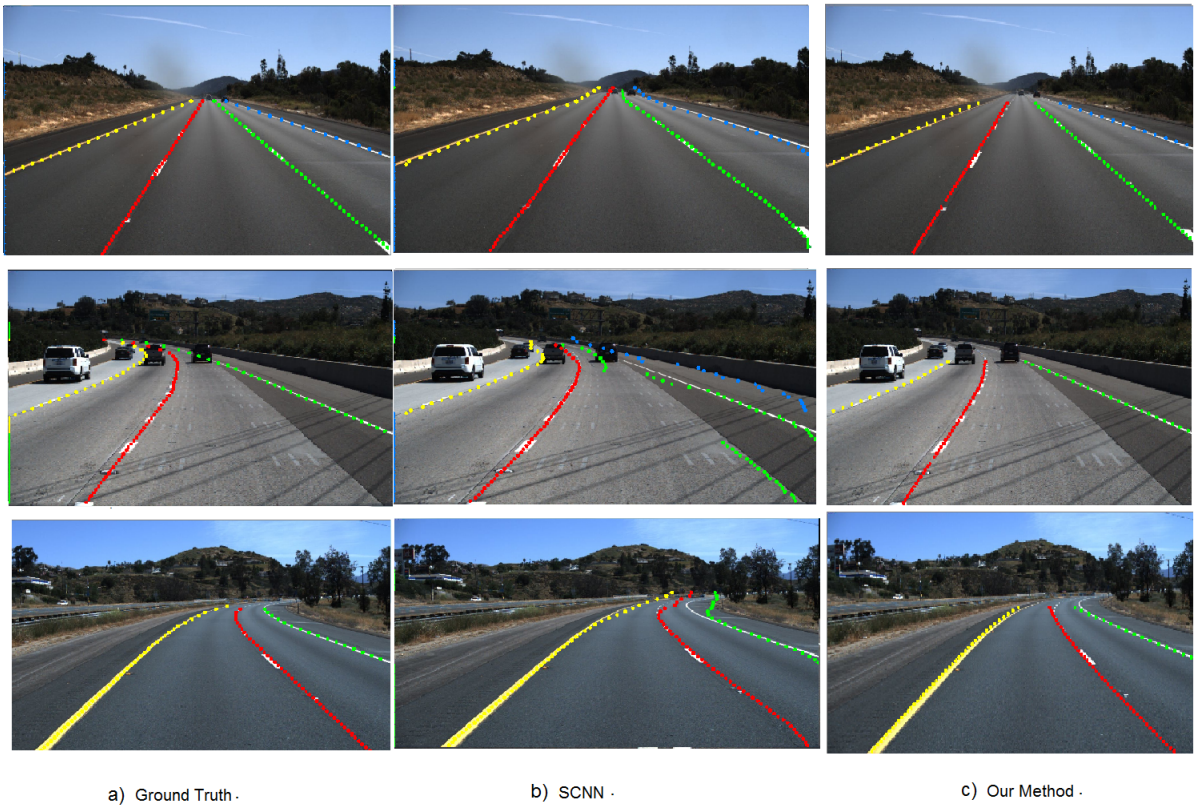


Figure 2.11: Result of the SCNN algorithm in curved and straight situations.

Based on the formula above, we also computed the rate of a false positive and false negative for the test results. False positive means the lane is predicted but not matched with any lane in ground truth. False negative means the lane is in the ground-truth but not matched with any lane in the prediction.

$$\text{FP} = \frac{F_{\text{pred}}}{N_{\text{pred}}}$$
$$\text{FN} = \frac{M_{\text{pred}}}{N_{\text{gt}}}, \tag{2.7.2}$$

where F_{pred} is the number of incorrectly predicted lanes, N_{pred} is the number of all predicted lanes. M_{pred} is the number of missed ground truth lanes in the predictions, and N_{gt} is the number of all ground truth lanes.

We also compared our running time with the SCNN algorithm on a normal CPU. In [68], running time with normal CPU is reported as about 5.6 frames per second, but the algorithm run time on our computer (Intel i7 CPU, clocked at 2.9 GHz with 16 GB RAM) was substantially lower, 0.71 frames per second. The results are shown in Table 4.1.

Table 2.4: Comparison of Proposed Approach with SCNN Fully Supervised Method

	SCNN Method in [68]				Proposed			
	FPR	FNR	Accuracy	Frame/s	FPR	FNR	Accuracy	Frame/s
Straight Highway#1	0.166	0.250	0.855	0.71	0.160	0.250	0.880	6.5
Curvy Highway#2	0.479	0.416	0.825	0.72	0.166	0.250	0.861	6.5
Overall	0.375	0.361	0.836	0.71	**0.160**	**0.250**	**0.869**	**6.5**

2.8 Conclusions

In this paper, we proposed a novel approach for multilane detection with expanding estimation into the time domain using an IPDA filter. By using the intensity feature in conjunction with the Hough transform, we first formulated an algorithm to cluster and group multiple line segments belonging to each lane-line. By using these lane-lines as an extended target (spline), we then identified a set of control points, which then were tracked on every frame. This multitarget tracking-based approach outperformed other model-based and fully supervised state-of-the-art results in the literature using a realistic dataset. In particular, for two other model-based methods, the proposed

approach was able to offer as much as 5%, 2%, and 2% improvements on the true positive, false positive, and false positives per frame rates, respectively, compared to the second-best approach. Furthermore, for one of the newest fully supervised lane detection methods (SCNN), our method has about 30%, 50%, and 3% improvements in overall on the false positive, false negative, and accuracy, respectively, when applied to both curved and straight roads. For normal CPU and without GPU, the frame rates for our algorithm were significantly lower at 6.5 Hz, compared to SCNN, which was 0.71 Hz. We believe that our algorithm frame rate can be even further enhanced by accelerating the algorithms using appropriate embedded platforms.

Chapter 3

Lane Tracking Using Dependent Extended Target Models and Multi-Output Spatiotemporal Gaussian Processes

3.1 Introduction

Extended Target Tracking (ETT) is the process of estimating both the shape of a target and the evolving kinematic states using a sequence of noisy measurements ([34]). It is sometimes considered similar to Group Target Tracking (GTT), when states of closely grouped single-point targets moving with similar dynamics are estimated. Extended target tracking is using in many modern applications, including

robot navigation, tracking of people and cars, chemical and biological reaction estimation, disease epidemics, and pollution evolution ([76]). It is also considered an integral part of many Advanced Driver Assistance Systems (ADAS) and autonomous or self-driving cars in urban traffic ([52]). New advanced sensors can give information about the shape of objects in addition to their kinematics. The process of tracking targets with these multiple detected measurements is known as Multiple Detection Target Tracking (MDTT). MDTT requires nonlinear estimation methods that possibly deal with clutter and data association ([6]; [36]). Typical extended targets can be modeled with simple shapes like circles, ellipsoids, or rectangles ([48]; [33]). Random Matrices (RM) was the first Bayesian method for tracking extended targets ([48]). The formulation of the RM technique assumes that the extended target is ellipsoidal. However, in some practical applications, this assumption doesn't hold. To describe and estimate the non-ellipsoidal shape of an extended target, several techniques, such as the multi-ellipsoid RM model ([53]) and random hyper-surface model ([9]) have been proposed.

A Gaussian process (GP) is another excellent nonparametric Bayesian modeling method that has been used recently for tracking extended targets ([82];[35]; [90]; [1]). A GP is a distribution over an unknown and nonlinear function in a continuous domain. The observed values of these functions can be used to predict the values at unobserved points. The kernel approach for GP regression was introduced in [69]. The best characteristic of the kernel-based approach is the the ability to encode the prior assumption of the process into a kernel function.

In real-time applications, the kernel-based GP regression is not feasible because it is a batch process and needs inversion of a covariance matrix with cubic complexity

with respect to the number of inputs. In this case, a recursive GP regression model can be used ([75]). In [82], GP was used for the first time to model extended targets. A recursive GP regression called GP-KF was used to estimate the shape and kinematics of extended targets in linear time. The evolution of the shape was considered Markovian with a forgetting factor for the temporal dynamic, similar to the Huber method in 2014 ([41]).

Since the characteristics of GP are summarized in its covariance function, it is possible to model the evolution of the covariance function using a state-space model representation. This is done by augmenting the states with a sufficient number of time derivatives; further, a smoother needs to be used in addition to a filter. In [38] it is shown that for many stationary temporal kernels, there exists a state-space model that can achieve the exact same results as a GP regression. Sarkka and colleagues introduced the recursive Spatio-Temporal Gaussian Process (STGP), solvable with an infinite dimension Bayesian filter ([77]). STGP can model correlation in space and time. In tracking extended targets when the shape of the object is changing in time, the spatial part of the object is correlated in both the space and the time domains. In this case, considering the evolving shape only in the spatial domain will compromise performance. In [1], STGP was used for the first time to track extended targets, using a technique called STGP-KF that uses the idea in [74] to factorize the power spectral density function of the STGP covariance function to find the transfer function. This technique was used to track rigid objects in radial space.

In this paper we exploit the correlation between outputs to model the interfering of targets with the Multi-output Gaussian Process (MOGP) method ([14]). In MOGP, one single input (index point) has a set of correlated outputs. [2] demonstrated that

the overall covariance function could be parameterized as a combination of an output similarity's matrix and the input spatial kernel. In fact, instead of viewing each output separately, this model corresponds to a scenario where only one Gaussian process exists, and each task simply shares this latent process with various weights plus additive white noise ([72]).

In lane detection, the subject of this paper, many studies have used extended target models ([59]). Most of the proposed methods perform their task in two steps: clustering and grouping lane features and then modeling and tracking lane lines. The feature extraction step has primarily been accomplished through use of the Hough transform, Line Segment Detection (LSD), RANdom SAmple Consensus (RANSAC), and other parametric regression models ([3]; [64]). For the modeling, polynomial or spline model fitting is typically used as a measurement model combined with a simple random walk dynamic model for extends. These models are parametric, highly sensitive to prior information, and not flexible for modeling dependency. Since the lanes are parallel and the road's shape is designed to be smooth enough for safe driving, extracted features of lane markings tend to be spatially and temporally correlated together between frames. In [83] GP was used to cluster the lane marking features using 3D LIDAR and to model the shape of the road. They used GP regression to predict hyperparameters for the road curvature model and a naive technique of maximum a posterior estimation to predict the new parameters using prior information and incoming data.

Fully supervised methods like Neural Network (NN), Convolutional NN, and Deep NN have also been used recently for detection and classification of extended targets. To add inference, NN could be combined with other Bayesian modeling approaches.

In [80] Convolutional NN (CNN) has been used for the classification of tracked object results of GP-KF [82]. In some applications like lane tracking, the lack of distinctive features tends to make the algorithms confused by other objects with a similar local appearance. NN works perfectly for detecting objects with enough distinctive features. For example, Convolutional Patch Network (CPN) can be used for detecting road surfaces and Region Of Interest (ROI) and You-Only-Look-Once (YOLO) networks for detecting and removing other patterns on the road or cars ([60]). Many approaches have been introduced to take advantage of dependency in NN and overcome the problem of lack of distinctive features for lane markings. In [68] a Spatial Convolutional NN (SCNN) was used to extract spatial correlation between rows and columns. These relationships help with semantic objects that have dependent parts, like lane markings. The other approach that can be used for this purpose is a two-step learning method. In [86] a real-time network called "Lanenet," based on a two-stage Deep NN for lane detection, was employed. One network detected lane marking edges and the other grouped and localized lane markings. Note that in this paper we call a group of lane markings a "lane-line." To have a proper output of fully supervised learning, we need thousands of training data points, especially in challenging scenarios with clutter, because simple shapes like lanes can be mistaken for targets.

In this work, we propose a novel kernel-based estimation approach based on a joint probabilistic data association coupled filter called K-JPDACF that is used for clustering and feature extraction. The K-JPDACF is a semi-supervised version of JPDACF with a trained kernel prior and coregionalization matrix. The result of K-JPDACF is a set of extracted features and clustered measurements.

After the features belonging to each lane line are clustered, a novel recursive approach based on MOGP and STGP is proposed for estimating and fitting lane markings. We model dependency between lane markings in each frame and then between frames with separable spatial and temporal kernels. Our method is an extension of the method introduced in [1] to track multiple dependent extended targets in real time. This procedure, called the Multi-output Spatio-Temporal Gaussian Process Kalman Filter (MO-STGP-KF), is Bayesian and non-parametric, and requires a very low amount of training data.

The contributions of this work include:

1. Introducing a Kernel-based Joint Probabilistic Data Association Coupled Filter (K-JPDACF) for clustering features belonging to lane-lines in linear time.

2. Introducing a dependency model based on MOGP for training lane-lines and initializing the filter.

3. Introducing recursive MO-STGP-KF for tracking dependent multiple extended targets and implementing it for tracking multiple lane lines.

The remainder of this paper is organized as follows: In Section 3.2 we provide background on a number of aspects that this work builds upon, including GP and MOGP regression and Recursive STGP. In Section 3.4, we formulate the overall problem, and in section 3.5, we discuss our approach for solving each of the subproblems. The results of our evaluations are then presented in Section 4.4 and we discuss conclusions in Section 4.5.

Table 3.1: Symbols used in this manuscript.

Symbol	Description
$\mathbf{z_k}$	Measurements at time step k, $\mathbf{z}_k = \{u_i, z_i\}_{i=1}^{M}$
$\mathbf{u}$	Index points or input vector corresponding to measurements
$\mathbf{x}_k$	state vector corresponding to GP function $\mathbf{f}(\mathbf{u}, k)$
$\mathbf{w}_i$	Process Gaussian noise with variance Q
v_i	Measurement Gaussian noise for single measurement with variance R
$K(\mathbf{u}, \mathbf{u}')$	Covariance matrix
$\kappa(u_i, u_j)$	Covariance function or kernel function
k	Discrete time step
$\mathbf{Q}$	Covariance of the process noise
$\mathbf{F}$	Discrete state transition matrix of Extend
$\mathbf{H}$	Measurement matrix

3.2 Background

3.2.1 A Review of Gaussian Process

A GP is a collection of random variables for which any finite subsets have a joint Gaussian distribution ([69]). It can be described as a generalization of Gaussian distribution over function space. A GP can be fully described by its mean $\mu(\mathbf{u})$ and covariance matrix $K(\mathbf{u}, \mathbf{u}')$. The covariance of a GP can be implemented through a kernel function. A kernel function is a positive-definite function $\kappa(u_i, u_j)$ that demonstrates the connection of two latent outputs $f(u),\ f(u')$ based on their index points $u,\ u'$ ([69]). The parameters of the mean and the covariance kernel are called hyperparameters. The optimal values of the hyperparameters can be determined by maximizing the likelihood of the GP on a given set of training data. This process is called learning. Unknown complicated nonlinear functions can be learned with a GP. In most of the application $\mu(\mathbf{u})$ can be a zero matrix. A rigid or nonrigid irregular extended target that maps a constant index points (u) to spacial output function $f(u)$

can be modeled by a GP. A GP is defined as:

$$\mathbf{f}(\mathbf{u}) \sim \mathcal{GP}(\mu(\mathbf{u}), K(\mathbf{u}, \mathbf{u}')), \tag{3.2.1}$$

$$\mu(\mathbf{u}) = E[f(\mathbf{u})], \qquad K(\mathbf{u}, \mathbf{u}') = E[(f(\mathbf{u}) - \mu(\mathbf{u}))(f(\mathbf{u}') - \mu(\mathbf{u}'))^\top], \tag{3.2.2}$$

where $\mathbf{u} = [u_1, u_2, ..., u_N]$ represents the key-points or inputs of the function $f(.)$, which identifies the latent output. According to the GP assumptions, the function values at a finite number of index points are normally distributed,

$$\begin{bmatrix} f(u_1) \\ \vdots \\ f(u_N) \end{bmatrix} \sim \mathcal{N}(\mu, K), \tag{3.2.3}$$

where mean and covariance matrix without considering noise are:

$$\mu(\mathbf{u}) = \begin{bmatrix} \mu(u_1) \\ \vdots \\ \mu(u_N) \end{bmatrix}, \quad K(\mathbf{u}, \mathbf{u}') = \begin{bmatrix} \kappa(u_1, u_1') & ... & \kappa(u_1, u_N') \\ ... & \ddots & ... \\ \kappa(u_N, u_1') & ... & \kappa(u_N, u_N') \end{bmatrix} \tag{3.2.4}$$

Gaussian Process Regression: The distribution of a nonlinear function can be estimated with the Gaussian Process Regression (GPR). We can use GPR to estimate the unobserved parts of the object using a kernel function prior and observed parts, as shown in Fig. 3.1.

Suppose we have a set of noisy observation z_i of the unknown latent function $f(.)$,

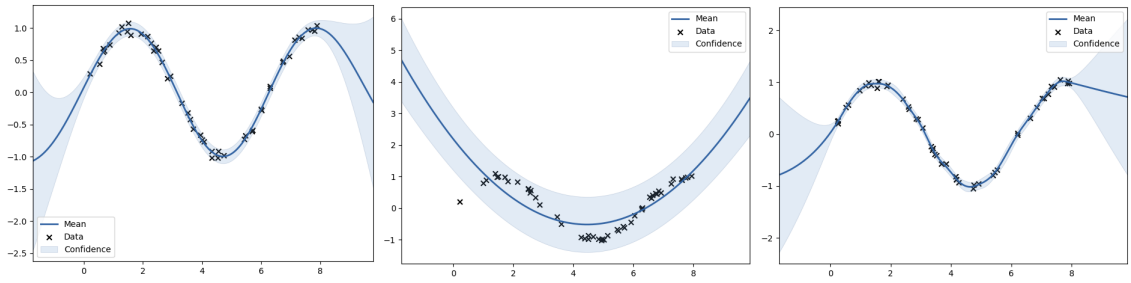


Figure 3.1: Effect of prior kernel in GP regression. (A) Squared exponential kernel; (B) Second degree polynomial kernel; (C) Matern32.

described with measurement model:

$$z_i = f(u_i) + v_i, \quad v \sim \mathcal{N}(0, R), \tag{3.2.5}$$

where v_i is a zero-mean Gaussian random variable with variance R. Our goal is to estimate the function values evaluated at some arbitrary inputs with considering observations z_i. In tracking applications, the latent function vector $\mathbf{f}^f = [f(u_1^f), ..., f(u_{Nf}^f)]^\top$ can be considered a state of the object for corresponding index points $\mathbf{u}^f = [u_1^f, ..., u_{Nf}^f]^\top$. The model is learned by using measurements $\mathbf{z} = [z_1, ...z_M]^\top$ and the corresponding index points $\mathbf{u} = [u_1, ..., u_M]^\top$. Using 3.2.3 and 3.2.5, the joint distribution of the measurements and the function values stay at a normal distribution, as:

$$\begin{bmatrix} \mathbf{z} \\ \mathbf{f} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{u},\mathbf{u}) + I_M \otimes R & K(\mathbf{u},\mathbf{u}^f) \\ K(\mathbf{u}^f,\mathbf{u}) & K(\mathbf{u}^f,\mathbf{u}^f) \end{bmatrix}\right), \tag{3.2.6a}$$

where $\otimes$ is the Kronecker product ([40]) and:

$$K(\mathbf{u}, \mathbf{u}^{\mathbf{f}}) = \begin{bmatrix} \kappa(u_1, u_1^f) & .... & \kappa(u_1, u_{Nf}^f) \\ ... & \ddots & ... \\ \kappa(u_N, u_1^f) & ... & \kappa(u_N, u_{Nf}^f) \end{bmatrix}. \tag{3.2.6b}$$

From the joint distribution equation 3.2.6a, the conditional distribution $p(\mathbf{f}|\mathbf{z})$ can be computed similarly to [69]:

$$p(\mathbf{f}|\mathbf{z}) \sim \mathcal{N}(A\mathbf{z}, P), \tag{3.2.7a}$$

where:

$$A = K(\mathbf{u}^{\mathbf{f}}, \mathbf{u})[K(\mathbf{u}, \mathbf{u}) + I_M \otimes R]^{-1}, \tag{3.2.7b}$$

$$P = K(\mathbf{u}^{\mathbf{f}}, \mathbf{u}^{\mathbf{f}}) - K(\mathbf{u}^{\mathbf{f}}, \mathbf{u})[K(\mathbf{u}, \mathbf{u}) + I_M \otimes R]^{-1} K(\mathbf{u}, \mathbf{u}^{\mathbf{f}}) \tag{3.2.7c}$$

The normal GPR is a batch process and computing the inverse matrix $[K(\mathbf{u}, \mathbf{u}) + I_M \otimes R]^{-1}$ is a bottleneck with an order of $O(N^3)$ complexity. Using Cholesky decomposition of a matrix to find the inverse would be faster and numerically more stable ([40]).

3.2.2 Multi-Output Gaussian Process (MOGP)

If we have multiple outputs corresponding to one index point, the dimension of the output expands from 1-dimensional space into D-dimensional space, where for a given index input point u_i, the output becomes a vector $\overline{\mathbf{f}}(u_i) = [f_1(u_i), f_2(u_i), ..., f_D(u_i)]^\top$ of size D. Here $f_d(u_i)$ denotes the d-th latent output for index point u_i.

If latent outputs are correlated, then a Linear Model of Coregionalization (LMC) can be applied. In LMC, the d-th output of any input u is expressed as a linear combination of $\mathcal{Q}$ independent of latent process $U_q(u)$. For each output we can write ([2]):

$$f_d(u) = \sum_{q=1}^{\mathcal{Q}} a_{d,q} U_q(u), \tag{3.2.8}$$

where each $U_q(u)$ represents an underlying latent process with different and independent GP priors. Here each latent function arises from a unique and independent latent GP. From equation 3.2.8, a computing covariance function between coupled outputs $\overline{\mathbf{f}}(u) = [f_1(u), ..., f_D(u)]^\top$ can be expressed as:

$$\text{cov}(\overline{\mathbf{f}}(u), \overline{\mathbf{f}}(u')) = \sum_{q=1}^{\mathcal{Q}} \mathbf{a}_q \mathbf{a}_q^\top \mathbb{E}\{U(u)U(u')^\top\} - \sum_{q=1}^{\mathcal{Q}} \mathbf{a}_q \mathbf{a}_q^\top \mathbb{E}\{U(u)\}\mathbb{E}\{U(u')^\top\} \tag{3.2.9}$$

$$\text{cov}(\overline{\mathbf{f}}(u), \overline{\mathbf{f}}(u')) = \sum_{q=1}^{\mathcal{Q}} \mathbf{a}_q \mathbf{a}_q^\top \text{cov}(U_q(u), U_q(u)) \tag{3.2.10}$$

where $\mathbf{a_q} = [a_{q,1}, ..a_{q,D}]$. After substituting we have:

$$\text{cov}(\overline{\mathbf{f}}(u), \overline{\mathbf{f}}(u')) = \sum_{q=1}^{\mathcal{Q}} B_q \kappa_q(u, u'), \tag{3.2.11}$$

where the $B_q = \mathbf{a}_q \mathbf{a}_q^\top$ with size of $D \times D$ is a positive definite (rank=1) matrix called a coregionalization matrix for D outputs, and $\kappa_q(u, u')$ is the covariance function of the underlying process. The kernel in Eq. (3.2.11) is a mixture of $\mathcal{Q}$ different kernels and is sometimes called the Sum Of Separable (SOS) kernels ([2]). The SoS kernel corresponds to an assumption that outputs are from a mixture of $\mathcal{Q}$ latent processes

([72]). Considering all the index points $\mathbf{u}$ and using the Kronecker product, we can write the covariance matrix as:

$$\mathbb{K}(\bar{\mathbf{f}}(\mathbf{u}), \bar{\mathbf{f}}(\mathbf{u})) = \sum_{q=1}^{\mathcal{Q}} B_q \otimes K(\mathbf{u}, \mathbf{u}). \tag{3.2.12}$$

A special case of LMC is when $\mathcal{Q} = 1$, which means that all targets share the same prior kernel. This simplest case is called the Intrinsic Coregionalization Model (ICM) with a covariance matrix as follows:

$$\mathbb{K}(\bar{\mathbf{f}}, \bar{\mathbf{f}}') = B \otimes K(\mathbf{u}, \mathbf{u}') \tag{3.2.13}$$

Fig. 3.2 shows the regression using ICM for two correlated outputs.

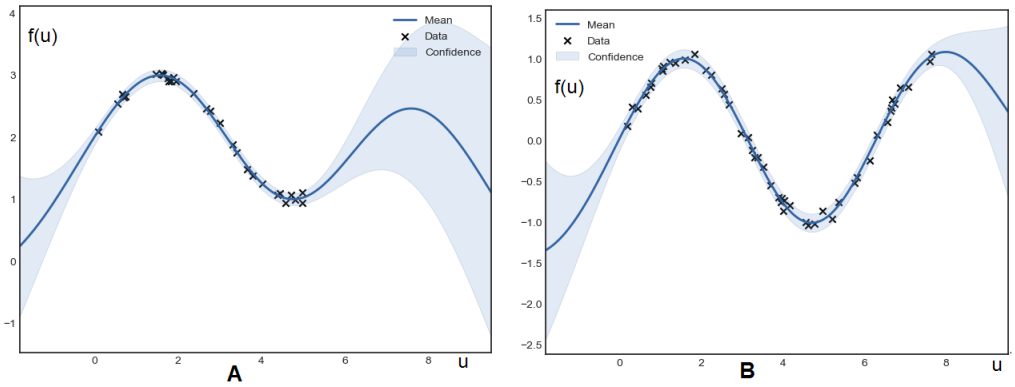


Figure 3.2: MOGPR can predict the part of A that is not observed, using the visible parts in B.

In isotopic cases, when the outputs share the same inputs, we can stack outputs and compute the non-zero off-diagonal covariance matrix as Eq. 3.2.13. Expanding the kernel on the coregionalization matrix creates a prediction process that is similar

to a normal GP regression.

$$\begin{bmatrix} \mathbf{f_1} \\ \mathbf{f_2} \\ \vdots \\ \mathbf{f_D} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} K(\mathbf{f_1}, \mathbf{f_1}) & \dots & K(\mathbf{f_1}, \mathbf{f_D}) \\ \vdots & \ddots & \\ K(\mathbf{f_D}, \mathbf{f_1}) & \dots & K(\mathbf{f_D}, \mathbf{f_D}) \end{bmatrix} \right), \tag{3.2.14}$$

Learning Process: For the stacked training data $\{\mathbf{u}_d, \mathbf{z}_d\}_{d=1}^{D}$ finding the hyper-parameters (θ) is similar to a normal GP, only we need to add extra parameters for the coregionalization matrix B. The approach normally taken is to minimize the log-likelihood of marginal likelihood via optimization techniques such as maximizing marginal likelihood or the gradient descent method ([69]; [16]).

3.2.3 Spatio Temporal Gaussian Process (STGP)

One type of GP that can model the evolution of a process in both space and time is called STGP. Since the information about the shape of the target in GP is summarized in a covariance function, we can use STGP to model the evolution of the covariance function and the shape of the target ([1]).

In fact, an STGP is a stochastic process model for systems evolving in both space and time, specified by:

$$f(u,t) \sim \mathcal{STGP}(\mu(u,t), \kappa(u,u';t,t')), \tag{3.2.15}$$

$$z_i = \mathcal{H}_i f(u_i, t_i) + v_i, \tag{3.2.16}$$

where u_i is the spacial index points or inputs, t is the time, and $\mu(u,t)$ and $\kappa(u,u';t,t')$ represent the mean and the covariance function of the STGP model, respectively. The STGP regression is similar to the normal GP explained in Eqs. 3.2.7 by when the temporal information is augmented in the covariance function. Using normal GP needs regression in both domains; the time complexity of determining an STGP regression on a model with T time steps and N input points will be $O(N^3T^3)$.

Recursive STGP

In many real-time applications, cubic time complexity is a problem. GP regression can be represented as a state space model, as shown in Eq. (4.2.1). To use linear methods and the Markov property to solve this differential equation we can augment the states with a sufficient number of time derivatives, as shown in Eq. (4.2.2) ([74]).

$$a_n \frac{d^n f(t)}{dt^n} + ... + a_1 \frac{df(t)}{dt} + a_0 f(t) = \mathbf{w}(t) \tag{3.2.17}$$

$$d\mathbf{f}(t) = \mathcal{A}\mathbf{f}(t) + \mathbf{L}\mathbf{w}(t), \tag{3.2.18}$$

where $\mathbf{f} = [f, df/dt, ...d^{s-1}f/dt^{s-1}]^\top$ and $\mathbf{w}(t)$ is a Wiener process.

Bayesian Kalman Filtering is a linear way to solve this group of the differential equation (4.2.2). However, filtering only provides the forward-time posteriors of the process: to get the full posterior a linear smoother must be used.

[74] has shown that, by considering some constraints in the kernel function, a covariance function can be modeled by an inverse Fourier transform of the power spectral density of the temporal kernel. They have shown that the corresponding

state-space model could make the exact solution as a kernel base GP using a recursive filter combined with a linear Rauch-Tung-Strieble (RTS) Smoother. To simplify the result, the covariance kernel was considered separable, because the correlation structure obeys different dependencies through space and time.

$$\kappa(u, u'; t, t') = \kappa_u(u, u')\kappa_t(t, t'), \tag{3.2.19}$$

where $\kappa_u(u, u')$ and $\kappa_t(t, t')$ are the spatial and temporal covariance functions, respectively. To model the evolution of the covariance using a state-space model, the following conditions are necessary:

1. The temporal covariance kernel needs to be stationary.

   $\kappa_t(t, t') = \kappa_t(t - t')$

2. Power Spectral Density (PSD) of the process or covariance function is or is approximated to a rational form.

   $S(w_u, w_t) = \mathcal{F}[\kappa(u, u'; t, t')] = \frac{\text{constant for } w_t}{\text{polynomial in } w_t^2}$,

   where $S(.)$ represents the PSD of the process, w_u and w_t represent the Fourier frequency in the u and t domains, respectively, and $\mathcal{F}[.]$ denotes the Fourier transform.

3. The order of the temporal PSD is a multiple of 2.

   $S(w_u, w_t) = \frac{q_t S(w_u)}{S(w_t^2)}$, where q_t denotes the spectral density of the white noise process driving the temporal dynamics.

4. The spectral factorization of PSD gives a stable transfer function:

   $S(w_u, w_t) = G(iw_t)S(w_u)G(-iw_t)$,

where $G(iw_t)$ and $G(-iw_t)$ represent the unstable and the stable transfer function components, respectively, and iw_t represents the complex Fourier frequency.

As a result, the corresponding GP covariance matrices are also separable ([1]). Because the transfer function $G(iw_t)$ does not contain the variable w_u at all, the feedback operator $\mathcal{A}_t$ will actually be just an ordinary matrix and the only spatial coupling will come from the spectral density of $\mathbf{w}_t(\mathbf{u}, t)$. Under the above conditions, the temporal stochastic process can be equivalently represented by an infinite dimensional dynamic system, given below ([74]):

$$\frac{\partial \mathbf{f}(\mathbf{u}, t)}{\partial t} = \mathcal{A}_t \mathbf{f}(\mathbf{u}, t) + \mathbf{L}_t \mathbf{w}_t(\mathbf{u}, t) \tag{3.2.20}$$

$$\mathcal{A}_t = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_0 & \dots & -a_{s-2} & a_{s-2} \end{bmatrix}, \quad \mathbf{L}_t = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{3.2.21}$$

where $\mathbf{f}(\mathbf{u}, t) = [f, \partial f/\partial t, ...\partial f^{s-1}/\partial t^{s-1}]^\top$ is a state at time t. The function $\mathbf{f}(\mathbf{u}, t)$ consists of the function and a suitable number of its time derivatives, $\mathcal{A}_t$ is the continuous state transition with a matrix form, and $\mathbf{L}_t$ represents the noise effect and $\mathbf{w}_t(\mathbf{u}, t)$ is a zero mean continuous time white noise process.

This model is now an infinite-dimensional Markovian type of model that allows for linear time inference solvable with a Bayesian filter and smoother. In most applications, measurements arrive at discrete times. A finite collection of discrete spatial points of interest $\mathbf{u} = \{u_i\}_{i=1}^N$ in Hilbert space is used so an evolving model for extent

of the target can be written as a discrete form:

$$\mathbf{f}(\mathbf{u}, t_k) = \mathbf{F}_k \mathbf{f}(\mathbf{u}, t_{k-1}) + \mathbf{L}_k \mathbf{w}_k(\mathbf{u}) \tag{3.2.22}$$

$$\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k(\mathbf{u}, \mathbf{u}'; T_s)), \tag{3.2.23}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{f}(\mathbf{u}, t_k) + \mathbf{v}_k \tag{3.2.24}$$

$$\mathbf{v}_k \sim \mathcal{N}(0, R) \tag{3.2.25}$$

The corresponding states would be a finite matrix (a multi-output model for input t_k) and the dynamics are uncoupled.

$$\mathbf{f}(t_k) = [[f(u_1, t_k), \dot{f}(u_1, t_k), ...], [f(u_2, t_k), \dot{f}(u_2, t_k), ...], [f(u_{N^f}, t_k), \dot{f}(u_{N^f}, t_k), ...]]^\top \tag{3.2.26}$$

To drive the process we can expand temporal model matrices using the Kronecker product. The discrete transition matrix, $\mathbf{F}_k = I \otimes e^{\mathcal{A}T_s}$, $T_s = t_k - t_{k-1}$, $\mathbf{L}_k$ and $\mathbf{Q}_k(\mathbf{u}, \mathbf{u}'; T_s)$, which is the separable process noise covariance matrix for N index points with a spatial structure.

$$\mathbf{Q}(\mathbf{u}, \mathbf{u}'; T_s) = K(\mathbf{u}, \mathbf{u}')[I_N \otimes \tilde{\mathbf{Q}}(T_s)], \tag{3.2.27}$$

$$\tilde{\mathbf{Q}}(T_s) = \int_0^{T_s} \mathbf{F}(T_s - \tau) \mathbf{L} q_t \mathbf{L}^\top \mathbf{F}(T_s - \tau)^\top d\tau \tag{3.2.28}$$

This integral can be solved using matrix fraction decomposition ([74]). If the model (and the corresponding covariance function) is stationary, the SDE has a stationary state $f_\infty \sim \mathcal{N}(0, P_\infty)$. The stationary state corresponds to the state that the model stabilizes to infinity. It can be represented by the stationary covariance of $f(t)$ that is the solution to:

$$\frac{d\mathbf{P}_\infty}{dt} = \mathbf{F}\mathbf{P}_\infty + \mathbf{P}_\infty\mathbf{F}^\top + \mathbf{L}q_t\mathbf{L}^\top = 0. \tag{3.2.29}$$

The stationary state is invariant to the choice of input location and describes the state the process defaults to. Therefore, for stationary models the initial (prior) state is given by the stationary state covariance, $\mathbf{P}_0 = \mathbf{P}_\infty$. Provided that $\mathbf{P}_\infty$ exists and is known, the following relation is a computationally lightweight way to solve the integral (3.2.28) and obtain the process noise covariance:

$$\mathbf{Q}_k = \mathbf{P}_\infty - \mathbf{F_k}\mathbf{P}_\infty{\mathbf{F_k}}^\top. \tag{3.2.30}$$

The initial covariance in this case is the Kronecker product of a spatial covariance matrix with elements of $\kappa(u_i, u_j)$ and temporal initialized covariance, $\mathbf{P}_0 = K \otimes \mathbf{P}_{0,t}$. This model allows a sequential inference using a multi-variant recursive Bayesian filter and smoother with computation order of $O(N^3T)$, where N is the state dimension and T is time.

3.2.4 Infinite Kalman Filtering and Smoothing

Solving the infinite dimension stochastic dynamic, Eq. (3.2.20) needs an infinite dimension Kalman filtering and smoothing process. In discrete cases for a finite

collection of spatial points of interest, we can estimate the states as a multivariate Kalman filter with states $\mathbf{x}_k = \mathbf{f}(\mathbf{u}, t_k)$, using Eq. (3.2.26), and covariance of process noise expands on the kernel matrix $\mathbf{P} = K \otimes \mathbf{Q}_t$. The recursion using the GP priors, which is encoded into the power spectral density of covariance function, results in the filter matrices. The state prediction and update are similar to a normal Kalman filter ([77]):

The prediction step is:

$$\mathbf{x}_{k+1|k} = \mathbf{F}_k \mathbf{x}_{k|k} \tag{3.2.31a}$$

$$\mathbf{P}_{k+1|k} = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^\top + \mathbf{Q}_k \tag{3.2.31b}$$

The update step is:

$$\mathbf{v}_k = \mathbf{z}_k - \mathbf{H}_k \mathbf{x}_{k+1|k} \tag{3.2.32a}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k+1|k} \mathbf{H}_k^\top + \mathbf{R}_k \tag{3.2.32b}$$

$$K_k = \mathbf{P}_{k+1|k} \mathbf{H}_k^\top \mathbf{S}_k^{-1} \tag{3.2.32c}$$

$$\mathbf{x}_{k+1|k+1} = \mathbf{x}_{k+1|k} + K_k \mathbf{v}_k \tag{3.2.32d}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - K_k \mathbf{S}_k K_k^\top \tag{3.2.32e}$$

Fixed Lag RTS Smoother: In state estimation, smoothing is defined as a process

where the current measurements are used to improve estimates of past states. In fact, smoothing estimates backwards posterior density. The general case of smoothing that can be used in real-time applications is called a fixed lag smoother. It smooths the trajectory for an interval of s states.

Fixed interval smoothing requires a backward iteration after the (forward) filtering. The results of filtering, $\mathbf{x}_{k|k}, \mathbf{x}_{k+1|k}, \mathbf{P}_{k|k}, \mathbf{P}_{k+1|k} \quad k = k, k-1, ..., k-s$, need to be stored to use in backward iteration.

In fixed interval smoothing, when the smoothing depth, (s) is fixed, the smoother gain $\mathbf{G}_k$, the smoothed state $\tilde{x}_k$, and the state error covariance $\tilde{P}_k$ are recursively estimated using the Rauch-Tung-Strieble (RTS) recursion ([70]; [6]):

$$\mathbf{G}_k = \mathbf{P}_{k|k}\mathbf{F}^\top(\mathbf{P}_{\mathbf{k+1|k}})^{-1} \tag{3.2.33a}$$

$$\tilde{\mathbf{x}}_{k|\mathbb{N}} = \mathbf{x}_{k|k} + \mathbf{G}_k[\tilde{\mathbf{x}}_{k+1|N} - \mathbf{x}_{k+1|k}] \tag{3.2.33b}$$

$$\tilde{\mathbf{P}}_{k|\mathbb{N}} = \mathbf{P}_{k|k} + \mathbf{G}_k[\tilde{\mathbf{P}}_{k+1|N} - \mathbf{P}_{k+1|k}]\mathbf{G}_k^\top \tag{3.2.33c}$$

The smoother is initialized at the current time step k as $\tilde{\mathbf{x}}_{k|\mathbb{N}} = \mathbf{x}_{k|k}$ and $\tilde{\mathbf{P}}_{k|\mathbb{N}} = \mathbf{P}_{k|k}$. The result of smoothing function $f(.)$ at time k is summarized in the mean and covariance $\tilde{\mathbf{x}}_{k|\mathbb{N}}$ and $\tilde{\mathbf{P}}_{k|\mathbb{N}}$, conditioned to the measurements $\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_{\mathbb{N}}$.

3.3 Measurement Preparation and Lane-Line Clustering Using K-JPDACF

Preparation of measurements in ETT primarily involves extracting and grouping features that belong to each target. In this work on lane-line detection, we used edge and color features, similar to ([65]; [37]). Finding the ROI and removing the other objects from the scene is crucial. In some papers, supervised methods like the Neural Network or Cascade Classifier ([30]) is used to detect roads and cars and handle clutter ([60]). To eliminate the perspective effects of the camera, normal Inverse Perspective Mapping (IPM) / Birds Eye View is used ([67]; [3]). This transformation makes the dependency of lane-lines easier to model; otherwise, a more complex coregionalization model needs to be used. Fig. 3.3 shows the steps for preprocessing the frame before clustering.

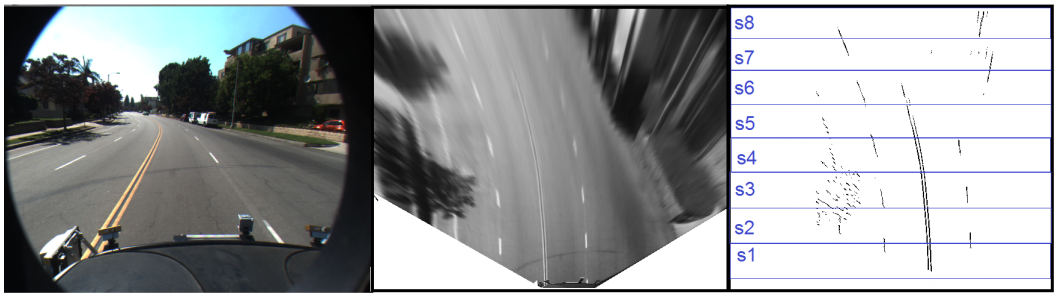


Figure 3.3: Preprocessing of images, including (B) IPM transformation and (C) edge-color extraction and sectioning.

In grouping and clustering the features, we introduced a novel kernel-based estimation approach using a joint probabilistic data association coupled filter called K-JPDACF. The idea behind K-JPDACF is based on the fact that Bayesian filters

like KF can be modeled with GP models ([74]). As we know, a characteristic of GP is its ability to be summarized in a kernel function. The parameters of the kernel function can be trained from a limited amount of trained data. Because JPDACF is a Bayesian filter, it can be considered to be kernel-based, which allows us to model the motion model and dependency of tracks using trained data (see section 3.4.1).

In fact, the K-JPDACF is a semi-supervised version of JPDACF ([8]) with a dynamic and dependency model learnable from the trained data. A recursive JPDACF is similar to the Probabilistic Data Association Filter (PDAF), itself a statistical approach to the problem of measurement to target assignment in target tracking algorithms. In cases when clutter or miss-detection are not present, the most likely measurements to a target can be chosen using a Nearest Neighbor method. Unlike the Nearest Neighbor, which makes a hard decision, the PDAF makes a soft decision when estimating an expected value, which is the minimum mean square error (MMSE) estimate ([6]). The PDAF is designed to track a single target or multiple targets that are not interfering or sharing measurements. In contrast, the Joint Probabilistic Data Association Filter (JPDAF) can handle multiple targets that may share measurements. In the JPDAF we assumed that the target states and thus the target-originated measurements are independent. On the other hand, for the dependent targets, a statistical dependence of their estimation can be taken into account by calculating the state cross-covariances. The resulting algorithm is called the Joint Probabilistic Data Association Coupled Filter (JPDACF) ([7]; [8]; [79]; [13]). The motion model in JPDACF is a parametric prior information that must be considered precisely. More details about the proposed K-JPDACF can be found in chapter 4.2.

In this paper, we introduce K-JPDACF after adding track management to estimate the cross-covariance of each track; it is used to cluster measurements belonging to each lane-line. Filter parameters and matrices come from the factorization of power spectral density of a trained kernel function and coregionalization matrix B, similar to section 3.5. Note that the K-JPDACF can be used to modeling lane branch and merge situations as well, when targets share similar measurements. The result of K-JPDACF is a set of tracks including states, covariance, and validated measurements. Each cluster considers the overall validated measurements for each track.

Polynomial, Wiener velocity ([6]) or squared exponential ([69]) kernels can be used to model lane markings. The state-space representation for these kernels is similar to that in [74].

The clustering procedure starts by sectioning measurements according to their horizontal positions (Z^k), as shown in Fig. 3.3 part (C). The purpose of sectioning is to decrease the number of clustering steps in real-time applications. The section sizes depend on the desired precision and height of the image. We can initialize tracks based on prior information and the first horizontal group of measurements. In each step, we use K-JPDACF to estimate marginal posterior density, mean, and covariance to identify the most likely measurements belonging to each track.

For track management, a method identical to the one used in [62] is implemented. The dynamic model for the coupled targets is as follows:

$$\bar{x}_{k+1} = \bar{F}_k \bar{x}_k + \bar{w}_k, \quad \bar{w}_k \sim \mathcal{N}(0, \bar{Q}_k), \tag{3.3.1}$$

$$\bar{z}_k = \begin{cases} FA_k & \text{False alarm} \\ \bar{H}_k \bar{x}_k + \bar{v}_k, \quad \bar{v}_k \sim \mathcal{N}(0, \bar{R}_k) & \text{Otherwise,} \end{cases} \tag{3.3.2}$$

where the coupled vectors/matrices with D number of targets at time k and coregionalization matrix B_D are as follows:

$$\bar{x}_k = \left[x_k^1, x_k^2, ..., x_k^D \right]^\top, \quad \bar{P}_k = B_D \otimes P_k \tag{3.3.3}$$

$$\bar{F}_k = I_D \otimes F_k, \quad \bar{Q}_k = B_D \otimes Q_k \quad \bar{H}_k = I_D \otimes H_k \quad \bar{R}_k = I_D \otimes R_k \tag{3.3.4}$$

As can be seen in the algorithm (4), the procedure starts by receiving measurements sequentially from each section. In the case when no track exists, we create new tracks with the first measurements, using the One-Point initialization method ([5]). Similar to the procedure in [62], if still some non-associated measurements exist after the update step, we group them as potential new tracks. The result of our algorithm is that all tracks and their associated measurements satisfy the minimum thresholds. Details and functions of the algorithm are explained in section 4.2. In Fig. 3.4, we can see the result of the algorithm for estimating covariance and finding measurements belonging to each lane-line.

Algorithm 4 Clustering using K-JPDACF.

▷ Inputs: $\mathbf{Z}, \kappa(.), B$
▷ Output: $\Gamma = \{(\Gamma_t.x, \Gamma_t.P, \Gamma_t.\mathbf{z}_t)\}_{t=0}^{D}$
$\Gamma = \{\ \}$ # Initialize tracks
for k in range(Sections) **do**
  $\mathbf{z}_k \leftarrow$ getMeasInSection$(\mathbf{Z}, k)$
  if $\Gamma.size > 0$ **then**
    $\bar{F}, \bar{H}, \bar{x}, \bar{P}, \bar{Q}_c \leftarrow$ MakeVectorMatrixes $(\Gamma_k, \kappa(.), B)$
    $\hat{\bar{x}}, \hat{\bar{P}}, \hat{\bar{z}} \leftarrow$ Predict $(\bar{F}, \bar{H}, \bar{x}, \bar{P}, \bar{Q}_c)$
    $\bar{V}\mathbf{z} \leftarrow$ ValidatedMeasurements $(\hat{\bar{x}}, \hat{\bar{P}}, \hat{\bar{z}}, \mathbf{z}_k)$
    $\Gamma_k \leftarrow$ K-PDA-Update $(\hat{\bar{x}}, \hat{\bar{P}}, \bar{V}\mathbf{z})$
    $NonAssMeas \leftarrow \{\mathbf{z} - \bar{V}\mathbf{z}\}$
    $\Gamma \leftarrow$ AddNewTracks $(\{\Gamma\}, NonAssMeas)$
  else
    $\Gamma \leftarrow$ AddNewTracks $(\{\Gamma\}, \mathbf{z})$
  end if
end for
return Γ

3.4 Problem Formulation for MO-STGP-KF

Common multiple lane detection algorithms based on line or curve detection lack flexibility in urban traffic when a part of a lane-line is missing. In this situation, we can still predict the unobserved parts of one lane-line using a dependency with other visible lane-lines. In this section, we will show how the dependency of lane markings can be modeled using the MOGP framework and coregionalization matrix. Our approach uses a combined variation of models represented in [1] and [82] for multiple outputs.

3.4.1 Extended Target Model

In our application, extended targets are defined as dependent straight or curved lines that are modeled with a GP framework. The GP framework of extended targets includes a limited number of index points $\mathbf{u}^f$ and the latent function $\mathbf{f}(\mathbf{u}^f)$. We

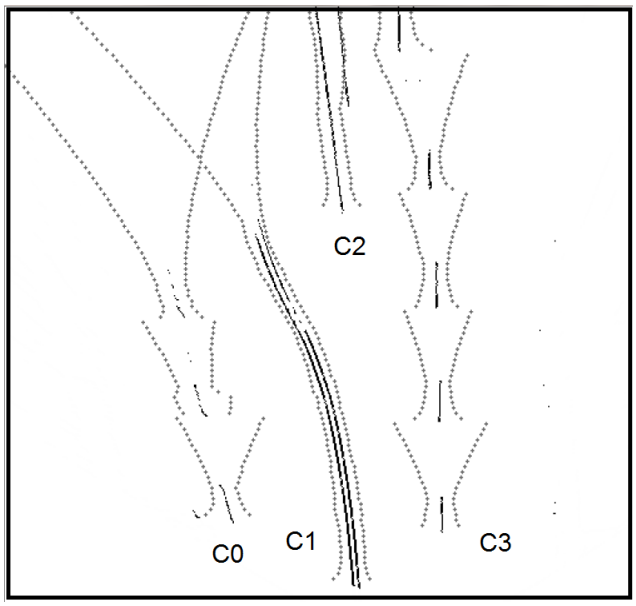


Figure 3.4: Use of the covariance prediction of tracks to cluster the measurements belonging to each lane-line.

use the MOGP framework and ICM model of coregionalization to model dependency between targets. We stack all the latent functions and consider them a single state vector. We also change the notation of latent function $\mathbf{f}(\mathbf{u}^f)$ to $\mathbf{x}$ to be more similar to state-space model representation. The dynamics of extent are designed as separable kernels, which satisfy conditions described in section 3.2.3.

$$\kappa(u, u'; t, t') = \kappa_u(u, u')\kappa_t(t, t'), \tag{3.4.1}$$

where $\kappa(u, u'; t, t')$ represents the spatio-temporal covariance kernel, $\kappa_u(u, u')$ represents the spatial covariance kernel, and $\kappa_t(t, t')$ represents the temporal covariance kernel. A polynomial or RBF ([69]) covariance kernel can be used to model $\kappa_u(u, u')$. For the temporal covariance kernel $\kappa_t(t, t')$, we primarily use the Whittle-Matern

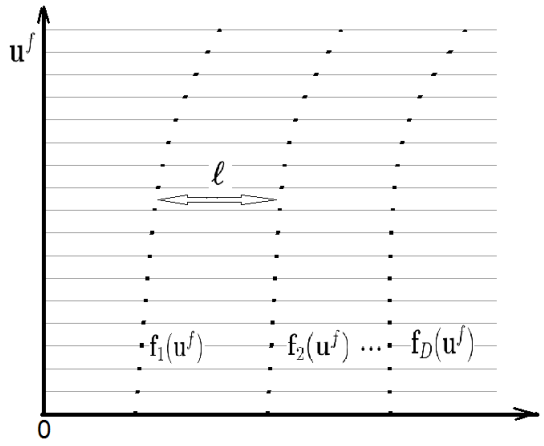


Figure 3.5: MOGP model for three lane-lines.

kernel ([74]). The proposed model is converted to a transfer function form and subsequently to an equivalent state space representation using steps given in subsection 3.2.3. The evolution of stacked extended targets states is modeled as follows:

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{F}}_k \bar{\mathbf{x}}_k + \bar{\mathbf{w}}_k \tag{3.4.2}$$

$$\bar{\mathbf{w}}_k \sim \mathcal{N}(0, \bar{\mathbf{Q}}_k), \tag{3.4.3}$$

$$\bar{\mathbf{x}}_k = \begin{bmatrix} \mathbf{f}_1(\mathbf{u}^f) \\ \mathbf{f}_2(\mathbf{u}^f) \\ \cdots \\ \mathbf{f}_D(\mathbf{u}^f) \end{bmatrix} \tag{3.4.4}$$

where $\bar{\mathbf{x}}_k$ is stacked latent outputs in which each corresponds to N distinct index points $\mathbf{u}^f = \{u_1^f, u_2^f, ...u_N^f\}$. The stacked version of the state transition matrix is

denoted by $\bar{\mathbf{F}}_k$, and $\bar{\mathbf{w}}_k$ is a zero-mean white Gaussian noise with covariance $\bar{\mathbf{Q}}_k$ for stacked states. Note that all coordinates are in Cartesian space with a single global origin, the same as Fig. 4.1. The cardinality and value of index points are related to the precision and shape of the extended target. For lane markings, input indexes would be N distinct points uniformly distributed across the ROI height. In Fig. 4.1 the model is represented for three lane-lines. Overall, the MOGP model can be represented as follows:

$$\bar{\mathbf{f}}(\mathbf{u}^f) \sim \mathcal{MOGP}(\bar{\mu}(\mathbf{u}^{\mathbf{f}}), \bar{\mathbf{K}}(\mathbf{u}, \mathbf{u}')), \tag{3.4.5}$$

where $\bar{\mu}$ and $\bar{\mathbf{K}}$ are mean and kernel matrices for the stacked states. Under the Gaussian assumption of $\bar{\mathbf{Q}}_k$, the propagated distribution of the state in Eq. (4.3.2) remains Gaussian. We train the model similarly to a normal GP, minimizing log marginal likelihood of measurements ([83]). The training results include the kernel parameters and coregionalization matrix. We assume that the first frame includes enough information for optimizing hyperparameters; otherwise we can use more frames or prior information coming from offline training.

3.4.2 Measurement Model

The result of the lane-line clustering is a set of measurements or scattering points D belonging to the D lane-line, $\bar{\mathbf{z}} = [\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_D]^\top$. Usually, the cardinality and position of the measurements are both unknown and time varying. In fact, the index points of measurements $\bar{\mathbf{u}} = [\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_D]^\top$ are not isotopic. In addition, in some applications the measurement origin uncertainty (i.e., missed detection and clutter) must be taken into account. In section (3.3) we used the K-JPDACF algorithm for

clustering; this algorithm will remove some outliers in a naive way. Considering the ICM model for coregionalization (see section 3.2.2) given a set of N distinct input indexes $\mathbf{u}^f = [u_1^f, u_2^f, ...u_N^f]^\top$ for states, we want to estimate the D set of outputs or states, $\bar{\mathbf{x}} = [\mathbf{f}_1(\mathbf{u}^f), \mathbf{f}_2(\mathbf{u}^f), ..., \mathbf{f}_D(\mathbf{u}^f)]^\top$, corresponding to these D sets of noisy measurements $\bar{\mathbf{z}}$. the measurement model is given by:

$$\bar{\mathbf{z}}_k(\bar{\mathbf{u}}_k) = \bar{\mathbf{H}}_k^f(\bar{\mathbf{u}}_k)\bar{\mathbf{x}}_k + \mathbf{v}_k, \qquad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k^f), \tag{3.4.6}$$

where $\bar{\mathbf{H}}_k^f(\bar{\mathbf{u}}_k)$ is a measurement transition matrix and $\mathbf{v}_k$ is zero mean Gaussian measurement noise. In order to use the same prediction model as a normal GP regression we need to stack clustered measurements vertically into $\bar{\mathbf{z}}$. For simplification, we add another dimension to the input index $\bar{\mathbf{u}}$, which represents the cluster ID. For example, for cluster number d the index points are $\mathbf{u}^d = [[u_1^d, d], [u_2^d, d], ..., [u_{N_d}^d, d]]^\top$, and the result of stacking is as follows:

$$\bar{\mathbf{u}} = \begin{bmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \\ ... \\ \mathbf{u}^D \end{bmatrix} = \begin{bmatrix} u_1^1 & 1 \\ \vdots & \vdots \\ u_{N_1}^1 & 1 \\ ... & ... \\ u_1^D & D \\ \vdots & \vdots \\ u_{N_D}^D & D \end{bmatrix} \tag{3.4.7a}$$

$$\bar{\mathbf{f}} = \begin{bmatrix} \mathbf{f}_1(\mathbf{u}^1) \\ \mathbf{f}_2(\mathbf{u}^2) \\ \cdots \\ \mathbf{f}_D(\mathbf{u}^D) \end{bmatrix} \tag{3.4.7b}$$

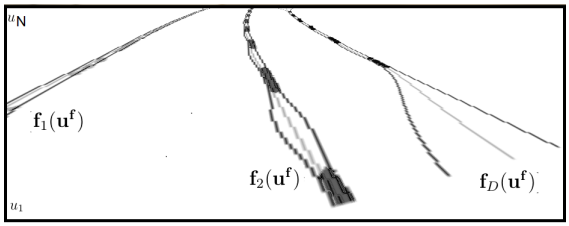


Figure 3.6: MOGP model for lane markings; $\mathbf{u}^f$ and $\mathbf{f}_d(\mathbf{u}^f)$ are input indexes and corresponding output.

The measurement likelihood is given for the D set of measurements according to the joint distribution probability density of GP.

$$p(\bar{\mathbf{z}}_k|\bar{\mathbf{f}}) \sim \mathcal{N}(\bar{\mathbf{z}}_k; \bar{\mathbf{H}}_k^f(\bar{\mathbf{u}}_k)\bar{\mathbf{f}}, \bar{\mathbf{R}}^f(\bar{\mathbf{u}}_k)) \tag{3.4.8}$$

Measurement transition matrix $\bar{\mathbf{H}}_k^f$ and measurement covariance $\bar{\mathbf{R}}^f$ will derive from Eqs. (3.5.13) and (3.5.14) in the next section. Based on the extended target model and the measurement model discussed above, we can predict unobserved response values $\mathbf{f}_d(\mathbf{u}^f)$ at index points $\mathbf{u}^f$ using not only measurements belonging to that target but *all* observed measurements (see Fig. 3.6. The objective then is to simultaneously estimate the state kinematics of lane markings from the total measurements set in the stream of frames.

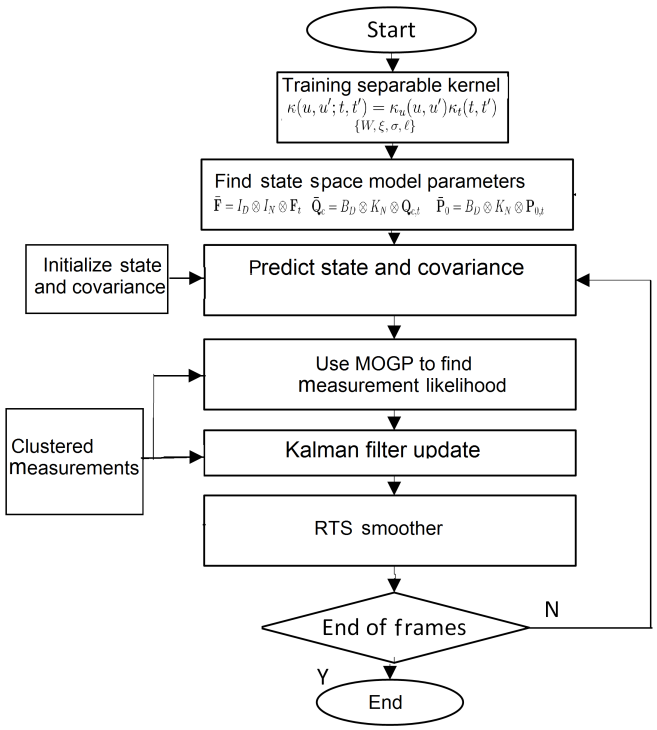


Figure 3.7: Geproneral pipeline.

3.5 Recursive MO-STGP-KF

Because most ETT applications require real-time processing, the estimation of the state-space model and measurement likelihood derived above need to be done recursively. A real-time recursive filter equivalent to a full GP regression has also been proposed in [82], [1], and [39]. In this section, we will show how we can change recursive STGP-KF for a single extended target, introduced in [1], to track multiple dependent extended targets. The basic framework for this is shown in Fig. 3.7. The main contributions of MO-STGP-KF address two aspects of the technique. First, it expands the recursive STGP-KF for tracking multiple extended targets. Second,

it implements dependency between targets, incorporating it into the process noise covariance. Similar to the STGP-KF, we consider separable kernels for the temporal and spatial parts of the model.

To find the state-space model, we first factorize the power spectrum density of the temporal covariance function. Most of the GP priors can be represented with a simple and convenient form of the linear and time-invariant State-Space Differential Equation (SDE) ([74]). Given a model of the form (4.3.2), and a recursive Kalman filter and smoother (see section 3.2.4), we make a linear estimation of state and covariance at each time step. A fixed lag smoother can be enrolled in real-time cases. Similar to flowchart 3.7, the overall recursive MO-STGP regression can be reformulated as follows:

1. Use the temporal kernel and priors to predict the extended targets' states and covariances.

2. Use the spatial kernel to compute measurement likelihood and predicted measurements.

3. Update the state and covariance with new observations, using Kalman filtering.

4. Estimate the full posterior, using the fixed lagged RTS smoother.

Temporal Covariance Kernel: Most temporal covariance functions either have an exact linear time-invariant state-space representation or can be approximated with such a model ([77]). In our application, we used the Matérn covariance function for the temporal part. The general form of Matérn is given as:

$$\kappa_t(\tau) = \sigma_t^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\frac{\sqrt{2\nu}}{l}\tau)^{\nu} \kappa_{\nu}(\frac{\sqrt{2\nu}}{l}\tau), \tag{3.5.1}$$

where σ_t^2 is a magnitude scale hyperparameter, l is the characteristic length-scale, and ν is a smoothness parameter. The modified Bessel function of the second kind is denoted by $\kappa_\nu(.)$. For this class, the corresponding process is s-times differentiable if $\nu > s$ ([74]). For example, if $\nu = 1/2$, the covariance function would be the same as the normal exponential with the following continuous dynamic information:

$$\kappa_t(\tau)_{1/2} = \sigma^2 exp(-\frac{\tau}{l}) \tag{3.5.2}$$

$$\mathcal{A}_t = -\frac{1}{l}, \quad \mathbf{L}_t = 1, \quad \mathbf{P}_\infty = \mathbf{P}_0 = \sigma^2, \quad \mathbf{Q}_c = 2\sigma^2/l, \tag{3.5.3}$$

where $\frac{1}{l} \geq 0$ will determine the speed of the dynamics, which in some papers is called a ''forgotten factor" ([82]). In this work, we considered $\nu = 3/2$, with a continuous and once differentiable process. In this case, the covariance function is simplified to:

$$\kappa_t(\tau)_{3/2} = \sigma^2(1 + \frac{\sqrt{3}\tau}{l})exp(-\frac{\sqrt{3}\tau}{l}) \tag{3.5.4}$$

The continuous system matrix and the noise effect vector of the corresponding state-space model are derived as follows:

$$\mathcal{A}_t = \begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix}, \quad \mathbf{L}_t = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{P}_\infty = \mathbf{P}_0 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \lambda^2\sigma^2 \end{bmatrix}, \tag{3.5.5}$$

where $\lambda = \frac{\sqrt{3}\tau}{l}$ and the spectral density of the Gaussian white noise process $w(t)$ is $\mathbf{Q}_c = 4\lambda^3\sigma^2$. The measurement model matrix is $\mathbf{H} = [1 \;\; 0]$. The state of each target

can be defined as:

$$\mathbf{f}_i = [[f(u_1)_k, \dot{f}(u_1)_k], ..., [f(u_D)_k, \dot{f}(u_D)_k]] \tag{3.5.6}$$

Recall that the dynamics still stay uncoupled in the temporal domain: we just need to expand dimension across the state dimension and number of targets D. In a discrete case, $\mathbf{H}_k = \mathbf{H}_t, \quad \mathbf{L}_k = \mathbf{L}_t, \quad \mathbf{F}_k = e^{\mathcal{A}_t T_s}$, so we have:

$$\bar{\mathbf{F}}_k = I_D \otimes I_N \otimes \mathbf{F}_k \quad \bar{\mathbf{L}}_k = I_D \otimes I_N \otimes \mathbf{L}_t \quad \bar{\mathbf{H}}_k = I_D \otimes I_N \otimes \mathbf{H}_t \tag{3.5.7}$$

Subscript t denotes the temporal continuous model matrices in Eq. (4.3.7). The spectral density and the initialized state covariance have spatial structure.

$$\bar{\mathbf{Q}}_k = \bar{\mathbf{K}} \otimes (\mathbf{P}_\infty - \mathbf{F}_k \mathbf{P}_\infty \mathbf{F}_k^\top), \quad \bar{\mathbf{P}}_0 = \bar{\mathbf{K}} \otimes \mathbf{P}_{0,t} \tag{3.5.8}$$

In cases where outputs are correlated and considering a GP prior over the latent function $\mathbf{f}(\mathbf{u})$, as we saw in section 3.2.2, the coupling can be defined with coregionalization matrix B. In the ICM case, the covariance matrix for stacked data is as follows ([2]):

$$\bar{\mathbf{K}}(\mathbf{u}, \mathbf{u}') = B \otimes K(\mathbf{u}, \mathbf{u}'), \tag{3.5.9}$$

where B is a positive semidefinite rank-one coregionalization matrix that specifies the inter-output similarities, and $K(\mathbf{u}, \mathbf{u}')$ is a covariance matrix of the underlying process. In this paper the two individual inputs (u, u') spatial kernel can be a polynomial

or Radial Basis Function (RBF) kernel.

$$\kappa(u,u')_{pol} = (\sigma^2 + u.u')^p, \quad \kappa(u,u')_{RBF} = \sigma^2 \exp\left(-\frac{1}{2}\frac{(u-u')^2}{\ell^2}\right) \tag{3.5.10}$$

The parameters of the coregionalization matrix include vector W, which is the same size as the output and constant ξ.

$$\mathbf{B} = WW^\top + \text{diag}(\xi) \tag{3.5.11}$$

The Kronecker product of $B \otimes K$ can be written as:

$$\bar{\mathbf{K}}(\bar{\mathbf{f}}, \bar{\mathbf{f}}') = \begin{bmatrix} B_{11}K(\mathbf{u_1}, \mathbf{u_1}') & .... & B_{1D}K(\mathbf{u_D}, \mathbf{u_D}') \\ ... & \ddots & ... \\ B_{D1}K(\mathbf{u_1}, \mathbf{u_1}') & ... & B_{DD}K(\mathbf{u_D}, \mathbf{u_D}') \end{bmatrix} \tag{3.5.12}$$

The covariance of the joint Gaussian process over $\bar{\mathbf{z}}$ is not a diagonal matrix, because of the dependency of the outputs. In fact, the observations of one target can affect the predictions for another target.

Using a stacked version of last states and covariance and transition matrices in Eq. (4.3.9), we can predict state and covariance similar to a normal Kalman filter in section 3.2.4. For the prediction of measurements, we project the states into measurement space using measurement likelihood (3.4.8) with a GP framework. We just need to expand the kernel matrix on the coregionalization matrix B.

$$\bar{\mathbf{H}}_k^f(\bar{\mathbf{u}}_k) = (B \otimes K(\mathbf{u}_k, \mathbf{u}^f))[B \otimes K(\mathbf{u}^f, \mathbf{u}^f)]^{-1}, \tag{3.5.13}$$

$$\bar{\mathbf{R}}^f(\bar{\mathbf{u}}_k) = (B \otimes K(\mathbf{u}_k, \mathbf{u}_k)) + I_M \otimes R - (B \otimes K(\mathbf{u}_k, \mathbf{u}^f))[B \otimes K(\mathbf{u}^f, \mathbf{u}^f)]^{-1}(B \otimes K(\mathbf{u}^f, \mathbf{u}_k)) \tag{3.5.14}$$

Predicted measurements are:

$$\bar{\mathbf{z}}_{k-1|k} = \bar{\mathbf{H}}_k^f(\bar{\mathbf{u}}_k)\bar{\mathbf{f}} + \mathbf{v}_k, \qquad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k^f) \tag{3.5.15}$$

The Kalman filter update part is similar to a normal Kalman filter in section 3.2.4.

3.6 Experiments and Evaluations

We compared the results of our estimation with two main groups of lane detection algorithms: model-based Bayesian ETT and fully supervised learning methods.

3.6.1 Compare to Other Recursive ETT and Batch Models

The estimates of the proposed method are compared with two recursive methods, the GP-KF ([82]) and STGP-KF ([1]), and two batch methods, GP-Regression and MOGP-Regression, over 100 Monte Carlo runs in three scenarios, for the real experiments. Traditional GP-KF and STGP-KF methods are designed for tracking closed nonrigid extended targets, so for tracking lane markings we changed their measurement model from polar to Cartesian, similar to section 4.3.2. The GP-KF dynamic model uses a ''forgotten factor" model with $\alpha = 0.01$, whereas STGP-KF uses a dynamic model result of power spectrum density factorization of the temporal kernel. In our method, we used a similar dynamic as STGP-KF, considering dependency in

the spatial kernel (see section 3.5). Note that in our algorithm and in STGP-KF we used only the filter and not the smoother.

For all the algorithms, we used the same kernels. For the spatial kernel, we used polynomial, and for the temporal kernel, we used Matern 3/2 with constant parameters, $\sigma = 3$ and $l = 10$. The ICM model of coregionalization was considered for coupling in our approach. Parameters σ, l, B were trained similarly to typical GP optimization methods. In this part, we only used the first frame for training. For feature extraction and clustering, the K-JPDACF method was enrolled for all algorithms similarly (see section (3.3). Three different road shapes and situations, numbering a total of $K = 100$ [1] frames are selected from the Caltech data set ([3]).

We evaluated performance, confidence of detection (volume of covariance region), and complexity by comparing our estimates to the true data. The performance evaluation parameters are the positional root mean square errors (RMSE) of the state vector, defined as follows:

$$\text{RMSE}_{k,x}^{f} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_j - \hat{x}_j)^2}, \tag{3.6.1}$$

where $RMSE_{k,x}^{f}$ represents the RMSE of the extend at time step k, x represents the true, $\hat{x}$ represents the estimated value, and N is the number of scattering points. If $RMSE_{k,x}^{f}$ is represented by vector b and those of our estimates by c, then the corresponding percentage improvement d and the mean percentage improvement d_μ

[1] For measurement standard deviation about 1.2, confidence level %95 and precision 0.05 the 100 number of Monte Carlo run for two lane markings in each frame and 96 degree of freedom is statistically significance

are given below. Note that K is the number of frames.

$$d = \frac{b-c}{b}, \quad d_\mu = \frac{d}{K} \times 100 \tag{3.6.2}$$

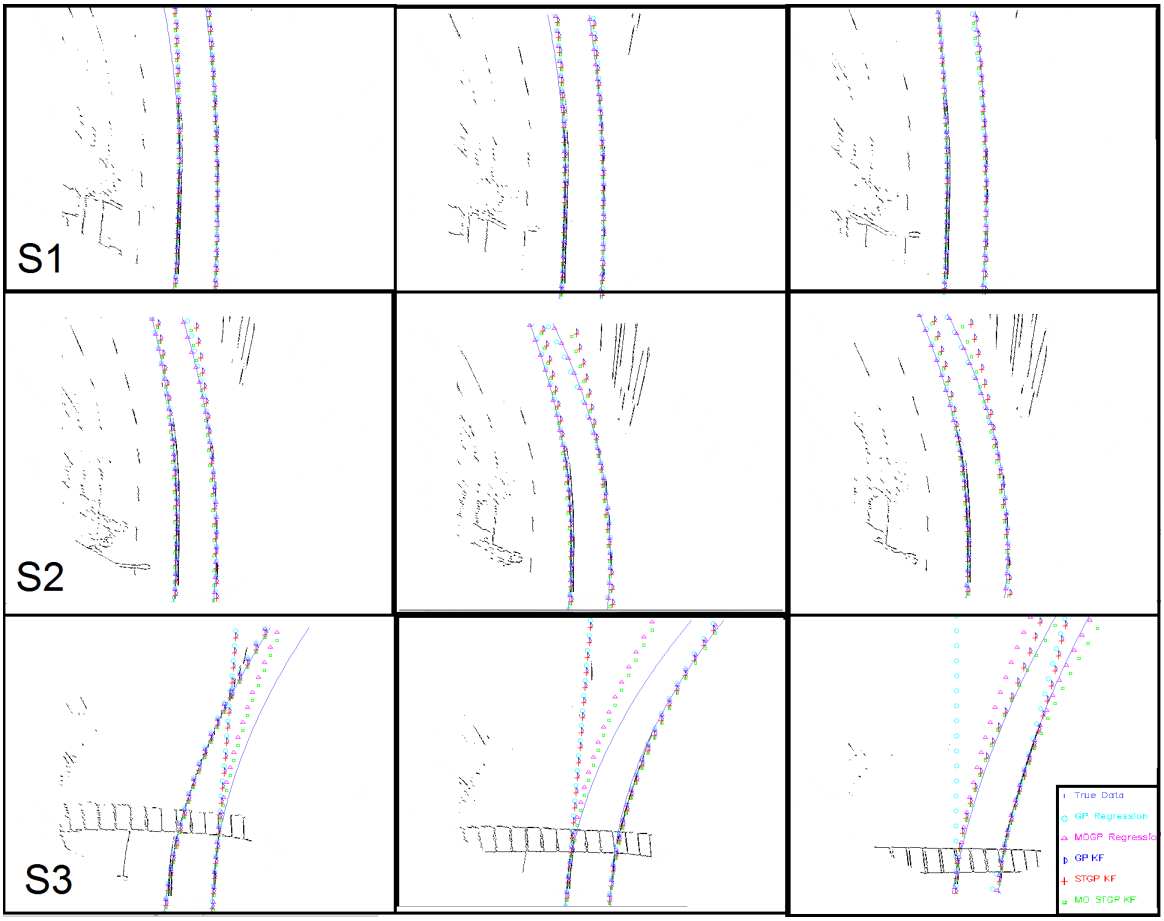


Figure 3.8: Comparing algorithms for 3 different scenarios in the sequential times.

Results: The RMSE values and the percentage improvement from 100 Monte Carlo runs for the three scenarios is given in Tables II and III. These number of run can be enough to obtain a confidence region for the performance assuming that its distribution is, in view of the central limit theorem, approximately normal.

The tables show that the performance of the STGP-KF and GP-KF is improved in all three cases using the multi-output structure. However, in S1 and S2 cases, when a false alarm is not an issue, the performance of our algorithm is slightly less than the batch GP regression and MOGP regression. In the case where false alarms exist (S3), the proposed algorithm is about 51 percent more accurate than corresponding batch regressions (GP, MOGP). Most of our improvement can be seen in the curved situation or where false alarms exist. Also, the volume of covariance of confidence value for our proposed algorithm is reasonably lower than other algorithms. Note that for straight lane markings GP-KF worked slightly better in both precision and the confidence region. Fig. 3.8 shows the snapshots of tracking of a two-lane line, for the three scenarios at the selected time steps. It can be observed that both the GP-KF and STGP-KF shape estimates are less accurate as compared to MO-STGP-KF.

Table 3.2: Shape Models.

Lane-line situation	Description
S1	Straight lane
S2	Curved line to the left
S3	Very curvy to right with false alarm

Table 3.3: RMSE Values for All Algorithms, (Pixels).

	GP	MOGP	GP-KF[82]	STGP-KF[1]	MOSTGP-KF
S1	3.46	3.16	3.19	4.25	3.47
S2	4.86	5.18	25.01	20.54	9.49
S3	38.02	36.44	19.40	26.56	18.26
Overall	15.45	14.93	15.86	17.12	10.41

Table 3.4: Confidence Values for All Algorithms, (Pixels).

	GP	MOGP	GP-KF ([82])	STGP-KF ([1])	MOSTGP-KF
S1	226	293	137	1101	195
S2	304	433	384	2171	384
S3	1467	1838	445	1697	400
Overall	666	855	322	1657	326

Table 3.5: Percentage Improvement Compared to GP-KF and STGP-KF.

	GP-KF ([82])	STGP-KF ([1])
S1	-0.08	0.18
S2	0.62	0.53
S3	0.05	0.31
Overall	0.34	0.39

3.6.2 Compare with Fully Supervised Methods

Fully supervised methods have also recently been used for detecting lane markings in environments with clutter. In [68], the Spatial Convolutional NN (SCNN) has been used to detect lane-lines and extract spatial correlation between rows and columns. This structure enables message passing between pixels across rows and columns in a layer. These relationships help with learning semantic objects with dependency between parts. [68] evaluated their method running on CULane and TuSimple datasets ([68], [18]). They achieved a great result detecting the lane markings in clutter, especially when the lane-lines were straight. They used probability maps (probmaps) of lane markings resulting from SCNN, and after verification, filled them with a cubic spline, which made the final prediction. The SCNN works well when the lanes are straight with enough feature cues, but in situations with a lack of distinctive features, miss clustering and wrong detection can be observed (see Fig. 3.10). This happens mostly at curved lane-lines when the symmetric structure has been missed. In our

proposed method, the combination of Bayesian K-JPDACF clustering of lane-lines with MO-STGP-KF will use the correlation of lane markings and the stream of frames to make a proper prediction. Note that MO-STGP-KF can be attached to any supervised lane detection algorithm to remove false alarms and add more semantics to the result using time domain information.

We evaluated our method with the SCNN method described in [68], using the TuSimple dataset ([81]), particularly their trained model weights. The TuSimple dataset has about 7,000 one-second-long video clips of 20 frames each. The ground-truth result has been prepared for the last frame (Frame 20), including height (h-sample) and width values corresponding to lane-lines (see Fig. 3.9).

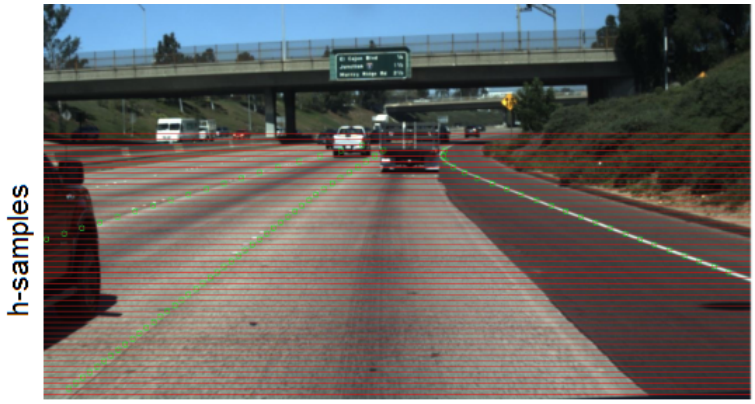


Figure 3.9: A sample of TuSimple ground truth format.

The TuSimple dataset covers most of the challenging situations, including curves, shadows, and splitting and merging on highways. We tested our algorithm against the SCNN algorithm in simple straight and curved lane-lines only. We tested on five different videos overall: 100 frames for straight and 100 frames for curved lanes [2].

[2]For measurement standard deviation about 1.2, confidence level %95 and precision 0.05 and 96 degree of freedom, overall 200 frames and 600 lane markings test is statistically significance

We saw 1.33,36 and 20 percent improvement in the accuracy, false positive rates, and negative rates in the curved and straight situation when we used our methods, compared to the SCNN method. We used the same accuracy formula as the TuSimple benchmark, which is:

$$\text{accuracy} = \frac{\sum_{\text{clip}} C_{\text{clip}}}{\sum_{\text{clip}} S_{\text{clip}}}, \tag{3.6.3}$$

where C_{clip} is the number of correct points in the last frame of the clip, and S_{clip} is the number of requested points in the last frame of the clip. If the difference between the width of ground truth and prediction is less than a threshold, the predicted point is a correct one. We will evaluate the values of all heights in h-sample.

Based on the formula above, we also computed the rate of a false positive and false negative for the test results. False positive means the lane is predicted but not matched with any lane in ground truth. False negative means the lane is in the ground truth but not matched with any lane in the prediction.

$$FP = \frac{F_{pred}}{N_{pred}}$$
$$FN = \frac{M_{pred}}{N_{gt}}, \tag{3.6.4}$$

where F_{pred} is the number of wrongly predicted lanes, N_{pred} is the number of all predicted lanes. M_{pred} is the number of missed ground truth lanes in the predictions, and N_{gt} is the number of all ground truth lanes.

We also compared our running time with the SCNN algorithm on a normal CPU. Our algorithm run-time in our computer (Intel i7 CPU, clocked at 2.9 GHz with 16 GB RAM) for that scenario was 4.8 times faster than the SCNN presented in [68].

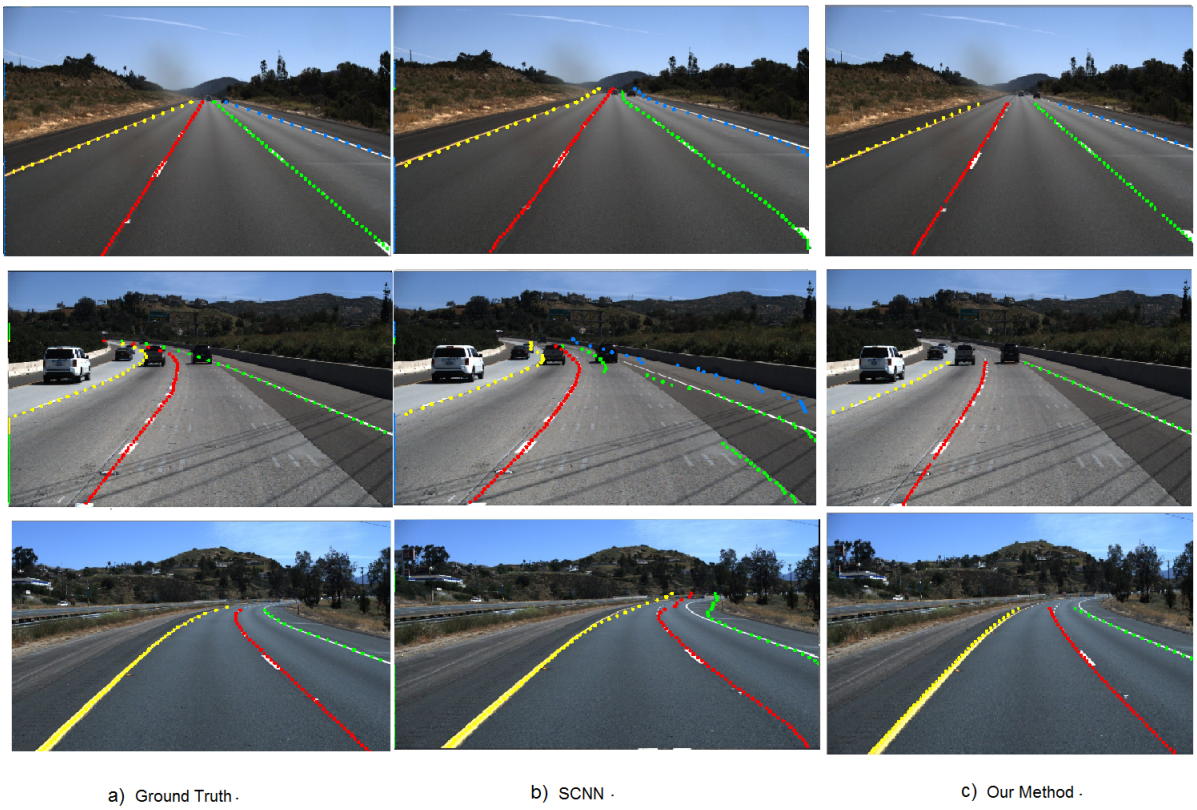


Figure 3.10: Result of SCNN algorithm in curved and straight situations.

The results are shown in Table 4.1.

3.7 Conclusions

In this work, we proposed a new approach for tracking multiple dependent targets recursively using an MO-STGP framework. The proposed method uses MOGP to model the coupling of extended targets and simultaneously estimate the states. To associate measurements and remove clutter, we proposed a kernel-based JPDACF

Table 3.6: Comparison of Our Approach with the SCNN Fully Supervised Method.

	SCNN Method in [68]				Proposed Method			
	FPR	FNR	Accuracy	Frame/s	FPR	FNR	Accuracy	Frame/s
Straight Highway#1	0.13	0.13	0.911	0.71	0.183	0.183	0.895	3.84
Curved Highway#2	0.366	0.266	0.886	0.72	0.13	0.13	0.901	5.02
Overall	0.250	0.199	0.899	0.71	**0.158**	**0.158**	**0.911**	**4.13**

algorithm for clustering measurements belonging to each target. For recursive inference, we developed recursive MO-STGP-KF to handle spatial dependency in the state-space model formulation. We implemented our method for the lane detection problem of detecting degraded lane markings in a traffic situation. The performance and capabilities of the algorithm are demonstrated through real data experiments. The result shows significant improvements in the estimation of unobserved lane markings compared to recently published semi-supervised or supervised methods.

Chapter 4

Spatio-Temporal Joint Probabilistic Data Association (ST-JPDA) For Extended Target Tracking

4.1 Introduction

One of the main assumptions in ordinary target tracking problems is that each target can generate at most one measurement per scan. Thus we call these problems single point target tracking problems. With the recent developments in high-resolution sensors, tracking the shape of the target in addition to its kinematics is attracting attention in ground, water, and air surveillance applications. The process of tracking the shape and kinematics of an object simultaneously is called Extended Object

Tracking (EOT). Nowadays, EOT has become an essential part of many autonomous systems and self-driving cars. The main focus of these technologies is currently the improvement of robustness and safety. Various types of sensors, such as camera, radar, and LiDAR (light detection and ranging), are being employed to fuse information for detecting and tracking roads and obstacles covered with different types of clutter. The focus of EOT has been mostly on measurement models, shape estimation, and data association. A typical approach to an EOT problem is to estimate the kinematics of the center of the object (CoO) and model the extent as a parametric or nonparametric function that is unknown and nonlinear, using a sequence of noisy measurements.

In [82] a Gaussian process (GP) was used for the first time to model the extent of the target. A GP is a distribution over an unknown and nonlinear function, in the continuous domain. The observed values of these functions (measurements) can be used to predict the values at unobserved points. Since the GP is a batch process and cannot be used in real-time applications, [82] have propose a recursive version of GP, where the change in the shape is modeled using a forgetting factor model. To combine evolving the shape with a kernel-based GP platform, [1] introduced the recursive Spatio Temporal Gaussian Process (STGP), in which they considered separable covariance functions for spatial and temporal dependency. They used a factorization of the power spectral density function of the covariance function to track the extent evolving in the temporal and spatial domain. They used the idea of converting a covariance function to a state-space model solvable with a smoother ([74]).

In the case where multiple targets are in the scene, associating the proper measurements to the target in clutter (data association) is an issue. The Probabilistic Data Association (PDA) technique is a robust method in a situation with clutter; it

employs a weighted sum of the latest set of measurements to update the state. When clutter and miss-detections do not exist, the most likely measurements to a target can be chosen using a Nearest Neighbor method. Unlike the Nearest Neighbor, which makes a hard decision, the PDAF makes a soft decision when estimating an expected value, which is the minimum mean square error (MMSE) estimate ([6]; [5]).

Using the PDAF for extended targets through the use of GP was introduced in [92]. They used a GP measurement and the random walk dynamic state model to estimate the scattering point evolution of a single target. Generally, PDAF is designed to track a single target or multiple targets that are not interfering or sharing measurements. In contrast, the Joint Probabilistic Data Association Filter (JPDAF) can handle multiple targets that may share measurements. In the JPDAF, we assumed that the target states and thus the target-originated measurements are independent. For the dependent targets, a statistical dependence of their estimation can be taken into account by calculating the state cross-covariances. The resulting algorithm is called the Joint Probabilistic Data Association Coupled Filter (JPDACF) ([7]; [8]; [79]; [13]). A motion model in the traditional JPDACF is parametric and needs to be considered precisely at the design stage.

We used the idea explained in [1] by adding a new joint probabilistic data association measurement model instead of the GP measurement model. This idea enables us to track the shape of multiple spatiotemporal extended targets in a clutter situation where targets share measurements. We used ST-JPDA to track lane-lines, especially in merging or splitting situations.

4.1.1 Contributions

We make the following key contributions:

1. We combine clustering and regression of multiple extended targets in linear time, using the K-JPDACF augmented with an RTS smoother.

2. We introduce a new algorithm for tracking multiple interfering extended targets using the ST-JPDA in the clutter.

3. We model dependency between extended targets using the ST-JPDA coupled version.

4. We use ST-JPDA practically to track lane-lines in merging or splitting situations.

4.2 The Gaussian Process vs JPDA

JPDAF and JPDACF have been used for tracking multiple independent or dependent single-point targets when targets interfere with each other. In chapter 3.2.3, we saw that with a sufficient number of time derivatives, kernel-based GP regression can be represented as a state-space model ([74]) (see Eq. (4.2.1).

$$a_n \frac{d^n f(t)}{dt^n} + ... + a_1 \frac{df(t)}{dt} + a_0 f(t) = \mathbf{w}(t) \tag{4.2.1}$$

$$d\mathbf{f}(t) = \mathcal{A}\mathbf{f}(t) + \mathbf{L}\mathbf{w}(t), \tag{4.2.2}$$

where $\mathbf{f} = [f, df/dt, ...d^{s-1}f/dt^{s-1}]^\top$ and $\mathbf{w}(t)$ is a Wiener process.

Like the Bayesian Kalman Filter, JPDAF can be considered to be a linear approach to solving this group of differential equations (4.2.2) when clutter is present. However, filtering only provides the forward-time posteriors of the process: to get the full posterior, one needs to use a linear smoother. In fact, the state-space model equation (4.2.2) could make the exact solution a kernel-based GP or a recursive JPDAF filter combined with a linear Rauch-Tung-Strieble (RTS) smoother. So we can use K-JPDACF not just for clustering interfering targets in the spatial domain, but also to make pseudo measurements. In fact, adding a smoother to the result of K-JPDACF will make a solution similar to a GP regression.

4.2.1 K-JPDACF for Clustering Extended Targets

The general recursive algorithm for clustering with K-JPDACF uses the following steps:

1. *Make stacked vectors/matrices*: For D number of targets in time-step k, stacked states, covariance, and filter matrices are as follows:

$$\bar{x}_k = \left[x_k^1, x_k^2, ..., x_k^D\right]^\top, \quad \bar{P}_k = B_D \otimes P_k \tag{4.2.3}$$

$$\bar{F}_k = I_D \otimes F_k, \quad \bar{Q}_k = B_D \otimes Q_k \quad \bar{R}_k = I_D \otimes R_k, \tag{4.2.4}$$

where B_D is a coregionalization matrix that shows target dependency. In independent situations, $B_D = I_D$. Note that in the case when D is unknown, the integrated version of JPDAF or IJPDAF is employed ([62]).

2. *Prediction*: We use the prior information coming from the spatial kernel function $(\bar{F}_k, \bar{H}_k, \bar{Q}_k)$ and last estimates $(\hat{\bar{x}}_{k|k}, \bar{P}_{k|k})$ to predict the new states and covariance, similar to a normal KF prediction (Eq. (4.2.5), and results are $(\hat{\bar{x}}_{k+1|k}, \bar{P}_{k+1|k})$. Note that predicted covariance is the cross-covariance with off-diagonal blocks.

$$\bar{\mathbf{x}}_{k+1|k} = \bar{\mathbf{F}}_k \bar{\mathbf{x}}_{k|k} \tag{4.2.5a}$$

$$\bar{\mathbf{P}}_{k+1|k} = \bar{\mathbf{F}}_k \bar{\mathbf{P}}_k \bar{\mathbf{F}}_k^\top + \bar{\mathbf{Q}}_k \tag{4.2.5b}$$

3. *Measurement validation*: Measurement validation first needs to be performed for each target separately. Let S_k^d denote the $n_z \times n_z$ sub-block diagonal of $\bar{S}_k$ for a single predicted measurement $\hat{z}_k^d$. The measurement z_k^j can be validated if and only if :

$$[z_k^j - \hat{z}_k^d][S_k^d]^{-1}[z_k^j - \hat{z}_k^d]^\top < \gamma, \tag{4.2.6}$$

where γ is an appropriate threshold.

4. *Computing the joint probabilities*: The conditional probability for a joint association event in JPDACF is as follows:

$$P\{A_{k+1}|Z^{k+1}\} = \frac{1}{c} p[z_k|A_k, m_k, Z^{k-1}] P\{A_{k+1}|Z^k, m_k\}, \tag{4.2.7}$$

where we have:

$$\begin{aligned} P\{A_{k+1}|Z^k, m_k\} &= P\{A_{k+1}|\delta_A, \phi_A, m_k\} P\{\delta_A, \phi_A|m_k\} \\ &= (A^{m_k}_{m_k-\phi})^{-1} \prod_t (P_D^t)^{\delta_t} (1-P_D^t)^{1-\delta_t} \frac{\mu_{F,\phi}}{P\{m_k\}} \\ &= \frac{\phi!}{m_{k+1}!} \mu_{F,\phi} \prod_t (P_D^t)^{\delta_t} (1-P_D^t)^{1-\delta_t}, \quad (4.2.8) \end{aligned}$$

where ϕ is the number of false measurements in the event A, $(A^{m_k}_{m_k-\phi})$ is an arrange of m_k over $(m_k - \phi)$ measurements, P_D is the probability of detection, $\mu_{F,\phi}$ is prior pmf of the clutter model, and the binary variable δ_t is the the detection indicator for target t (equal to one if target t is assumed detected in event A). The states of the target are dependent and the conditional measurement likelihood in this case cannot be reduced to the marginal form:

$$p[z_k|A_k, m_k, Z^{k-1}] = V^{-\phi} f_{t_{j1}, t_{j2}, ..}[z_{j,k}, j : \tau_j = 1] \quad (4.2.9)$$

where volume V and $f_{t_{j1}, t_{j2}, ..}$ is the joint pdf of the measurements of the targets under consideration, and t_{j1} is the target to which $z_{j1,k}$ is associated in event A. τ_j is the target association indicator for measurement j in event A_{k+1} and c is the normalization constant.

5. *Update*: For coupled filtering, the joint probabilities are not reduced to the marginal association probabilities. Instead, these joint probabilities are used directly in a coupled filter as follows:

$$\hat{\bar{x}}_{k+1|k+1} = \hat{\bar{x}}_{k+1|k} + \bar{G}_{k+1} \bar{v}_{k+1}, \quad (4.2.10)$$

where

$$\bar{v}_{k+1} = \sum_{A} P\{A_{k+1}|Z^{k+1}\}\bar{v}_{A,k+1}, \tag{4.2.11}$$

$$\bar{v}_{A,k+1} = [\bar{z}_{k+1,A} - \hat{\bar{z}}_{k+1|k}], \tag{4.2.12}$$

$$\bar{z}_{k+1,A} = \left[z_{j1,A}^{k+1}, z_{j2,A}^{k+1}, ..., z_{jD,A}^{k+1}\right]^{\top} \tag{4.2.13}$$

and $_{jt,A}$ is the index for the measurement associated with target t in event A at time $k+1$. The conditional measurement matrix $H_A(k)$ in this case needs to take care of the cases where some of the targets are not detected.

$$\bar{H}_A(k) = Diag[\delta_A^1, \delta_A^2, ..., \delta_A^D] \otimes H(k), \tag{4.2.14}$$

where $[\delta_A^1, \delta_A^2, ..., \delta_A^D]^{\top}$ is a target detection indicator vector. The filter gain in (4.2.11) is computed by inverting the covariance of residual $\bar{S}_k = \bar{H}_{k+1}\bar{P}_{k+1|k}\bar{H}_{k+1}^{\top} + \bar{R}_{k+1}$:

$$\bar{G}_{k+1} = \bar{P}_{k+1|k}\bar{H}_{k+1}^{\top}[\bar{H}_{k+1}\bar{P}_{k+1|k}\bar{H}_{k+1}^{\top} + \bar{R}_{k+1}]^{-1} \tag{4.2.15}$$

The predicted stacked measurement vector is:

$$\hat{\bar{z}}_{k+1|k} = \bar{H}_{k+1}\hat{\bar{x}}_{k+1|k}, \tag{4.2.16}$$

Having all the matrices above, the state covariance update is as follows:

$$\bar{P}_{k+1|k+1} = \beta_{0,k+1}\bar{P}_{k+1|k} + [1-\beta_{0,k+1}]\bar{P}^{\mathcal{C}}_{k+1|k+1} + \tilde{\bar{P}}_k \tag{4.2.17}$$

With probability $\beta_{0,k+1} \triangleq P\{A_0|Z^k\}$, none of the measurements is correct, in which case there is no update of the state estimate. With probability $1-\beta_{0,k+1}$, the correct measurement is available, and the updated covariance is $\bar{P}^{\mathcal{C}}_{k+1|k+1}$.

$$\bar{P}^{\mathcal{C}}_{k+1|k+1} = \bar{P}_{k+1|k} - \bar{G}_k\bar{S}_k\bar{G}_k^\top \tag{4.2.18}$$

However, since it is not known which of the $m(k)$ validated measurements is correct, the term $\tilde{\bar{P}}_k$, which is positive semidefinite, increases the covariance of the updated state. The spread of the innovations term is:

$$\tilde{\bar{P}}_k \triangleq \bar{G}_{k+1}[\sum_A P\{A_{k+1}|Z^{k+1}\}\bar{v}_{A,k+1}\bar{v}_{A,k+1}^\top - \bar{v}_{k+1}\bar{v}_{k+1}^\top]\bar{G}_{k+1}^\top \tag{4.2.19}$$

4.2.2 Fixed Lag RTS Smoother

In state estimation, smoothing is a process where current measurements are used to improve estimates of past states. In fact, smoothing estimates backwards posterior density. The general case of smoothing that can be used in real-time applications is called a fixed lag smoother. A fixed lag smoother smoothes the trajectory for an interval of s states. Fixed interval smoothing requires a backward iteration after the (forward) filtering is complete. When the targets are coupled, the results of filtering are in the stacked form, $\bar{\mathbf{x}}_{k|k}, \bar{\mathbf{x}}_{k+1|k}, \bar{\mathbf{P}}_{k|k}, \bar{\mathbf{P}}_{k+1|k} \quad k = k, k-1, ..., k-s$, and need to

be stored for use in the backward iteration. In fixed interval smoothing, when the smoothing depth (s) is fixed, the smoother gains $\bar{\mathbf{G}}_k$, the smoothed state $\tilde{\bar{x}}_k$, and the state error covariance $\tilde{\bar{P}}_k$ are recursively estimated using the Rauch-Tung-Strieble (RTS) recursion ([70]; [6]):

$$\bar{\mathbf{G}}_k = \bar{\mathbf{P}}_{k|k}\bar{\mathbf{F}}^{\top}(\bar{\mathbf{P}}_{\mathbf{k+1|k}})^{-1} \tag{4.2.20a}$$

$$\tilde{\bar{x}}_{k|\mathbb{N}} = \bar{\mathbf{x}}_{k|k} + \bar{\mathbf{G}}_k[\tilde{\bar{x}}_{k+1|N} - \bar{\mathbf{x}}_{k+1|k}] \tag{4.2.20b}$$

$$\tilde{\bar{P}}_{k|\mathbb{N}} = \bar{\mathbf{P}}_{k|k} + \bar{\mathbf{G}}_k[\tilde{\bar{P}}_{k+1|N} - \bar{\mathbf{P}}_{k+1|k}]\bar{\mathbf{G}}_k^{\top} \tag{4.2.20c}$$

The smoother is initialized at the current time step k as $\tilde{\bar{x}}_{k|\mathbb{N}} = \bar{\mathbf{x}}_{k|k}$ and $\tilde{\bar{P}}_{k|\mathbb{N}} = \bar{\mathbf{P}}_{k|k}$. The result of smoothing function $f(.)$ at time k is summarized in the mean and covariance $\tilde{\bar{x}}_{k|\mathbb{N}}$ and $\tilde{\bar{P}}_{k|\mathbb{N}}$, conditioned to the measurements $\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_{\mathbb{N}}$.

4.3 Problem Formulation

In this section, the extended target model and the measurement model are briefly reviewed.

4.3.1 Extended Target Model

In our application, extended targets are defined as continuous functions. To simplify the design, we considered targets to be functions in Cartesian space with a global origin that is the bottom left part of the frame. In radial space, when input indexes

are angles, these curved functions could transfer to the 2D star-convex objects ([82]). To implement independent and dependent targets in a similar framework, we stack all the latent functions and consider them a single state vector. We change the notation of latent function $\mathbf{f}(\mathbf{u}^f)$ to $\mathbf{x}$ to be more similar to state-space model representation. The dynamics of extent are designed as separable kernels, which satisfy the conditions

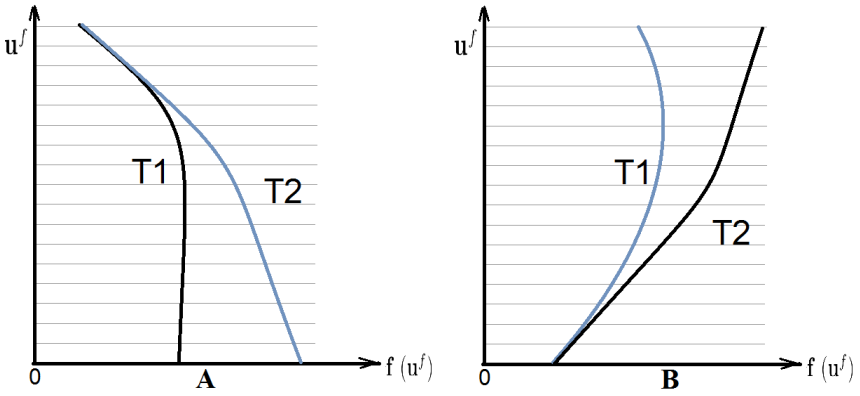


Figure 4.1: Extended target model for (A) merging and (B) splitting for two targets.

described in chapter 3.2.3.

$$\kappa(u, u'; t, t') = \kappa_u(u, u')\kappa_t(t, t') \tag{4.3.1}$$

The evolution of stacked extended target states is modeled as follows:

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{F}}_k \bar{\mathbf{x}}_k + \bar{\mathbf{w}}_k \tag{4.3.2}$$

$$\bar{\mathbf{w}}_k \sim \mathcal{N}(0, \bar{\mathbf{Q}}_k), \tag{4.3.3}$$

$$\bar{\mathbf{x}}_k = \begin{bmatrix} \mathbf{f}_1(\mathbf{u}^f) \\ \mathbf{f}_2(\mathbf{u}^f) \\ \dots \\ \mathbf{f}_D(\mathbf{u}^f) \end{bmatrix} \tag{4.3.4}$$

where $\bar{\mathbf{x}}_k$ is stacked latent outputs that each corresponds to N distinct index points $\mathbf{u}^f = \{u_1^f, u_2^f, ...u_N^f\}$. $\bar{\mathbf{F}}_k$ is stacked version of the state transition matrix and $\bar{\mathbf{w}}_k$ is a zero-mean white Gaussian noise with covariance $\bar{\mathbf{Q}}_k$ for stacked states. cardinality and value of index points are related to the precision and shape of the extended target. Under the Gaussian assumption of $\bar{\mathbf{Q}}_k$, the propagated distribution of the state in Eq. (4.3.2) remains Gaussian.

We train the model similar to a normal GP, minimizing the log marginal likelihood of measurements ([83]). The training results include the kernel parameters and coregionalization matrix. Note that in cases where extended targets are independent, the coregionalization matrix is an identity matrix.

Temporal Covariance Kernel: In our application, we used the Matérn covariance function for the temporal part. The general form of Matérn is given as:

$$\kappa_t(\tau) = \sigma_t^2 \frac{2^{1-\nu}}{\Gamma(\nu)} (\frac{\sqrt{2\nu}}{l}\tau)^\nu \kappa_\nu (\frac{\sqrt{2\nu}}{l}\tau), \tag{4.3.5}$$

where σ_t^2 is a magnitude scale hyperparameter, l is the characteristic length-scale, and ν is a smoothness parameter. $\kappa_\nu(.)$ denotes the modified Bessel function of the second kind. For this class, the corresponding process is s-times differentiable if $\nu > s$, [74].

We considered $\nu = 3/2$, with a continuous and once differentiable process. In this

case, the covariance function is simplified to:

$$\kappa_t(\tau)_{3/2} = \sigma^2(1 + \frac{\sqrt{3}\tau}{l})exp(-\frac{\sqrt{3}\tau}{l}) \tag{4.3.6}$$

The continuous system matrix and the noise effect vector of the corresponding state-space model are derived as follows:

$$\mathcal{A}_t = \begin{bmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{bmatrix}, \quad \mathbf{L}_t = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{P}_\infty = \mathbf{P}_0 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \lambda^2\sigma^2 \end{bmatrix}, \tag{4.3.7}$$

where $\lambda = \frac{\sqrt{3}\tau}{l}$ and the spectral density of the Gaussian white noise process $w(t)$ is $\mathbf{Q}_c = 4\lambda^3\sigma^2$. The measurement model matrix is $\mathbf{H} = [1 \;\; 0]$. The state of each target can be defined as:

$$\mathbf{f}_i = [[f(u_1)_k, \dot{f}(u_1)_k], ..., [f(u_D)_k, \dot{f}(u_D)_k]] \tag{4.3.8}$$

In a discrete case, $\mathbf{H}_k = \mathbf{H}_t, \;\; \mathbf{L}_k = \mathbf{L}_t, \;\; \mathbf{F}_k = e^{\mathcal{A}_t T_s}$ so we have:

$$\bar{\mathbf{F}}_k = I_D \otimes I_N \otimes \mathbf{F}_k \quad \bar{\mathbf{L}}_k = I_D \otimes I_N \otimes \mathbf{L}_t \quad \bar{\mathbf{H}}_k = I_D \otimes I_N \otimes \mathbf{H}_t \tag{4.3.9}$$

Subscript t denotes the temporal continuous model matrices in Eq. (4.3.7). The spectral density and the initialized state covariance have spatial structure.

$$\bar{\mathbf{Q}}_k = \bar{\mathbf{K}} \otimes (\mathbf{P}_\infty - \mathbf{F}_k\mathbf{P}_\infty\mathbf{F}_k^\top), \quad \bar{\mathbf{P}}_0 = \bar{\mathbf{K}} \otimes \mathbf{P}_{0,t} \tag{4.3.10}$$

In the case where outputs are dependent, the coupling can be defined with coregionalization matrix B. In the ICM case, the covariance matrix for stacked data is

as follows ([2]):

$$\bar{\mathbf{K}}(\mathbf{u}, \mathbf{u}') = B \otimes K(\mathbf{u}, \mathbf{u}'), \tag{4.3.11}$$

where B is a positive semidefinite rank-one coregionalization matrix that specifies the inter-output similarities. $K(\mathbf{u}, \mathbf{u}')$ is a covariance matrix of the underlying process. The parameters of the coregionalization matrix include vector W, which is the same size as the output and constant ξ .

$$\mathbf{B} = WW^{\top} + \text{diag}(\xi) \tag{4.3.12}$$

The Kronecker product of $B \otimes K$ can be written as :

$$\bar{\mathbf{K}}(\bar{\mathbf{f}}, \bar{\mathbf{f}}') = \begin{bmatrix} B_{11}K(\mathbf{u_1}, \mathbf{u_1}') & .... & B_{1D}K(\mathbf{u_D}, \mathbf{u_D}') \\ ... & \ddots & ... \\ B_{D1}K(\mathbf{u_1}, \mathbf{u_1}') & ... & B_{DD}K(\mathbf{u_D}, \mathbf{u_D}') \end{bmatrix} \tag{4.3.13}$$

When the B matrix is not identity, the covariance of the joint Gaussian process over $\bar{\mathbf{z}}$ is not a diagonal matrix, because of the dependency of the outputs. In fact, the observations of one target can affect the predictions for another target.

4.3.2 Measurement Model

The cardinality and position of the measurements are usually both unknown and time-varying. Furthermore, measurement origin uncertainty (i.e., missed detection and clutter) must be taken into account. In fact, index points of measurements, $\bar{\mathbf{u}} = [\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_D]^{\top}$ are not isotopic.

Based of the K-JPDACF discussed in section 4.2.1, given a set of N distinct input

indexes $\mathbf{u}^f = [u_1^f, u_2^f, ...u_N^f]^\top$ for the model, we want to estimate a D unknown set of outputs or states, $\bar{\mathbf{x}} = [\mathbf{f}_1(\mathbf{u}^f), \mathbf{f}_2(\mathbf{u}^f), ..., \mathbf{f}_D(\mathbf{u}^f)]^\top$ corresponding to M noisy measurements $\mathbf{z}$. The pseudo measurements are the result of K-JPDACF after applying the RTS smoother. Note that $\mathbf{z}$ is not clustered. For the application of lane-line tracking, the spatial kernel for two individual inputs (u, u') can be a polynomial or Radial Basis Function (RBF) kernel.

$$\kappa(u, u')_{pol} = (\sigma^2 + u.u')^p, \quad \kappa(u, u')_{RBF} = \sigma^2 \exp\left(-\frac{1}{2}\frac{(u - u')^2}{\ell^2} \right) \tag{4.3.14}$$

The parameters of the K-JPDACF filter come from the power spectral density of the spatial kernel function.

4.3.3 ST-JPDAF for Tracking Extended Targets

Similar to flowchart 4.2, the overall recursive MO-STGP regression can be reformulated as follows:

1. Use the temporal kernel and priors to predict the extended targets' states and covariances.

2. Use the spatial kernel and K-JPDACF to compute measurement likelihood and predicted measurements.

3. Update the state and covariance with new observations using Kalman filtering.

Using a stacked version of last states and covariance and transition matrices in Eq. (4.3.9), we can predict state and covariance similar to a normal Kalman filter,

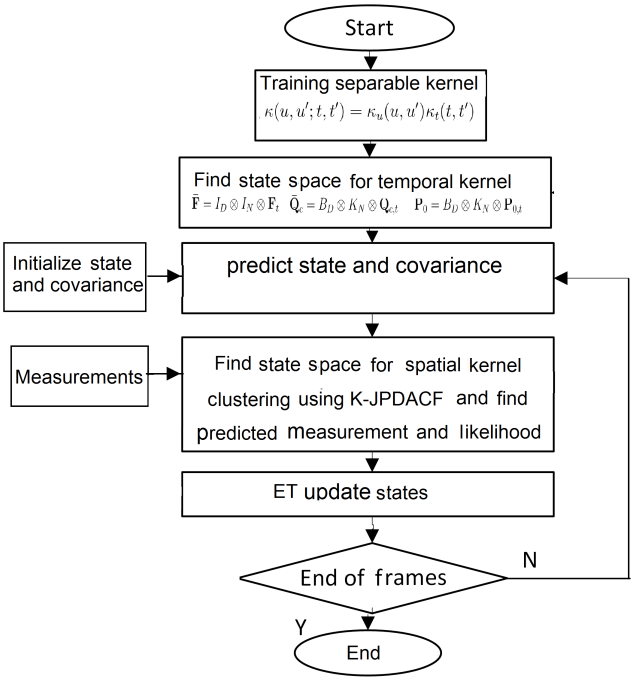


Figure 4.2: General pipeline for tracking ET using ST-JPDAF.

discussed in section 3.2.4.

4.4 Experiments and Evaluations

In multiple lane-line tracking applications we compared our algorithm with a fully supervised lane detection method recently published in [68]. In [68], the Spatial Convolutional NN (SCNN) was used to detect lane-lines and extract spatial correlation between rows and columns. The proposed method was run on the CULane and TuSimple datasets for evaluation ([68]; [18]). This method achieved good results detecting lane markings in the presence of clutter, especially when the lane-lines are

straight. The approach uses probability maps (probmaps) of lane markings resulting from SCNN, and after verification, fills the maps with a cubic spline, which makes the final prediction. The SCNN works well when the lanes are straight with enough feature cues, but in situations with a lack of distinctive features, it produces miss clustering and incorrect detection (see Fig, 4.3). This occurs most frequently at curved lane-lines, splitting, and merging when the symmetric structure missed.

We evaluated our method against the SCNN method proposed in [68] on the TuSimple dataset ([81]) using their trained model weights. The TuSimple dataset has about 7,000 one-second-long video clips of 20 frames each. They prepared the ground-truth result for the last frame (Frame 20), including height (h-sample) and width values corresponding to lane-lines.

The TuSimple dataset covers most of the challenging situations on highways: curves, shadows, splitting, and merging. We tested our algorithm against the SCNN algorithm in merging and splitting videos only. We tested on 5 different videos, amounting to a total of 100 frames [1]. Our method yielded 20, 36, and 20 percent improvement in accuracy, false positive, and false negative rates, respectively, in merging and splitting situations over the SCNN method. We used the same accuracy formula as the TuSimple benchmark:

$$\text{accuracy} = \frac{\sum_{\text{clip}} C_{\text{clip}}}{\sum_{\text{clip}} S_{\text{clip}}}, \tag{4.4.1}$$

where C_{clip} is the number of correct points in the last frame of the clip, and S_{clip} is the number of requested points in the last frame of the clip. If the difference between

[1] For measurement standard deviation about 1.2, confidence level %95 and precision 0.05 and 96 degree of freedom for each lane marking, overall 100 frames and 300 lane markings test is statistically significance

the width of ground truth and prediction is less than a threshold, the predicted point is a correct one. We further evaluate the values of all heights in the h-sample.

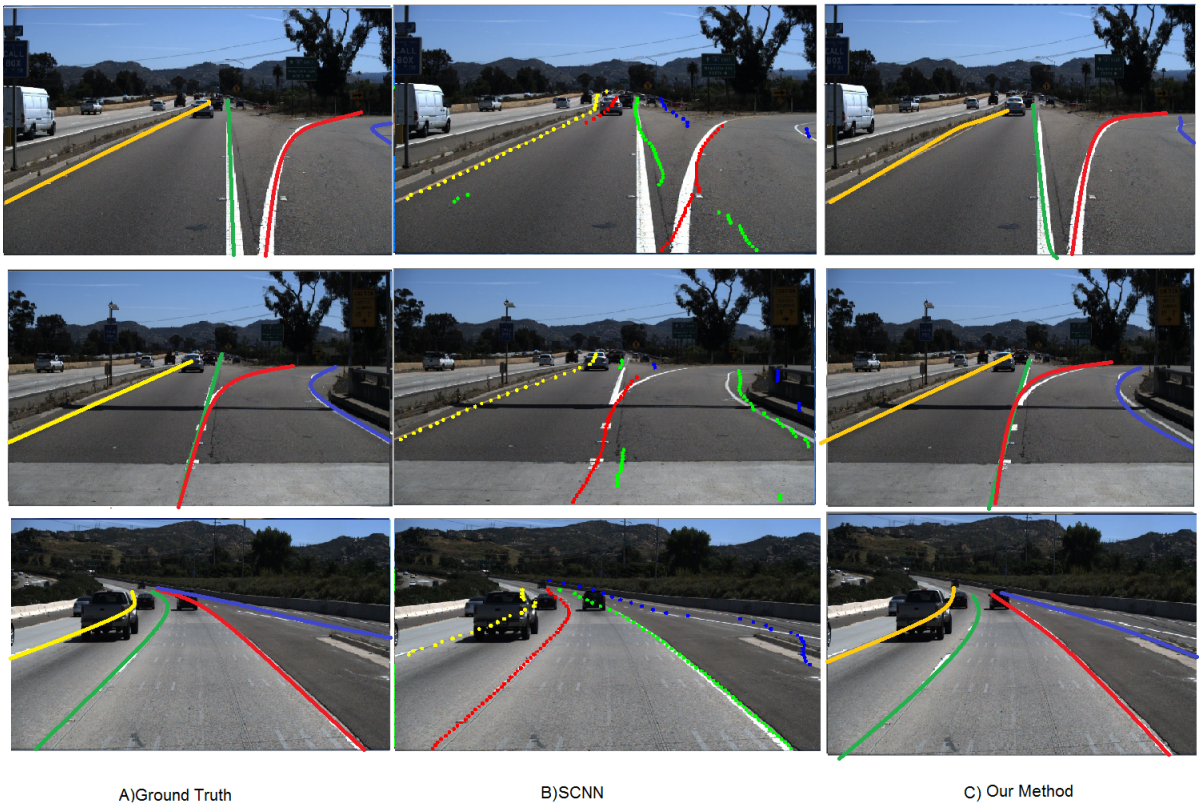


Figure 4.3: Result of SCNN algorithm in curved and straight situations.

Based on the formula above, we will also compute the rate of a false positive and false negative for the test results. False positive means the lane is predicted but not matched with any lane in ground truth. False negative means the lane is in the ground truth but not matched with any lane in the prediction.

$$\text{FP} = \frac{F_{\text{pred}}}{N_{\text{pred}}}$$

$$\mathrm{FN} = \frac{M_{\mathrm{pred}}}{N_{\mathrm{gt}}}, \tag{4.4.2}$$

where F_{pred} is the number of wrongly predicted lanes, N_{pred} is the number of all predicted lanes. M_{pred} is the number of missed ground-truth lanes in the predictions, and N_{gt} is the number of all ground-truth lanes.

We also compared our running time with the SCNN algorithm on a normal CPU. Our algorithm's run-time in our computer (with an Intel i7 CPU, clocked at 2.9 GHz with 16 GB RAM) for that scenario was five times faster than that of SCNN, as presented in [68]. The results are shown in Table 4.1.

Table 4.1: Comparison of Our Approach with SCNN Fully Supervised Method

	SCNN Method in [68]				Proposed Method			
	FPR	FNR	Accuracy	Frame/s	FPR	FNR	Accuracy	Frame/s
Merging and Splitting	0.35	0.266	0.877	0.71	0.183	0.183	0.895	5

4.5 Conclusions

In this work, we introduced ST-JPDAF for tracking multiple extended targets. We introduced a kernel-based JPDAF augmented with an RTS smoother and used it as a measurement model. We managed dependency of measurements in space (inside a frame) and time (between frames) using different kernel functions. To address target dependency, the coupled version of ST-JPDAF was introduced. The kernel functions can be learned using the training data. This extension can be used to track the shape and dynamics of start-convex nonparametric dependent extended targets in the presence of clutter when targets share measurements in linear time.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, the tracking of multiple independent or dependent ETs in the presence of clutter has been studied. In general, the tracking multiple objects is categorized into two main parts, namely, clustering features into pseudo-measurements and tracking pseudo-measurements in the temporal domain. We developed new Bayesian clustering methods based on MAP and KJPDACF to cluster line and point shape features in the spatial domain. In the time domain, we developed new tracking methods based on PDAF, JPDAF and MO-STGP-KF to associate, track and manage multiple ETs. We also implemented our algorithms for lane detection and presented the results in three different papers.

In the first work, we expanded estimations into the time domain using the IPDA filter. By using the intensity feature in conjunction with the Hough transform, we first formulated an algorithm for clustering and grouping multiple line segments belonging to each lane-line. By using these lane-lines as an extended target (spline), we identified

a set of control points, which were then tracked on every frame.

In the second work, we proposed a new approach for tracking multiple dependent targets recursively using the MO-STGP framework. The proposed method uses MOGP to model the coupling of extended targets, simultaneously estimating the states. To associate measurements and remove clutter, we proposed a kernel-based JPDACF algorithm to cluster measurements belonging to each target. For recursive inference, we developed the recursive MO-STGP-KF technique to handle spatial dependency during the formulation of the state-space model. We implemented our method for lane detection problems to detect degraded lane markings in a high-traffic situation.

In the last work, we expanded JPDA to track multiple dependent extended targets, developing a Spatio-Temporal Joint Probabilistic Data Association Filter (ST-JPDAF). We managed dependency of measurements in space (inside a frame) and time (between frames) using different kernel functions, which can be learned using the trained data. This extension can be used to track the shape and dynamics of nonparametric dependent extended targets in the presence of clutter when targets share measurements. Considering with last two approaches ST-JPDAF can cover all challenges, dealing with clutter, dependency and interfering in linear time with approximately similar precision as batched MO-STGP.

The performance and capabilities of each of these methods were demonstrated through experiments using real data. The results show significant improvements in the estimation of unobserved lane markings compared to recently published semi-supervised or supervised methods.

5.2 Future Work

METT is still considered an open problem and further research is needed to address the remaining challenges and make the algorithms more efficient and more accurate. Lane-line tracking requires precise shape estimation because of the road safety implications. To improve computational efficiency parallel and GPU computing techniques have to be utilized. Thus, the following are some potential areas for future research:

1. Expanding the measurement model for MO-STGP-KF from a continues-function to 2D or 3D space to track rigid and nonrigid objects. This will improve the general applicability of our work.

2. Combining fully supervised methods with ETT models. The objective here is to reduce false alarms.

3. Using Deep Gaussian Processes (DGP) in METT problems. The objective is to take advantage of parallel and GPU computing techniques to improve computational efficiency.

4. Going beyond the Gaussian models (e.g., Student's-t Processes) to track multiple non-Gaussian objects with sharp angles. In this case, the Student's-t Process can be used instead of GP. This can improve the tracking of highly nonlinear lane markings.

.

Appendix A

Association Probability for Lane Measurements

Assuming there are m detections at time k, when finding the association probability β_j between a measurement and a target (lane j), the overall event $\epsilon(j)$ comprises two mutually exclusive events: either $\epsilon(j)$ is such that the measurement $\psi(j)$ is from the target, for $j = 1, \ldots, m$, or all measurements are false.

Here, the association set $\{\beta_j\}$ is defined as the probability of the events $\{\epsilon_j\}$ given all the measurements Ψ, $\beta_j = p\{\epsilon_j|\Psi\}$. This appendix demonstrates how β_j is related to the feature likelihood ratio of the measurement e_j.

Let the set of validated measurements at time k be denoted as $\Psi_k = \{\psi_k(j)\}$, for $i = 1, \ldots, m_k$. In general, the following criteria must be satisfied:

$$(\psi - \bar{x})'(R + Q)^{-1}(\psi - \bar{x}) \leq \gamma, \tag{A.0.1}$$

where the γ is chi-square distributed with n_z degrees of freedom. The association

probability of β_j for a set of gated measurements can then be stated as $\beta_j = p\{\epsilon_j|\Psi, m_k\}$. Using Bayes' rule, this can be expressed as

$$\beta_j = \frac{1}{c_1} p\{\Psi|\epsilon_j, m_k\} p\{\epsilon_j|m_k\}. \tag{A.0.2}$$

Assuming that the false measurements are uniformly distributed within the validation region, and asumming the correct measurement location to be Gaussian with mean $\bar{x}$ and covariance of $S = R + Q$, the probability density function for the correct measurement without the intensity feature is

$$\mathcal{L}_j = p\{\Psi|\epsilon_j, m_k\} = P_G^{-1}\mathcal{N}(\psi_j\bar{x}, S), \tag{A.0.3}$$

where P_G is the gating probability. Since the intensities are independent of location within the validation region, the probability density function of a single correct measurement, including the intensity feature, can be expressed as the product of the intensity likelihood ratio with $\mathcal{L}_j$ as follows:

$$p\{\psi|\epsilon_j, m_k\} = P_1^{\tau}(f_j)\mathcal{L}_j \tag{A.0.4}$$

and for incorrect measurements

$$p\{\psi|\epsilon_j, m_k\} = P_0^{\tau}(f_j)V^{-1}, \tag{A.0.5}$$

where V is the volume of the validation region. Thus, for the all measurements, the

probability density function is

$$p[\mathbf{\Psi}|\epsilon_j, m_k] = P_1^{\tau}(f_j)[\prod_{j=1}^{m} p_0^{\tau}(f_j)]V^{-m+1}e_j \tag{A.0.6}$$

for $j = 1, \ldots, m_k$ and

$$p[\mathbf{\Psi}|\epsilon_j, m_k] = V^{-m}[\prod_{j=1}^{m} p_0^{\tau}(f_j)] \tag{A.0.7}$$

for $j = 0$.

Using a non-parametric model considering

$$p[\epsilon_j|m_k, x] = P_D P_G m^{-1} \tag{A.0.8}$$

for $j = 1, \ldots, m_k$ and

$$p[\epsilon_j|m_k, x] = 1 - P_D P_G \tag{A.0.9}$$

for $j = 0$, the association probability β_j can be expressed as

$$\beta_j = P_1^{\tau}(f_j)[\prod_{j=1}^{m} p_0^{\tau}(f_j)]V^{-m+1}e_j P_D P_G m_k^{-1} \tag{A.0.10}$$

for $j = 1, \ldots, m_k$ and

$$\beta_0 = V^{-m}[\prod_{j=1}^{m} p_0^{\tau}(f_j)](1 - P_D P_G) \tag{A.0.11}$$

for $j = 0$.

Since the set of events $\{\epsilon_j\}$ are mutually exclusive and exhaustive,

$$\beta_0 + \sum_{j=1}^{m_k} \beta_j = 1. \tag{A.0.12}$$

With some simplification, the overall results can be stated as

$$\begin{aligned} \beta_j &= p\{\epsilon_j|\psi_j, m_k\} \\ &= \begin{cases} \frac{\mathcal{L}_i e_i}{1-P_D P_G+\sum_{j=1}^{m_k} \mathcal{L}_j e_j}, \wedge j \neq 0 \\ \frac{1-P_D P_G}{1-P_D P_G+\sum_{j=1}^{m_k} \mathcal{L}_j e_j} \wedge j = 0 \\ . \end{cases} \end{aligned} \tag{A.0.13}$$

Bibliography

[1] W. Aftab, R. Hostettler, A. De Freitas, M. Arvaneh, and L. Mihaylova. Spatio-temporal gaussian process models for extended and group object tracking with irregular shapes. *IEEE Transactions on Vehicular Technology*, 68(3):2137–2151, March 2019. ISSN 0018-9545. doi: 10.1109/TVT.2019.2891006.

[2] M. A. Alvarez, L. Rosasco, and N. D. Lawrence. Kernels for vector-valued functions: A review. *CoRR, Machine learning*, 2011.

[3] M. Aly. Real time detection of lane markers in urban streets. *Computational Vision Lab, Electrical Engineering, California Institute of Technology*, 2008.

[4] D. C. Andrade, F. Bueno, F. R. Franco, R. A. Silva, J. H. Z. Neme, E. Margraf, W. T. Omoto, F. A. Farinelli, A. M. Tusset, S. Okida, M. M. D. Santos, A. Ventura, S. Carvalho, and R. d. S. Amaral. A novel strategy for road lane detection and tracking based on a vehicle's forward monocular camera. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2018.

[5] Y. Bar-Shalom and L. Xiao-Rong. *Multitarget multisensor tracking: Principles and techniques*. Storrs, CT: Yaakov Bar-Shalom.

[6] Y. Bar-Shalom, X. Rong.Li, and T. Kirubarajan.

[7] Y. Bar-Shalom, K. C. Chang, and H. A. P. Blom. Tracking of splitting targets in clutter using an interacting multiple model joint probabilistic data association filter. *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 2043–2048 vol.2, 1991.

[8] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *IEEE Control Systems*, 29, Issue: 6:82–100, December 2009.

[9] M. Baum and U. D. Hanebeck. Random hypersurface models for extended object tracking. *2009 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 178–183, 2009.

[10] M. Bertozzi and A. Broggi. Gold: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7 (1):62–81, January 1998. ISSN 1057-7149. doi: 10.1109/83.650851.

[11] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, K. Redmill, and U. Ozguner. Lane change and merge maneuvers for connected and automated vehicles: A survey. *IEEE Transactions on Intelligent Vehicles*, 1(1):105–120, March 2016.

[12] C. Bila, F. Sivrikaya, M. A. Khan, and S. Albayrak. Vehicles of the future: A survey of research on safety issues. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1046–1065, May 2017.

[13] H. A. P. Blom and E. A. Bloem.

[14] E. V. Bonilla, K. M. Chai, and C. Williams. Multi-task gaussian process prediction. *Advances in Neural Information Processing Systems 20*, pages 153–160, 2008.

[15] A. Borkar, M. Hayes, and M. T. Smith. A novel lane detection system with efficient ground truth generation. *IEEE Transactions on Intelligent Transportation Systems*, 13:365–374, March 2012. ISSN 1524-9050.

[16] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441.

[17] E. Catmull and R. Rom. A class of local interpolating splines. *Computer aided geometric design. Academic Press.*, pages 317–326, 1974.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.

[19] P. Coulombeau and C. Laurgeau. Vehicle yaw, pitch, roll and 3d lane shape recovery by vision. *IEEE in Intelligent Vehicle Symposium*, 2:619–625, 2002.

[20] B. Csébfalvi. Fast catmull-rom spline interpolation for high-quality texture sampling. *Computer Graphics Forum*, 37(2):455–462, 2018.

[21] X. Dai, A. Kummert, and S. B. Park. A warning algorithm for lane departure warning system. *IEEE*, Intelligent Vehicles Symposium:431–435, 2009.

[22] R. Danescu and S. Nedevschi. Probabilistic lane tracking in difficult road scenarios using stereovision. *IEEE Transactions on Intelligent Transportation Systems*, 10(2):272–282, June 2009.

[23] R. Dang, J. Wang, S. E. Li, and K. Li. Coordinated adaptive cruise control system with lane-change assistance. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2373–2383, Oct 2015.

[24] A. Daniyan, S. Lambotharan, A. Deligiannis, Y. Gong, and W. Chen. Bayesian multiple extended target tracking using labeled random finite sets and splines. *IEEE Transactions on Signal Processing*, 66(22):6076–6091, 2018.

[25] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. Ass. Comp. Match*, 15:11–15, 1972.

[26] Z. Feihu, S. Hauke, C. Chao, B. Christian, and K. Alois. A lane marking extraction approach based on random finite set statistics. *Intelligent Vehicles Symposium (IV)*, pages 1143–1148, 2013.

[27] L. Fletcher, N. Apostoloff, L. Petersson, and A. Zelinsky. Vision in and out of vehicles. *IEEE Intelligent Systems*, 18(3):12–17, May 2003.

[28] V. Gaikwad and S. Lokhande. Lane departure identification for advanced driver assistance. *IEEE Transaction On Inteligent Transportation Systems*, 16:2, April 2015.

[29] C. Galamhos, J. Matas, and J. Kittler. Progressive probabilistic hough transform for line detection. 1:560, 1999.

[30] J. Gama and P. Brazdil. Cascade generalization. *Springer,Machine Learning*, 41:315 – 343, 2000.

[31] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 4th edition, 2017.

[32] R. Gopalan, T. Hong, M. Shneier, and R. Chellappa. A learning approach towards detection and tracking of lane markings. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1088–1098, Sept 2012.

[33] K. Granström, C. Lundquist, and U. Orguner. Tracking rectangular and elliptical extended targets using laser measurements. *14th International Conference on Information Fusion*, pages 1–8, 2011.

[34] K. Granstrom, M. Baum, and S. Reuter. Extended object tracking: Introduction, overview and applications. *CoRR, Computer Vision and Pattern Recognition*, 2016.

[35] Y. Guo, Y. Li, R. Tharmarasa, T. Kirubarajan, M. Efe, and B. Sarikaya. Gp-pda filter for extended target tracking with measurement origin uncertainty. *IEEE Transactions on Aerospace and Electronic Systems*, 55(4):1725–1742, Aug 2019. ISSN 0018-9251. doi: 10.1109/TAES.2018.2875555.

[36] B. Habtemariam, R. Tharmarasa, T. Kirubarajan, D. Grimmett, and C. Wakayama. Multiple detection probabilistic data association filter for multistatic target tracking. *14th International Conference on Information Fusion*, 01 2011.

[37] R. Haralick and L. Shapiro. *Computer and Robot Vision*. Wesley Publishing Company, 1st edition edition, 1992.

[38] J. Hartikainen and S. Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 379–384, 2010.

[39] T. Hirscher, A. Scheel, S. Reuter, and K. Dietmayer. Multiple extended object tracking using gaussian processes. *2016 19th International Conference on Information Fusion (FUSION)*, pages 868–875, 2016.

[40] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991. doi: 10.1017/CBO9780511840371.

[41] M. Huber. Recursive gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45, 03 2014. doi: 10.1016/j.patrec.2014.03.004.

[42] J. Hur, S. N. Kang, and S. W. Seo. Multi-lane detection in urban driving environments using conditional random fields. pages 1297–1302, June 2013. ISSN 1931-0587. doi: 10.1109/IVS.2013.6629645.

[43] P. L. J. l. Yang and H. w. Ge. Extended target shape estimation by fitting b-spline curve. *Journal of Applied Mathematics*, 2014.

[44] R. Kastner, T. Michalke, J. Adamy, J. Fritsch, and C. Goerick. Task-based environment interpretation and system architecture for next generation ADAS. *IEEE Intelligent Transportation Systems Magazine*, 3(4):20–33, October 2011.

[45] H. Kaulbersch, J. Honer, and M. Baum. A cartesian b-spline vehicle model for extended object tracking. pages 1–5, 2018.

[46] Z. Kim. Robust lane detection and tracking in challenging scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):16–26, March 2008.

[47] N. Kiryati, Y. Eldar, and A. Bruckstein. A probabilistic Hough transform. *Pattern Recognition*, 24(4):303 – 316, 1991.

[48] J. W. Koch. Bayesian approach to extended object and cluster tracking using random matrices. *IEEE Transactions on Aerospace and Electronic Systems*, 44 (3):1042–1059, 2008.

[49] D. H. U. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. *SIGGRAPH Comput. Graph.*, 18(3):33–41, jan 1984.

[50] H. Kong, J. Y. Audibert, and J. Ponce. General road detection from a single image. *IEEE Transactions on Image Processing*, 19(8):2211–2220, August 2010.

[51] J. G. Kuk, J. H. An, H. Ki, and N. I. Cho. Fast lane detection amp: Tracking based on hough transform with reduced memory requirement. *13th International IEEE Conference on Intelligent Transportation Systems*.

[52] F. Kunz, D. Nuss, J. Wiest, H. Deusch, S. Reuter, F. Gritschneder, A. Scheel, M. Stübler, M. Bach, P. Hatzelmann, C. Wild, and K. Dietmayer. Autonomous driving at ulm university: A modular, robust, and sensor-independent fusion approach. *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 666–673, 2015.

[53] J. Lan and X. R. Li. Tracking of maneuvering non-ellipsoidal extended object or target group using random matrix. *IEEE Transactions on Signal Processing*, 62(9):2450–2463, 2014.

[54] C. Lee and J. Moon. Robust lane detection and tracking for real-time applications. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–6, 2018.

[55] Y. Lee and H. Kim. Real-time lane detection and departure warning system on embedded platform. pages 1–4, September 2016.

[56] D. Lerro and Y. Bar-Shalom. Automated tracking with target amplitude information. *1990 American Control Conference*, pages 2875–2880, May 1990.

[57] Q. Li, L. Chen, M. Li, S. Shaw, and A. Nüchter. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Transactions on Vehicular Technology*, 63(2):540–555, Feb 2014.

[58] B. Lin, Y. Lin, L. Fu, P. Hsiao, L. Chuang, S. Huang, and M. Lo. Integrating appearance and edge features for sedan vehicle detection in the blind-spot area. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):737–747, June 2012.

[59] C. Lundquist, U. Orguner, and F. Gustafsson. Extended target tracking using polynomials with applications to road-map estimation. *IEEE Transactions on Signal Processing*, 59(1):15–26, 2011.

[60] Y. Ma, V. Havyarimana, J. Bai, and Z. Xiao. Vision-based lane detection and lane-marking model inference: A three-step deep learning approach. *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 183–190, 2018.

[61] M. Mallick and B. L. Scala. Comparison of single-point and two-point difference track initiation algorithms using position measurements. *Acta Automatica Sinica*, 34(3), March 2008.

[62] D. Mušicki, R. Evans, and S. Stankovic. Integrated probabilistic data association. *IEEE Transactions on Automatic Control*, 39(6):1237–1241, 1994.

[63] M. Nieto, L. Salgado, F. Jaureguizar, , and J. Cabrera. Stabilization of inverse perspective mapping images based on robust vanishing point estimation. *IEEE Intelligent Vehicles Symposium, Istanbul, Turkey*, pages 13–15, 2007.

[64] M. Nieto, L. Salgado, F. Jaureguizar, and J. Arrospide. Robust multiple lane road modeling based on perspective analysis. *2008 15th IEEE International Conference on Image Processing*, pages 2396–2399, 2008.

[65] J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao. Robust lane detection using two-stage feature extraction with curve fitting. *Pattern Recognition*, 59:225 – 233, 2016.

[66] J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao. Robust lane detection using two-stage feature extraction with curve fitting. *Pattern Recognition*, 59(1):225 – 233, 2016.

[67] M. Oliveira, V. Santos, and A. D. Sappa. Multimodal inverse perspective mapping. *Information Fusion*, 24:108 – 121, 2015. ISSN 1566-2535.

[68] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial as deep: Spatial cnn for traffic scene understanding. *CoRR, Computer Vision and Pattern Recognition*, 2017.

[69] C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. MIT Press, Cambridge, 1st edition edition, 2006.

[70] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965. doi: 10.2514/3.3166.

[71] M. Reichenbach, L. Liebischer, S. Vaas, and D. Fey. Comparison of lane detection algorithms for adas using embedded hardware architectures. pages 48–53, October 2018.

[72] W. Ruan and E. L. Miller. Ensemble multi-task gaussian process regression with multiple latent processes. *CoRR, Machine learning*, 2017.

[73] N. Sanchez, J. Alfonso, J. Torres, and J. M. Menendez. ITS-based cooperative services development framework for improving safety of vulnerable road users. *IET Intelligent Transport Systems*, 7(2):236–243, June 2013.

[74] S. Sarkka, A. Solin, and J. Hartikainen. Spatiotemporal learning via infinite-dimensional bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. *IEEE Signal Processing Magazine*, 30(4):51–61, July 2013. ISSN 1053-5888. doi: 10.1109/MSP.2013.2246292.

[75] M. Schürch, D. Azzimonti, A. Benavoli, and M. Zaffalon. Recursive estimation for sparse gaussian process regression. *CORR Machine Learning*, 2019.

[76] F. Septier, A. Carmi, and Godsill.

[77] A. Solin and S. Särkkä. Infinite-dimensional bayesian filtering for detection of quasiperiodic phenomena in spatiotemporal data. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 88:052909, 11 2013. doi: 10.1103/PhysRevE.88.052909.

[78] W. Song, Y. Yang, M. Fu, Y. Li, and M. Wang. Lane detection and classification for forward collision warning system based on stereo vision. *IEEE Sensors Journal*, 18(12):5151–5163, June 2018.

[79] J. K. Tugnait. Tracking of multiple maneuvering targets in clutter using multiple sensors, imm, and jpda coupled filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 40(1):320–330, 2004.

[80] B. Tuncer, M. Kumru, and E. Özkan. Extended target tracking and classification using neural networks. *CoRR, Machine learning*, 2020.

[81] TuSimple. Tusimple benchmark. 2017. Available at http://benchmark.tusimple.ai.

[82] N. Wahlstrom and E. Ozkan. Extended target tracking using gaussian processes. *IEEE Transactions on Signal Processing*, 63(16):4165–4178, 2015.

[83] D. Wang, J. Xue, D. Cui, and Y. Zhong. A robust submap-based road shape estimation via iterative gaussian process regression. *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1776–1781, 2017.

[84] Y. Wang. *Smoothing Splines: Methods and Applications*. Chapman and Hall/CRC, New York, 1st edition edition, June 2011.

[85] Y. Wang, E. K. Teoh, and D. Shen. Lane detection and tracking using b-snake. *Image and Vision Computing*, 22(4):269 – 280, 2004.

[86] Z. Wang, W. Ren, and Q. Qiu. Lanenet: Real-time lane detection networks for autonomous driving. *CoRR, Machine learning*, 2018.

[87] S. Wu, S. Decker, P. Chang, T. Camus, and J. Eledath. Collision sensing by stereo vision and radar sensor fusion. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):606–614, Dec 2009.

[88] W. Yang, B. Fang, and Y. Y. Tang. Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):755–766, May 2018.

[89] J. H. Yoo, S. W. Lee, S. K. Park, and D. H. Kim. A robust lane detection method based on vanishing point estimation using the relevance of line segments. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–13, 2017.

[90] H. Yu, W. An, and R. Zhu. Extended target tracking and feature estimation for optical sensors based on the gaussian process. *Sensors (Basel)*, 2019.

[91] C. Yuan, H. Chen, J. Liu, D. Zhu, and Y. Xu. Robust lane detection for complicated road environment based on normal map. *IEEE Access*, 6:49679–49689, September 2018.

[92] M. Yue, X. Hou, X. Zhao, and X. Wu. Robust tube-based model predictive control for lane change maneuver of tractor-trailer vehicles based on a polynomial trajectory. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–9, 2018.

[93] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen. A novel lane detection based on geometrical model and gabor filter. *IEEE in Intelligent Vehicles Symposium (IV)*, pages 59–64, 2010.

[94] B. Zhu, S. Yan, J. Zhao, and W. Deng. Personalized lane-change assistance system with driver behavior identification. *IEEE Transactions on Vehicular Technology*, pages 1–1, 2018.