Laser-Based Detection and Tracking of Dynamic Objects

Dominic Zeng Wang

St John's College

Supervisors:

Professor Paul Newman and Professor Ingmar Posner

Mobile Robotics Group

Department of Engineering Science

University of Oxford

October 2014

Dominic Zeng Wang
St John's College

Doctor of Philosophy
October 2014

Laser-Based Detection and Tracking of Dynamic Objects

Abstract

In this thesis, we present three main contributions to laser-based detection and tracking of dynamic objects, from both a model-based point of view and a model-free point of view, with an emphasis on applications to autonomous driving. A segmentation-based detector is first proposed to provide an end-to-end detection of the classes car, pedestrian and bicyclist in 3D laser data amongst significant background clutter. We postulate that, for the particular classes considered, solving a binary classification task outperforms approaches that tackle the multi-class problem directly. This is confirmed using custom and third-party datasets gathered of urban street scenes. The sliding window approach to object detection, while ubiquitous in the Computer Vision community, is largely neglected in laser-based object detectors, possibly due to its perceived computational inefficiency. We give a second thought to this opinion in this thesis, and demonstrate that, by fully exploiting the sparsity of the problem, exhaustive window searching in 3D can be made efficient. We prove the mathematical equivalence between sparse convolution and voting, and devise an efficient algorithm to compute exactly the detection scores at all window locations, processing a complete Velodyne scan containing 100K points in less than half a second. Its superior performance is demonstrated on the KITTI dataset, and compares commensurably with state of the art vision approaches. A new model-free approach to detection and tracking of moving objects with a 2D lidar is then proposed aiming at detecting dynamic objects of arbitrary shapes and classes. Objects are modelled by a set of rigidly attached sample points along their boundaries whose positions are initialised with and updated by raw laser measurements, allowing a flexible, nonparametric representation. Dealing with raw laser points poses a significant challenge to data association. We propose a hierarchical approach, and present a new variant of the well-known Joint Compatibility Branch and Bound algorithm to handle large numbers of measurements. The system is systematically calibrated on real world data containing 7.5K labelled object examples and validated on 6K test cases. Its performance is demonstrated over an existing industry standard targeted at the same problem domain as well as a classical approach to model-free tracking.

Statement of Authorship

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Dominic Zeng Wang, St John's College

Funding

The work described in this thesis was in large funded by the Clarendon Fund. The author was also in receipt of funding from Nissan Motor Co., Ltd. for the past year.

Acknowledgements

First, I would like to say my biggest "thank you" to my supervisors Professor Paul Newman and Professor Ingmar Posner. During the course of the past four years, there were moments of joy and moments of bitterness. It is their never-ending support and encouragements that provided me with the confidence to pursue directions to my heart's desire. Without their guidance, I could never have hoped to achieve what I have today.

I would also like to pass my thanks to all the lovely folks at the Mobile Robotics Group where I worked, lived and laughed in the past four years, including both the veterans, the past members and the new comers. It is a great pleasure and honour to work amongst such a fun and ingenious bunch of people. It is a place filled with fresh ideas.

Finally, thanks to my wife Yurika, for her love and support, especially when the writing of this thesis coincided with such an important moment of our lives. Without her help and understanding, I may never have glimpsed the light at the end of this tunnel. Thanks also go to my mother, who traveled all the way from the other side of the globe to provide help during the last days of my writing when I needed it the most. Also, it is the long-term love, support and understanding of my family during all these years behind the scenes that have given me the chance to devote myself to my passion. Thanks to you all.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 The Challenge

Autonomous driving has become a prominent application domain for robotics research. This is witnessed by a cornucopia of publications in this area (McNaughton et al., 2011; Levinson et al., 2007; Huang and Teller, 2010). The success of the DARPA Grand- (Thrun et al., 2006) and Urban Challenges (Urmson et al., 2008) as well as Google's endeavour to promote autonomous driving (Fairfield and Urmson, 2011) has heightened expectations that autonomous cars will be able to operate in environments of realistic complexity. Our community's aspiration to create self-driving cars has further served to highlight the importance of – and to focus efforts within – environment understanding.

Much research effort is being expended on the detection and classification of objects pertinent to navigating a realistic road environment, both using vision and laser data. In particular, it is absolutely critical to be able to identify and subsequently predict the motion states of *dynamic* objects on the road for an autonomous vehicle to make safe decisions and traverse successfully through the busy urban streets today.

Common players on the road such as cars, pedestrians and bicyclists exhibit a large variety of motion patterns. Their appearance changes constantly as they manoeuvre in the environment, interacting with neighbouring objects in a complex manner. Even within a single object category, the intra-class variation is significant – consider the diversity of human body shapes and the colours and appearance of the clothes they wear, or the different sizes and designs of cars. Even for a single object instance, non-rigid objects such as pedestrians may deform giving rise to a changing appearance regardless of what sensor modality they are being observed in.

The challenge is greater in the context of autonomous driving because in this scenario, the sensors mounted on the vehicle are *themselves* moving through the environment, making all observations to any objects *relative* to the moving sensor. In addition to making motion estimation of objects challenging without an estimate of the sensor's own motion, this also creates large changes to the appearance of *static* objects due to viewpoint changes and occlusion, exhibiting dynamic-like behaviours. This leads to significant ambiguity for dynamic object detection.

1.2 The Common Approaches

Existing approaches to the detection and tracking of dynamic objects differ significantly in the sensor modality used, the emphasis on *detection* or *tracking*, and the specific underlying method used.

1.2.1 Variation on Sensor Modalities

The most popular and most successful sensor modalities for object detection and tracking deployed in the robotics community are arguably cameras and lasers. Camera-based detection methods benefit greatly from the large body of work studied in the Computer Vision community of generic object detection and tracking. The advan-

(a) (b)

Figure 1.1: Our research platforms deployed in this thesis. (a) The Bowler Wildcat research platform, equipped with a Velodyne HDL-64E S2 3D scanner (highlighted in red), and a Bumblebee 2 stereo camera (highlighted in green). (b) The RobotCar autonomous driving platform equipped with a SICK LDMRS laser range finder (highlighted in cyan).

tages in vision lie in the fact that objects are described by dense texture information, making appearance a rich cue to suggest object locations. However, the projective nature of the image forming process means information loss is inevitable. In particular, it is challenging to recover the scale and depth of a given object from a single image alone. Methods using stereo cameras such as the Bumblebee 2 camera depicted in Figure 1.1(a) may recover depth and scale by triangulation, however the reconstruction becomes noisy as the distance to the camera increases. Hence tracking is usually constrained to the projected motion within the image plane. Because of these limitations, vision methods tend to focus on the problem of object *detection*.

A laser measures the environment by directing a beam of light (usually in the infrared range of spectrum) at a particular angle, and observing the amount of time for the light to bounce back from a solid surface. Based on the time of travel, the distance to the reflecting surface can be obtained, together with the reflectivity of the surface calculated based on how much light that has been bounced back. The laser beam may then sweep through a set of different angles to give a complete scan. Each scan is thus composed of a set of angle and range pairs (also possibly with

the reflectance value), from which point locations (in cartesian frame if so desired) can be worked out. A laser scanner is thus commonly referred to as a range-and-bearing device. An example laser scanner is the highlighted SICK LDMRS scanner in Figure 1.1(b).

If multiple beams are placed at different angles of elevation, and rotated around a common vertical axis, points gathered by each laser beam in a single revolution may be combined to form a *3D* laser scan. The Velodyne HDL-64E S2 scanner is one of such 3D scanners as depicted in Figure 1.1(a). The Velodyne HDL-64E S2 scanner has 64 vertically spaced laser beams. The 3D laser scan formed is a collection of points in the 3D space, and is sometimes referred to as a "point cloud".

Because distances are directly measured in laser data, depths and scales are explicit, making tracking a dynamic object's motion in the 3D world more readily achievable. However, compared with vision methods, laser data is inherently sparse, the only appearance information available is the reflectance value of the reflecting surface, which is far less descriptive than the dense, full colour texture information available in an image. Because of these difficulties, laser-based methods tend to focus on *shape* information, and operation directly on segments of points.

1.2.2 Variation on Methods

Methods on dynamic object detection and tracking may be broadly divided into two categories. The model-based methods, and model-free methods.

In model-based methods, a set of object classes is first identified, for example, cars, pedestrians and bicyclists. Then for each class, class-specific patterns are extracted, either by a supervised machine learning approach to learn shape- or appearance-based object detectors, or derive class-specific motion models for accurate prediction of object motion.

In model-free methods, on the other hand, no prior knowledge is assumed on

the object being sought after, other than that they *move*. Thus detection is purely *motion-driven*. In this case, successful detection relies on a good motion estimate. Thus model-free methods tend to focus on object *tracking*.

Both model-based and model-free methods have their advantages. Model-based methods, because detection is usually based on shape or appearance cues (rather than motion), are able to detect both *instantaneously* moving objects and stationary objects of the specified class which may *potentially* move. Model-free methods, on the other hand, because of the lack of a semantic model, cannot identify stationary objects.

Given the knowledge of the object class being sought after, the detection system may *actively* look for the existence of such an object, reducing the chance of missing an object because of, for example, segmentation error. Also, a sophisticated motion model specifically designed for the object class may be used for tracking to achieve better motion prediction. Model-free methods, in the absence of semantic knowledge, have to resort to generic segmentation algorithms and generic motion models.

However, model-free methods detect and track dynamic objects regardless of their identities. The users of the road are far from common classes such as cars, pedestrians and bicyclists, for examples, buses, trucks, motorcycles are also abundant. To name all possible dynamic objects one may encounter in an urban driving scenario including the most unexpected cases is challenging if not impossible, not alone designing class-specific models to each of them. The model-based approaches when faced with such a real-life diversity of object classes, soon become unscalable. This is the main advantage of model-free methods over their model-based counterparts.

1.3 Our Approach

The ability to tell the precise distance to a moving object, and estimate its velocity in the 3D world with relative ease is an important requirement of an autonomous driving system. While the 3D reconstruction accuracy of stereo vision-based methods is working their way to being commensurate with laser measurements, lasers remain to be the most reliable modality for object detection and tracking in this application domain. For this reason, in this thesis, we restrict our attention to laser-based methods.

Figure 1.1 shows our research platforms deployed in this thesis: the Bowler Wildcat equipped with a Velodyne HDL-64E S2 3D laser scanner, and the RobotCar – a modified Nissan LEAF – equipped with the SICK LDMRS multilayer 2D laser scanner designed for object tracking on automotive platforms. The works described in this thesis cover our contributions to both the model-based and model-free ends of dynamic object detection and tracking.

There are three parts to the main contributions of this thesis:

1. An end-to-end pipeline is proposed that, when fed by a raw 3D laser scan, produces distinct clusters of points corresponding to object instances of the classes *car*, *pedestrian* and *bicyclist*. In contrast to other segmentation-based methods, we do not rely on a correct segmentation to be given, instead, we start with an over-segmentation, and group the segments into distinct object instances once their identities are known. In particular, we postulate that, for the specific classes considered, solving a binary classification task (i.e. separating the data into foreground and background first) outperforms approaches that tackle the multi-class problem directly. This is confirmed using custom and third-party datasets gathered of urban street scenes. We also present a new variant of clustering algorithms based on the Euclidean Minimum Span-

ning Tree with a RANSAC-based edge selection criterion for robustly grouping segments to object instances.

2. A novel sliding window object detector in pure 3D is proposed. We demonstrate its feasibility by taking full advantage of the sparse nature of 3D point clouds, and prove the mathematical equivalence of sliding window detection with linear classifiers and voting on a sparse feature grid. We then show both its efficiency and superior performance on a large scale public benchmarking dataset, and compare its performance quantitatively with the previously proposed segmentation-based detector and qualitatively with the best vision-based detectors to-date on the *car* class. To the best of our knowledge, this is the first time the sliding window approach to object detection is taken to 3D data.

3. A new approach to detection and tracking of moving objects with a *2D* laser scanner is presented. Objects are modelled with a set of rigidly attached sample points along their boundaries whose positions are initialised with and updated by *raw* laser measurements, thus allowing a nonparametric representation that is capable of representing objects independent of their classes and shapes. In addition, we propose a hierarchical data association algorithm to assign raw laser measurements to potential state updates, and present a variant of the Joint Compatibility Branch and Bound (JCBB) algorithm (Neira and Tardos, 2001) that is suitable for associating a large number of measurements, and derive an alternative set of recursive update rules based on the triangular form representation of positive definite matrices for its efficient and numerically stable computation.

1.4 Thesis Outline

The thesis is structured as follows. First, we review existing approaches to object detection and tracking in Chapter 2, including both vision-based and laser-based methods, from both the model-based and model-free perspectives. Then, we take a look at model-based detection of *potentially* moving objects. The most important players on the road are cars, pedestrians and bicyclists. Our proposed segmentation-based system for their detection will be the subject of Chapter 3.

The sliding window approach to object detection is arguably the most popular method for object detection in the Computer Vision community (as will be discussed in Section 2.2). However, it received little attention for laser-based object detection, possibly due to its perceived computational inefficiency. We argue in Chapter 4 that by exploiting the sparsity of the problem, the sliding window approach can be made efficient for object detection in 3D, and demonstrate its superior performance.

To raise the situational awareness of the autonomous vehicle beyond the classes *car*, *pedestrian* and *bicyclist*, we turn our attention to model-free tracking of *all* moving objects in Chapter 5, and propose a nonparametric object representation that is capable of representing objects of arbitrary shapes and classes.

Finally, in Chapter 6, we conclude the thesis, summarise our contributions, and point at possible future directions to take advantage of the insights gained in our studies for further improvements.

1.5 Publications

The material presented in Chapter 3 has been published in (Wang et al., 2012). The model-free approach to dynamic object detection and tracking described in Chapter 5 has been presented at the International Symposium on Robotics Research (ISRR) 2013 as (Wang et al., 2013a), and was subsequently invited for submission

to the ISRR 2013 Special Issue of the International Journal of Robotics Research (IJRR). The journal submission titled "Model-Free Detection and Tracking of Dynamic Objects with 2D Lidar" is currently under review. A publication based on the material covered in Chapter 4 is under preparation.

Chapter 2

Object Detection and Tracking

2.1 Introduction

In this chapter, we survey existing approaches to object detection and tracking, including both successful approaches to image-based object detection and laser-based object detection and tracking. Again, our emphasis in this chapter will be the detection and tracking of *dynamic* objects.

By *dynamic* objects, we include both *potentially* moving objects and the objects that are *actually* moving. The *potentially* moving objects are inevitably defined in a semantic sense, i.e. objects of certain classes that are dynamic in nature such as the classes *car*, *pedestrian* and *bicyclist*. Even instantaneously stationary, the state of these objects may change unexpectedly, therefore it is important to be aware of them in all situations. The *potentially* moving objects, given the semantic nature of their definition, are more suited to model-based methods.

On the other hand, the *actually* moving objects include *anything* that has some instantaneous absolute speed (either linear or rotational), *independent* of shapes and classes. Thus this class of moving objects are more suited to model-free methods.

2.2 Image-Based Object Detection

Image-based object detectors commonly involve sliding a detection window through the image at a set of possible object locations. Because of the ambiguity to the object scale due to the projective nature of the image forming process as discussed in Section 1.2.1, the searching process has to be repeated at each possible object scale.

The human face detectors proposed by Sung and Poggio (1998) and Viola and Jones (2001) are among the earliest examples of sliding window object detectors. Sung and Poggio (1998) train a model for face patterns and a model for non-face patterns by fitting a fixed number of Gaussian clusters to the training data with a modified K-means algorithm. Then each window location is converted to a feature vector by computing two distance measures for each of the clusters in the face and non-face models respectively. The feature vector is then passed through a neural network to classify whether the current window contains a face or not. Viola and Jones (2001), on the other hand, apply the AdaBoost algorithm to select weak classifiers based on simple features extracted from example face and non-face image patches during training. A window is then slid across all locations and scales during testing to use the trained AdaBoost classifier to detect faces. The features are defined as differences of sums of rectangular subregions in the local patch, the Integral Image is proposed for the first time to compute the features efficiently. Both methods handle scale ambiguity by passing windows of different scales through the test image.

Dalal and Triggs (2005) propose a novel dense feature referred to as the Histograms of Oriented Gradients (HOG) for human detection. Here each test image is first converted into a dense feature grid, and a linear SVM classifier is used to classify each window location for the existence of a human or not. Felzenszwalb et al. (2010) extend this work to include deformable parts attached to the central

object window. Locations of the the deformable parts are included as latent variables and optimised away when computing the detection score for the central object window. In both works, searching through object scales are taken care of by scaling the *image* in contrast to scaling the detector compared with the methods to face detection above.

Finally, we mention an important approach to object detection *different* to the commonly encountered sliding window approach in vision. Variants of the Implicit Shape Model (ISM) proposed by Leibe and colleagues (Leibe et al., 2008; Lehmann et al., 2011) take a feature centric point of view. During training, sparse features are extracted around the object with an interest point detector, then a codebook of visual words are built. In addition, for each word, the relative location and scale of the object at each occurrence of the word in the training set are noted. Then when presented with a new test image, the image is first processed with the same interest point detector to extract sparse features, and the features are then matched to the codebook. Each matched feature then places "votes" in the continuous search space of all possible object locations and scales based on the remembered object locations and scales relative to the visual word that has been observed in the training set before. The object in the image (if any) are finally found by a mode finding algorithm (e.g. mean shift) from the distribution of the votes cast by the matching features in the image as the object location and scale that have the most support.

2.3 Laser-Based Object Detection in 3D

In this section we turn our attention to object detection methods in 3D laser data. Existing work on object detection and recognition in 3D laser data can be coarsely divided into two categories.

The first commonly assumes that point clouds representing *entire* objects have

already been segmented out of the data and, therefore, focus mainly on classification. Examples include (Teichman et al., 2011), who in addition assume not only the availability of segmented object clusters but that the segments are also correctly tracked across frames and classifies a complete track of an object into one of the *car*, *pedestrian*, *bicyclist* and *background* classes, and have later extended the framework into a semi-supervised set-up (Teichman and Thrun, 2011). Lai et al. (2011b) combine advantages of both shape and appearance with a Kinect-style sensor to classify indoor objects using sparse distance metric learning with a Group-Lasso regulariser. In this case the segmentation task is facilitated by the controlled environment the objects are placed in (see (Lai et al., 2011a) for a description of their dataset). Endres et al. (2009) on the other hand take an unsupervised approach to discover object categories in the presented segments using Latent Dirichlet Allocation (LDA).

The segmentation of desirable objects from amongst an often large amount of background clutter in 3D laser data is a pivotal precursor to such systems. Existing work includes that by Douillard et al. (2011), where the existence of a ground plane is assumed and object segments are derived in an unsupervised fashion from non-ground data only. Klasing et al. (2008) perform clustering based on Euclidean distance between individual laser points, implicitly assuming that objects are not connected by scene clutter.

The second class of methods label a scene directly into regions belonging to object classes (with possibly a *background* class), but do not distinguish separate object instances. Anguelov et al. (2005), for example, take a supervised approach based on a Markov Random Field (MRF) using local features computed at individual data points to produce globally consistent labels. Triebel et al. (2010) follow an unsupervised approach, where a scene is first over segmented, and the resulting segments are taken as the basic elements of manipulation. Then two graphs are constructed in the Euclidean space and feature space (with one feature vector

extracted per segment) respectively. Then probabilistic inference is performed on the graph in feature space by the use of a Conditional Random Field (CRF), the results are in turn used to perform a second inference stage in the Euclidean space to determine the final class labels. The scene segmentation obtained often represent object categories that correspond to repeated patterns.

Both classes of approaches are notably different to common practices in vision-based object detection surveyed in the previous section. Object detection in 3D laser is largely driven by segmentation.

2.4 Laser-Based Object Detection and Tracking in 2D

In this section, we survey the bodies of work covering object detection *and tracking* in 2D laser data, and focus on approaches with an emphasis on *dynamic* objects. Whereas the detection frameworks discussed in the previous sections fall largely within the class of model-based methods, the approaches we examine in this section cover both the model-based and model-free classes of methods.

2.4.1 Model-Based Approaches

In a model-based approach to object detection and tracking, the class of the objects to be detected is known a priori, and objects are first detected based on a parametric model of its shape and then tracked as separate entities. Examples of this class of methods include the work by Granström (2012), who detects and tracks rectangular and elliptical targets with a Probability Hypothesis Density (PHD) filter. The line of works by Arras et al. (2007, 2008) focus on people detection. They train a boosting classifier to detect legs of people and the detected legs are grouped into

individual persons and tracked with a Multi-Hypothesis Tracker (MHT). The work by Schulz et al. (2001) is similar in that they also identify legs of people in 2D range scans and apply the Joint Probabilistic Data Association Filter (JPDAF) for robust tracking. However, in their work, only moving persons are considered. Topp and Christensen (2005) extend the work by removing this restriction, and extends the model for a person to also include people whose legs are not directly visible. Zhao and Thorpe (1998), on the other hand, restrict their attention to vehicles. In their work, an Interactive Multiple Model (IMM) filter is proposed for vehicle tracking that consists of three different motion models to cover a full range of motions for the tracked vehicles. Another example of model-based vehicle detection and tracking is the work presented by Vu and Aycard (2009), where a box model for vehicles is assumed, and detection and tracking are solved simultaneously by optimising over the best trajectories of the vehicles over a temporal sequence of laser scans using a Data-driven Markov Chain Monte Carlo (DDMCMC) algorithm.

2.4.2 Model-Free Approaches

In a *model-free* approach, detection is based on motion cues. Examples of model-free dynamic obstacle detection and tracking include systems deployed in the DARPA Urban Challenge (Leonard et al., 2008; Mertz et al., 2013), which usually function by first segmenting measurements from multiple laser range finders, and then extracting geometric features from the segments. These geometric features are used to compile a list of object hypotheses, dynamic objects are then extracted as objects having a significant manoeuvring speed. Another body of work jointly estimates a static map of the environment alongside the detection and tracking of moving objects. Examples include Toyota's tracking system (Miyasaka et al., 2009) and Wang's system (Wang et al., 2003) (later extended by Montesano et al. (2005) and Vu et al. (2007)) that combines SLAM with dynamic object tracking. Both approaches take

an occupancy grid representation of the environment, and use knowledge of occupancy probabilities from the map to propose likely moving object detections. Biswas et al. (2002) also take an occupancy grid representation, and detect non-stationary objects by map differencing. The individual objects are then identified with an EM algorithm. Wolf and Sukhatme (2005) keep two occupancy grid maps, one for the static part and one for the dynamic part of the environment. However, the focus there is mapping in dynamic environments, dynamic objects are not tracked as separate entities. Yang and Wang (2011) propose a system that jointly estimates the vehicle pose and moving object detections using a variant of RANSAC, and track merging and splits are handled via a decision tree based on spatiotemporal consistency tests. Work by Tipaldi and Ramos (2009) and van de Ven et al. (2010) focuses on the detection part of the problem, and formulates it under a joint Conditional Random Field (CRF) framework for solving both the data association and moving object detection problems. Finally, the work by Hahnel et al. (2003) is also relevant where the authors formulate an EM algorithm to solve for a set of hidden indicator variables for each laser point to determine whether it is static or dynamic.

2.5 Conclusions and Discussions

The detection and tracking of dynamic objects is a challenging problem. Existing state of the art approaches tackle different aspects of its full complexity by making simplifying assumptions. For example, most of the model-based approaches described above assume dynamic players of the world fall within a set of fixed object classes, and their appearances are captured by a set of fixed canonical models. In an environment where the assumptions taken by a perception system hold with high probability, the system is expected to function well. In a different environment, different assumptions need to be made to handle the specific situation at hand. For

example, a deer detector (Zhou and Wang, 2011) is required if deer are frequently occurring enough to warrant attention. In a model-free approach, simplifying assumptions are usually made to the motion of the objects. For examples, objects are usually assumed to follow motions independent of each other. Without the knowledge of the object class, complex interactions between the objects are usually difficult to capture.

In the first parts of the thesis, we make the assumption that the dynamic objects of interest in an urban driving scenario are dominated by a set of fixed semantic objects such as the classes *car*, *pedestrian* and *bicyclist*, and focus on their detection. Existing approaches for object detection with 3D laser sensors taking similar assumptions commonly follow a segmentation-based approach. In Chapter 3 we contribute to the state of the art by studying the effectiveness of running multi-class classification at the level of pre-segments, and document our findings that a binary classification at the segment-level performs better for the task because segments (parts) of objects do not contain enough information to reveal its class identity.

Segmentation-based approaches commonly fail of segmentation errors due to unmodelled close interactions between objects. Chapter 4 provides a solution to this problem by bypassing the segmentation step all together – the object location is search exhaustively in the 3D space. In addition, we show that this apparently expensive operation can be done very efficiently.

In Chapter 5, we make the assumptions that objects consist of anything that has an absolute motion, and that their motions are constrained on the 2D ground plane, and, finally, objects can be well-modelled as 2D rigid bodies. We contribute to the state of the art by giving a novel representation of objects based on raw sensor measurements under these assumptions that models well the variety of shapes and classes of objects in a generic fashion.

Other authors focus on other aspects of the problem that fall outside the as-

sumptions we make in this thesis. For example, the aforementioned work by Zhou and Wang (2011) focuses on the detection of deer. Work by Luber and Arras (2013) focuses specifically on the modelling of pedestrians, pedestrian groups and their interactions. A large body of recent approaches study the interactions between objects (Pellegrini et al., 2009; Yao and Fei-Fei, 2012; Koppula and Saxena, 2013; Sun et al., 2014), which are not modelled explicitly by the methods described in this thesis.

Chapter 3

Segmentation-Based Detection

3.1 What *Could* Move?

In this chapter, we take a model-based point of view, and focus on detecting *potentially* dynamic objects – that is objects which *could* move – since their presence and potential change of state will influence the planning of actions and trajectories. We present an end-to-end system that detects instances of cars, pedestrians and bicyclists from a raw stream of 3D laser data for autonomous driving applications.

The point detections produced by a 3D laser scanner is often unstructured, and at its raw data stream format is merely a collection of three-dimensional position values that do not contain any semantic information about the scene. It is important, therefore, to be able to extract semantic information from such collections of point locations by examining the *shape* of local point distributions. We wish to infer the existence and location of operational-critical objects in the scene, achieving the aim of detection.

However, before determining the identity of a particular subset of points contained in the scene that may correspond to an object of interest, the interesting subset itself must first be identified from the unstructured collection of points pro-

duced by the sensor which may contain a large amount of background clutter that is irrelevant to the system. Thus before solving the *classification* problem, the *segmentation* problem needs to be addressed.

While a significant body of work targeted at the *classification* task exist, they often assume (explicitly or otherwise) that a suitable segmentation of the 3D point cloud into *complete entities* of interest is already available (Teichman et al., 2011) or is straight-forward to obtain (Lai et al., 2011b). However, obtaining such a segmentation is widely acknowledged to be a hard problem (Teichman et al., 2011; Lai et al., 2011b; Endres et al., 2009) since the number of objects in the scene is commonly not known and only a small proportion of the data contain relevant class information.

In this chapter, we restrict ourselves to the detection of *cars*, *pedestrians* and *bicyclists* directly from an unstructured stream of 3D laser data obtained from sensors commonly deployed on autonomous vehicles, using shape information alone. Our objective here is to group salient subsets of the raw data stream into contiguous and *complete* entities corresponding to objects of interest without prior knowledge of the number and location of the objects present. Since we have knowledge of the object classes of interest (and the set of them is comparatively small) we employ a supervised approach. We investigate the application of graph-based techniques to the problem, and establish that, for the specific classes considered in this work, solving a binary classification task (i.e. separating the data into foreground and background first) outperforms approaches that tackle the multi-class problem directly.

First, we introduce the graph-based clustering algorithm used in this work in Section 3.2. A system pipeline for the extraction of foreground data from a stream of raw 3D laser points is then detailed in Section 3.3. After that, the performance of the proposed system is evaluated in Section 3.4 and finally we conclude the chapter in Section 3.5.

3.2 Graph-Based Clustering

For the purposes of *detection*, each *instance* of objects (entities) of interest must be correctly identified in the scene. However, the number of such objects existing in a single scene is often unknown a priori. In this section we formulate this problem as a clustering task and present an algorithm that adapts a popular approach in the literature to cluster the scene into instances of single entities independent of the number of objects contained in the scene.

Unsupervised data clustering has been an active area of research for decades and many methods exist which circumvent the lack of prior information, such as the number of clusters present. Variational Bayesian methods (Bishop, 2006), for example, provide an attractive mechanism but are often plagued with convergence issues. Jenssen et al. (2003) use an information theoretic measure for model selection to determine the optimal number of clusters from amongst various possibilities.

The graph-based segmentation algorithm developed by Shi and Malik (2000) in its basic form solves a binary clustering problem by optimising a measure called the normalised cut, which is a normalised measure of inter-cluster similarity. It is shown (Shi and Malik, 2000) that minimising this measure of normalised cut simultaneously maximises a corresponding measure of intra-cluster similarity. The proposed algorithm optimises the normalised cut approximately via a generalised eigen-decomposition of the Laplacian matrix of the graph. Although in its basic form, the Normalised Cut algorithm clusters points into two partitions, an extension of the algorithm to the N-clusters case functions by recursively applying two-way partitions starting from the full point set until the normalised cuts of the sub-partitions reach a certain maximum allowed value (Shi and Malik, 2000). Although in this form, the Normalised Cut algorithm handles the clustering of a point set into an unknown number of clusters, the recursive nature of the process may introduce

undesirable computational burdens.

An alternative set of popular graph-based clustering methods based on graph cuts (Kolmogorov and Zabin, 2004; Isack and Boykov, 2012) formulates clustering as an energy minimisation problem. The basic form again deals with partitioning of a point set into two clusters. The insight to efficient optimisation of such energy functions in the bipartitioning case lies in realising that, under certain conditions (sub-modularity (Kolmogorov and Zabin, 2004)), by introducing additional fictitious nodes to the point set (including a source s and a sink t), finding the global minimum of the energy function corresponds exactly to solving the minimum cut problem of the constructed graph, thus allowing the global minimum to be found in polynomial time. Unfortunately, an extension to the N-partition problem is NP-hard (Kolmogorov and Zabin, 2004). The α-Expansion algorithm proposed by Boykov et al. (2001) is an effective algorithm for finding a local minimum of the energy function in such a scenario. The PEARL algorithm (Isack and Boykov, 2012) builds on α-Expansion to fit multiple models to a point set in the presence of a significant level of outliers, given geometric models for the clusters sought after. Unfortunately, to take advantage of the algorithm, one needs to define geometric models for the classes of interest *car*, *pedestrian* and *bicyclist*, which is far from a straightforward task.

Another popular approach to graph-based clustering uses a Euclidean Minimum Spanning Tree (EMST) constructed from the data (Duda et al., 2000). EMST-based techniques made their appearance in the literature in as early as the 1970's (Zahn, 1971) and are often used when cluster boundaries are expected to be irregular. We build upon this class of clustering algorithms by combining its strengths and simplicity with the RANSAC paradigm (Fischler and Bolles, 1981), to effectively tackle the lack of prior information on the number of objects in the scene.

Before we introduce the proposed clustering algorithm for entity segmentation, we first briefly review the EMST-based clustering algorithms and the RANSAC

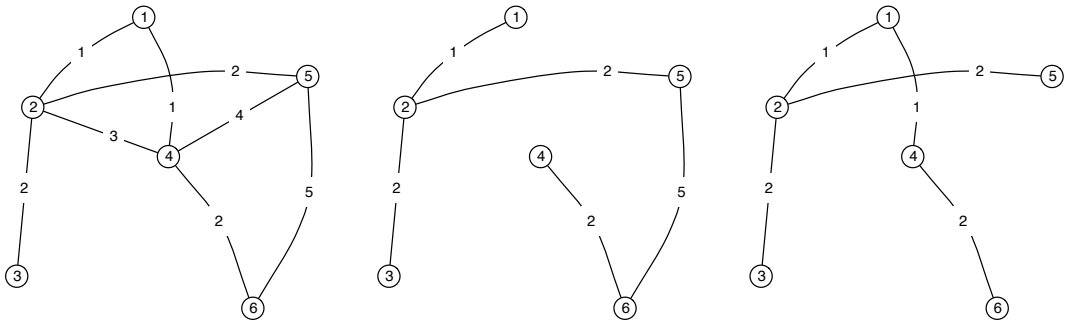


Figure 3.1: (a) A simple example of a weighted undirected graph $\mathcal{G}$. (b) A spanning tree of the graph $\mathcal{G}$. (c) The Minimum Spanning Tree of the graph $\mathcal{G}$. In all cases, numbers in circles denote nodes in the graph, and numbers on edges specify the edge weights.

paradigm in the following sections.

3.2.1 EMST-Based Clustering Algorithms

An undirected graph is defined by an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a finite set of vertices (nodes) and $\mathcal{E}$ is a set of unordered pairs (i.e. subsets of size 2) of the elements of $\mathcal{V}$ that forms the edges on the graph. Figure 3.1(a) shows a simple example of a graph. A spanning tree over a graph $\mathcal{G}$ is defined as a subgraph $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$ of $\mathcal{G}$ that is connected and contains no cycles. If a weight function $w : \mathcal{E} \to \mathbb{R}$ is defined over the edge set, for each possible spanning tree of a graph $\mathcal{G}$, an overall weight for the entire tree can be defined as the sum of the edge weights contained in the tree. Then amongst all possible spanning trees of $\mathcal{G}$, the one with the least overall weight is termed the Minimum Spanning Tree (MST). Thus Figure 3.1(b) is a spanning tree of the graph in Figure 3.1(a) but not the MST. The MST for the graph in Figure 3.1(a) is given by the graph in Figure 3.1(c).

Given a finite point set $\mathcal{P} \subset \mathbb{R}^d$, EMST-based algorithms first compute the Minimum Spanning Tree over the complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{i : p_i \in \mathcal{P}\}$ and $\mathcal{E} = \{\{i, j\} : p_i \in \mathcal{P}, p_j \in \mathcal{P}, i \neq j\}$, with edge weights given by the pairwise

Euclidean distances (thus the terminology *Euclidean* Minimum Spanning Tree). Let us denote $\mathcal{T} \subset \mathcal{E}$ as the edge set of the computed MST. A subset $\mathcal{S} \subset \mathcal{T}$ is then selected from $\mathcal{T}$ and removed from the tree. The resulting clusters are given by the disconnected components of the graph $(\mathcal{V}, \mathcal{T} \backslash \mathcal{S})$.

Variants of the EMST-based clustering algorithms differ in the way $\mathcal{S}$ is formed. For example, it was shown in (Asano et al., 1988) that, by removing the $K-1$ longest edges in the EMST, a clustering is obtained which maximises the minimum inter-cluster distance in the space of all possible disjoint partitions of the point set into K groups. That is, if an arbitrary partition of the node set $\mathcal{V}$ into K disjoint subsets is denoted by $P = \{C_i : i \in \mathbb{N}, i \leq K\}$ where $C_i \subset \mathcal{V}$ denotes individual *components* of the partition, taking $\mathcal{S}$ to be the set containing the $K-1$ longest edges in $\mathcal{T}$ produces a partition (clustering) P^* that maximises the measure $\min_{i,j} \min\{||\mathbf{p}_k - \mathbf{p}_l|| : k \in C_i, l \in C_j\}$. When K is unknown, as in our case, heuristics are used to determine which edges to remove.

For example, Zahn (1971) defines inconsistency measures using local statistics of the edge weights in the MST and removes edges which violate any one of a set of consistency criteria. Grygorash et al. (2006) propose a sequence of edge removal operations such that the standard deviation of the edge weights is minimised. The optimal number of clusters is found as the first local minimum in the standard deviation reduction.

We explicitly take advantage of the characteristics of the sensor to select the edge set $\mathcal{S}$ to be removed from the tree. For this purpose we need a popular paradigm for robust model fitting and outlier detection which we review briefly in the next section.

3.2.2 The RANSAC Paradigm

Random Sample Consensus (RANSAC) is a paradigm introduced by Fischler and Bolles (1981) to effectively tackle the problem of model fitting in the presence of outliers (i.e. samples arising from a different distribution to that generated by the model).

The paradigm assumes that the majority of data is dominated by a single model $\mathcal{M}$ from which the data was generated with noise (e.g. points that lie within a plane), within which embedded outliers arising from different physical processes or systematic errors (e.g. matching errors for a Computer Vision application).

For each iteration, a hypothesis generation phase is carried out. First, a minimum number of data points that uniquely determine a model are randomly sampled from the data. For the example of plane fitting in $\mathbb{R}^3$, this is three non-colinear data points. This minimum set is then used to generate uniquely a hypothesis $\mathcal{M}_h$ of the model $\mathcal{M}$. Inliers of $\mathcal{M}_h$ are given by the data points that fall within a support width w of $\mathcal{M}_h$ determined with a defined distance function from a data point to the model $\mathcal{M}_h$. Again, taking plane fitting as an example, an appropriate distance measure would be the distance from the data point to the hypothetical plane.

This process of hypothesis generation is then repeated, and the hypothesis with the greatest inlier support (the number of inliers to the hypothetical model $\mathcal{M}_h$) is chosen, whose inliers are taken as the true inliers contained in the data.

Finally, a robust estimator is applied only to the inliers found in the data to produce the final model estimate $\mathcal{M}^*$. In the example of plane fitting, this could be a least squares fit to the inliers found.

The benefits of RANSAC not only include estimating robustly the model $\mathcal{M}$ given data in the presence of both noise and outliers, but it also finds the set of outliers in the data as a by-product of the process. This is the key for its application to our clustering problem.

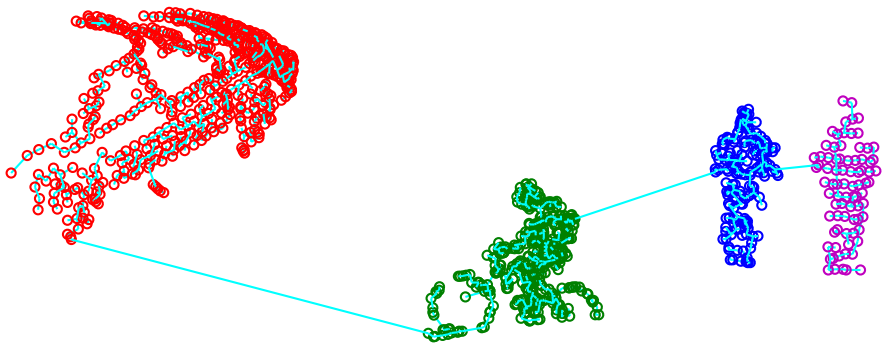


Figure 3.2: The output of the EMST-RANSAC clustering algorithm when applied to a synthetic scene containing fours objects of interest: a car, two pedestrians and a bicyclist (all examples from real data). Different colours denote different clusters produced. Cyan line segments show edges in the EMST.

3.2.3 The EMST-RANSAC Clustering Algorithm

For the particular case of our application, we observe that, as a result of the formation process of the MST, edges connecting points of the same object instance tend to be of similar lengths (up to sensor noise) corresponding to the sample width of the sensor. Edges linking individual object instances, on the other hand, tend to be comparatively long.

Figure 3.2 conceptually illustrates this situation, where clustering is performed on a synthetic scene containing a car, two pedestrians and a bicyclist (all examples from a real dataset).

Hence we adopt a model $\mathcal{M}$ for the edge weights within object instances to be

$$w_e = \text{const} \ , \tag{3.1}$$

where w_e denotes the weight of edge $e \in \mathcal{T}$, and apply the RANSAC technique to the set $\mathcal{T}$ of edges contained in the MST. The resulting set of outliers are taken as the set $\mathcal{S}$ to be removed from $\mathcal{T}$.

In subsequent sections we refer to this clustering algorithm as the EMST-RANSAC algorithm. It can be used to segment a point cloud into multiple entities without prior knowledge of the number of objects contained in the scene. However, since the algorithm clusters data based on Euclidean distance, unwanted points belonging to background clutter (i.e. anything other than the object classes of interest) must be removed before the algorithm can be applied. In the next section, we formulate this removal of background clutter as a supervised classification task. As a closing remark, we note that the procedures we are about to describe are agnostic to the specific clustering algorithm chosen, however the EMST-RANSAC algorithm is preferred for clustering 3D laser data under clutter-free conditions for its simplicity and effectiveness.

3.3 Supervised Foreground Extraction

3.3.1 Overview

Assume we are given a set of points $\mathcal{P} \subset \mathbb{R}^3$ generated by a 3D laser scanner. In order to apply the EMST-RANSAC algorithm described in the previous section to recover instances of objects of interest, the objects have to be separated by distances greater than the sensor's spatial sample width to be correctly identified as outliers. However, this is not the case in a real 3D scan, where points belonging to background clutter connect the individual objects.

Therefore it is necessary to split $\mathcal{P}$ into the set of *foreground* data, $\mathcal{P}_f \subset \mathcal{P}$ – i.e. those belonging to object classes of interest – and its complement, the set of *background* data $\mathcal{P}_b = \mathcal{P} \backslash \mathcal{P}_f$. Then the EMST-RANSAC algorithm will be applicable on $\mathcal{P}_f$ to recover object instances.

We employ a bottom-up approach, starting by preprocessing the point cloud to obtain an *over-segmentation* in the form of a set of point cloud patches. While we

do not require the segmentation to be perfect, it is necessary for each segment to span no more than a single class of interest. Features are then extracted for each patch and later used for classification of each patch into either belonging to $\mathcal{P}_f$ or $\mathcal{P}_b$. The steps in the pipeline described can be summarised as:

- Pre-processing to prepare for patch segmentation.
- Over segmentation of data input into patches.
- Extract features from each patch to project it into a fixed dimensional feature vector.
- Classify the feature vectors into either the *foreground* class or the *background* class.

We describe each of the steps of the system in detail in the following sections.

3.3.2 Preprocessing

In common with other works we perform an off-the-shelf pre-segmentation step based on point normal estimates in the point cloud to obtain a set of super-voxels[1] as atomic inputs for our entity segmentation approach.

The super-voxel segmentation algorithm requires as input an estimate of the normal to the local surface at each data point and also requires the points $\mathcal{P}$ be linked to form a graph (see Section 3.3.3 for details of the segmentation algorithm).

[1]The term "super-voxels" derives from the corresponding concept of "super-pixels" in the Computer Vision community, where an image is over segmented into regions that become the atomic elements of manipulation in later stages of processing. Since it replaces the role of ordinary pixels as atomic elements of manipulation but is a sort of "super set" of the pixels, they are termed "super-pixel" segments. Use of the word "voxel" here is appropriate because in the 3D space, points span a *volume*.

3.3.2.1 Robust Normal Estimation

A popular approach for normal computation finds the local set of N nearest neighbours for each datum $\mathbf{p}_i$ and then, assuming local planarity, performs a least squares fit to the neighbourhood.

Specifically, let us denote the set of N nearest neighbours of point $\mathbf{p}_i$ by $\mathcal{N}_i$. Assuming the size of the local neighbourhood is sufficiently small that local surface curvature can be ignored, the points $\mathbf{p}_j \in \mathcal{N}_i$ lie on a plane

$$P : \mathbf{n}^\top \mathbf{p} = c \text{ s.t. } \mathbf{n}^\top \mathbf{n} = 1 \ . \tag{3.2}$$

Then defining a distance function from a datum $\mathbf{p}_j$ to the plane P by

$$d^2(\mathbf{p}_j, P) = (\mathbf{n}^\top \mathbf{p}_j - c)^2 \ , \tag{3.3}$$

the normal $\mathbf{n}_i$ at point $\mathbf{p}_i$ is estimated as

$$(\mathbf{n}_i^*, c^*) = \arg\min_{\mathbf{n},c} \sum_{\mathbf{p}_j \in \mathcal{N}_i} d^2(\mathbf{p}_j, P) \text{ s.t. } \mathbf{n}^\top \mathbf{n} = 1 \ . \tag{3.4}$$

By using a Lagrange multiplier, it can be shown that the solution to Equation (3.4) is given by an eigenvalue decomposition of the sample covariance matrix

$$M_i = \frac{1}{N} \sum_{\mathbf{p}_j \in \mathcal{N}_i} (\mathbf{p}_j - \bar{\mathbf{p}})(\mathbf{p}_j - \bar{\mathbf{p}})^\top \ , \tag{3.5}$$

where $\bar{\mathbf{p}}$ denotes the sample mean of $\mathbf{p}_j \in \mathcal{N}_i$, and $\mathbf{n}_i^*$ is given by the (normalised) eigenvector of M_i corresponding to the least eigenvalue.

This approach has been shown empirically to perform best in terms of the trade-off between robustness and computation overhead (Klasing et al., 2009).

3.3.2.2 Linking

The segmentation algorithm we use for producing point cloud patches (super-voxel segments) is a graph-based algorithm (see Section 3.3.3 for a detailed description). Thus an edge set is needed for the algorithm to determine the connectivity of data points that will also capture spatial relationships of the data (i.e. the existence of an edge represents a sense of spatial closeness between its end nodes).

For these reasons, we employ N nearest neighbour linkage. That is

$$\mathcal{E} = \{\{i, j\} : \mathbf{p}_i \in \mathcal{P},\, \mathbf{p}_j \in \mathcal{N}_i^N\}\ , \tag{3.6}$$

where $\mathcal{N}_i^N$ denotes the set of N nearest neighbours of the point $\mathbf{p}_i$, *excluding* the point itself. So every point is connected to its N nearest neighbours.

3.3.3 Patch Segmentation

To obtain the initial patch segmentation, we follow the approach proposed by Triebel et al. (2010) who adapted the popular segmentation algorithm introduced by Felzenszwalb and Huttenlocher (2004) to operate on normal estimates for points in $\mathcal{P}$. We review briefly this segmentation algorithm in the next section.

3.3.3.1 The EGBIS Segmentation Algorithm

The Efficient Graph-Base Image Segmentation (EGBIS) algorithm introduced by Felzenszwalb and Huttenlocher (2004) takes as input a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with the edge weights representing a dissimilarity measure between adjacent points. The vertex set $\mathcal{V}$ is the set to be segmented, and the edge set $\mathcal{E}$ represent likely similarities between nodes to be considered for segmentation.

If the weight function on $\mathcal{E}$ is denoted by w, then given a segmentation of $\mathcal{V}$ into components $C_i \subset \mathcal{V}$ (C_i being non-empty, disjoint and $\bigcup C_i = \mathcal{V}$), an *internal*

difference for a given component C_i is defined as

$$\mathrm{Int}(C_i) = \max_{e \in \mathrm{MST}(C_i, E_i)} w(e) \ , \tag{3.7}$$

where $E_i = \{e \in \mathcal{E} : e \subset C_i\}$ is the subset of $\mathcal{E}$ that fall within the vertices covered by the component C_i, $\mathrm{MST}(C_i, E_i)$ denote the Minimum Spanning Tree of the subgraph of $\mathcal{G}$ formed by C_i and E_i. Thus the *internal difference* for a component is given by the maximum edge weight on the MST computed over the component.

A difference *between* two components C_i and C_j is also defined, and is given by

$$\mathrm{Dif}(C_i, C_j) = \min_{u \in C_i, v \in C_j, \{u,v\} \in \mathcal{E}} w(\{u, v\}) \ , \tag{3.8}$$

that is, the minimum edge weight over all edges connecting the two components.

The algorithm starts with each vertex $v \in \mathcal{V}$ as a single segment, and traverses the edges in order of non-decreasing weight, if the edge currently under consideration is denoted by $e = \{u, v\}$, and the components the nodes u and v currently belong to by C_i and C_j, then if $C_i \neq C_j$ (that is, if u and v are not already in a single component) they are merged into a single component if the criterion

$$w(e) \leq \mathrm{MInt}(C_i, C_j) \tag{3.9}$$

holds. $\mathrm{MInt}(C_i, C_j)$ is the minimum internal difference between C_i and C_j defined by

$$\mathrm{MInit}(C_i, C_j) = \min(\mathrm{Int}(C_i) + k/|C_i|, \mathrm{Int}(C_j) + k/|C_j|) \ , \tag{3.10}$$

where k is the (only) parameter of the algorithm for controlling the coarseness (the average size of a component) of the resulting segmentation.

It is proved in (Felzenszwalb and Huttenlocher, 2004) that the output of the algorithm is one that is optimal in the sense that there is evidence of a boundary

between any pair of components in the segmentation, and any refinements of the segmentation by subdividing any of its components result in a segmentation that violates the above statement. Evidence of a boundary between a pair of components C_i and C_j is defined by the condition

$$\mathrm{Dif}(C_i, C_j) > \mathrm{MInt}(C_i, C_j) \ . \tag{3.11}$$

The resulting segmentation often represent regions where the dissimilarity measure is uniform or slowly varying. Region boundaries often occur at abrupt changes of the dissimilarity measure.

3.3.3.2 Application of EGBIS to Patch Segmentation

We follow the approach proposed by Triebel et al. (2010) to apply the EGBIS segmentation algorithm in the case of laser data, and produce segmented patches of the point cloud to serve as atomic units of manipulation in foreground extraction.

Consider the set of vertices $\mathcal{V} = \{i : \mathbf{p}_i \in \mathcal{P}\}$ and the set of edges $\mathcal{E}$ as given by Equation (3.6). The dissimilarity measure (edge weight function) w is defined by

$$w(\{i, j\}) = 1 - |\mathbf{n}_i \cdot \mathbf{n}_j| \ , \tag{3.12}$$

where $\mathbf{n}_i$ denotes the normal estimated at point $\mathbf{p}_i$. Intersections between smooth surfaces will thus give rise to segmentation boundaries. A sample result of applying the EGBIS segmentation algorithm on the graph structure defined above is shown in Figure 3.3.

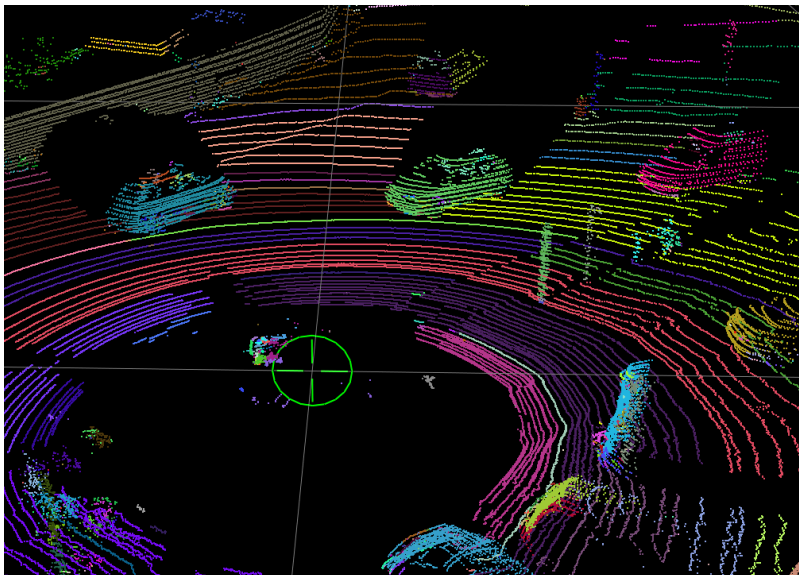


Figure 3.3: Patch segmentation result for a sample scene. Points are coloured by segment label. Segment colours were selected at random.

3.3.4 Feature Extraction

In order to determine if a given patch belongs to the *foreground* class or the *background* class, features have to be extracted from it that are invariant to translation, rotation about the vertical axis, sampling density (due to the fact that laser detectors are range-bearing devices, structures that are further away to the sensor tend to be sampled more sparsely than those that are closer), and sensor noise. Thus each patch in a training set is projected down into a point (vector) in the D-dimensional feature space. By using invariant features, each patch, hence each feature vector in the training set, serves as an example for a large class of similar patches seen from different viewpoints and at different distances to the sensor so that their class can be determined correctly from their proximity to the training example.

Therefore, for each patch, we construct a 120-dimensional feature vector by concatenating five sets of common invariant descriptors extracted from the patch. The descriptors consist of 50-dimensional spin images (Johnson and Hebert, 1999), two 32-dimensional shape distributions (Osada et al., 2002), three-dimensional shape

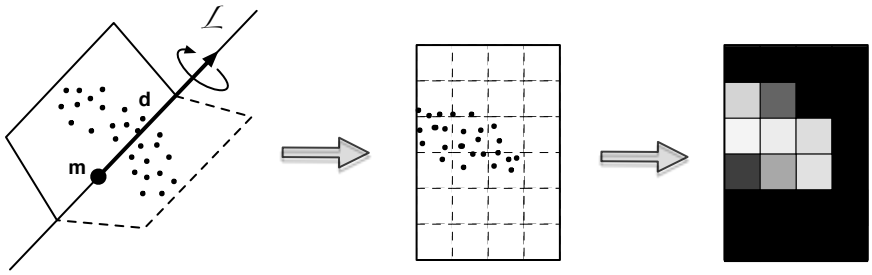


Figure 3.4: Schematic illustration for spin image computation. A hypothetical plane is first revolved around an axis of rotation given by the line $\mathcal{L}$, collecting points as they hit the plane. Dashed plane denotes the starting location of the plane and solid plane denotes its current location. Resulting points collected on the plane is then binned into a normalised 2D histogram plotted here as a greyscale image.

factors (Westin et al., 1997), and the three dimensions of the bounding box of the patch. In the following sections, we briefly review these features, and detail specific modifications to our application.

3.3.4.1 Spin Images

Spin images were proposed by Johnson (Johnson, 1997; Johnson and Hebert, 1999), and are popular invariant descriptors and one of the most commonly applied to the domain of 3D laser data. The backbone of the feature extraction process is illustrated in Figure 3.4, which consists of fixing one side of a hypothetical rectangular sheet at a line $\mathcal{L}$ in space passing through a point $\mathbf{m}$ with orientation $\mathbf{d}$. Then the sheet is revolved about $\mathcal{L}$ collecting laser points along its way. After a full revolution, the sheet is discretised into a regular grid and points collected in each grid cell are counted. The result forms a two-dimensional histogram. This histogram is then normalised by the total number of points fallen on the sheet such that the cell values sum up to one.

We compute a 10 by 5 histogram for each patch taken at the mean location of a patch $\mathbf{m} = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \mathbf{p}$, where P denotes the set of points belong to a patch, along the Z direction, which is defined by convention to be the vertical axis. This gives us,

after vectorising the 2D histogram entries, a 50-dimensional feature vector. Because the spin images are taken along the Z-axis, the feature vector is invariant to any rotation about Z.

3.3.4.2 Shape Distributions

Shape distributions are a class of invariant descriptors for 3D models proposed by Osada et al. (2002). In the construction of the feature, pairs of points are randomly selected from the model (the point cloud patch in our case), then a *shape function* is defined to map each pair of points selected to a scalar. The scalar outputs of the *shape function* are then binned into a histogram with bin counts normalised to sum up to one.

We define two shape functions f and g as

$$f(i,j) = ||\mathbf{p}_i - \mathbf{p}_j||_2 \tag{3.13}$$

and

$$g(i,j) = |\mathbf{n}_i \cdot \mathbf{n}_j| \ , \tag{3.14}$$

i.e. the Euclidean distance between the pair of points and the magnitude of the dot product between the normals estimated at those points. Because these measures are relative, the features computed are invariant to translation and rotation.

3.3.4.3 Shape Factors

Shape factors, though originally introduced in the context of medical imaging (Westin et al., 1997), are closely related to Principal Component Analysis (PCA). Briefly, a PCA finds principal directions local to a group of data points corresponding to stationary values in the *variance* of the data as a function of direction.

Specifically, a PCA finds stationary points with respect to $\mathbf{n}$ of the cost function

$$C(\mathbf{n}) = \frac{1}{N}\sum_{i=1}^{N}((\mathbf{x}_i - \bar{\mathbf{x}})^\top \mathbf{n})^2 \tag{3.15}$$

for a set of points $\mathbf{x}_i$, where $\bar{\mathbf{x}}$ denotes the mean of the point set, subject to the constraint $\mathbf{n}^\top\mathbf{n} = 1$ (because $\mathbf{n}$ is a direction).

Again, using a Lagrange multiplier similar to Section 3.3.2.1, the solution to Equation (3.15) is given by the eigenvectors of the covariance matrix

$$M = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top \ . \tag{3.16}$$

The eigenvalues of M give the value of the variance along the eigenvector directions. Hence the eigenvector corresponding to the largest eigenvalue represents the direction of largest variance while the one corresponding to the least eigenvalue gives the direction of least variance.

Shape factors are normalised versions of the eigenvalues of M in $\mathbb{R}^3$. Specifically, the three shape factors are defined as

$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \ , \ c_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3} \ , \ c_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \ , \tag{3.17}$$

where λ_1, λ_2 and λ_3 denote the eigenvalues of M in descending order.

In a case when $\lambda_1 \gg \lambda_2 \approx \lambda_3$, $c_l \approx 1$, $c_p \approx 0$ and $c_s \approx 0$. This corresponds to a *linear* case, where the point set has a *rod-like* distribution in space. In a case when $\lambda_1 \approx \lambda_2 \gg \lambda_3$, $c_l \approx 0$, $c_p \approx 1$ and $c_s \approx 0$. This corresponds to a *planar* case, where the point set has a *disc-like* distribution in space. Finally, in a case when $\lambda_1 \approx \lambda_2 \approx \lambda_3$, $c_l \approx 0$, $c_p \approx 0$ and $c_s \approx 1$. This corresponds to a *spherical* case, where the point set is *spherically* distributed in space. Any distribution in between produces fractional shape factors representing components of the three cases present

in the shape.

3.3.4.4 Bounding Box Dimensions

We also include the dimensions of a bounding box for the patch as part of the feature set. In order to achieve rotational invariance, we take the *width*, *height* and *depth* dimensions of a *tight-fit* bounding box for the patch along a local set of axes given by the PCA eigenvectors.

3.3.5 Patch Classification

Now each patch has been mapped into a vector in the 120-dimensional feature space, standard off-the-shelf classifiers can be applied to determine the membership of $\mathcal{P}_f$ of a novel patch. We apply a well-known and effective classifier from the literature, the Support Vector Machine (SVM) with the Radial Basis Function (RBF) kernel for the classification task. Readers unfamiliar with SVM classification is referred to Appendix A.1 for a brief summary.

3.4 Evaluation of Three Segmentation and Classification Strategies

3.4.1 Three Different Schemes

In our particular application, since it is relatively straight-forward to provide the actual class label of a patch (if it is part of a *car*, a *pedestrian* or a *bicyclist*, rather than it belongs to an object of interest), we propose two schemes of merging the three foreground classes in the patch classification stage to produce a clean foreground-background segmentation of the scene so that the EMST-RANSAC algorithm is applicable.

	STC	Wildcat	Total
Car	13998	580	14578
Pedestrian	4619	0	4619
Bicyclist	4601	2	4603
Background	10000	30500	40500
Total	33218	31082	64300

Table 3.1: Details of the evaluation set composition for patch classification in units of patches obtained from the initial pre-segmentation. Columns denote the source of the data. See text for details.

F/B binary: in this scheme the three foreground classes are pooled into a single class and a SVM classifier is trained to separate them.

F/B N-class: here, N one-versus-all SVM classifiers are deployed for the *car*, *pedestrian*, *bicyclist* and *background* classes, respectively. After classification is performed the outputs for the three foreground classes are merged into a single set.

These schemes are also benchmarked against a third, **N-class**, where the individual foreground classes are treated separately up to, and including, the object detection level.

We evaluate our segmentation approach using both a publicly available dataset as well as data gathered using our own autonomous vehicle. In particular, we make use of the Stanford Track Collection (STC) dataset released to the public with (Teichman et al., 2011). The STC contains a significant number of labelled objects of interest (cars, pedestrians and bicyclists) and has the added advantage of being gathered using the same sensor as deployed on our car. However, the dataset was originally produced for the task of track classification and therefore contains only instances of trackable objects. Scene clutter is especially underrepresented (see Table 3.1). For this work we therefore augment the STC with data gathered using our Bowler Wildcat research platform (Figure 1.1(a)) equipped with a Velodyne HDL-64E S2 laser range finder.

3.4.2 Patch Classification

The performance of the three patch classification schemes was evaluated using the data detailed in Table 3.1. Our approach is agnostic to the patch segmentation scheme employed as long as it produces an *over*-segmentation of the data with respect to the classes of interest. The parameters for the patch segmentation algorithm used here (cf. Section 3.3.3) were determined empirically based on a qualitative evaluation of performance on a small number of scenes. For classifier training and evaluation, 70% of the data were selected at random to form the training set. The remainder were used as a hold-out set for classifier evaluation. For scheme *F/B binary* a single binary SVM classifier is trained for the *foreground* and *background* classes. For schemes *F/B N-class* and *N-class*, four individual SVM classifiers are trained in a one-versus-all configuration for each of the *car*, *pedestrian*, *bicyclist* and *background* classes. Final class decisions are made greedily such that the winner takes all.

We follow the common practice in the machine learning community and use Precision-Recall curves to characterise quantitatively the performance of the three schemes up to the task of patch classification. Readers unfamiliar with the Precision and Recall evaluation metrics are referred to Appendix A.2 for a detailed description.

Figure 3.5 shows Precision-Recall curves generated for the three schemes using the held-out data. Curves for *F/B binary* and each of the one-versus-all classifiers of *N-class* were generated by varying the threshold on prediction scores (distances to the decision hyperplane) returned by the binary SVM classifier, effectively shifting the location of the decision boundary. The curve for the *F/B N-class* was generated using a score defined as follows: let $\mathbf{s} = [s_1, s_2, s_3, s_4]^\top$ be scores returned by the four one-versus-all classifiers with s_1, s_2, s_3, s_4 representing prediction scores for the *car*, *pedestrian*, *bicyclist* and *background* classifiers respectively. We define a score

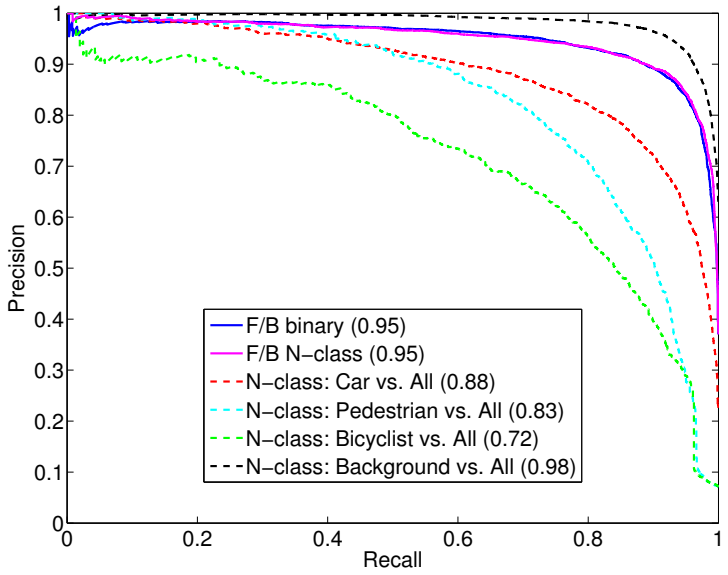


Figure 3.5: Precision-Recall curves for schemes *F/B binary*, *F/B N-class* and *N-class*. For the latter, individual curves are shown for each one-versus-all classifier. Numbers in brackets represent the area under the curve (AUC).

for the *F/B N-class* classifier to be

$$s = \max\{s_1, s_2, s_3\} - s_4 \ . \tag{3.18}$$

Thus a decision boundary of $s = 0$ results in the same classification results as taking the class label to be the one corresponding to the maximum of the N one-versus-all prediction scores. And more positive values correspond to more confident positive classification results in one or more of the three foreground classes and more confident negative classification results in the background class, and vice versa.

It is evident from Figure 3.5 that the binary *foreground/background* separation in this particular case presents an easier task for the classifiers than separating the data into the individual classes *car*, *pedestrian*, *bicyclist* and *background*. When combining the output of the classifiers for the three individual foreground classes into a single class for the *F/B N-class* scheme the performance is almost identical to that of the binary *foreground/background* classifier of the *F/B binary* scheme. Note

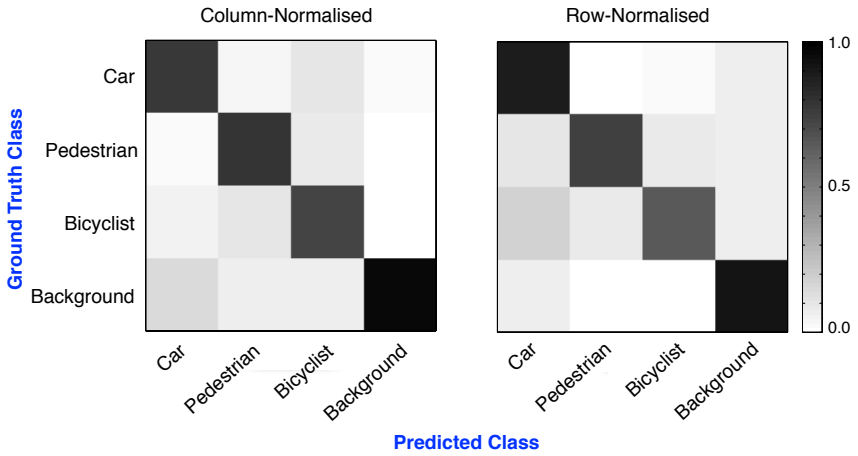


Figure 3.6: Confusion matrices for the *N-class* scheme normalised to Precision (left) and Recall (right) along the diagonal.

also that, by definition, the performance of the background-versus-all classifier of the scheme *N-class* is identical to that of the binary *foreground/background* classifier evaluated for the *background* class. This indicates that the separate classification of the *car*, *pedestrian*, *bicyclist* classes introduces significant confusion amongst only these classes which is remedied by collating them into a single *foreground* class. Further evidence of this can be found in the confusion matrices for the *N-class* scheme depicted in Figure 3.6. These imply that the biggest confusion between *foreground* and *background* is caused by *background* data being mistakenly classified as *car*. On the other hand, significant confusion exists amongst the individual *foreground* classes. These results indicate that, for the task of separating *car*, *pedestrian* and *bicyclist* from *background* in 3D laser data, the predominantly shape-based features employed here are not sufficient. This lends further support to the intuitive notion that over-segmented patches do not carry enough shape information to be classified correctly. Formulating the task as a binary classification problem, on the other hand, remedies this problem as the *foreground* and *background* classes appear much more amenable to separation when characterised by shape features.

3.4.3 Overall System Evaluation

In this section we evaluate the performance of the overall system starting with a raw data stream as input and performing preprocessing, patch classification and EMST-RANSAC clustering. However, in order to demonstrate the efficacy of the F/B-based schemes against the *N-class* scheme, the clusters obtained have to be classified into one of the three foreground object classes. For this purpose we trained a N-class SVM classifier using the same set of features listed in Section 3.3.4, now computed over *entire entity clusters* (as opposed to patches representing object *parts*) returned by the *F/B binary* scheme. For the *F/B N-class* scheme, since it uses an N-class classifier in an intermediate stage, we retain the prediction scores from patch classification and determine the class of an entity cluster by a majority vote amongst the constituent laser points. The votes are weighted by prediction confidence. For the *N-class* scheme, detections are obtained by running the EMST-RANSAC algorithm over the three foreground regions independently and entity cluster classes are self-evident.

The EMST-RANSAC algorithm involves only a single parameter: the inlier support width w to evaluate hypotheses (cf. Section 3.2). This parameter was trained on 200 frames extracted from the STC dataset that are disjoint from those used in producing the training data for patch classification. For the *F/B binary* and *F/B N-class* schemes, a single value for w was determined since the EMST-RANSAC algorithm is applied only once on patches belonging to the *foreground* class. For the *N-class* scheme, EMST-RANSAC is applied to each of the object classes, resulting in a three-element vector $\mathbf{w}$. We trained the three support widths independently for the *N-class* scheme.

To evaluate the performance of the system, we hand-labelled 100 randomly chosen frames from a busy urban scene taken at a local town centre. These data are entirely independent from those used during any of the training phases. A set of

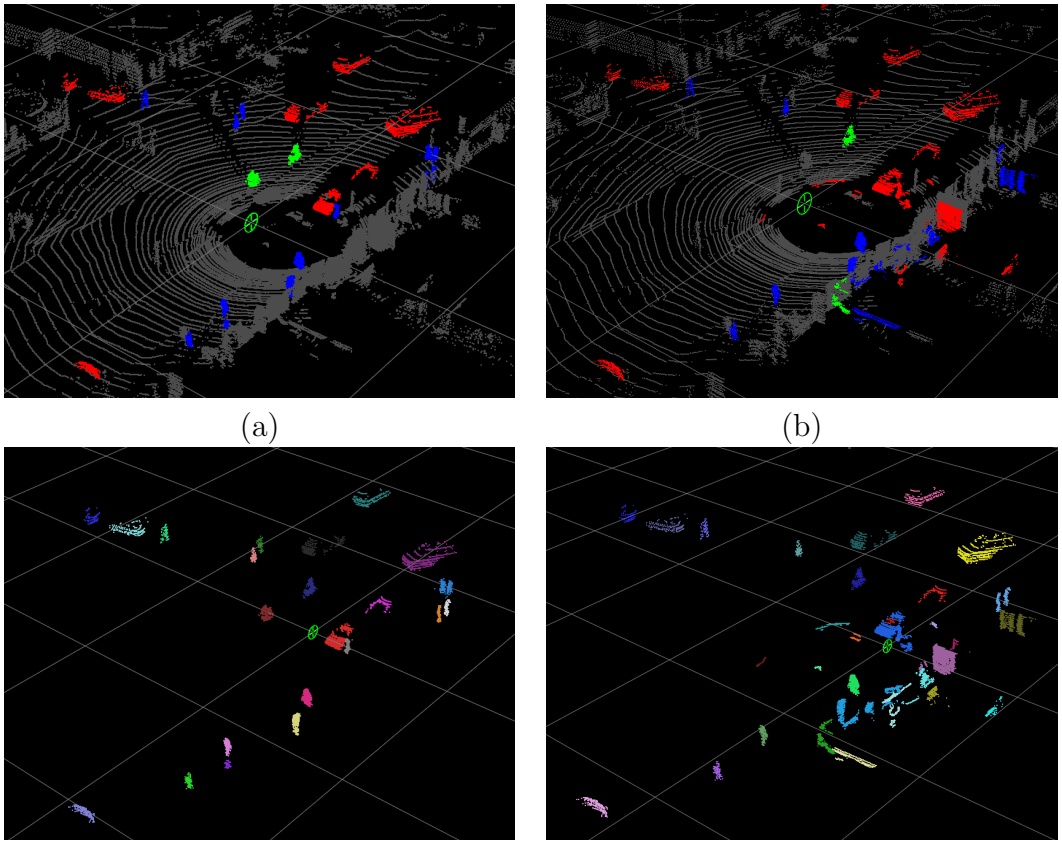


Figure 3.7: Sample frame showing results of the *F/B N-class* scheme. (a) Ground truth scene labels. (b) Objects detected by *F/B N-class*. (c) Ground truth objects in the scene. (d) Object clusters produced by *F/B N-class*. In both (a) and (b), regions coloured red, blue, green and grey correspond to points belonging to the *car*, *pedestrian*, *bicyclist* and *background* classes, respectively. In both (c) and (d), different colours denote different object instances, with colours chosen at random.

qualitative results on a sample frame for the *F/B N-class* scheme is shown in Figure 3.7. For quantitative analysis we adopt evaluation metrics derived from those used in a popular object detection challenge in the Computer Vision community, the PASCAL Visual Object Classes Challenge (Everingham et al., 2010). In particular, a detection is marked as correct if it overlaps with a ground-truth annotation more than 50%. The measure of overlap is computed as

$$a_o = \frac{|\mathcal{C}_\mathrm{p} \cap \mathcal{C}_\mathrm{gt}|}{|\mathcal{C}_\mathrm{p} \cup \mathcal{C}_\mathrm{gt}|}, \tag{3.19}$$

where $\mathcal{C}_{\mathrm{p}}$ and $\mathcal{C}_{\mathrm{gt}}$ denote sets of points belonging to the predicted and ground-truth object clusters respectively. Each detection is assigned to at most one ground-truth object, and multiple detections are treated as false positives. Table 3.2 thus lists evaluation results for each of the foreground object classes obtained on the 100 evaluation frames containing, in total, 818 cars, 899 pedestrians and 39 bicyclists.

It is evident from observing the F_1-measures (See Appendix A.2 for a definition of F-measures) in Table 3.2 that the schemes based on a binary formulation of the problem (i.e. clustering based on separating *foreground* from *background*) outperform the *N-class* scheme. However, the latter still does surprisingly well considering the findings in the patch classification evaluation. This observation can be explained by the two extra degrees of freedom found in the EMST-RANSAC algorithm for this scheme. For example, the system has the freedom to learn that instances of people tend to be closer together than instances of cars. Performance differences between the *F/B binary* and *F/B N-class* schemes are attributed to the difference in entity classification schemes.

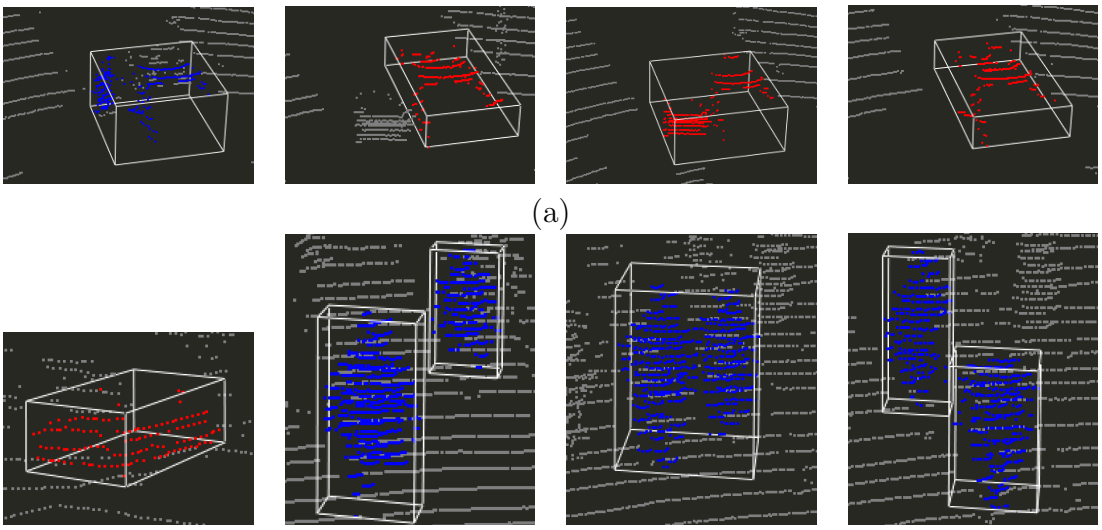
3.4.3.1 Failure Cases

In this section, we examine some interesting scenarios that often cause the detector to fail. As an illustrative case, we show detection results returned by the *F/B N-class* scheme. Similar qualitative performance is observed from the other two schemes in these challenging scenarios.

Figure 3.8(a) shows the detection results in a case where a pedestrian gets in a car. As shown in the figure, when the pedestrian approaches the car, the car and the pedestrian are erroneously merged into one object. After the pedestrian opens the door, the opened door may or may not be identified as one part of the same car (see the middle two frames of Figure 3.8(a)). Finally when the door is closed, the car is identified successfully. Figure 3.8(b) shows a case where a parked car has

		Car (818)			Pedestrian (899)			Bicyclist (39)		
		P	R	F_1	P	R	F_1	P	R	F_1
N-class		0.1856	**0.5122**	0.2725	0.4415	**0.5751**	0.4995	0.0294	0.4359	0.0551
F/B	binary	**0.2795**	0.4401	**0.3419**	0.4696	0.3782	0.4190	0.0256	0.4103	0.0483
	N-class	0.2102	0.5037	0.2966	**0.5877**	0.4360	**0.5006**	**0.0989**	**0.4615**	**0.1629**

Table 3.2: System evaluation results by class. P, R and F_1 stand for Precision, Recall and the F_1-measure, respectively. Numbers in brackets denote the ground-truth number of objects of a given class in the evaluation data. Bold numbers denote the largest entry for the column.



(a) (b) (c)

Figure 3.8: Challenging scenarios from the 100 evaluation frames where detection may fail. Points coloured red belong to detected cars whereas points coloured blue belong to detected pedestrians. Separate object instances determined by the detector are marked with separate bounding boxes. Most cases are due to complex interactions between objects. Detection shown is given by the *F/B N-class* scheme. (a) A scenario of a pedestrian getting in a car. (b) A parked car with an open boot. (c) Segmentation failure erroneously returned two pedestrians passing each other as one object instantaneously.

its boot open. In this case, the opened boot has been left out of the detected car. Figure 3.8(c) shows two pedestrians passing by each other. Due to segmentation failures, they are temporarily merged as one object when they are close together.

Most of these challenging scenarios arise from complex interactions of objects, for example between a pedestrian and a car, and between pedestrians. It is these interactions that are not well modelled in the current approach, where we assume objects are spatially separated.

3.4.4 Timing

A prototype of the proposed system has been implemented in C++ and Matlab and was deployed on a vanilla MacBook Pro equipped with a dual core Intel i5 processor (2.4GHz) with 4GB of RAM. For point normal estimation our implementation takes advantage of the facilities provided in the Point Cloud Library (PCL) (Rusu and Cousins, 2011). SVM training and prediction were carried out using LIBSVM (Chang and Lin, 2011). For efficient EMST computation, we implemented the fast EMST algorithm proposed by March et al. (2010). Based on measurements from our 100 evaluation frames (containing of the order of 100,000 points per frame), the per-frame run-time is dominated by the EMST-RANSAC clustering step (3.3s) and the normal computation (1.7s).

3.5 Conclusions and Discussions

In this chapter, we presented an approach to segmenting objects of interest from a raw data stream as commonly obtained from a 3D laser range finder. We focus on the supervised extraction of potentially dynamic objects such as cars, pedestrians and bicyclists for autonomous driving applications. The output of the system are clusters of points representing *entire* objects, which is often assumed to be available

by work on object classification in 3D point clouds. We show that, for the specific classes considered, solving a binary classification task (i.e. separating the data into foreground and background first) outperforms approaches that tackle the multi-class problem directly. This is primarily the case because *parts* of objects, as commonly obtained by a pre-segmentation step, do not contain enough shape information to be robustly categorised as belonging to the classes considered here. While our pipeline is agnostic to the graph-based clustering algorithm used we explore the use of the EMST algorithm, and extend it by a RANSAC-based outlier rejection step to automatically determine the number of clusters present in a scene. In doing so, we explicitly exploit the sampling characteristics of the laser range finder. While the results on patch classification presented here are particular to the popular *car*, *pedestrian*, *bicyclist* and *background* classes, the EMST-RANSAC approach is agnostic to the choice of classes and, therefore, more generally applicable.

The proposed segmentation-based object detector has also been successfully applied to building high quality maps free of cars, pedestrians and bicyclists to improve navigation (Stewart and Newman, 2012; McManus et al., 2013).

In the next chapter, we take a different approach to object detection, and investigate the applicability of the sliding window technique that is so ubiquitous in vision-based detection methods to the case of 3D laser data.

Chapter 4

Efficient Sliding Window Object Detection in 3D

4.1 The Idea

The sliding window approach to object detection, as we have surveyed in Section 2.2, is arguably the most frequently deployed paradigm for object detection in the Computer Vision community. However, it has been largely neglected so far for laser-based object recognition. In fact, the same paradigm seems to be equally applicable to a 3D point cloud as it is to a 2D image. The conceptual difference is not significant, one only need to first discretise the space into a 3D voxel grid, and slide a window through all three-dimensions instead of two in the case of images.

Perhaps what has been discouraging is the worry of the extra computational burden introduced by the additional dimension, and dismissing sliding window approaches as intractable in 3D.

In this chapter, we attempt to give a second thought to this opinion, and noting that there is one fundamental difference to the structure of a 3D point cloud and that of a 2D image – a 3D point cloud is sparse in that most space is simply

unoccupied. Could we exploit this observation to our advantage and make sliding window approaches tractable or even efficient in 3D? In fact, it turns out this is achievable.

We show in Section 4.5 there exists a duality between sliding window detection with linear classifiers and voting from *only* the occupied cells, reducing the amount of computation to the bare minimum while at the same time maintaining exact mathematical equivalence. This is our key contribution in this chapter.

To the best of our knowledge, this is the first time the sliding window approach has been taken to object detection in 3D data.

4.2 Related Works

Perhaps most similar to our work is a body of work for monocular object detection with 3D pose estimation (Hedau et al., 2010; Fidler et al., 2012). Objects are characterised by 3D bounding boxes, and its location is slid in 3D. However, instead of building a 3D feature grid, detection is achieved by projecting the image fronto-parallel to each visible face of the object bounding cuboid and 2D features are then extracted for that face from the projected image.

A line of work by Oliveira and colleagues (Oliveira et al., 2010; Oliveira and Nunes, 2013) slides a window in 3D to aid image-based pedestrian detection as follows. From the mounting position of their 2D laser scanner, the location of the ground plane can be estimated to be at a fixed height below the laser and parallel to the laser scanning plane. Then a 2D window of a fixed size fronto-parallel to the camera imaging plane is anchored on the ground plane in 3D and slid through discrete locations on the ground plane. The window at each location is then back-projected into the image plane to bound a pedestrian hypothesis.

An alternative approach to object detection in 3D is to combine a 3D point cloud

acquired with a laser scanner with an image taken by a camera, and project *all* laser points to the image plane to form a depth image (Premebida et al., 2014; Quigley et al., 2009; Lai et al., 2011a). Then the sliding window approach is applied on both the ordinary image and the depth image in 2D.

The approach we propose in this chapter does not require an image. We take solely the 3D laser data, slide a 3D detection window across a three-dimensional feature grid. There is no projection involved.

4.3 Overview

The steps required in our sliding window detector is conceptually analogous to an image-based sliding window detector. Figure 4.1 illustrates the process with a toy example – a small section of a real 3D laser scan containing an object of interest, a car in this case.

The input to detection is the 3D laser scan represented as a list of point locations, together with reflectance values for each point. The locations of the points provide the shape cues while the reflectance values provide some information about the appearance of the object.

First, the point cloud is converted into a feature grid as follows. The 3D space is discretised according to a fixed grid size, and each occupied cell (by occupation, we mean at least one point of the point cloud falls within the bounds of the cell) is converted into a fixed-dimensional feature vector. Details of our feature representation are explained in Section 4.6. Cells that are not occupied by any points map to zero feature vectors (i.e. a vector of all zero elements). This definition is critical for exploiting the sparsity of the problem. For example, as an illustration, the middle left diagram of Figure 4.1 visualises the feature grid extracted over the section of point cloud shown at the top left of the same figure. Here each coloured ball rep-

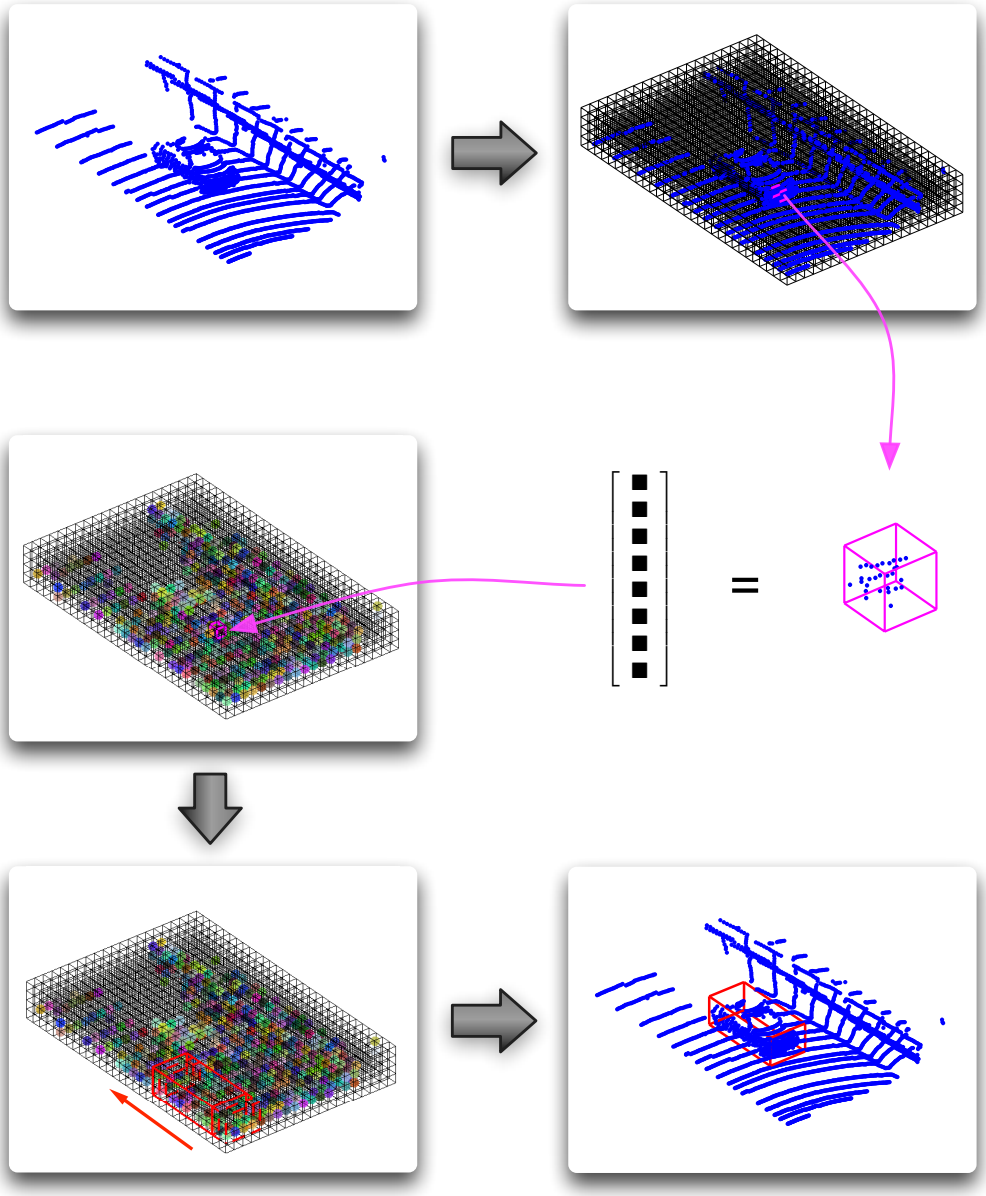


Figure 4.1: An illustration of the detection process. The point cloud (top left) is first discretised into a 3D grid (top right). For each occupied cell, points that fall within the cell, together with their reflectance values, are mapped to a fixed-dimensional feature vector (middle right gives an example of such an occupied cell highlighted in both top right and middle left). Unoccupied cells are mapped to zero feature vectors by definition. Thus the point cloud is converted to a feature grid (middle left, each coloured ball represents a feature vector extracted for an occupied cell). A 3D detection window then slides, in all three-dimensions, through the feature grid (bottom left), a classifier evaluates each window location for the evidence of an object. The point cloud with the detected object is shown at the bottom right. The process repeats for each angle of rotation.

resents a feature vector extracted for an occupied cell, the absence of a ball means the cell is unoccupied and therefore its feature vector is zero. Note the sparsity of the feature grid – coloured balls only occupy a small subset of the entire grid. In particular, they lie only on a 2D surface of the world that the laser traces out.

Then conceptually, a three-dimensional detection window of a fixed size is placed at one corner of the feature grid and slides down the x-direction then the y-direction and then the z-direction. At each location of the detection window, the feature vectors contained within its bounds are stacked up into a single long vector and passed to a classifier (for example, an SVM classifier). The classifier then decides whether the current location of the detection window bounds an object of interest or not by means of returning a detection score (a higher score meaning more confident that it bounds an object of interest). Section 4.5 is devoted to a mathematical "trick" that makes this step tractable.

Finally, just as is the case for image-based sliding window detectors, the classifier may fire multiple times centred around the true object of interest. Non-maximum suppression must be applied over returned object windows to suppress duplicate detections. Our strategy to non-maximum suppression is detailed in Section 4.7.

In contrast to image-based detectors, scale is not an issue here, because the absolute scale (in metres) is known in 3D. However, rotation *is* a problem. Assuming objects of interest are generally upright, that is, any rotation is constrained to be about the vertical axis, in order to be able to detect objects of our interest in arbitrary orientations, we discretise the full 360° into N orientation bins and run the same detection process (cf. Figure 4.1) N times on the rotated *point cloud* for each orientation bin.

4.4 Linear SVM versus Nonlinear SVM

For our classifier, we choose the linear SVM (we refer readers unfamiliar with the SVM classifier to Appendix A.1 for a review of SVM classification). This is far from an arbitrary choice.

In the sliding window case, the feature vector for input to an SVM classifier is the stacked feature vector composed of features extracted for each cell in the detection window (which is itself a 3D grid, albeit small). For example, if the dimensions of the detection window is (l, w, h), and the dimension of the feature vector for each cell is n, then the overall dimension of the stacked up feature vector will be $lwhn$. Even for moderate values of these numbers, e.g. let's say $(l, w, h) = (8, 4, 4)$ and $n = 5$, this translates into a feature vector of 640 dimensions to go into SVM classification. This is a quite large number of dimensions.

The above line of reasoning means that the feature vectors in the case of sliding window detectors usually live in a high dimensional space already, making the extra effort of the kernel trick that lifts the features up into a higher dimensional space less justifiable. It is true that additional performance gains may be expected with kernelised (i.e. nonlinear) SVM's. In particular, classification using the Radial Basis Function (RBF) kernel performs at least as well as the linear SVM (Hsu et al., 2003). However, this statement only holds true if the hyperparameters of the SVM cost function and the RBF kernel have been selected via cross-validation (Hsu et al., 2003).

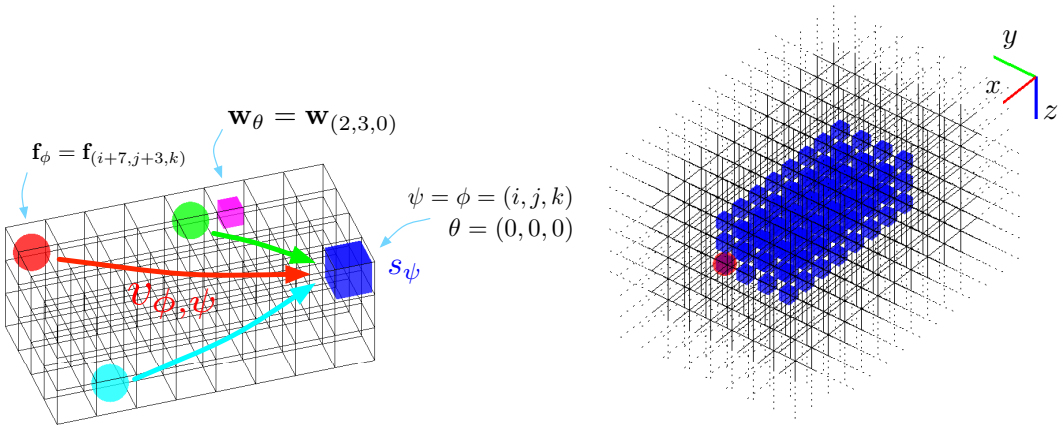
Another *key* difference of linear SVM's compared with nonlinear SVM's in the case of sliding window detection is that the application of linear models to a sliding window on a feature grid can be viewed as a convolution, and therefore standard techniques such as the Fast Fourier Transform (FFT) can be applied to compute the detection scores efficiently (Dubout and Fleuret, 2012). Unfortunately, this tech-

nique does not apply in our case in 3D. Sparsity in the spatial domain does not imply sparsity in the frequency domain, thus the Fourier transform of the sparse feature grid will be dense. However, as we show in Section 4.5, we can view convolution on a sparse feature grid from a different angle. This leads to an efficient way of computing the detection scores fully exploiting the sparse nature of the problem to our advantage. We take this to be our main contribution in this work. Note again, the technique we are about to describe is *only* applicable in the case of a linear classifier.

4.5 The Duality between Sparse Convolution and Voting

We prove, in this section, the key result of this chapter. That is, sparse convolution is mathematically equivalent to the process of voting. Before we begin the derivation, some mathematical formality has to be set up.

The feature grid is naturally four-dimensional – there is one feature *vector* per cell, and cells span a three-dimensional grid. Let us denote the l'th feature at cell location (i, j, k) by f^l_{ijk}. Alternatively, we will find it convenient to refer to all features computed at location (i, j, k) collectively as a vector $\mathbf{f}_{ijk}$. To keep the presentation simple and clear, and to avoid cluttering the mathematics with multiple subscripts, we refer to the tuple (i, j, k) by a single variable, e.g. $\phi = (i, j, k)$. If the grid dimension is (N^G_x, N^G_y, N^G_z) then define the set $\Phi = [0, N^G_x) \times [0, N^G_y) \times [0, N^G_z)$, thus $\phi \in \Phi$. Here the notation $[m, n)$ is to be understood as the standard half open interval defined over the set of integers, i.e. $[m, n) = \{q \in \mathbb{Z} : m \leq q < n\}$, and "$\times$" denotes the set Cartesian product. In this notation $\mathbf{f}_{ijk}$ can be written in the cleaner form $\mathbf{f}_\phi$ (this indexing notation is illustrated in Figure 4.2(a)). Recall that by definition $\mathbf{f}_\phi = \mathbf{0}$ if the cell at ϕ is not occupied. We can capture this concept



(a) (b)

Figure 4.2: (a) An illustration of the duality between convolution and voting. This location of the detection window happens to include only three occupied cells (represented by the three coloured spheres). The origin (anchor point) of the detection window is highlighted by the big blue cube at the corner, which happens to coincide with the cell location $\psi = \phi = (i, j, k)$ on the feature grid. Being the origin of the detection window, the anchor point has coordinates $\theta = (0, 0, 0)$ on the detection window. The feature vector for the occupied cell at grid location $\phi = (i+7, j+3, k)$ is shown as an illustration. The weights from the linear classifier are dense, and four-dimensional. The weight vector for an example location $\theta = (2, 3, 0)$ is highlighted by a small magenta cube. All three occupied cells cast votes to the window location ψ, contributing to the score s_ψ. (b) An illustration of the votes that a single occupied cell casts. The location of the occupied cell is indicated by the red sphere and the origins of detection windows that receive votes from it are represented by blue cubes. This example is for a $8 \times 4 \times 3$ window.

by defining a subset $\Phi^* \subset \Phi$ that represents the subset of cell locations that are *occupied*. Thus $\phi \in \Phi \setminus \Phi^* \implies \mathbf{f}_\phi = \mathbf{0}$. The feature grid is *sparse*.

Similarly, if the dimensions of the detection window is (N_x^W, N_y^W, N_z^W), then define the set $\Theta = [0, N_x^W) \times [0, N_y^W) \times [0, N_z^W)$, we may denote the weights associated with location $\theta \in \Theta$ as $\mathbf{w}_\theta$ (an example is also illustrated in Figure 4.2(a)). In contrary to the feature grid, the weights can be dense.

Finally, to save us from worrying about boundary conditions, we define the feature vectors and weight vectors to be zero if its index is outside the bounds. For

example, $\mathbf{w}_\theta = \mathbf{0}$ if $\theta = (-1, 0, 0)$. This extends the set of indices in both cases (feature and weights) to the full $\mathbb{Z}^3$. We are now in a position to derive the main result of this section.

Theorem 4.1. *The detection score s_ψ for the detection window with origin placed at grid location ψ can be written as a sum of votes from occupied cells that fall within the detection window.*

Proof. We begin by writing down the explicit form for the detection score s_ψ according to the linear classifier

$$s_\psi = \sum_{\theta \in \Theta} \mathbf{f}_{\psi+\theta} \cdot \mathbf{w}_\theta \ , \tag{4.1}$$

where "$\cdot$" denotes the vector dot product. Since $\mathbf{w}_\theta = \mathbf{0}$ whenever $\theta \notin \Theta$, the summation can be extended to the entire $\mathbb{Z}^3$, then after a change of variables $\phi = \psi + \theta$ we arrive at

$$\begin{aligned} s_\psi &= \sum_{\theta \in \mathbb{Z}^3} \mathbf{f}_{\psi+\theta} \cdot \mathbf{w}_\theta && (4.2)\\ &= \sum_{\phi \in \mathbb{Z}^3} \mathbf{f}_{\phi} \cdot \mathbf{w}_{\phi-\psi} && (4.3)\\ &= \sum_{\phi \in \Phi} \mathbf{f}_{\phi} \cdot \mathbf{w}_{\phi-\psi} && (4.4)\\ &= \sum_{\phi \in \Phi^*} \mathbf{f}_{\phi} \cdot \mathbf{w}_{\phi-\psi} \ . && (4.5) \end{aligned}$$

Equation (4.4) follows from Equation (4.3) because $\mathbf{f}_\phi = \mathbf{0}\ \forall \phi \notin \Phi$, and Equation (4.5) then follows from Equation (4.4) because $\mathbf{f}_\phi = \mathbf{0}$ for unoccupied cells by definition.

Now, we note also, that $\mathbf{w}_\theta = \mathbf{0}\ \forall \theta \notin \Theta$, this implies that the summation in

Equation (4.5) further reduces to

$$s_\psi = \sum_{\phi \in \Phi^* \cap \Gamma_\psi} \mathbf{f}_\phi \cdot \mathbf{w}_{\phi-\psi} \ , \tag{4.6}$$

where $\Gamma_\psi = \{\phi \in \mathbb{Z}^3 : \phi - \psi \in \Theta\} = \{\phi \in \mathbb{Z}^3 : \exists \theta \in \Theta, \phi = \psi + \theta\}$.

If we now define the *vote* from the occupied cell at location ϕ to the window at location ψ as $v_{\phi,\psi} = \mathbf{f}_\phi \cdot \mathbf{w}_{\phi-\psi}$, Equation (4.6) becomes

$$s_\psi = \sum_{\phi \in \Phi^* \cap \Gamma_\psi} v_{\phi,\psi} \ . \tag{4.7}$$

This completes the proof. □

Theorem 4.1 gives sliding window detection on a sparse grid a second view, in that each detection window location is voted by its contributing occupied cells. This is illustrated in Figure 4.2(a). Indeed, we can also imagine votes being cast from each occupied cell for *different* detection window locations in support of the existence of an object of interest at those particular window locations. This view of the voting process is summarised by the next corollary.

Corollary 4.1. *The three-dimensional score array s can be written as a sum of arrays of votes one from each occupied cell.*

Proof. First, we note that s is a function that maps elements in $\mathbb{Z}^3$ to real numbers (the detection scores at different window locations), that is $s : \mathbb{Z}^3 \to \mathbb{R}$.

With this view in mind, we turn our attention back to Equation (4.5), with our previous definition of the vote $v_{\phi,\psi} = \mathbf{f}_\phi \cdot \mathbf{w}_{\phi-\psi}$, we arrive at

$$s_\psi = \sum_{\phi \in \Phi^*} v_{\phi,\psi} \ . \tag{4.8}$$

Now, v is defined for each $\phi, \psi \in \mathbb{Z}^3$. Given a fixed ϕ, with some abuse of notations,

we define a function $v_\phi : \mathbb{Z}^3 \to \mathbb{R}$ such that $v_\phi(\psi) = v_{\phi,\psi}\ \forall \psi \in \mathbb{Z}^3$. It is now obvious that the three-dimensional score array s can be written as

$$s = \sum_{\phi \in \Phi^*} v_\phi \ . \tag{4.9}$$

□

We now turn our attention to the structure of the 3D array v_ϕ. By definition, $v_\phi(\psi) = v_{\phi,\psi} = \mathbf{f}_\phi \cdot \mathbf{w}_{\phi-\psi}$, this implies that $v_\phi(\psi) = 0$ whenever $\phi - \psi \notin \Theta$. Noting that ϕ specifies the "id" of the occupied cell where the votes originate from, and ψ the window location a vote is being cast to, this means only windows at locations satisfying $\phi - \psi \in \Theta$ receive possibly a non-zero vote from the cell. Now given a fixed ϕ, define the set $\Lambda_\phi = \{\psi \in \mathbb{Z}^3 : \phi - \psi \in \Theta\} = \{\psi \in \mathbb{Z}^3 : \exists \theta \in \Theta, \psi = \phi - \theta\}$. Then the argument above limits the votes from cell ϕ to the subset of window locations given by Λ_ϕ. Λ_ϕ includes all window locations whose origins are located in a window of the same size as the detection window but *going backwards* from the cell location ϕ. Referring to Figure 4.2(b), the red sphere in the figure represents the location of the occupied cell ϕ and blue cubes indicate window locations that will receive votes from ϕ, that is, the set Λ_ϕ.

With the insight of the structure of voting gained, Corollary 4.1 readily translates into an efficient algorithm – Algorithm 4.1 – to compute the array of detection scores s by voting. The new set of indices $\Psi \subset \mathbb{Z}^3$ introduced in Algorithm 4.1 is the set of window locations that possibly receive a non-zero score, that is, $\Psi = [1 - N_x^W, N_x^G) \times [1 - N_y^W, N_y^G) \times [1 - N_z^W, N_z^G)$. Note that the real computation happens inside the double loop where the dot product $\mathbf{f}_\phi \cdot \mathbf{w}_\theta$ is computed for all $\phi \in \Phi^*$ and $\theta \in \Theta$. This, in fact, can be thought of as a single matrix-to-matrix multiplication as follows. First, all the feature vectors for the occupied cells are stacked horizontally to form a feature matrix $\mathbf{F}$ that is of size $d \times N$, where d is the

```
1  Function ComputeScoreArray(w,f)
       Input: Weights of the classifier w and the feature grid f.
       Output: The array of detection scores s.
2      // Initialise the score array with zero values.
3      for ψ ∈ Ψ do
4          s_ψ ← 0;
5      end
6      // Begin voting.
7      for φ ∈ Φ* do
8          for θ ∈ Θ do
9              s_{φ−θ} ← s_{φ−θ} + f_φ · w_θ;
10         end
11     end
12     return s;
13 end
```

Algorithm 4.1: Computing the score array given weights of the classifier and the feature grid. See text for details.

dimension of the feature vector *per cell*, and N is the total number of *occupied* cells. Then the weights of the classifier are arranged in a weight matrix $\mathbf{W}$ of size $M \times d$, where M is the total number of cells of the detection window. That is, each row of $\mathbf{W}$ corresponds to the transposition of some $\mathbf{w}_\theta$ for some $\theta \in \Theta$. Now all the votes from all occupied cells can be computed in one go as $\mathbf{V} = \mathbf{WF}$. The $M \times N$ *votes* matrix $\mathbf{V}$ then contains for each column the votes going to the window locations Λ_ϕ for some occupied cell $\phi \in \Phi^*$.

However, despite the elegance of this way of computing all the votes, it is in practice more advisable to compute individual columns of $\mathbf{V}$ as $\mathbf{v}_i = \mathbf{W}\mathbf{f}_i$, where with some abuse of notations, here $\mathbf{v}_i$ denotes the i'th column of $\mathbf{V}$ and similarly $\mathbf{f}_i$ the i'th column of $\mathbf{F}$, and add these votes to the score matrix at each iteration in a batch. The reason being that the size of the entire matrix $\mathbf{V}$ is $M \times N$, that is, the total number of cells in the detection window (which can be in the order of a thousand) by the number of all occupied cells in the entire feature grid (a fraction

of the total number of cells in the feature grid). In most practical cases, $\mathbf{V}$ is simply too large to be stored in memory.

Finally, as an aside, the next corollary verifies that sliding window detection with a linear classifier is indeed equivalent to convolution.

Corollary 4.2.

$$s_\psi = \sum_{\phi \in \mathbb{Z}^3} \tilde{\mathbf{w}}_{\psi-\phi} \cdot \mathbf{f}_\phi \tag{4.10}$$

for some $\tilde{\mathbf{w}}$ *related to* $\mathbf{w}$.

Proof. The proof is trivial. Looking at Equation (4.3), we may define a *reversed* array of weights $\tilde{\mathbf{w}}$ by having $\tilde{\mathbf{w}}_\theta = \mathbf{w}_{-\theta}$ for all $\theta \in \mathbb{Z}^3$. Equation (4.10) then follows from Equation (4.3). □

We are probably not the first to notice the duality between sliding window detection with linear classifiers and voting. Lehmann et al. (2011) use a similar argument to justify the voting process in the Implicit Shape Model (ISM). In their framework named Principled Implicit Shape Model (PRISM), it is argued that the Implicit Shape Model is in fact equivalent to the sliding window detector – they are two sides of the same coin.

However, there are three main differences of the derivation presented in this section to the PRISM framework (other than that they are applied to different sensor modalities – Lehmann et al. (2011) focus on image-based object detection):

1. The "votes" are not cast into a continuous search space, they vote directly for the discrete locations of the sliding window.

2. There are no codebooks generated, feature vectors are not matched to any exemplars. Instead, votes are simply the scalar product between the feature vector and the corresponding weight vector.

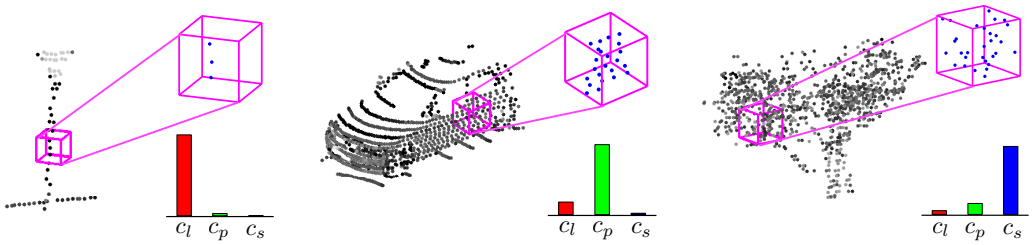
3. Finally, and most importantly, instead of a conceptual equivalence, what we demonstrate in this section, is an *exact mathematical equivalence* for convolution on a sparse feature grid.

As closing remarks, the theoretical results derived in this section can be transferred to the simpler 2D case with minimal changes (in fact, one only needs to define the index sets Φ, Θ and Ψ to be subsets of $\mathbb{Z}^2$ instead of $\mathbb{Z}^3$). Thus these theoretical results may also prove useful for applications such as a sliding window detector for 2D laser scans, or sliding window object detection with sparse image features such as, for example, edge maps.

4.6 Feature Extraction

This section is concerned with the middle right block of Figure 4.1, that is, given an occupied cell containing scattered 3D points with reflectance values, how it is mapped to a fixed, finite-dimensional feature vector.

Throughout the experiments for this work, we fix the grid size to be 20cm. Since this is a small scale, points contained within occupied cells appear to have simple and local distributions. Broadly speaking, they may appear to be rod-like, plane-like, a scatter of points in space or a mixture of the three. This is exactly what the shape factors (cf. Section 3.3.4.3) cover. Figure 4.3 illustrates three situations with examples from real 3D laser scans where the shape factors will be the most discriminative. Vertical shafts such as the shaft of the sign post shown in Figure 4.3(a) will give a high linear score c_l. Planar patches on a car will give cells having a high planar score c_p. The crown of a tree typically appear as a scatter of points in space in a laser scan and thus gives cells with high spherical scores c_s. Of course in general shape factors computed for an occupied cell may have any arbitrary proportions of linear, planar and spherical scores, making the shape factors far more descriptive



(a) (b) (c)

Figure 4.3: Motivation for shape factors in sliding window detection in 3D. The most useful cases: (a) vertical shafts (sign posts as shown here or column poles) have cells that are locally rod-like, (b) cars are mainly composed of locally planar patches, and (c) trees tend to leave a scatter of points that appear randomly distributed in the entire space giving rise to a high spherical score. All three cases are taken from real 3D laser scans. Typical example cells are indicated in each case and magnified at the top right with the actual computed shape factors for the linear (c_l), planar (c_p) and spherical (c_s) scores displayed at the bottom right as bar charts. A generic cell will have shape factors that are a blend of these three canonical cases.

than the simple canonical cases illustrated here.

To capture the appearance information provided by the reflectance values, we also include in our feature set the mean and variance of the reflectance values of points contained in the cell. These simple features may not appear to be descriptive when considering just a *single* cell. However, we note again that the cell size is typically small, thus the mean and variance are usually sufficient to capture the most important aspects of the handful of points falling within it. And considering an object is described by a collection of cells (and that the relative positions of these cells *do* matter), the overall descriptive power of these apparently simple features can be rich.

Finally, we also include a binary occupancy feature that is 1 for a cell that is occupied and 0 if it is not. This gives a total of 6 features for each cell.

We stress here that designing the best features for a sliding window detector in 3D is not the main focus of this work. However, the simple feature set we have chosen gives a good detection performance as is demonstrated in Section 4.8.

4.7 Non-Maximum Suppression

Each object may trigger multiple detections in its close vicinity. To remove duplicate detections, we apply a non-maximum suppression procedure analogous to the standard technique commonly applied in Computer Vision (Dalal and Triggs, 2005; Felzenszwalb et al., 2010; Neubeck and Van Gool, 2006). Specifically, we follow the greedy approach described in (Felzenszwalb et al., 2010).

The result of the voting procedure described in Section 4.5 is a score array giving a detection score for each window location. All window locations with a detection score (strictly) greater than a given threshold σ are then classified as positive. Because voting is conducted for each angle of rotation about the vertical axis in N discretised orientation bins (cf. Section 4.3), a set of such positive detection windows will be collected for *each* orientation bin. All the positive detection window locations (including its orientation) paired with their detection scores are then pooled into an array $\mathcal{P}$.

The detection threshold σ is restricted to be non-negative, that is, $\sigma \geq 0$. This requirement is to ensure that the empty window, which will have a zero score, is always classified as negative. The selection of σ is usually conducted with a dataset independent of the dataset used for training the linear classifier. We defer the specific strategy used to select the threshold value until Section 4.8.5.

The non-maximum suppression then proceeds as outlined in Algorithm 4.2. The positive windows are first sorted in descending order of their detection scores. Then they are taken one-by-one in that order, and compared with the current list of accepted window locations (initialised to be empty). A window location is accepted and added to the list of accepted windows if it doesn't overlap with any of the previously accepted object windows by more than a given threshold. This overlap threshold is a system parameter that is set based on intuition.

```
1  Function NonMaximumSuppression(P, t_o)
       Input: P, an array of scored oriented positive detection windows, and t_o
              the overlap threshold.
       Output: D, a subarray of P selected as detection results.
2      // Sort the detection windows according to their scores in
          descending order.
3      SortInDescendingScore(P);
4      // Initalise D as an empty array.
5      D ← [ ];
6      for i ← 1 to Length(P) do
7          overlap ← false;
8          for j ← 1 to Length(D) do
9              o ← Intersection(P_i, D_j) / Union(P_i, D_j);
10             if o > t_o then
11                 overlap ← true;
12                 break;
13             end
14         end
15         if not overlap then
16             AppendToArray(P_i, D);
17         end
18     end
19     return D;
20 end
```

Algorithm 4.2: The greedy approach to non-maximum suppression. The list of positive detection windows are first sorted in descending order of their detection scores. Each window is then taken in turn, added to the array of windows for output if it does not overlap with any previously selected windows for more than a certain threshold.

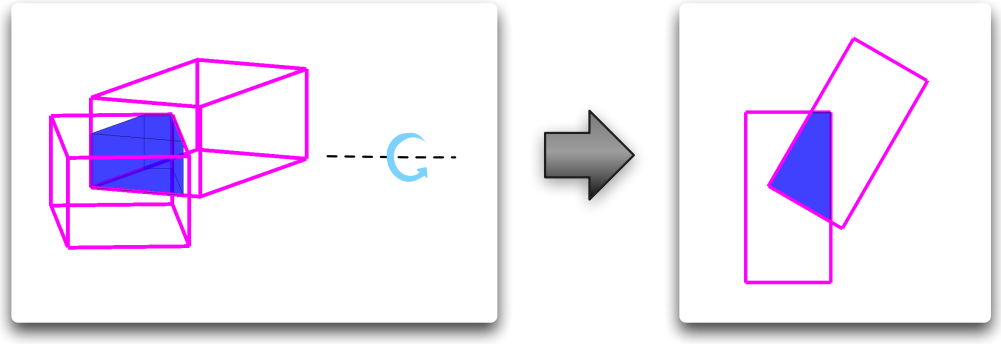


Figure 4.4: An illustration of two intersecting extruded boxes. The volume of intersection is highlighted in blue in both the 3D view and the bird's-eye view. The concept of intersection for extruded boxes is closely related to that of their cross-sections.

Overlap between two object windows is computed as the ratio of the volume of intersection of the windows (treated geometrically as orientated 3D boxes) over the volume of their union.

4.7.1 Efficient Computation of Overlap between Extruded Boxes

There is not much to be said about computing the overlap between two axis-aligned 2D boxes (albeit different sizes). This is commonly the case encountered for image-based sliding window detectors. The corresponding case in 3D, however, requires computing the overlap between *oriented* (i.e. non-axis-aligned) 3D boxes. Computing the intersection between two *arbitrarily oriented* 3D boxes efficiently is quite a complex problem (Gottschalk, 2000).

Fortunately, in our case, the orientation is not arbitrary – rotation is constrained to be about the vertical axis. Each box may be treated as an arbitrarily oriented box *in 2D*, and then extruded along the vertical direction to make the full 3D detection window (See Figure 4.4 for an illustration). The intersection between two arbitrary polygons, of which oriented 2D boxes are special cases, is a well studied problem

```
1 Function Intersects(W_1, W_2)
      Input: W_1 = (B_1, z_l^1, z_h^1) and W_2 = (B_2, z_l^2, z_h^2), the pair of oriented
             detection windows to be tested for intersection.
      Output: intsc, a boolean value indicating the result of the test.
2     if z_l^1 > z_h^2 or z_h^1 < z_l^2 then
3         intsc ← false;
4     else
5         intsc ← Intersects2D(B_1, B_2);
6     end
7     return intsc;
8 end
```

Algorithm 4.3: An algorithm for testing for intersection of two extruded boxes. First, quick tests are done using the range of extrusion along the vertical axis to determine simple cases of non-intersection, then if these tests are indeterminate, the 2D intersection test `Intersects2D` is called.

in computational geometry, and efficient implementations are readily available (for example, the Boost Geometry library[1] includes efficient implementations for both the test for intersection and the computation of the area of intersection for polygons).

Each oriented detection window is an extruded box represented by a tuple $W = (B, z_l, z_h)$, where B is the oriented 2D box, and $[z_l, z_h]$ gives the range of extrusion along the vertical axis. Algorithm 4.3 then gives an algorithm for testing intersection of two such extruded boxes based on the assumption that test of intersection for the 2D case is available. The geometry of the problem is illustrated by Figure 4.4. Provided their ranges of extrusion overlap, the two boxes intersect if and only if their 2D cross-sections intersect. Now we can easily test whether two extruded boxes intersect, we can build on that to compute their volume of intersection as illustrated by Algorithm 4.4. Again here we assume we have available a method of computing the area of intersection of two oriented 2D boxes. With reference again to Figure 4.4, the volume of intersection of two extruded boxes is itself an extruded shape of the 2D intersection of the boxes' cross-sections.

[1] `www.boost.org`

Function `Intersection`(W_1, W_2)

  Input: $W_1 = (B_1, z_l^1, z_h^1)$ and $W_2 = (B_2, z_l^2, z_h^2)$, the pair of oriented detection windows whose volume of intersection is to be computed.

  Output: v the volume of intersection.

  if not `Intersects`(W_1, W_2) **then**

    $v \leftarrow 0$;

  else

    $v \leftarrow$ `Intersection2D`$(B_1, B_2) * ($`Min`$(z_h^1, z_h^2) -$ `Max`$(z_l^1, z_l^2))$;

  end

  return v;

end

Algorithm 4.4: An algorithm for calculating the volume of intersection for two extruded boxes. It is assumed that computing the area of intersection for two oriented 2D boxes is easy (given by the function `Intersection2D`).

Function `Union`(W_1, W_2)

  Input: $W_1 = (B_1, z_l^1, z_h^1)$ and $W_2 = (B_2, z_l^2, z_h^2)$, the pair of oriented detection windows whose volume of union is to be computed.

  Output: v the volume of union.

  $v \leftarrow$ `Volume`$(W_1) +$ `Volume`$(W_2) -$ `Intersection`(W_1, W_2);

  return v;

end

Algorithm 4.5: A straightforward algorithm for computing the volume of union of two extruded boxes.

Finally, for completeness, we give the straightforward computation of the volume of union for two extruded boxes in Algorithm 4.5, based on their volume of intersection (cf. Figure 4.4).

4.8 Evaluation

To facilitate supervised learning, we take advantage of the publicly available KITTI dataset (Geiger et al., 2013). The object detection benchmark from the KITTI dataset supplies synchronised camera and Velodyne frames, with objects annotated

in both image and laser data. Specific to our interests is that the annotations in the laser data are given as complete oriented 3D bounding boxes bounding the object of interest in a canonical orientation.

We evaluate the performance of the proposed 3D sliding window detector on the *car* class, and demonstrate its superior performance in terms of both the absolute performance based on common evaluation metrics and the relative performance compared both quantitatively with the previously proposed segmentation-based detector in Chapter 3, and qualitatively with state of the art vision-based methods from a practical point of view.

Details related to training are given in Section 4.8.1. In Section 4.8.2 we explain our evaluation strategy, specifically, how the ground truth labels are divided into three difficulty levels: *hard*, *moderate* and *easy* similar to the original split in the image annotations of the KITTI dataset (Geiger et al., 2013) but now according to a criterion suitable to laser-based detections. Then in Section 4.8.3 we present our main results of the detection performance on the car class. Section 4.8.4 investigates the relative importance of the features chosen. Section 4.8.5 gives an evaluation of the computation time taken by the proposed detector. In Section 4.8.6, we compare the sliding window detector with the previously proposed segmentation-based detector aiming more at a better understanding of the advantages and disadvantages of the two approaches in general than simply looking at the performance differences due to the specific implementations. Finally, in Section 4.8.7 we compare qualitatively with current state of the art vision-based car detectors reported on the KITTI dataset.

4.8.1 Training

The standard KITTI object detection benchmark contains a labelled training set and a labelled test set. However, the labels on the test set are held back for evaluation

	No. of Frames	No. of Cars		
		Hard (Total)	Moderate	Easy
All	7481	28742	18971	12611
Training (80%)	5985	22802	15028	9941
Testing (20%)	1496	5940	3943	2670

Table 4.1: Details of data splits for training and testing. The first row shows information about *all* data with annotation that is publicly available (this is what the KITTI object detection benchmark calls "training" data). The second and third rows show information about the subsets of data that has been selected for training and testing respectively. We take an 80/20 random split for training and testing. The first column shows the number of frames in the dataset. The following three columns show the number of annotated ground-truth *hard*, *moderate* and *easy* cars respectively contained in the dataset. See Section 4.8.2 for how these difficulty levels are classified. Note the number of *hard* cars is also the *total* number of cars present in the dataset.

purposes. Since what we are interested in here is a fair evaluation of the performance of the sliding window detector on 3D data, whereas KITTI is primarily a vision dataset, we create our own training and test datasets from the labelled data in KITTI that is publicly available (i.e. the original "training" dataset) by randomly splitting it into two parts and then test the detector's performance based on metrics that are more suitable for evaluating detections in 3D (cf. Section 4.8.2).

Specifically, we randomly split the 7481 labelled frames available into 80/20 proportions for training and testing respectively. The numbers of frames contained in the resulting training and test sets, together with other information, are tabulated in Table 4.1.

For training the linear SVM classifier, we use the LIBLINEAR library (Fan et al., 2008). An initial set (equal to the number of positive examples) of negative examples are randomly sampled from the training data taking care not to overlap with any positive examples. Taking this initial set of training examples, we adopt the standard hard negative mining technique from image-based object detectors (e.g. Sung and Poggio, 1998; Dalal and Triggs, 2005; Felzenszwalb et al., 2010).

Specifically, a classifier is first trained on the initial training set. After training, the classifier is applied back on all the *training* frames. All false positive detections from this classifier on all the training frames are collated, and sorted in descending order of the detection score. The first N (or all of the false positives if there are less than N of them) are then taken and added to the set of negative examples. The classifier is then retrained with this updated training set and this process may iterate for a predefined number of rounds. In all our experiments that follow, we fix N to be 10000 and conduct 20 rounds of hard negative mining.

A disadvantage of sliding window approaches is the artefacts introduced during the discretisation process. Because window locations are only searched on the discretised feature grid (and the discretised angle of rotation), it is unlikely an object is captured in the detection window in precisely its canonical pose. However, the positive examples for training are extracted from manual labels, the objects contained are therefore centred and facing forward. To compensate for this discrepancy, for each positive example, we randomly sample 10 slightly translated and rotated (about the vertical axis) versions of it, and append them to the set of positive examples for training.

4.8.2 Evaluation Strategy

The object labels provided by the KITTI dataset on the 3D laser data are comprehensive in the sense that, as well as obvious object instances, challenging objects that are heavily occluded or very sparsely sampled due to being at a large distance from the sensor are also included. The included objects may at times be as challenging as being described by only a handful of laser measurements (see, for example, the left column of Figure 4.5).

This motivates us to divide the labelled car instances into different difficulty levels similar to the original KITTI specification (Geiger et al., 2013), to respect the

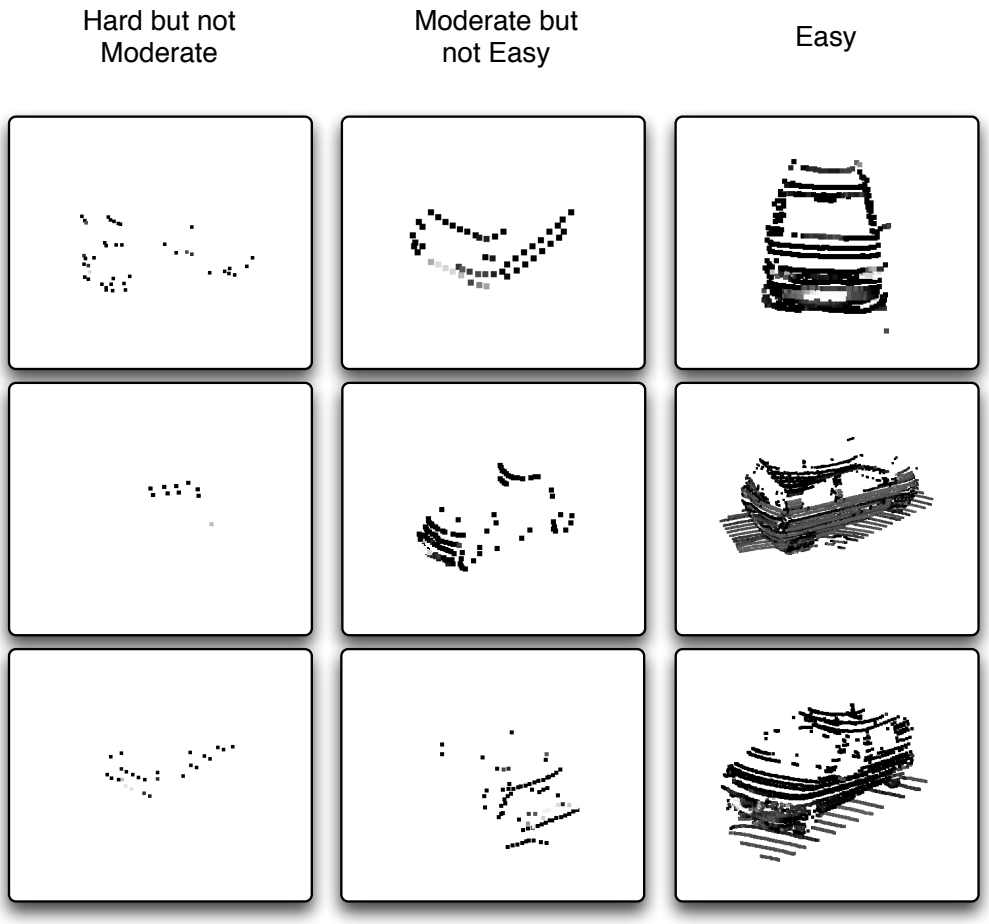


Figure 4.5: Examples of labelled car instances from the training set of different difficulties. Left column: hard but not moderate, instances containing numbers of measurements $m < 50$. Middle column: moderate but not easy, instances containing numbers of measurements $50 \leq m < 150$. Right column: easy, instances containing numbers of measurements $m \geq 150$. See text for details.

complete set of labels from the dataset at the same time not to place unreasonable demands to the detection system.

The original KITTI specification is tailored specifically to vision-based detection systems. Here, we first take a closer look into the dataset for the types of labelled car instances provided in the 3D *laser* data, and based on that, devise suitable criteria for dividing the objects into the *easy*, *moderate* and *hard* difficulty levels.

Figure 4.5 presents examples of labelled car instances from the KITTI Velodyne data. As can be noted, from left to right, the identity of the object ranges from very difficult to judge, to being obvious (as far as a human perceiver is concerned) that it is a car. The left column displays example ground truth labels that contain only less than 50 laser measurements, the middle column shows examples that contain between 50 and 150 laser measurements, whereas the right column gives examples that have over 150 measurements on them. Examples in the left column contain insufficient measurements for even a human observer to tell its identity. On closer inspection, a human observer may be able to identify the examples of cars in the middle column. Finally, the features of a car are much better defined for the examples in the right column. Given the observations above, we define the *easy* car instances as instances described by over 150 laser measurements, the *moderate* car instances as instances described by over 50 laser measurements, and the *hard* car instances including *all* labelled instances provided in the dataset. Note the set of *hard* instances include the set of *moderate* instances, and similarly the set of *moderate* instances include the set of *easy* instances. Table 4.1 gives the number of labelled car instances of each difficulty level contained in the KITTI dataset and our splits.

We use the standard Precision and Recall metrics to evaluate the detector's performance on the test dataset (readers unfamiliar with the Precision and Recall metrics are referred to Appendix A.2 for a brief summary). Specifically, Recall for each difficulty level is computed as the ratio of the number of car instances *belonging*

to that difficulty level that are successfully detected over the total number of car instances *of that difficulty level*. Precision is computed independently of difficulty levels as the usual ratio of true detections (of cars of any difficult level) over the total number of detections.

Detections are assigned to ground truth labels in a matter similar to that described in Section 3.4.3. In this specific case, in addition to the overlap being required to be greater than 0.5 between the detection and the ground truth label, the detection has to match the angle of orientation of the ground truth object, that is, the angle of rotation about the vertical axis between the detected object box and the ground truth object box must be within $\pm\frac{\Delta}{2}$ where Δ is the angular resolution. Each detection is assigned to at most one ground truth object, and duplicate detections to the same ground truth object are taken as false positives.

4.8.3 Detection Performance

We present the key results of this chapter in this section. The sliding window detector described in this chapter is trained with the training set according to the procedure outlined in Section 4.8.1, and evaluated on the test set. There are only three parameters to the detector, the grid size δ, the number of angular bins N and the overlap threshold t_o for non-maximum suppression (cf. Section 4.7). In all our experiments, we set $\delta = 0.2$m, $N = 8$ and $t_o = 0.01$.

With the division of ground truth labels into difficulty levels defined in the previous section, in addition to *evaluating* the performance of a certain classifier at each difficulty level, we may also investigate the effect of *training* on only the ground truth objects of a particular difficulty level. For example, if all we care about is good performance with respect to detecting *easy* cars, we may reflect this in the training stage by training on only *easy* car examples. Without the confusion introduced with ambiguous cases abundant in higher difficulty levels, the performance on *easy* cars is

expected to increase. But because at no stage of training is the classifier presented with *hard* (hard but not moderate) cars for example, the performance at the *hard* difficulty level is somewhat undefined.

Naturally there arise three schemes of training based on the three difficulty levels – training on only the *easy*, *moderate* (including all easy cases) and *hard* (including all labelled cars) positive examples respectively. Figure 4.6 presents results from an experiment where the detector is trained on the training set according to the three different training schemes, each evaluated on the three different difficulty levels on the *test* set. The Precision-Recall curves presented in the figure are generated by varying the detection threshold σ (cf. Section 4.7). In general, the performance of the detector increases as the number of rounds of hard negative mining increases until convergence as one would expect. And regardless of the training scheme, the detector performs better as the evaluation difficulty decreases, with the best performance noted on the *easy* cars. According to the bottom row, the detector trained only on the *easy* car instances perform poorly compared with the other two training schemes on the *hard* and *moderate* difficulties confirming our reasoning above. However training on only cases of a certain difficulty level does not seem to increase the detection performance for that difficulty level. For example, training on the *easy* positive examples gives similar performance compared with the other two schemes on the *easy* difficulty level, and training on the *moderate* cases produces slightly worse performance at the *moderate* difficulty level than training on the *hard* cases (that is, all of the positive examples). This suggests, for example, focusing training on *only* the easy cars does not necessarily increase performance on detecting easy car instances – the detector is capable of accommodating more difficult car instances in addition to handling well the easy cases. All three training schemes perform equally well evaluated according to the *easy* difficulty level.

We note here again that the three difficulty levels *hard*, *moderate* and *easy* are

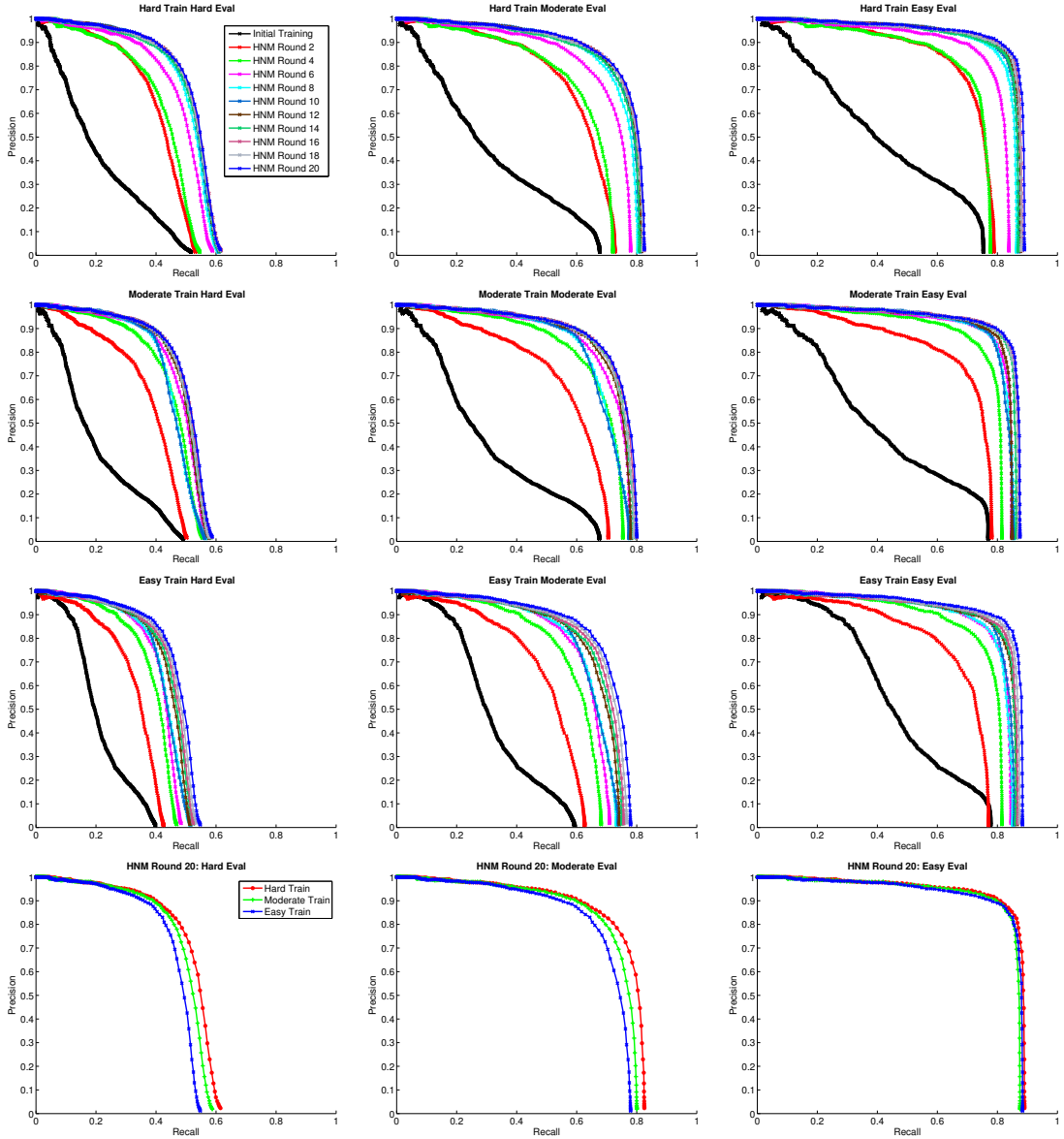


Figure 4.6: Precision-Recall curves as the number of rounds of hard negative mining (HNM) goes up. The top row gives Precision-Recall curves for training on all the *hard* positive examples and evaluating for the *hard*, *moderate* and *easy* difficulties respectively (from left to right). Similarly, the second and third row present the corresponding results for training on only the *moderate* and *easy* training examples respectively. Precision-Recall curves are shown for every two rounds of hard negative mining. The bottom row compares the relative performances of the three different training strategies at the end of training (HNM Round 20) on common axes, again evaluated for the *hard*, *moderate* and *easy* difficulties respectively (from left to right). All Precision-Recall curves are generated on the *test* dataset.

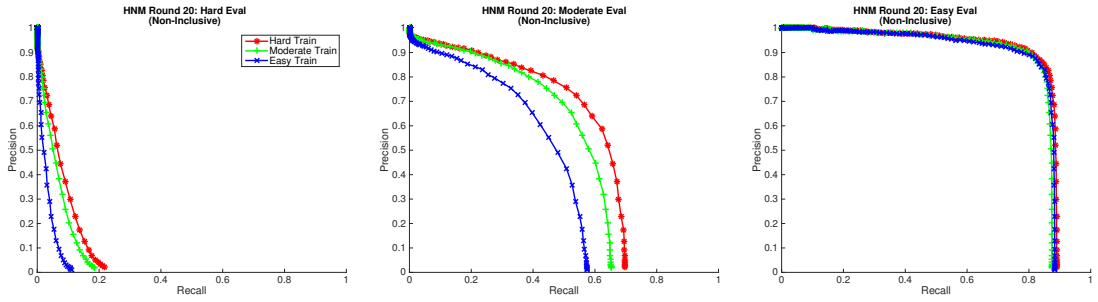


Figure 4.7: Precision-Recall curves of three training schemes similar to the bottom row of Figure 4.6, but evaluated on *non-inclusive* difficulty levels, for example, hard and only hard positive examples (i.e. hard but not moderate). Thus the plot for evaluating on the easy difficulty is the same as the corresponding plot in the bottom row of Figure 4.6, reproduced here for completeness.

inclusive, in the sense that all *hard* positive examples include all *moderate* positive examples, and all *moderate* examples in turn include all *easy* examples. We may also examine system performance over a set of *non-inclusive* difficulty levels, for example, the *hard* and only *hard* examples (hard but not moderate). Figure 4.7 shows plots similar to the bottom row of Figure 4.6 for the three training schemes, but evaluated on *non-inclusive* difficulty levels. As can be noted from the figure, all training scheme performs poorly on the non-inclusive hard difficulty, confirming the challenging situation revealed by the left column of Figure 4.5. In particular, the scheme of training on hard examples performs better than training on the other difficulty levels. This makes intuitive sense because there are no examples of hard (non-inclusive) cars in these latter training schemes during training. Similarly, both schemes trained on hard and moderate cars outperform the scheme trained on easy cars at the non-inclusive moderate difficulty. However, training on moderate cars does not improve performance at the non-inclusive moderate difficulty level over training on hard cars. Noting again that the hard examples for training include all moderate examples, this observation suggests that there may be significant overlap in appearance of the hard (hard only) and moderate (moderate only) examples (also, see Figure 4.5), and that the additional hard only examples also help in

discriminating moderate only cars too.

In the remaining experiments we focus on the performance of the proposed detector on the *moderate* and *easy* difficulty levels, because, referring back to Figure 4.5, requiring any detection system to reliably detect car instances belonging to the hard but not moderate difficulty level without incurring a large number of false detections is beyond our reach. For this purpose, the *moderate* training strategy (i.e. training on all *moderate* positive examples) is adequate judging from the bottom row of Figure 4.6, and will be followed in all our experiments hereafter. Though less well-performing at the *moderate* difficulty level compared with training on all *hard* positive examples, this choice strikes a balance between performance and resource requirements at training (from Table 4.1, moderate training instances account for only 66% of the total labelled examples).

4.8.4 How Useful are the Features?

In Section 4.6 we have chosen six simple features for the proposed sliding window detector. Experiments in the previous section demonstrated their effectiveness (see also Section 4.8.6 and Section 4.8.7 for comparative studies with other methods). However, how important are these features relative to each other? Could some of the features be redundant or confusing in discriminating the positive and negative cases? To seek answers to these questions, we plot, in Figure 4.8, a measure of relative importance of each feature in the car detection task.

To compute the measure of relative feature importance, we take the weights of the final classifier trained (i.e. after the 20th round of hard negative mining) on all *moderate* positive examples (cf. Section 4.8.3), and collate, for each feature, the corresponding weights at all cell locations of the detection window. The feature's relative importance is then computed as the mean absolute value of these weights.

Figure 4.8 suggests that the most useful features for car detection are the appear-

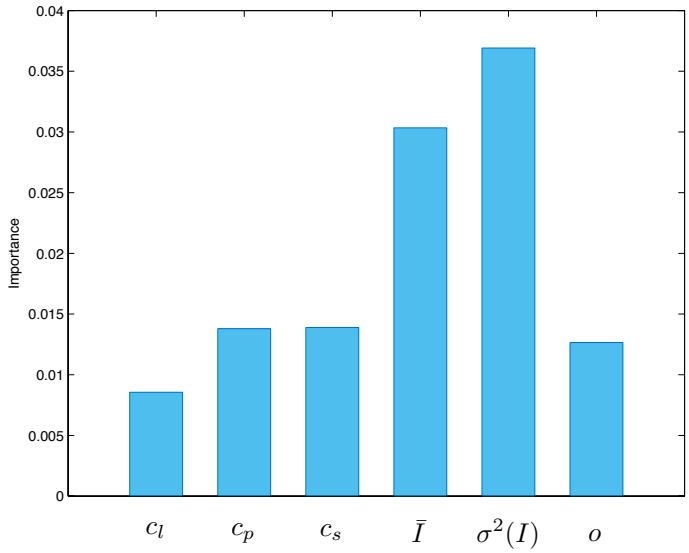


Figure 4.8: A plot of a measure of relative importance of features on the task of car detection. Features are denoted by: c_l, the linear shape factor, c_p, the planar shape factor, c_s, the spherical shape factor, $\bar{I}$, the mean of reflectance values, $\sigma^2(I)$, the variance of reflectance values, and o, the binary occupancy indicator. See Section 4.6 for detailed feature definitions.

ance features. The shape features have similar relative importance values other than the linear shape factor, which has a lower importance value. The lower importance value of the linear shape factor makes intuitive sense, because by definition the three shape factors always sum to one, given values of the other two shape factors, the linear shape factor is redundant.

To confirm the insights gained by studying Figure 4.8, we trained the detector with different feature selections, again on all *moderate* positive examples in the training dataset for up to 20 rounds of hard negative mining, and plot the Precision-Recall curves of these different variants of the detector in Figure 4.9 for the *moderate* and *easy* difficulty levels. Each Precision-Recall curve is generated with the final classifier trained (i.e. after the 20th round of hard negative mining) evaluated on the *test* dataset.

As the figure suggests, without the linear shape factor feature, the detector per-

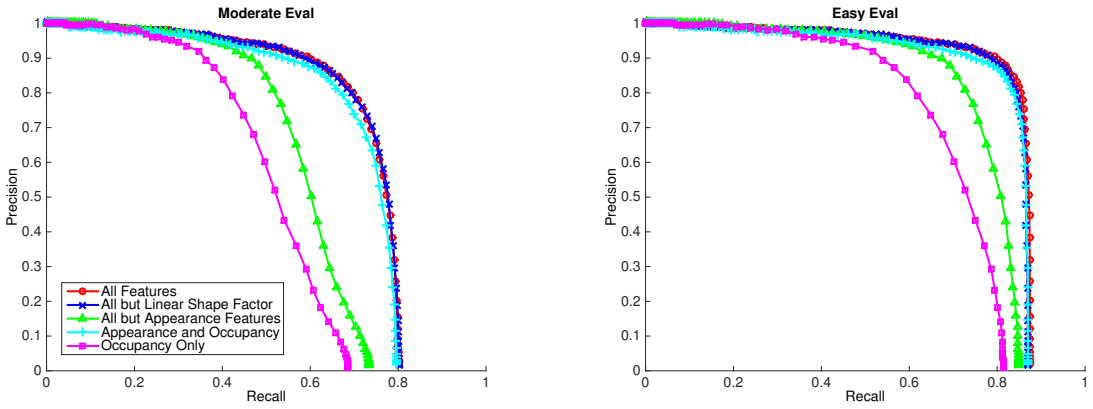


Figure 4.9: Precision-Recall curves of detector trained with different feature selections on all *moderate* positive examples in the training set evaluated at the *moderate* and *easy* difficulty levels on the *test* set. The Precision-Recall curves shown are generated with the final classifier – the classifier trained after the 20th round of hard negative mining.

forms equally well compared with the original variant of using all the features. Removing the appearance features, however, greatly degrades detection performance, confirming their dominating importance noted from Figure 4.8. Retaining only the appearance features (in addition to the binary occupancy feature, which is a compulsory feature since it is also used as an indicator for occupancy of the cell) gives only a slight decrease in performance, making one question the necessity of the shape factor features. Thus finally, we reduce the feature set to its bare minimum taking only the simplest binary occupancy feature. Compared with using the occupancy feature and the shape factors (all but appearance features), this variant performs significantly worse, suggesting shape factors do provide additional shape-cues on top of simple occupancy.

Finally, we note that although using only the binary occupancy feature decreases the system performance significantly compared with the full feature set, its performance is still reasonable in its own right, demonstrating the power of the proposed sliding window approach for 3D object detection (at least in the case of car detection).

4.8.5 Timing

In this section, we empirically analyse the computational efficiency of the proposed sliding window detector from a practical perspective. The proposed sliding window detector is implemented as a C++ library. We note the computation for each orientation bin (cf. Section 4.3) is completely independent of each other, therefore it falls within the "embarrassingly parallelisable" paradigm. We hence take full advantage of modern CPUs' multi-core architectures by treating the computation for each orientation bin as an independent job unit, which may be executed on different threads.

In what follows, we evaluate the timing aspects of our implementation on a MacBook Pro equipped with a quad-core 2GHz Intel i7 CPU and 8GB of RAM. We take the classifier trained with the full feature set on all *moderate* positive examples as the base classifier, and select the detection threshold as our point of operation from the Precision-Recall curve evaluated on the *test* dataset at the *easy* difficulty level (cf. Figure 4.6) as the threshold value that gives the highest Recall while maintaining a Precision of over 0.9. To ensure our analysis is not biased (because the detection threshold selected this way is based on the test dataset), all results quoted except otherwise stated are obtained on a third independent sequence data that is completely *unlabelled* from the KITTI dataset containing 1170 frames of Velodyne laser scans.

Figure 4.10(a) shows the computation time taken per frame of complete Velodyne scan (each scan contains about $100,000$ points) versus frame number while Figure 4.10(b) arranges the same information as a histogram plot to examine its distribution. As can be noted from Figure 4.10(a), the time taken by the detector is highly dependent on the scene structure. This is expected as both feature computation and voting depend on the number of *occupied* cells of the feature grid, cluttered scenes tend to take longer to process. Nonetheless, each frame takes at

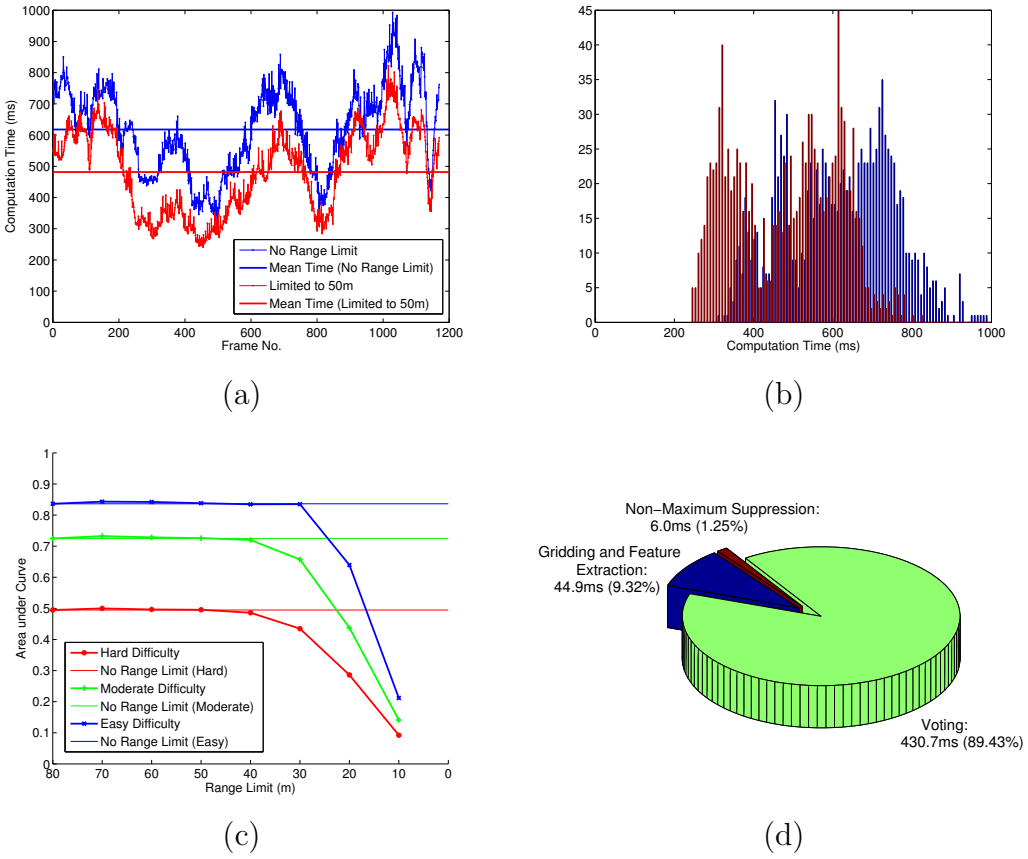


Figure 4.10: (a) Computation time versus frame number on an independent sequence data. Also shown on common axes is a similar plot restricting the range of detection to 50m. In both cases, the mean computation time per frame is given by a horizontal line. (b) A histogram plot putting the distributions of computation time per frame of the two variants (with, or without range limit) in (a) under comparison. (c) A plot of performance on the *test* dataset as the range limit decreases, evaluated at all three difficulty levels. Horizontal lines denote performance at no range limit. (d) A pie chart showing a decomposition of computation time per frame into the major system components, evaluated on the independent sequence data with a limited detection range of 50m. The time quoted for each component is the time taken for that component per frame averaged over the sequence.

most a second to process and on average only 618ms, demonstrating the feasibility of the sliding window approach to object detection in 3D.

Although the quoted maximum range of the Velodyne HDL-64E sensor[2] is as far as up to 120m (depending on surface reflectivity), in practice the useful information is contained only within a range of about 80m. Additionally, measurements at far ranges are so sparse due to the range-bearing nature of the device that reliable object detection is challenging if not impossible. By restricting the detection range, we may focus on the (relatively) close-range objects whose detection is more reliable with an additional gain of computational speed. This argument is backed up by Figure 4.10(c) where we show a plot of performance degradation as the detection range is gradually reduced (by running the detector on only the laser points that fall within range). Here performance is measured by the area under the curve (AUC) computed over the full Precision-Recall curve generated on the *test* dataset by varying the detection threshold as before, but now restricting detection to a given range. Compared with the AUC without range limit, there is no notable difference to system performance up to 50m, then the performance evaluated according to the *hard* difficulty starts to drop. The performance on the *easy* difficulty level, on the other hand, does not degrade until as close as 30m further confirming our arguments earlier that the challenging cases are largely due to sparse sampling when observed at long distances.

With this observation in mind, we plot the timing performance (on the same sequence data) when the detection range is restricted to 50m on common axes with the original timing results of the case without range limit in Figure 4.10(a) and Figure 4.10(b). The computation speed of the detector is greatly improved averaging to 482ms corresponding to an achievable processing rate of 2Hz (e.g. with buffering).

Finally, Figure 4.10(d) visualises the contributions of major components to the

[2] `http://www.velodynelidar.com/lidar/hdlproducts/hdl64e.aspx`

total computation time per frame as a pie chart. The majority of processing time is spent on voting, while a small proportion is spent in the computation of the feature grid. Non-maximum suppression takes the thinnest slice contributing to only 1.25% of the total time.

4.8.6 Comparison with the Segmentation-Based Approach

In this section, we quantitatively compare the sliding window object detector proposed with the segmentation-based detector described in Chapter 3, in hope to gain insights into how a sliding window approach to object detection in 3D compares with the segmentation-based approach that is popular among state of the art 3D detection methods.

To ensure a fair comparison with the existing results quoted in Chapter 3, we take care to follow exactly the same evaluation procedure and use exactly the same evaluation dataset on which results presented in Section 3.4.3 are obtained. In particular, each oriented object box as output from the sliding window detector is converted to a corresponding object segment by taking all points that fall within the window. Figure 4.11 presents the Precision-Recall curve in blue of the sliding window detector evaluated in this way, compared with the results quoted in Table 3.2 on the car class for the three different detection schemes proposed for the segmentation-based detector. The variant of the sliding window detector evaluated is the one trained with the full feature set on all *moderate* positive examples from the training set (from KITTI). As may be noted from the figure, the sliding window detector outperforms the segmentation-based detector by a significant margin. Because the segmentation-based detector is purely shape-based, it does not use appearance information from the reflectance values, to compare the two approaches on a common footing, we also include in Figure 4.11 the Precision-Recall curve of the variant with *only* the shape-based features from our feature set, also evaluated on

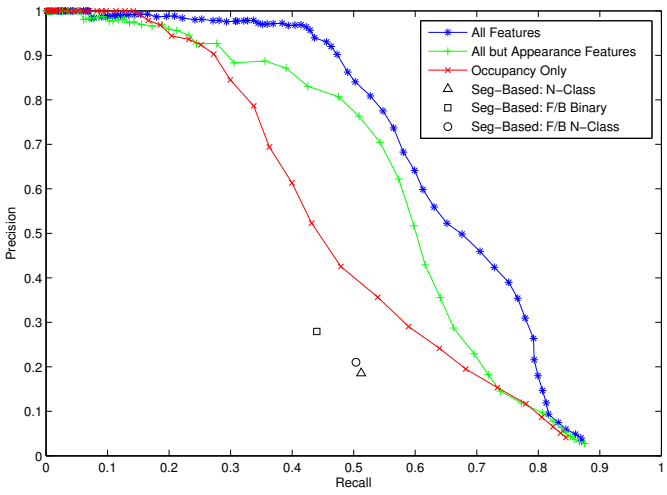


Figure 4.11: A comparative study of detection performance with the segmentation-based object detector proposed in Chapter 3. Results shown here are evaluated on the same dataset on which the segmentation-based detector is evaluated in Section 3.4.3. Performance metrics are also computed in the same manner to allow for an unbiased comparison. Results from Chapter 3 as presented in Table 3.2 for the *car* class of the three different detection schemes are placed here on common axes. See text for details.

the same dataset as the segmentation-based detector, in green. Though the performance compares less favourably with using the full feature set as one would expect, it still outperforms the segmentation-based detector. Is the success of the sliding window detector due to the benefits of taking a sliding window approach, or simply because we have been lucky enough to haven chosen a good feature set? To answer this question, we make a third comparison against the baseline sliding window performance – that is using the simplest possible feature, the binary occupancy feature *only*, and plot its Precision-Recall curve in Figure 4.11 in red. Despite such a disadvantage, the sliding window detector still outperforms the segmentation-based detector with all its sophisticated feature set (cf. Section 3.3.4). We accredit this to the rich description brought by the feature grid representation unique to sliding window approaches where an object is represented by, instead of a single feature vector, features extracted at different cell locations of the detection window, together

with the knowledge of their relative positions, even a feature as simple as the binary occupancy indicator can become quite powerful in describing an object.

Finally, we note that the comparative studies in this section are actually biased in favour of the segmentation-based detector. The classes *van* and *car* are treated separatedly in the KITTI dataset thus the sliding window detector trained will not trigger on a van. However, the object labels in the dataset of the segmentation-based detector do not make this distinction, making the performance of the sliding window detector evaluated on this dataset an underestimate.

4.8.7 A Practical Comparison with State of the Art Vision Methods

In this section, we give a qualitative comparison to the reported performance of state of the art vision methods to car detection. The bottom row of Figure 4.12 shows the published results of the top five car detectors on the KITTI object detection vision benchmarking website[3], and the top row of Figure 4.12 shows the same plots from the bottom row of Figure 4.6 reproduced here for easy comparison.

We must stress here, although quantitative results for each case are displayed in Figure 4.12, any quantitative comparison between the proposed sliding window detector and the image-based detectors is not meaningful because the evaluations are not based on common criteria. Specifically, the published results of vision methods are evaluated according to the original difficulty specification defined with respect to *vision*, whereas the results quoted in Figure 4.12 for the proposed sliding window detector is evaluated according to the *laser*-based difficulty specification defined in Section 4.8.2. However, because of the inherent difference in sensor modality, attempting to compare laser-based and vision-based methods on a completely fair basis is challenging if not impossible. For example, what is difficult in appearance

[3] `http://www.cvlibs.net/datasets/kitti/eval_object.php`

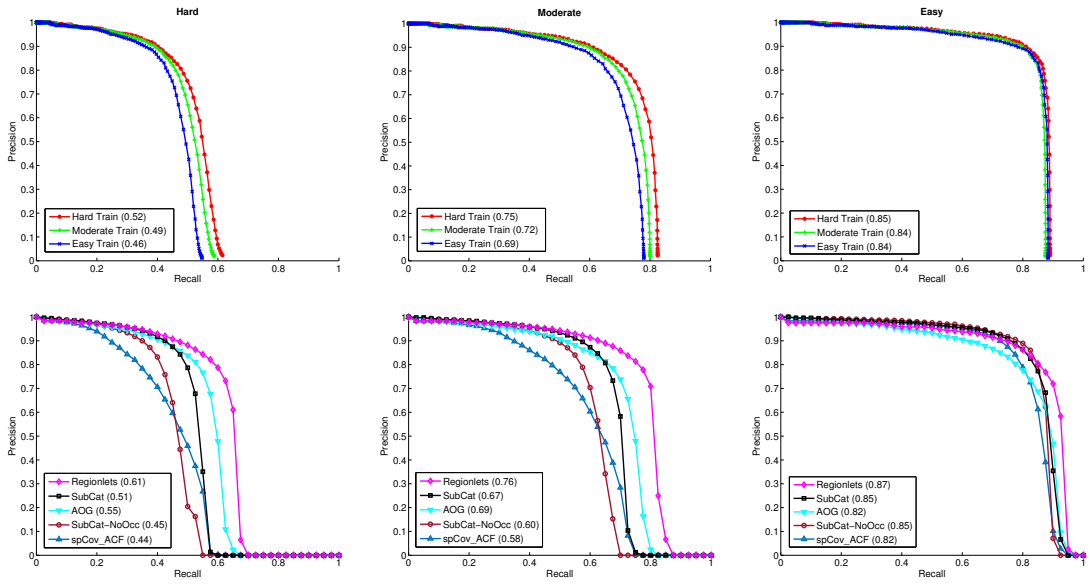


Figure 4.12: A qualitative comparison with state of the art vision-based car detectors evaluated on the KITTI dataset from a practical point of view. The top row reproduces the bottom row of Figure 4.6. The bottom row shows publicly reported Precision-Recall curves of the top five car detectors evaluated on the KITTI dataset. Numbers in brackets denote the area under the curve (AUC). Note because our proposed sliding window detector is evaluated according to the *laser* criteria whereas the vision methods are evaluated according to the *vision* criteria, it is not meaningful to draw any quantitative conclusions by placing the curves on common axes. Instead, we only note here that *qualitatively*, the performance of the proposed sliding window detector in 3D is commensurate to state of the art vision methods from a practical point of view. See text for further discussions.

in vision may not be difficult in laser and vice versa. Evaluating the performance of one sensor modality on a set of evaluation criteria designed for fair comparison for *another* sensor modality can only lead to non-meaningful results.

We may however compare qualitatively the performance of the proposed sliding window detector with the published results of current state of the art vision methods from a practical point of view. In both cases, the division to the *easy*, *moderate* and *hard* difficulty levels reflects what a human perceiver would find challenging or straight-forward to perceive the object by observation of data form the sensor modality alone (either laser or vision). Thus a qualitative comparison is meaningful because the Precision and Recall metrics in both cases evaluate how the respective detection system performs at achieving the perception task at a common indicative level of proficiency.

Hence we may note from Figure 4.12 that, from a practical point of view, the proposed sliding window detector is commensurate at the task of car detection with the current best vision-based car detectors. Note, only the *top five* vision methods are shown in Figure 4.12, the top ranked method, the "Regionlet" method (Wang et al., 2013b; Long et al., 2014) in particular, has only been submitted to the KITTI vision benchmark a few days prior to the writing of this thesis.

4.9 Conclusions

In this chapter, for the first time, we have taken the sliding window approach to object detection from vision to 3D data. The sparse nature of the problem is fully exploited to give an efficient method for exhaustively searching through all object locations at all orientations. Specifically, we proved the mathematical equivalence of convolution on a sparse feature grid and voting, and demonstrated the detector's superior performance with the KITTI dataset on the car class.

This concludes our studies on model-based objection detection. In the next chapter, we take a model-free perspective to dynamic object tracking, aiming at the detection of not only a limited set of classes of objects but any object that *moves*.

Chapter 5

Model-Free Tracking of Dynamic Objects

5.1 Motivation

A 3D laser scanner such as the Velodyne HDL-64E S2 sensor deployed during our studies in the previous chapters (cf. Figure 1.1(a)) produces a rich description of the environment, allowing for powerful model-based methods for object detection taking advantage of the shape and appearance information available unique to these sensors. However, despite a decreasing trend on the price of these 3D sensors in recent years, they remain prohibitively expensive. It is also inconceivable that each of the autonomous cars in the near future will have one of such devices spinning at 10Hz on its roof. The 3D laser scanner is fantastic to equip a dedicated *survey* vehicle, perhaps not the technology that common folks will find it friendly.

It is our desire to build light-weight, affordable and user-friendly self-driving technologies that motivates us to turn our attention away from the benefits of 3D laser scanners to its cheaper, smaller counterpart – the 2D range finders. However, since the 2D range finders scan the 3D world through only a single plane, significant

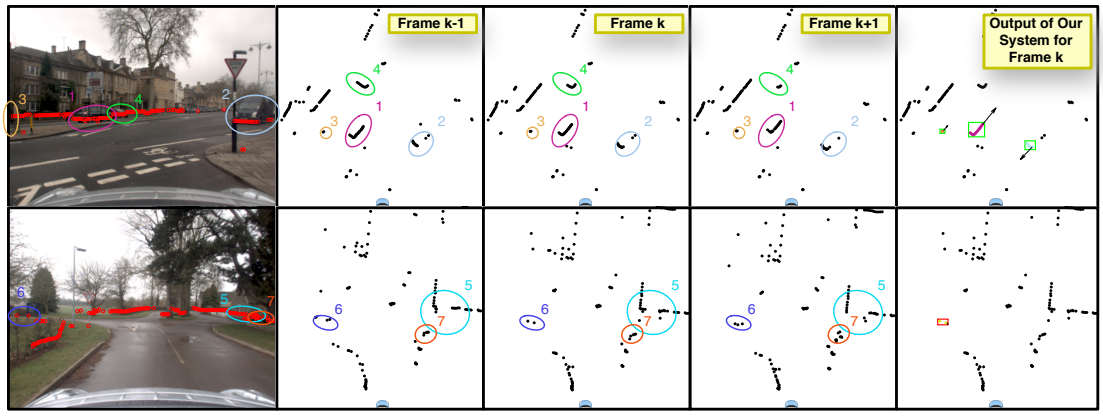


Figure 5.1: Examples of real laser scans in an outdoor driving scenario. Each row gives an example at a single instant (Frame k). The first column shows the scan data at Frame k reprojected into an image taken by an onboard camera. (The image is to provide some context for visualisation *only*. Our system does not require vision data to operate.) The next three columns show a sequence of consecutive raw laser scans at Frame $k-1$ (left), Frame k (middle) and Frame $k+1$ (right) respectively. The last column shows the output from our proposed system at the instant at Frame k (the middle scan). Here different objects detected are denoted by different colours and highlighted with boxes. A green box is a true detection whilst a red box is a false alarm. Arrows denote velocity estimates. See text for details.

challenges arise to the task of object detection.

5.2 The Challenge and Our Approach

There are several complicating factors to the successful detection and tracking of moving objects with a 2D laser scanner mounted on a *moving* platform (such as the SICK LDMRS mounted on our research vehicle – a modified Nissan LEAF, Figure 1.1(b)). First of all, because the sensor itself is moving, we have uncertainties to its motion, rendering simple frame differencing techniques ineffective. In addition, occlusion and viewpoint changes give the appearance of dynamic behaviours even in a purely static scene. This confusion makes the reliable detection of the *true* dynamic objects difficult without giving a high false alarm rate.

These difficulties are illustrated with real-world data in Figure 5.1. What is

being shown here are samples of data gathered with our research platform (cf. Figure 1.1(b)) as it was driven on the streets of Oxford. The first row gives a scenario of busy traffic, whilst the second row shows an example of laser data in an area where everything is stationary. In the first scenario (Row 1), it can be noted from the sequence of consecutive laser scans that, due to occlusion, it is ambiguous simply judging by appearance to tell whether an object is moving or not. Here, cars numbered 1 and 2 are traveling in opposite directions, whilst Car Number 4 is stationary. Note how Car Number 4 changes its appearance as Car Number 1 gradually occludes it. Number 3 is a pedestrian walking on the pavement. Even in a situation where there is *no* moving objects in the scene (Row 2), there is still much burden on a dynamic object detector. Because our vehicle is driving through the environment, static structures are being observed at different viewpoints. This gives rise to dynamic-like behaviours in the data. The corner of the building numbered 5 appears to be dynamic, bushes such as the ones circled as 6 and 7 appear to be moving too. Note how Bush Number 7 resembles more in its appearance to a group of walking pedestrians than the *true* walking pedestrian numbered 3 in the first row. These motivating examples illustrate the difficulties in motion detection from a moving 2D scanner. It is challenging even for a human to tell what is moving what is not from the raw data alone.

In this chapter, we focus on the motion cues, and take a model-free approach to the detection and tracking of moving objects, and present a principled framework that allows a flexible object representation capable of representing objects of any classes and shapes.

Many authors (for example, Miyasaka et al. (2009) and Wang et al. (2003)) observe that the problems of sensor pose estimation, map-building and detection and tracking of dynamic objects are closely related to each other. Removal of dynamic objects from the map-building process enhances the quality of the map, while

knowledge about the static structure of the environment helps significantly in the successful detection of dynamic objects. Both are in turn tightly coupled with sensor pose estimation because all observations are made relative to the sensor. To this end, our proposed system also estimates a joint state that includes the sensor pose, a local static background that maps the static structure around the sensor and the dynamic states of the tracked moving objects. However, we emphasise it is not our interest to map the static part of the environment, only *local* static information is kept and estimated as a part of our state for the purpose of dynamic object detection.

The last column of Figure 5.1 shows the output of our proposed system to the central frame (the middle column). Despite of the aforementioned difficulties, our system successfully detects the two moving vehicles and the walking pedestrian and rejects ambiguous static structures as moving entities (with one false alarm for Bush Number 6 in the static scene example, the second row).

We structure this chapter as follows. After stating our main contributions in this chapter in Section 5.3, Section 5.4 presents the core concept that is our model-free representation of objects. Following which we derive the prediction and observation models for the Bayes filter formulation in Section 5.5. In Section 5.6, we discuss the challenge posed to data association by our object representation and our solution to it. Finally, in Section 5.7, we evaluate the performance of the proposed system with real-world data, and show that it outperforms both an industry standard solution that was designed for the same problem domain of object tracking from a moving sensor and a classical approach to model-free tracking based on independent tracking of scan segments. Finally, we conclude the chapter in Section 5.8.

5.3 Contributions

Our main contribution in this chapter is the formulation of a unified framework that jointly estimates the pose of the sensor, a continuously updated local static background, and the motion states of dynamic objects, with the focus on reliable detection of moving objects. All three aspects are tightly coupled through a novel joint state representation that allows for objects of arbitrary shapes and sizes to be modelled and tracked. Our state representation is presented in Section 5.4.

In addition, we propose a hierarchical data association algorithm to assign raw laser measurements to potential state updates, and present a variant of the Joint Compatibility Branch and Bound (JCBB) algorithm (Neira and Tardos, 2001) that is suitable for associating a large number of measurements, and derive an alternative set of recursive update rules based on the triangular form representation of positive definite matrices for its efficient and numerically stable computation. The details of our approach to data association can be found in Section 5.6.

5.4 An Unusual State Representation

The system we propose is run within a Bayes filter implemented as a simple Extended Kalman Filter (for a comprehensive treatment of the Bayes filter and the Kalman filter, the reader is referred to the standard textbooks, for example, (Thrun et al., 2005) and (Bar-Shalom et al., 2002)). In this section, we describe in detail the representation of the system state. In particular, we motivate and describe how dynamic objects are represented to allow objects of any class and any shape to be modelled and tracked.

The motions of dynamic objects can be arbitrary and independent of each other. The sensor, however, does not observe their motions directly but ranges and bearings of points on the surface of the objects. Thus once conditioned on the measurements,

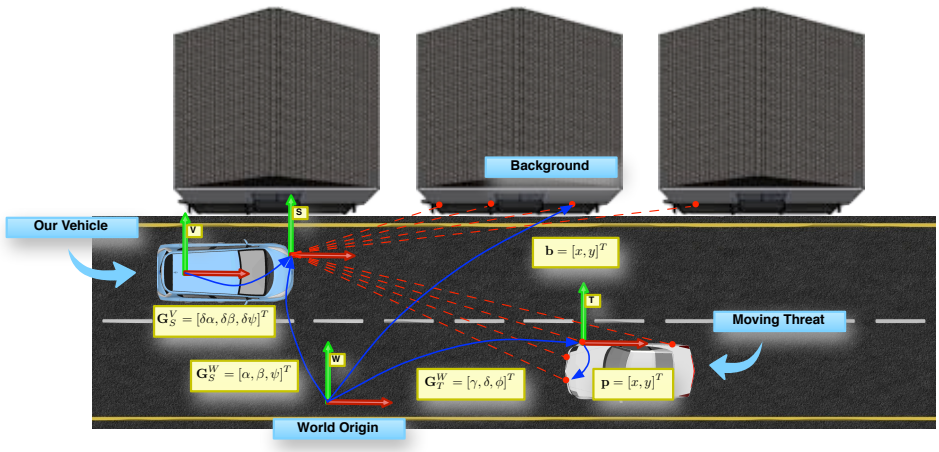


Figure 5.2: Illustration of frame conventions and variable definitions. Here $\mathbf{G}_A^B$ denotes an SE2 transform *from* frame A *to* frame B as an (x, y, θ)-triple. Coordinate frames W, V, S and T denote *world*, *vehicle*, *sensor* and *track* respectively. The sensor and objects' (tracks') motions are referenced to the fixed world origin W, and are denoted by the SE2 transforms $\mathbf{G}_S^W$ and $\mathbf{G}_T^W$. The sensor's extrinsic calibration parameters are denoted by the SE2 transform $\mathbf{G}_S^V$. Boundary points are referenced locally to the track's frame, and are denoted in the illustration by $\mathbf{p}$. Boundary points belonging to the *static* background are referenced globally to the world's frame, and are denoted in the illustration by $\mathbf{b}$. Subscripts are dropped to avoid clutter.

motions between different objects become correlated, due to the fact that these observations are taken from a moving sensor.

In order to correctly account for this correlation, the states of the objects and that of the sensor have to be estimated in a single joint distribution. A local static background is also simultaneously estimated as part of the joint state which is essential to distinguishing measurements belonging to dynamic objects from those on static objects. The state therefore consists of three parts: the sensor pose, the dynamic objects, and the static map. Figure 5.2 illustrates the relationships between the different parts of the joint state, and our object representation, and will be referred to extensively in what follows.

5.4.1 Sensor Pose Representation and Related Matters

The sensor pose (as part of the state vector $\mathbf{x}$) is represented by an SE2 transform $\mathbf{x}_S = \mathbf{G}_S^W = [\alpha, \beta, \psi]^T$ from the sensor's frame of reference to a stationary world frame of reference as depicted in Figure 5.2, which is updated by vehicle odometry measurements at the prediction stage, and as part of the measurement update stage with each new laser scan observation.

Since the holonomic constraints apply to the vehicle but not to the sensor directly, and odometry measurements are naturally referenced to the vehicle's frame of reference, the transform between the sensor and vehicle's frames of reference are required. To account for uncertainties in this estimated transform, we include it as part of the state as $\mathbf{x}_C = \mathbf{G}_S^V = [\delta\alpha, \delta\beta, \delta\psi]^T$. This is the SE2 transform that transforms points from the sensor frame into the vehicle frame, also illustrated in Figure 5.2.

5.4.2 Model-Free Object Representation

For convenience of description, in what follows, we will also refer to dynamic objects as "tracks", since their motion state is continuously being tracked. Each dynamic object i has its own set of axes T_i, and its motion state includes the SE2 transform $\mathbf{G}_{T_i}^W$ from the track's frame T_i to the world frame W, and its derivative, i.e. $\mathbf{x}_T^i = [\mathbf{G}_{T_i}^{W^T}, \dot{\mathbf{G}}_{T_i}^{W^T}]^T = [\gamma_i, \delta_i, \phi_i, \dot{\gamma}_i, \dot{\delta}_i, \dot{\phi}_i]^T$. This is illustrated by Figure 5.2 (the subscript i is dropped to avoid clutter). What is unusual about our representation is however, that *none* of these states are directly observed according to the observation model. Instead, each object has *additional* state parameters attached, named the "boundary point" coordinates, that are 2D cartesian coordinates represented *locally* to the object's frame of reference. It is these boundary points that are directly observed according to our observation model.

To understand the intuition behind boundary points, consider the case of a moving object being illuminated by the laser for the first time, for example, in the case illustrated in Figure 5.2. The set of raw range and bearing measurements Z is used to initialise a new track with its 6-vector states *plus* boundary points at the locations of the raw measurements in Z but transformed into the object's frame of reference (hence the name "boundary points" because the laser impinges on the boundary of the object). All subsequent measurements (laser illuminations) will be taken to be noisy observations of these boundary points on the object.

This model-free representation raises an interesting and central data association question. We must decide whether or not to extend the object's boundary by initialising additional boundary points with new raw laser measurements or simply associate the laser returns to the existing boundary points as it stands. Furthermore, which of the laser returns belong to the static background and hence have nothing to do with dynamic objects whatsoever? Our approach to data association lies at the heart of this work and is detailed in Section 5.6.

We make the assumption that dynamic objects observed in the 2D scanning plane of the sensor behave as rigid bodies. This assumption, although it does not hold strictly true due to deformable bodies such as a walking pedestrian, is a close approximation when observations are constrained to the 2D plane. Under this assumption, boundary points stay fixed relative to the object's frame of reference and hence their states have a trivial motion model.

With the introduction of boundary points, each object is thus parameterised with a partial outline of its perimeter allowing objects of arbitrary shapes and dimensions to be modelled under the same representation.

5.4.3 Static Background Representation

The representation for the static part of the state is simply a collection of boundary points as in the case of a dynamic object, except boundary points on the static background are represented with their global 2D cartesian coordinates in the *world*'s reference frame. See Figure 5.2 for an illustration.

5.4.4 The Complete State Structure

The complete state $\mathbf{x}$ consists of all parts described above, and is arranged as follows:

$$\mathbf{x} = [\mathbf{x}_S^T, \mathbf{x}_T^T, \mathbf{x}_b^T, \mathbf{x}_p^T, \mathbf{x}_C^T]^T \ , \tag{5.1}$$

where $\mathbf{x}_S$ is the sensor pose, $\mathbf{x}_T$ the collection of all 6-vector motion states of dynamic objects,

$$\mathbf{x}_T = [{\mathbf{x}_T^1}^T, {\mathbf{x}_T^2}^T, \ldots, {\mathbf{x}_T^{N_T}}^T]^T \ , \tag{5.2}$$

and $\mathbf{x}_b$ the collection of 2D coordinates of all boundary points belonging to the static part of the state,

$$\mathbf{x}_b = [\mathbf{b}_1^T, \mathbf{b}_2^T, \ldots, \mathbf{b}_{N_b}^T]^T \ , \tag{5.3}$$

and $\mathbf{x}_p$ the collection of 2D coordinates of all boundary points on all dynamic objects,

$$\mathbf{x}_p = [{\mathbf{p}_1^1}^T, {\mathbf{p}_2^1}^T, \ldots, {\mathbf{p}_{N_p^1}^1}^T, {\mathbf{p}_1^2}^T, {\mathbf{p}_2^2}^T, \ldots, {\mathbf{p}_{N_p^2}^2}^T, \ldots\ldots, {\mathbf{p}_1^{N_T}}^T, {\mathbf{p}_2^{N_T}}^T, \ldots, {\mathbf{p}_{N_p^{N_T}}^{N_T}}^T]^T \ . \tag{5.4}$$

Finally, $\mathbf{x}_C$ is the vector of the extrinsic calibration parameters of the sensor as described in Section 5.4.1.

```
1 Function ProcessMeasurement(x̂,P,Z)
      Input: The current estimate of the mean x̂ and covariance matrix P of
             the joint system state, and the measurement Z which may have
             type odometry or laser.
      Output: The updated mean x̂ and covariance matrix P of the joint
              system state.
2     if IsType(Z,odometry) then
3         (x̂, P) ← ProcessOdometryMeasurement(x̂,P,Z);
4     else
5         // ProcessLaserMeasurement is defined in Algorithm 5.2.
6         (x̂, P) ← ProcessLaserMeasurement(x̂,P,Z);
7     end
8     return (x̂, P);
9 end
```

Algorithm 5.1: Top-level algorithm executed upon each new measurement.

5.5 Detection and Tracking of Dynamic Objects

The mean and covariance matrix of the joint state vector are updated at each iteration according to the standard Bayes filter. The input to our system is a set of odometry measurements and a sequence of range and bearing laser scans. The inclusion of the odometry measurements is necessary because they provide the only *absolute* motion estimates. Without it, it is ambiguous to define what is *static* when all motion estimates are relative to the sensor. In this section, we derive the prediction and observation models for the joint state and present our approach to track initialisation which is key to the separation of dynamic objects from the static background.

Algorithm 5.1 lists the straightforward procedure carried out at each iteration when a new measurement is received. The mean $\hat{\mathbf{x}}$ and covariance matrix $\mathbf{P}$ of the joint state are updated differently according to the type of the measurement Z (odometry or laser). In general, odometry measurements arrive at a much higher frequency than laser measurements, they need to be processed very efficiently, and

```
1 Function ProcessLaserMeasurement(x̂,P,Z)
      Input: The current estimate of the mean x̂ and covariance matrix P of
             the joint system state, and the measurement Z which must be of
             type laser.
      Output: The updated mean x̂ and covariance matrix P of the joint
              system state.
2     (x̂, P) ← CleanUpStates(x̂,P);
3     (x̂, P) ← ForwardPredict(x̂,P);
4     // AssociateAndUpdate is defined in Algorithm 5.3.
5     (x̂, P) ← AssociateAndUpdate(x̂,P,Z);
6     (x̂, P) ← MergeTracks(x̂,P);
7     return (x̂, P);
8 end
```

Algorithm 5.2: Top-level algorithm executed upon receiving a new laser scan. See text for details.

therefore only forward-prediction of the sensor pose state taking the odometry measurement as a noisy control input is carried out in this case. In Section 5.5.1 we present the prediction model for this process. Algorithm 5.2 outlines the sequence of actions executed when a new laser scan is received. First, we do some house-keeping where out-of-date dynamic tracks and boundary points on the static background that have fallen out of the sensor's field of view are dropped. Next, the motion part of all dynamic tracks is forward-predicted according to an appropriate motion model as described in Section 5.5.2, and followed by data association and measurement updates. We derive observation models in Section 5.5.3 and defer the discussion of data association until Section 5.6. Finally, any tracks appear to be static are merged with the map, and adjacent tracks following the same rigid body motion are merged into a single track. The latter is to account for the situation that occasionally a large object is tracked as different "pieces", and this allows for the pieces to be put back into a single object. This merging procedure is described in Section 5.5.4.

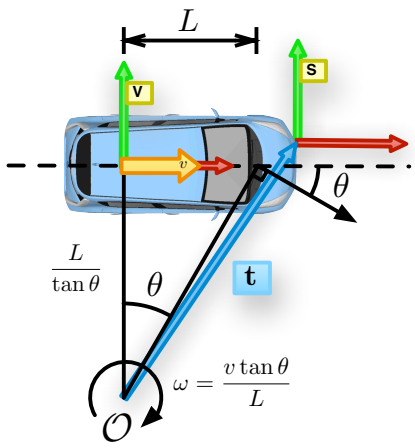


Figure 5.3: Illustration of the standard bicycle model. Here v denotes the vehicle's linear speed and θ the mean angle of the front wheels. The vehicle thus executes a coordinated turn about the point $\mathcal{O}$. Also shown are the vehicle's reference frame V against which the holonomic constraints are specified, and the sensor's reference frame S whose motion is to be predicted. See text for details.

5.5.1 Sensor Pose Prediction on Odometry Measurement

Each odometry measurement contains an estimate to the vehicle speed v, and the mean angle of the front wheels θ as illustrated in Figure 5.3. Here we assume the non-holonomic vehicle motion follows a bicycle model. What is nonstandard about our prediction model, however, is that the holonomic constraints of the bicycle model are specified with respect to the *vehicle*'s reference point, whereas it is the motion of the *sensor* that is to be predicted (and is part of the state). We derive the prediction model for the sensor pose in what follows.

Assuming both v and θ hold constant during the prediction period Δt, define the uncertain control input as

$$\mathbf{u}(v,\theta) = \begin{bmatrix} \Delta l \\ \Delta\psi \end{bmatrix} = \begin{bmatrix} v\Delta t \\ \omega\Delta t \end{bmatrix} = v\Delta t \begin{bmatrix} 1 \\ \frac{\tan\theta}{L} \end{bmatrix}, \tag{5.5}$$

where L is the distance between the front and rear wheel axles (see Figure 5.3), and

ω the angular speed of the vehicle $\omega = \frac{v \tan\theta}{L}$. In this model, we assume L is a fixed constant. After differentiating Equation (5.5), we obtain the Jacobian of the control input

$$\mathbf{U}(v,\theta) = \Delta t \begin{bmatrix} 1 & 0 \\ \frac{\tan\theta}{L} & \frac{v\sec^2\theta}{L} \end{bmatrix} . \tag{5.6}$$

Thus if the measured vehicle state $[\hat{v}, \hat{\theta}]^T$ has covariance matrix $\mathbf{V}$,

$$\mathbf{V} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} , \tag{5.7}$$

the control input has mean $\hat{\mathbf{u}} = \mathbf{u}(\hat{v}, \hat{\theta})$ and covariance matrix $\mathbf{Q} = \mathbf{U}\mathbf{V}\mathbf{U}^T$.

To arrive at a prediction model for the sensor pose $\mathbf{x}_S$, we note that, referring to Figure 5.3, the sensor's motion is a simple rotation about the same stationary point $\mathcal{O}$ as that of the vehicle. This is due to the fact that the sensor is attached rigidly to the vehicle.

The linear velocity $\mathbf{v}_S$ of the sensor frame origin is more easily obtained in the *vehicle* frame, and transformed into the global world frame later. To do this, note that the vector $\mathbf{t}$ from $\mathcal{O}$ to the origin of S in the *vehicle* frame V is given by

$$\mathbf{t} = \begin{bmatrix} 0 \\ \frac{L}{\tan\theta} \end{bmatrix} + \begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} . \tag{5.8}$$

Recall that $\mathbf{x}_C = \mathbf{G}_S^V = [\delta\alpha, \delta\beta, \delta\psi]^T$ is the SE2 transform from the sensor to the vehicle's frame. This gives the absolute velocity of the sensor frame origin (again, in the *vehicle* frame) as

$$\mathbf{v}_S = \omega\mathbf{R}(-\pi/2)\mathbf{t} = \omega \begin{bmatrix} \delta\beta \\ -\delta\alpha \end{bmatrix} + \begin{bmatrix} v \\ 0 \end{bmatrix} , \tag{5.9}$$

where $\mathbf{R}(\theta)$ is the 2D rotation matrix given an angle of rotation θ. Transforming $\mathbf{v}_S$ to the world frame and assuming small motion within the time step Δt, the new pose of the sensor $\mathbf{x}'_S$ is given by

$$\mathbf{x}'_S = \mathbf{x}_S + \Delta t \begin{bmatrix} \mathbf{R}(\psi')\mathbf{v}_S \\ -\omega \end{bmatrix}, \tag{5.10}$$

where $\mathbf{x}_S = [\alpha, \beta, \psi]^T$ is the pose of the sensor before the prediction and $\psi' = \psi - \delta\psi$ is the angle of rotation from the *vehicle* frame to the world frame. Substituting in Equation (5.9) and rearranging, we arrive at the discrete dynamic model for the sensor pose:

$$\mathbf{x}'_S = \mathbf{f}(\mathbf{x}_S, \mathbf{x}_C, \mathbf{u}) = \begin{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \mathbf{R}(\psi - \delta\psi) \left(\Delta\psi \begin{bmatrix} \delta\beta \\ -\delta\alpha \end{bmatrix} + \begin{bmatrix} \Delta l \\ 0 \end{bmatrix} \right) \\ \psi - \Delta\psi \end{bmatrix}. \tag{5.11}$$

We can now differentiate Equation (5.11) to obtain its Jacobian as $\mathbf{J} = [\mathbf{F}\ \mathbf{G}] = [\mathbf{F}_S\ \mathbf{F}_C\ \mathbf{G}]$, where

$$\mathbf{F}_S = \begin{bmatrix} \mathbf{I}_2 & \mathbf{R}(\psi - \delta\psi) \left(\Delta\psi \begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta l \end{bmatrix} \right) \\ \mathbf{0}_{1\times 2} & 1 \end{bmatrix} \tag{5.12}$$

is the Jacobian with respect to $\mathbf{x}_S$, and

$$\mathbf{F}_C = \begin{bmatrix} \Delta\psi \mathbf{R}(\psi - \delta\psi) \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & -\mathbf{R}(\psi - \delta\psi) \left(\Delta\psi \begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta l \end{bmatrix} \right) \\ \mathbf{0}_{1\times 2} & 0 \end{bmatrix} \tag{5.13}$$

is the Jacobian with respect to $\mathbf{x}_C$, and

$$\mathbf{G} = \begin{bmatrix} \mathbf{R}(\psi - \delta\psi) \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \mathbf{R}(\psi - \delta\psi) \begin{bmatrix} \delta\beta \\ -\delta\alpha \end{bmatrix} \\ 0 & -1 \end{bmatrix} \tag{5.14}$$

is the Jacobian with respect to $\mathbf{u}$ respectively, and $\mathbf{F}$ is defined by $\mathbf{F} = [\mathbf{F}_S \ \mathbf{F}_C]$.

Recall that $\hat{\mathbf{x}}$ and $\mathbf{P}$ denote the current mean and covariance matrix of the joint state respectively, it follows that the updates for the complete joint state is then given by

$$\hat{\mathbf{x}}' = [\mathbf{f}^T(\hat{\mathbf{x}}_S, \hat{\mathbf{x}}_C, \hat{\mathbf{u}}), \hat{\mathbf{x}}_r^T, \hat{\mathbf{x}}_C^T]^T \tag{5.15}$$

for the mean, and

$$\mathbf{P}' = \begin{bmatrix} \mathbf{F}\mathbf{P}_{s,c|s,c}\mathbf{F}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T & \mathbf{F}\mathbf{P}_{s,c|r,c} \\ \mathbf{P}_{r,c|s,c}\mathbf{F}^T & \mathbf{P}_{r,c|r,c} \end{bmatrix} \tag{5.16}$$

for the covariance matrix. Here a subscript of r denotes the remaining states other than $\mathbf{x}_S$ and $\mathbf{x}_C$, and the notation $\mathbf{P}_{s,c|r,c}$ denotes the sub-matrix of $\mathbf{P}$ that is formed by taking the rows belonging to the states $\mathbf{x}_S$ and $\mathbf{x}_C$ and columns belonging to the states $\mathbf{x}_r$ and $\mathbf{x}_C$ and so on. Note that the bottom right block of the covariance matrix is not touched and the top left block involves multiplications of matrices of fixed sizes (at most 6×6). This computation is therefore dominated by the off-diagonal updates, which is of $O(N)$, thus can be carried out very efficiently.

5.5.2 Dynamic Object Motion Prediction

At the prediction step after a new *laser* scan is received, all dynamic tracks are predicted forward according to a generic motion model before being updated with the measurements. A general motion model is desirable in this case because we do

not have information regarding to the object class (and hence its expected motion pattern). In an autonomous driving scenario, the sensor itself is constantly moving, this usually results in any particular object instance being observed for only a limited amount of time, rendering object-specific behaviour learning approaches also impractical. Therefore, in this work, we choose the classic constant velocity model. However, in our definition of the constant velocity model, in addition to the conventional linear velocity components, the angular velocity component is also modelled (which also follows a constant *angular* velocity motion decoupled to the linear components). The inclusion of the angular velocity component makes the model adequate at capturing the full range of 2D rigid body motion, which is the best we could hope for in a purely model-free approach.

Our constant velocity model is a linear model given by

$$\mathbf{x}'_T = \mathbf{F}\mathbf{x}_T + \mathbf{v} \ , \tag{5.17}$$

where $\mathbf{x}_T$ is the dynamic states of the object before the prediction, and $\mathbf{x}'_T$ the predicted states,

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \ , \tag{5.18}$$

and $\mathbf{v}$ is a zero mean additive noise with covariance matrix

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^3}{3}\mathbf{V} & \frac{\Delta t^2}{2}\mathbf{V} \\ \frac{\Delta t^2}{2}\mathbf{V} & \Delta t \mathbf{V} \end{bmatrix} \ . \tag{5.19}$$

Here $\mathbf{V}$ is the 3×3 covariance matrix for the zero-mean continuous linear and angular white noise accelerations (see (Bar-Shalom et al., 2002, Chapter 6) for more details).

To extend Equation (5.17) to multiple tracks, define the matrices

$$\tilde{\mathbf{F}} = \begin{bmatrix} \mathbf{F} & & & \\ & \mathbf{F} & & \\ & & \ddots & \\ & & & \mathbf{F} \end{bmatrix} \text{ and } \tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & & & \\ & \mathbf{Q} & & \\ & & \ddots & \\ & & & \mathbf{Q} \end{bmatrix}, \tag{5.20}$$

where the diagonals have an entry for each dynamic track. The update equations for the joint state mean and covariance matrix become:

$$\hat{\mathbf{x}}' = [\hat{\mathbf{x}}_S^T, (\tilde{\mathbf{F}}\hat{\mathbf{x}}_T)^T, \hat{\mathbf{x}}_r^T]^T , \tag{5.21}$$

and

$$\mathbf{P}' = \begin{bmatrix} \mathbf{P}_{s|s} & \mathbf{P}_{s|t}\tilde{\mathbf{F}}^T & \mathbf{P}_{s|r} \\ \tilde{\mathbf{F}}\mathbf{P}_{t|s} & \tilde{\mathbf{F}}\mathbf{P}_{t|t}\tilde{\mathbf{F}}^T + \tilde{\mathbf{Q}} & \tilde{\mathbf{F}}\mathbf{P}_{t|r} \\ \mathbf{P}_{r|s} & \mathbf{P}_{r|t}\tilde{\mathbf{F}}^T & \mathbf{P}_{r|r} \end{bmatrix} . \tag{5.22}$$

Here we follow the same subscripting convention as in Section 5.5.1, and $\mathbf{x}_T$ denotes the collection of dynamic states of *all* dynamic tracks as in Equation (5.2), and $\mathbf{x}_r$ the remaining states other than $\mathbf{x}_S$ and $\mathbf{x}_T$.

5.5.3 Observation Models for Raw Laser Measurements

In this section, we derive observation models for boundary points on the static background and dynamic objects respectively. All variables involved in what follows are defined in Section 5.4.

First we define the function and its Jacobian

$$\mathbf{u}(x, y) = \begin{bmatrix} r \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \tan^{-1} \frac{y}{x} \end{bmatrix} , \quad \mathbf{U}(x, y) = \frac{1}{r^2} \begin{bmatrix} rx & ry \\ -y & x \end{bmatrix} , \tag{5.23}$$

which converts a 2D point from cartesian coordinates into polar coordinates. This function (and its Jacobian) will be used extensively in what follows.

5.5.3.1 Boundary Points of Static Background

Each boundary point j on the static background may potentially generate a laser measurement $\mathbf{z} = [r, \theta]^T$, and hence its measurement model is the boundary point's location in polar coordinates in the *sensor*'s frame of reference:

$$\mathbf{h}_j(\mathbf{x}) = \mathbf{u}(\mathbf{g}(\mathbf{x}_S, \mathbf{b}_j)) \ , \ \ \mathbf{g}(\mathbf{x}_S, \mathbf{b}_j) = \mathbf{R}^T(\psi) \left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \right) \ . \tag{5.24}$$

Here $\mathbf{x}_S$ has been defined in Section 5.4.1, and $\mathbf{b}_j = [x_j, y_j]^T$ is the 2D cartesian coordinates of boundary point j in the world frame as described in Section 5.4.3 and illustrated by Figure 5.2.

The Jacobians of $\mathbf{g}$ is given by:

$$\begin{aligned} \mathbf{G}_S(\mathbf{x}_S, \mathbf{b}_j) &= \frac{\partial \mathbf{g}}{\partial \mathbf{x}_S} = \begin{bmatrix} -\mathbf{R}^T(\psi) & -\mathbf{R}\left(\frac{\pi}{2}\right) \mathbf{g}(\mathbf{x}_S, \mathbf{b}_j) \end{bmatrix} \ , \\ \mathbf{G}_b(\mathbf{x}_S, \mathbf{b}_j) &= \frac{\partial \mathbf{g}}{\partial \mathbf{b}_j} = \mathbf{R}^T(\psi) \ . \end{aligned} \tag{5.25}$$

This leads to the overall Jacobians for the measurement model $\mathbf{h}_j$ as:

$$\begin{aligned} \mathbf{H}_S^j(\mathbf{x}) &= \frac{\partial \mathbf{h}_j}{\partial \mathbf{x}_S} = \mathbf{U}(\mathbf{g}(\mathbf{x}_S, \mathbf{b}_j)) \mathbf{G}_S(\mathbf{x}_S, \mathbf{b}_j) \ , \\ \mathbf{H}_b^j(\mathbf{x}) &= \frac{\partial \mathbf{h}_j}{\partial \mathbf{b}_j} = \mathbf{U}(\mathbf{g}(\mathbf{x}_S, \mathbf{b}_j)) \mathbf{G}_b(\mathbf{x}_S, \mathbf{b}_j) \ . \end{aligned} \tag{5.26}$$

5.5.3.2 Boundary Points of Dynamic Objects

Each boundary point j on any dynamic track i may also give rise to a laser measurement, and the measurement model in this case is the 2D polar coordinates of

the boundary point in the *sensor*'s frame, and is given by:

$$\mathbf{h}_j^i(\mathbf{x}) = \mathbf{u}(\mathbf{g}(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i)) \ , \tag{5.27}$$

where

$$\mathbf{g}(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) = \mathbf{R}^T(\psi) \left(\mathbf{R}(\phi_i) \begin{bmatrix} x_j^i \\ y_j^i \end{bmatrix} + \begin{bmatrix} \gamma_i \\ \delta_i \end{bmatrix} - \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \right) \ . \tag{5.28}$$

Here $\mathbf{x}_T^i$ is the dynamic states of track i, and $\mathbf{p}_j^i$ the boundary point's cartesian coordinates in the *object*'s frame as discussed in Section 5.4.2 and illustrated by Figure 5.2.

To obtain the Jacobians to the measurement model, we again obtain first the Jacobians of $\mathbf{g}$:

$$\begin{aligned}
\mathbf{G}_S(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) &= \frac{\partial \mathbf{g}}{\partial \mathbf{x}_S} = \begin{bmatrix} -\mathbf{R}^T(\psi) & -\mathbf{R}\left(\frac{\pi}{2}\right) \mathbf{g}(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) \end{bmatrix} , \\
\mathbf{G}_T(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) &= \frac{\partial \mathbf{g}}{\partial \mathbf{x}_T^i} = \begin{bmatrix} \mathbf{R}^T(\psi) & \mathbf{R}^T(\psi)\mathbf{R}(\phi) \begin{bmatrix} -y_j^i \\ x_j^i \end{bmatrix} & \mathbf{0}_{2\times 3} \end{bmatrix} , \\
\mathbf{G}_p(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) &= \frac{\partial \mathbf{g}}{\partial \mathbf{p}_j^i} = \mathbf{R}^T(\psi)\mathbf{R}(\phi) \ .
\end{aligned} \tag{5.29}$$

Now the Jacobians of $\mathbf{h}_j^i$ follow:

$$\begin{aligned}
\mathbf{H}_S^{ij} &= \frac{\partial \mathbf{h}_j^i}{\partial \mathbf{x}_S} = \mathbf{U}(\mathbf{g}(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i))\mathbf{G}_S(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) \ , \\
\mathbf{H}_T^{ij} &= \frac{\partial \mathbf{h}_j^i}{\partial \mathbf{x}_T^i} = \mathbf{U}(\mathbf{g}(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i))\mathbf{G}_T(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) \ , \\
\mathbf{H}_p^{ij} &= \frac{\partial \mathbf{h}_j^i}{\partial \mathbf{p}_j^i} = \mathbf{U}(\mathbf{g}(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i))\mathbf{G}_p(\mathbf{x}_S, \mathbf{x}_T^i, \mathbf{p}_j^i) \ .
\end{aligned} \tag{5.30}$$

5.5.4 Track Initialisation and Merging

The initialisation of new dynamic tracks is non-trivial because we have to ensure that only new *dynamic* objects are initialised into new tracks and static objects are merged with the static background. To this purpose, we apply the technique of constrained initialisation (Williams, 2001), where each new track's motion status is deferred until it has accumulated enough evidence to make the correct decision. Specifically, a new track is first marked as "tentative" when initialised, and becomes "mature" only if it is continuously being observed for more than a fixed number of frames (otherwise it is dropped). Then it is tested against the static background, and each existing dynamic track in turn for merging. The test and merging are all handled consistently within the same Bayesian filtering framework. If all merging tests fail, it is declared "established" and added to the set of existing dynamic tracks.

In the case of testing against merging with the static background, we are interested in the hypothesis that the track's absolute velocity (linear and angular) is zero given the estimated uncertainty in its motion. Following (Williams, 2001), we take uncertainty into account by introducing a fictitious *perfect* (noiseless) measurement on the track's absolute velocity, and test the validity of a measured value of zero with the standard χ^2 test. Specifically, given a tentative track T with its motion state vector $\mathbf{x}_T = [\gamma, \delta, \phi, \dot{\gamma}, \dot{\delta}, \dot{\phi}]^T$ (in accordance with the notation introduced in Section 5.4.2), we define a fictitious measurement model

$$\mathbf{h}_1(\mathbf{x}_T) = \begin{bmatrix} \dot{\gamma} \\ \dot{\delta} \\ \dot{\phi} \end{bmatrix} , \tag{5.31}$$

and consider a measured value of $\hat{\mathbf{z}} = \mathbf{0}$ under the noise-free condition ($\mathbf{R} = \mathbf{0}$). That is to say, given an observer that *perfectly* observes the internal dynamics of

the object, what is the chance for it to say they are zero?

The Jacobian to Equation (5.31) is trivial and is given by

$$\mathbf{H}_T^1 = \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}_T} = [\mathbf{0}_3 \ \mathbf{I}_3] \ . \tag{5.32}$$

Thus noting that $\mathbf{R} = \mathbf{0}$, the innovation covariance of the fictitious measurement is given by simply $\mathbf{S} = \mathbf{P}_{v_T v_T}$, where $\mathbf{P}_{v_T v_T}$ is the sub-matrix of the joint covariance matrix $\mathbf{P}$ corresponding to the velocity of track T.

We then carry out a validation test on the measurement $\hat{\mathbf{z}} = \mathbf{0}$, to see if it falls within the validation gate, that is, if

$$(\hat{\mathbf{z}} - \mathbf{h}_1(\hat{\mathbf{x}}_T))^T \mathbf{S}^{-1} (\hat{\mathbf{z}} - \mathbf{h}_1(\hat{\mathbf{x}}_T)) \leq \chi^2_{d,\alpha} \ . \tag{5.33}$$

Here $\hat{\mathbf{x}}_T$ is the current estimate (the mean) of $\mathbf{x}_T$, and $\chi^2_{d,\alpha}$ is the χ^2 validation gate threshold of degree of freedom d ($d = 3$ in this case) and confidence level α.

If Equation (5.33) holds, the hypothesis that this tentative track is stationary is accepted, and the merge proceeds with a formal update to the state estimate as if $\hat{\mathbf{z}}$ were a *real* measurement. This propagates the information gathered with the tentative track so far to the rest of the system and sets its absolute velocity actually to zero. The track can then be safely marginalised out after copying over its boundary points to the static background to complete the merge.

A similar procedure applies to merging tests with an existing dynamic track. In this case, the fictitious measurement applies to the relative motion of the tentative track to the existing track under consideration. Let us denote the dynamic states of the tentative and existing tracks by $\mathbf{x}_T = [\gamma_T, \delta_T, \phi_T, \dot{\gamma}_T, \dot{\delta}_T, \dot{\phi}_T]^T$ and $\mathbf{x}_E = [\gamma_E, \delta_E, \phi_E, \dot{\gamma}_E, \dot{\delta}_E, \dot{\phi}_E]^T$ respectively, then the fictitious measurement we are interested in is the relative motion (both linear and rotational) of the tentative track

with respect to the existing track:

$$\begin{aligned}\mathbf{h}_2(\mathbf{x}_T, \mathbf{x}_E) &= \dot{\mathbf{G}}_T^E \\ &= \begin{bmatrix} \mathbf{R}(-\phi_E)\left(\begin{bmatrix} \dot{\gamma}_T \\ \dot{\delta}_T \end{bmatrix} - \begin{bmatrix} \dot{\gamma}_E \\ \dot{\delta}_E \end{bmatrix}\right) - \dot{\phi}_E \mathbf{R}(\tfrac{\pi}{2} - \phi_E)\left(\begin{bmatrix} \gamma_T \\ \delta_T \end{bmatrix} - \begin{bmatrix} \gamma_E \\ \delta_E \end{bmatrix}\right) \\ \dot{\phi}_T - \dot{\phi}_E \end{bmatrix}.\end{aligned} \tag{5.34}$$

Differentiating Equation (5.34), we obtain its Jacobians

$$\mathbf{H}_T^2 = \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}_T} = \begin{bmatrix} -\dot{\phi}_E \mathbf{R}(\tfrac{\pi}{2} - \phi_E) & \mathbf{0}_{2\times 1} & \mathbf{R}(-\phi_E) & \mathbf{0}_{2\times 1} \\ \mathbf{0}_{1\times 2} & 0 & \mathbf{0}_{1\times 2} & 1 \end{bmatrix}, \tag{5.35}$$

and

$$\begin{aligned}\mathbf{H}_E^2 &= \frac{\partial \mathbf{h}_2}{\partial \mathbf{x}_E} \\ &= \begin{bmatrix} \dot{\phi}_E \mathbf{R}(\tfrac{\pi}{2} - \phi_E) & \mathbf{D}\mathbf{h}_2(\mathbf{x}_T, \mathbf{x}_E) & -\mathbf{R}(-\phi_E) & \mathbf{R}(\tfrac{\pi}{2} - \phi_E)\left(\begin{bmatrix} \gamma_T \\ \delta_T \end{bmatrix} - \begin{bmatrix} \gamma_E \\ \delta_E \end{bmatrix}\right) \\ \mathbf{0}_{1\times 2} & 0 & \mathbf{0}_{1\times 2} & -1 \end{bmatrix},\end{aligned} \tag{5.36}$$

where $\mathbf{D}$ is a selection matrix given by

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}. \tag{5.37}$$

The merging test then proceeds in a similar fashion to the merging test in the static case.

The same merging procedure is also conducted at the end of each processing

cycle (Algorithm 5.2) for testing each existing track against merging with the static background or other existing tracks.

The procedure described in this section is important because it is the *only* means through which the static information in the state is kept up-to-date. It also allows a dynamic object (or a false detection) to transition into the static state. An up-to-date static information is the *only* direct influential factor to successful dynamic object *detection* because successful data association requires an accurate estimate of static boundary points and the classification of each new laser scan into a static part and a dynamic part is handled implicitly by the data association process. Data association is the main subject of our discussion in the next section.

5.6 Hierarchical Data Association

Not all state variables in the joint state are directly observable, for the ones that are, namely boundary points on either the static background or any dynamic object, it is ambiguous which is being observed, which is not, and indeed, whether a new boundary point needs to be initialised. Thus when new laser measurements arrive, it has to be determined for each measurement:

1. If it is an observation on a static object, then
   (a) is it an observation of an existing boundary point?
   (b) is it an observation of a new boundary point?
2. If it is an observation on an existing dynamic object, then
   (a) is it an observation of an existing boundary point on the object?
   (b) is it an observation of a new boundary point on the object?
3. Is it an observation on a new dynamic object?

In addition, in case 2, it has also to be determined to which of the existing dynamic objects the measurement belongs to, and in case 3, how many new tracks need to be initialised.

This data association problem naturally breaks down into two levels. The first level operates at the coarse scale, in which measurements are first divided into clusters, and each cluster is assigned to either the static background, or a dynamic object, or used to initialise a new dynamic track. At the fine level, for each object (or the static background), measurements from the associated clusters are further associated with its existing boundary points or used to initialise new boundary points.

5.6.1 Coarse Level Data Association

The measurements in a given laser scan are first divided into a set of clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_{|\mathcal{C}|}\}$. The clusters are then assigned to the static background and dynamic objects recursively with the help of the ICP algorithm (Besl and McKay, 1992).

We use a simple variant of ICP with outlier rejection. Specifically, to align two point sets P and Q, at each iteration, we conduct nearest neighbour search between the two point sets. A point in P is associated to its nearest neighbour in Q if their distance is within a certain threshold, otherwise it is discarded as an outlier for this iteration and become unassociated to any point in Q. All *associations* obtained in this way are used to estimate a transform that aligns the point set P to Q. The points in P are then updated to their new positions with the estimated transform and the loop continues until convergence. The association upon convergence is taken as the final association, *with outlier rejection* from P to Q.

With the ICP procedure defined, the coarse level data association proceeds as follows. First, boundary points on the static background are aligned to the set of

```
1  Function AssociateAndUpdate(x̂,P,Z)
       Input: The current estimate of the mean x̂ and covariance matrix P of
              the joint system state (after prediction), and the laser scan Z.
       Output: The updated mean x̂ and covariance matrix P of the joint
               system state.
2      C ← ClusterMeasurements(Z);
3      (x̂, P, A) ← AssociateAndUpdateWithStatic(x̂, P, C);
       // A = {C ∈ C : C is associated}
4      C ← C \ A;
5      for i ← 1 to N_T do
6          (x̂, P, A) ← AssociatedAndUpdateWithDynamic(x̂, P, C, i);
7          C ← C \ A;
8      end
9      for C ∈ C do
10         (x̂, P) ← InitialiseNewTrack(x̂, P, C);
11     end
12     return (x̂, P);
13 end
```

Algorithm 5.3: The coarse level data association algorithm. The raw laser measurements are first clustered. Then clusters are recursively assigned to the static background and each dynamic object in turn, with the assigned clusters removed from the list at each stage. The unassociated clusters initialise new tentative tracks. See text for details and relations to fine level data association.

measurements Z with ICP, and clusters in $\mathcal{C}$ which contains measurements matched to any boundary points on the static background in this way are associated to the static background, and used to update or initialise new boundary points at the fine level for the static backgournd. Then the associated clusters are removed from $\mathcal{C}$ and a similar procedure follows recursively for each dynamic track. The clusters that remain in $\mathcal{C}$ at the end of this process are thus not associated with any existing track (or the static background), and each cluster will initialise a new tentative dynamic track. This procedure is captured in Algorithm 5.3.

Coarse level data association makes intuitive sense because first the majority of the measurements belonging to the static part of the environment will be associated

with the static boundary points, leaving the outliers being mostly measurements on dynamic objects. The association of clusters instead of raw measurements at this level helps in extending object boundaries because according to Algorithm 5.3, all measurements in a given cluster will be associated to an object (or the static background) if *any* measurement in it is associated with a boundary point on the object with ICP. Provided the initial clustering is correct and ICP with outlier rejection gives a reasonable performance, the remaining measurements in the cluster can safely be assumed to be previously unobserved boundary points on the same object and be used to extend the object boundary.

ICP is known to perform poorly when the initial misalignment between the two point sets is large. However, if the two point sets start close-to-aligned, ICP is ideal. This is precisely the case here because data association is conducted *after* model prediction. The boundary points will be in their *predicted* locations instead of their previous locations in the last frame. Given a good motion estimate, the predicted locations of the boundary points will be already close to their actually observed values.

What remains is to ensure that a good set of clusters is produced in the clustering step. The details of our approach to clustering are deferred until Section 5.6.5 where we introduce the proposed EMST-EGBIS clustering algorithm that is designed to produce perceptually coherent clusters.

However, segmentation failure is inevitable in any unsupervised clustering procedure. In such an event, other components of the system such as the merging procedure introduced in Section 5.5.4 take over to resolve the issue if it is possible.

5.6.2 Fine Level Data Association

Given a set of clusters associated with a certain track (or the static background), the fine level data association must find a matching satisfying certain desirable

criteria that assigns measurements contained in the clusters to boundary points on the track. Correct assignment is critical to successful tracking, and, the stability of the system as a whole, due to the fact that correlation is introduced between all pairs of variables in the joint state. In particular, all state variables we would like to infer: the sensor pose, the dynamic states of the tracked objects, are not directly observed.

Joint Compatibility Branch and Bound (JCBB) (Neira and Tardos, 2001) is a well-known data association algorithm that takes into account the correlations between observations. Explained in our nomenclature, an association between the set of measurements and the set of boundary points is called a *feasible* association if:

1. Each measurement is associated to at most one boundary point, and no two measurements are associated to the same boundary point (one-one association).

2. Each matching of a measurement to a boundary point is individually compatible as described below (individual compatibility).

3. The overall data association is jointly compatible as described below (joint compatibility).

To clarify the concepts of individual and joint compatibilities, consider a boundary point whose observation model has the standard form $\mathbf{z}_j = \mathbf{h}_j(\mathbf{x}) + \mathbf{w}_j$. Here $\mathbf{x}$ is the joint state defined in Section 5.4, and $\mathbf{w}_j$ is the additive zero-mean measurement noise. Thus its innovation covariance matrix is $\mathbf{S}_j = \mathbf{H}_j\mathbf{P}\mathbf{H}_j^T + \mathbf{R}$. Here $\mathbf{H}_j$ is the Jacobian of the function $\mathbf{h}_j$ evaluated at the current state mean, and $\mathbf{R}$ is the measurement noise covariance matrix (we assume all measurements have the same noise covariance matrix).

5.6.2.1 Individual Compatibility

Individual compatibility requires the assigned measurement $\hat{\mathbf{z}}_i$ must fall within a certain confidence region of boundary point j's validation gate, i.e. an assignment of $\hat{\mathbf{z}}_i$ to $\mathbf{z}_j$ is individually compatible if:

$$(\hat{\mathbf{z}}_i - \mathbf{h}_j(\hat{\mathbf{x}}))^T \mathbf{S}_j^{-1} (\hat{\mathbf{z}}_i - \mathbf{h}_j(\hat{\mathbf{x}})) \leq \chi^2_{d,\alpha} \ , \tag{5.38}$$

where $\chi^2_{d,\alpha}$ is the χ^2 validation gate threshold of degree of freedom d and confidence level α. Here, d is the measurement dimension, hence $d = 2$, because each measurement contains a range and a bearing $\hat{\mathbf{z}} = [\hat{r}, \hat{\theta}]^T$.

5.6.2.2 Joint Compatibility

Under the assumption of independent observations, the joint observation model of a complete association σ is given by

$$\mathbf{h}_\sigma(\mathbf{x}) = [\mathbf{h}^T_{\sigma(1)}(\mathbf{x}), \mathbf{h}^T_{\sigma(2)}(\mathbf{x}), \ldots]^T \ , \tag{5.39}$$

with the innovation covariance

$$\mathbf{S}_\sigma = \begin{bmatrix} \mathbf{H}_{\sigma(1)}\mathbf{P}\mathbf{H}^T_{\sigma(1)} + \mathbf{R} & \mathbf{H}_{\sigma(1)}\mathbf{P}\mathbf{H}^T_{\sigma(2)} & \ldots \\ \mathbf{H}_{\sigma(2)}\mathbf{P}\mathbf{H}^T_{\sigma(1)} & \mathbf{H}_{\sigma(2)}\mathbf{P}\mathbf{H}^T_{\sigma(2)} + \mathbf{R} & \\ \vdots & & \ddots \end{bmatrix} \ , \tag{5.40}$$

where $\sigma(1)$ denotes the index of the boundary point associated to the first *assigned* measurement and so on. Thus the joint measurement has dimension $N_a d$ if the number of assigned measurements is N_a. An association σ is jointly compatible if

$$(\hat{\mathbf{z}}_\sigma - \mathbf{h}_\sigma(\hat{\mathbf{x}}))^T \mathbf{S}_\sigma^{-1} (\hat{\mathbf{z}}_\sigma - \mathbf{h}_\sigma(\hat{\mathbf{x}})) \leq \chi^2_{N_a d,\alpha} \ . \tag{5.41}$$

Here $\hat{\mathbf{z}}_\sigma$ is the collection of the measurements that are *assigned* to some boundary point according to association σ.

The JCBB algorithm then finds the *feasible* association that has the largest number of assigned measurements N_a^*. Since there are in general many feasible associations with $N_a = N_a^*$, the algorithm finds the association σ^* that gives the lowest joint Normalised Innovation Squared (jNIS, defined to be the expression to the left of the inequality in Equation (5.41)).

5.6.3 The JCBB-Refine Algorithm

Unfortunately, the JCBB algorithm is an exponential algorithm in the number of measurements to be assigned. This means it is not directly applicable to our application domain, since in our case observations are raw laser measurements and number in the 100's.

We introduce the JCBB-Refine algorithm, which instead of aiming to find the optimum assignment σ^*, we only find a *good* association $\tilde{\sigma}$ that is *feasible*. Of course, there are many *feasible* associations, a *good* association must be measured relative to some gauge. The JCBB-Refine algorithm we propose here takes an initial association σ_0 as a starting point, and finds a feasible association that has as many assigned measurements and as low a jNIS as possible in a greedy manner while respecting the initial association σ_0. The initial association σ_0 can be arbitrary, i.e. it does *not* have to be feasible. In fact, none of the feasibility conditions has to be satisfied.

Given σ_0, the algorithm first removes assignments that do not comply with individual compatibility (i.e. noncompliant measurements become unassociated with any boundary point), and then removes duplicate assignments with a single pass through the measurements. After these, the resulting association satisfies feasibility conditions 1 and 2. The algorithm then proceeds to iteratively removing the assignment that leads to the most jNIS reduction until condition 3 is satisfied. Starting

from this minimal set of assignments that is now feasible, the unassociated measurements are then tried in turn, and assigned to the boundary point (among the boundary points that are individually compatible and yet unassigned) that gives the lowest jNIS if the assignment does not violate joint compatibility. The resulting association is thus guaranteed to remain feasible.

The JCBB-Refine algorithm can be initialised with any sensible starting assignment σ_0. In our particular application, the assignment as a result of the ICP matching at the coarse level association is a natural starting point. The association after the refinement is then used to update the joint state with the associated measurements, and all unassociated measurements initialise new boundary points to extend the object boundary.

5.6.4 Recursive Updates in Triangular Form

It is shown (Neira and Tardos, 2001) that the innovation covariance matrix $\mathbf{S}$, its inverse, and the jNIS can be computed recursively as hypotheses are being tested. However in its direct form, the recursion suffers from numerical stability issues when the number of measurements becomes large because both $\mathbf{S}$ and $\mathbf{S}^{-1}$ have to be maintained to be symmetric and positive definite. We show the same computation can be achieved in the triangular form, which is a numerically stable representation for positive definite matrices.

To begin with, at step k, assume a decomposition for $\mathbf{S}_k$ is given such that $\mathbf{S}_k = \mathbf{U}_k^T \mathbf{U}_k$ for some upper triangular matrix $\mathbf{U}_k$, for example through Cholesky decomposition. Then its inverse is given by $\mathbf{S}_k^{-1} = \mathbf{U}_k^{-1}(\mathbf{U}_k^{-1})^T$. If we define a new matrix $\mathbf{G}_k = \mathbf{U}_k^{-1}$ so that $\mathbf{S}_k^{-1} = \mathbf{G}_k \mathbf{G}_k^T$, we obtain a decomposition also for $\mathbf{S}_k^{-1}$. In particular, $\mathbf{G}_k$ is also upper triangular.

Now the next iteration selects a new boundary point to be assigned to a measurement expanding the innovation covariance matrix (with reference to Equation (5.40))

to

$$\mathbf{S}_{k+1} = \begin{bmatrix} \mathbf{S}_k & \mathbf{W}_k^T \\ \mathbf{W}_k & \mathbf{N}_k \end{bmatrix} . \tag{5.42}$$

If we now define a new upper triangular matrix

$$\mathbf{U}_{k+1} = \begin{bmatrix} \mathbf{U}_k & \mathbf{R}_k^T \\ \mathbf{0} & \mathbf{F}_k \end{bmatrix} , \tag{5.43}$$

where $\mathbf{R}_k = \mathbf{W}_k\mathbf{G}_k$ and $\mathbf{F}_k = \text{chol}(\mathbf{N}_k - \mathbf{R}_k\mathbf{R}_k^T)$. It is then straightforward to verify that $\mathbf{S}_{k+1} = \mathbf{U}_{k+1}^T\mathbf{U}_{k+1}$ by direct evaluation.

Equation (5.43) establishes a recursion on the upper triangular matrix $\mathbf{U}_k$. A similar recursion on $\mathbf{G}_k$ can be obtained by explicitly inverting Equation (5.43). Note that Equation (5.43) expresses $\mathbf{U}_{k+1}$ in block form, hence we can apply the standard formula for matrix inversion in block form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix} . \tag{5.44}$$

This leads to

$$\mathbf{G}_{k+1} = \mathbf{U}_{k+1}^{-1} = \begin{bmatrix} \mathbf{U}_k & \mathbf{R}_k^T \\ \mathbf{0} & \mathbf{F}_k \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{U}_k^{-1} & -\mathbf{U}_k^{-1}\mathbf{R}_k^T\mathbf{F}_k^{-1} \\ \mathbf{0} & \mathbf{F}_k^{-1} \end{bmatrix} . \tag{5.45}$$

Noting $\mathbf{G}_k = \mathbf{U}_k^{-1}$ and define a new matrix $\mathbf{M}_k = \mathbf{F}_k^{-1}$ for notational convenience, we obtain the result

$$\mathbf{G}_{k+1} = \begin{bmatrix} \mathbf{G}_k & -\mathbf{G}_k\mathbf{R}_k^T\mathbf{M}_k \\ \mathbf{0} & \mathbf{M}_k \end{bmatrix} . \tag{5.46}$$

This gives a recursion on the decomposition of the inverse of the innovation covariance matrix $\mathbf{S}_k^{-1}$.

At any stage of the recursion, if the innovation covariance matrix $\mathbf{S}_k$ or its inverse is needed, it can be obtained by a straightforward evaluation $\mathbf{S}_k = \mathbf{U}_k^T \mathbf{U}_k$ or $\mathbf{S}_k^{-1} = \mathbf{G}_k \mathbf{G}_k^T$. Thus by maintaining a different recursion on $\mathbf{U}_k$ and $\mathbf{G}_k$, we keep an *implicit* representation for the innovation covariance matrix and its inverse. This triangular form does not suffer from numerical stability issues as keeping an explicit recursion for $\mathbf{S}_k$ and $\mathbf{S}_k^{-1}$ does because $\mathbf{U}_k$ and $\mathbf{G}_k$ are maintained to be upper triangular by definition and this automatically guarantees $\mathbf{S}_k$ and $\mathbf{S}_k^{-1}$ to be symmetric and positive definite.

The innovation $\boldsymbol{\nu}_k$ also needs to be expanded with the newly assigned measurement:

$$\boldsymbol{\nu}_{k+1} = \begin{bmatrix} \boldsymbol{\nu}_k \\ \tilde{\boldsymbol{\nu}}_k \end{bmatrix} , \tilde{\boldsymbol{\nu}}_k = \hat{\mathbf{z}}_k - \mathbf{h}_k(\hat{\mathbf{x}}) , \tag{5.47}$$

where $\hat{\mathbf{z}}_k$ is the newly assigned measurement, and $\mathbf{h}_k(\hat{\mathbf{x}})$ the predicted measurement given by the measurement model of the associated boundary point (as presented in Section 5.5.3).

Given our representation of the innovation covariance matrix and its inverse in the triangular form, a simpler formula for the jNIS can be obtained if we keep track of an alternative vector $\boldsymbol{\xi}_k = \mathbf{G}_k^T \boldsymbol{\nu}_k$ instead of $\boldsymbol{\nu}_k$. The recursion for $\boldsymbol{\xi}_k$ can be established by substituting Equation (5.46) and Equation (5.47) into $\boldsymbol{\xi}_{k+1} = \mathbf{G}_{k+1}^T \boldsymbol{\nu}_{k+1}$, so we have

$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} \mathbf{G}_k & -\mathbf{G}_k \mathbf{R}_k^T \mathbf{M}_k \\ \mathbf{0} & \mathbf{M}_k \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\nu}_k \\ \tilde{\boldsymbol{\nu}}_k \end{bmatrix} = \begin{bmatrix} \mathbf{G}_k^T \boldsymbol{\nu}_k \\ \mathbf{M}_k^T (\tilde{\boldsymbol{\nu}}_k - \mathbf{R}_k \mathbf{G}_k^T \boldsymbol{\nu}_k) \end{bmatrix} . \tag{5.48}$$

Noting $\boldsymbol{\xi}_k = \mathbf{G}_k^T \boldsymbol{\nu}_k$ and define $\boldsymbol{\mu}_k = \mathbf{M}_k^T (\tilde{\boldsymbol{\nu}}_k - \mathbf{R}_k \mathbf{G}_k^T \boldsymbol{\nu}_k) = \mathbf{M}_k^T (\tilde{\boldsymbol{\nu}}_k - \mathbf{R}_k \boldsymbol{\xi}_k)$ we arrive at

$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} \boldsymbol{\xi}_k \\ \boldsymbol{\mu}_k \end{bmatrix} . \tag{5.49}$$

Now with $\boldsymbol{\xi}_k$ defined, the jNIS at each iteration k has a remarkably simple form

$$\text{jNIS}_k = \boldsymbol{\nu}_k^T \mathbf{S}_k^{-1} \boldsymbol{\nu}_k = \boldsymbol{\nu}_k^T \mathbf{G}_k \mathbf{G}_k^T \boldsymbol{\nu}_k = \boldsymbol{\xi}_k^T \boldsymbol{\xi}_k \ , \tag{5.50}$$

a recursion for the jNIS therefore follows naturally from the recursion for $\boldsymbol{\xi}_k$ (Equation (5.49)):

$$\text{jNIS}_{k+1} = \text{jNIS}_k + \boldsymbol{\mu}_k^T \boldsymbol{\mu}_k \ . \tag{5.51}$$

Finally, we note that these recursive update rules based on an *implicit* triangular form representation of the innovation covariance matrix and its inverse have the same computational complexity as the explicit recursions introduced in (Neira and Tardos, 2001), but are more numerically stable.

5.6.5 EMST-EGBIS Clustering

As previously discussed in Section 3.2.1, Euclidean Minimum Spanning Tree (EMST) based clustering algorithms have a long history, and is known for being capable of detecting clusters with irregular boundaries (Zahn, 1971). We propose a novel variant of EMST based clustering algorithms in this section, which is efficient, and particularly suitable for 2D range-bearing measurements where point densities vary significantly at different distances from the sensor and across different scene geometries.

EGBIS (Felzenszwalb and Huttenlocher, 2004) as described previously in Section 3.3.3.1 is a popular graph-based segmentation algorithm, that is effective at producing perceptually coherent segments over a wide range of variation in the dissimilarity measure across the global graph structure. However, the graph structures over unstructured application domains such as laser point clouds are usually not trivial to define.

Our EMST-EGBIS algorithm combines the strengths of both algorithms. Specif-

ically, we first compute the EMST over the collection of the points, and take the obtained EMST as the input graph structure to EGBIS to compute the clusters. Edge weights of the EMST (Euclidean distances between points) are taken directly as the dissimilarity measure. We apply EMST-EGBIS to each incoming laser scan to obtain measurement clusters to associate to object tracks (and the static background) at the coarse level of data association (cf. Section 5.6.1).

5.7 System Evaluation

In this section, we evaluate the proposed system, both quantitatively and qualitatively, and compare its performance against two benchmarking object tracking approaches, of which one is an industrial standard solution. We note there exists a large body of work on similar application domains (for example Miyasaka et al., 2009; Mertz et al., 2013; Wang et al., 2003), however it is often difficult to obtain a fair quantitative comparison to the methods due to either a lack of quantitative results or difficulty of a direct comparison using a common dataset. This motivates the comparison to a commercial product – the only one we are aware of which identifies and tracks dynamic obstacles. In addition, we also place the proposed system in comparison to a classical tracking solution where each object is tracked with independent Kalman filters with greedy data association (details to follow in the next section).

5.7.1 Experiment Setup

Our experiment platform is a modified Nissan Leaf that is equipped with a SICK LDMRS laser scanner, which is a scanner targeted at object tracking applications on automotive platforms. It scans the environment in four vertically separated scanning planes at 12.5Hz and produces native object tracking information at the same time.

Odometry information is provided internally as part of the vehicle state at 100Hz. Figure 1.1(b) shows our experiment platform together with the sensor setup.

The main benefit of the multi-layered architecture of the SICK LDMRS scanner is that it can compensate for the vehicle pitch appropriately. To take advantage of the multi-layered scanner, we follow a standard procedure to remove ground strike measurements given multiple scanning layers described in (Leonard et al., 2008). Specifically, a grid is first built over the 2D scanning plane, measurements from each scanning layer are projected down to the scanning plane and associated with the grid cell that the measurement falls into. Then each occupied grid cell is taken in turn. Where a cell contains measurements from more than one scanning layers the measurements contained in that cell will be retained, otherwise they are discarded. Finally, a single 2D polar scan free of ground strikes is generated by taking all measurements that are retained, collecting range measurements corresponding to each discrete scan angle from this retained set, and generate a new range value for the scan angle by averaging. The resulting 2D scan has a number of measurements in the same order as any single scan layer. The intuition behind this technique is that obstacles are assumed to extrude out from the ground thus measurements from multiple layers when projected down to the scanning plane tend to be close. On the other hand, a ground strike due to vehicle pitching does not exhibit this behaviour because it is equivalent to measuring a ramp. In all that follows, we conduct experiments using this synthesised scan from all scanning layers.

We note that this procedure of combining the multiple layers of the LDMRS scanner is solely for the purpose of removing false measurements due to ground strikes. Our proposed tracking framework is not dependent on a multi-layered laser scanner. The framework applies to any 2D laser scan. In fact, from our experience, in the case of few ground strikes, using only a single layer from the LDMRS scanner produces similar tracking performance.

Dataset	No. Laser Frames	Duration (min)	Drive Length (km)	No. Objects
Training	3508	4.68	1.04	7517
Test	2151	2.87	0.82	5928

Table 5.1: Details of the training and test datasets. Here each count of an "object" is a single observation of an object instance in a single laser scan.

In addition to the LDMRS's native tracking system, we compare the proposed system with a classic model-free tracking approach. For this second baseline, the laser scan is first clustered with the EMST-EGBIS algorithm described in Section 5.6.5. Then each cluster centroid is tracked with an independent Kalman filter under the constant velocity model. Data association in this case is done by greedily assigning each observed cluster centroid to the first object track for which the observation falls within its validation gate. A new object track is initialised if a cluster cannot be associated with any existing object track in this way.

To evaluate the proposed system against the baselines, both quantitatively and qualitatively, we collected data of busy traffic at the centre of Oxford containing a variety of dynamic objects including pedestrians, cars, bicyclists, buses, trucks, motorcycles and so on, and extracted two busy sections of the log right at the centre of the city for evaluation. One dataset is used to find the best-performing parameter set, and is hence named the *training* set, and the other, the test set, is used to obtain unbiased test results running under the trained parameter set for fair comparison. Both datasets are hand-labelled to provide ground truth for quantitative evaluation. Figure 5.4 presents sample frames from the training and test datasets respectively to illustrate the complexity of the datasets, and Table 5.1 lists the relevant statistics of the datasets.

Note that, from Figure 5.4, a group of pedestrians is labelled in the datasets as a single dynamic object provided the group has a common heading. Taking a model-free approach, our goal here is to identify the dynamic hazard, to be able to describe and predict its motion and estimate its extend. All of these requirements

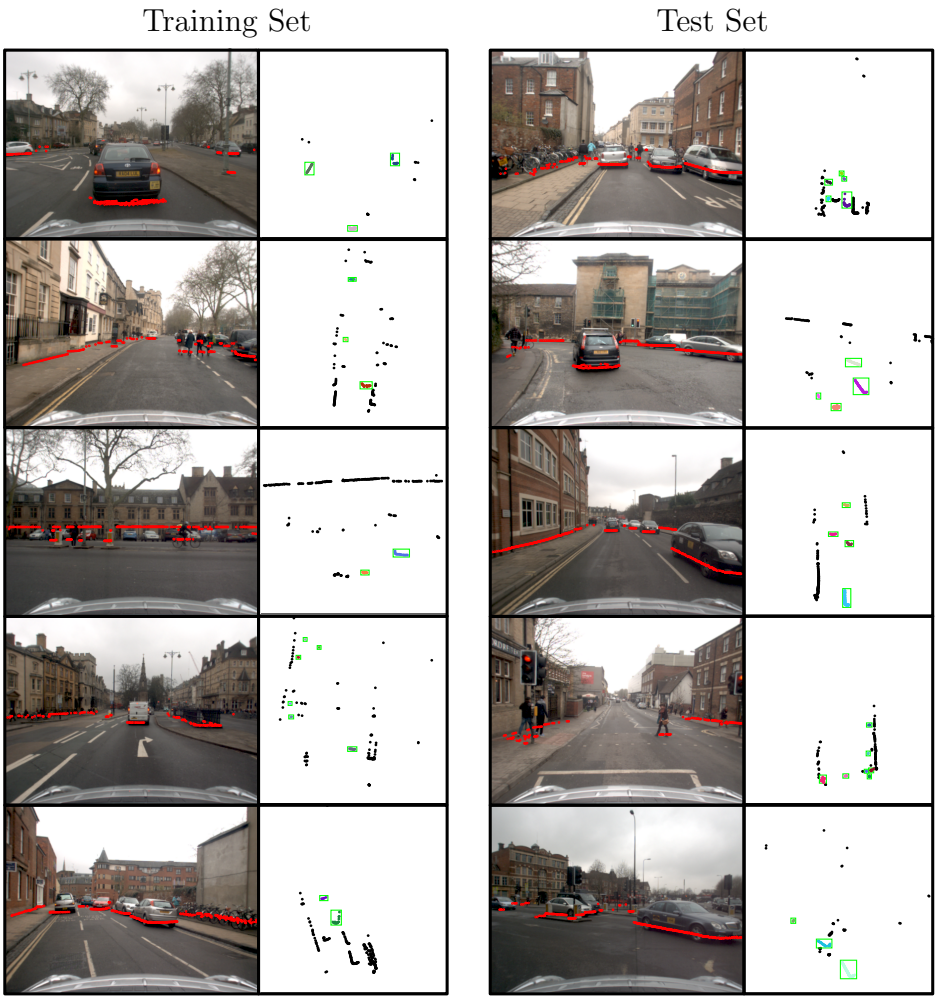


Figure 5.4: Example frames from the datasets used for training and testing, demonstrating the variety and complexity of dynamic scenarios covered by this challenging dataset. For each column, similar to Figure 5.1, the left panel shows the image from the onboard camera *for visualisation only*, with laser scan points projected into the image plane, and the right panel shows the corresponding laser scan with *ground truth* labels. Each ground truth object is highlighted by a green box. The left column shows five example challenging scenarios from the training set. From top to bottom: (1) a complex junction, (2) an area frequently traversed by pedestrians and groups of pedestrians, (3) a T-junction, also showing a cyclist travelling orthogonal to the vehicle heading, and a pedestrian waiting for crossing (note in this scenario the pedestrian is *stationary* hence is *not* included in the ground truth labelling), (4) a situation where a large number of pedestrians can be observed by the laser at a far distance, (5) a car manoeuvring from its parked location. Similarly, the right column gives five example challenging situations from the test set. From top to bottom: (1) a narrow street with pedestrians and manoeuvring cars, (2) a complex junction with oncoming cars turning at different rates (relative to our own vehicle motion), (3) a narrow road with significant oncoming traffic, (4) a busy pedestrian crossing, (5) a wide turn resulting in a large relative motion to oncoming vehicles.

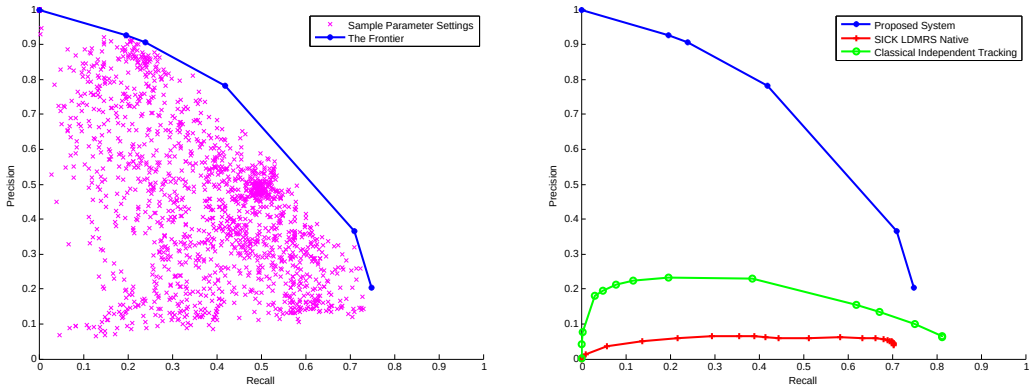
can be satisfied by treating the group as a single object. In fact, the semantic description that this single "object" is actually composed of several "pedestrians" is neither achievable nor necessary given a completely model-free approach. Also, more specifically, this imposes unrealistic demands to the clustering algorithm where the cluster boundaries in this case are really only defined semantically. Individuals who split from a group are labelled as separate objects as soon as their motion differs from the common motion of the group.

5.7.2 Evaluation Metric and System Training

We evaluate the system's ability to detect dynamic objects against the ground truth using the standard Precision and Recall metrics. Specifically, Precision and Recall are computed over the detected object boxes against the hand-labelled ground truth object boxes using the overlapping criterion as is commonly used in the Computer Vision community (Everingham et al., 2010). An object box is marked as a true detection if it overlaps with a ground truth object box by more than a fixed percentage threshold. In all our results, we use 0.5 as the percentage overlap threshold. And a detection is matched to at most one ground truth object, and multiple detections of the same ground truth object are treated as false positives.

To train the system for best performing parameter sets, we follow an approach similar to that described in (Gavrila and Munder, 2007) as follows: both Precision P and Recall R are functions of system parameters, thus if the number of system parameters exceeds one, the set of all feasible (R, P) pairs will in general occupy a continuous 2D space in the R-P plane. The best parameters are then the parameters that give rise to the (R, P) pairs at the *frontier* of the feasible region (conceptually corresponds to the top-right boundary of the feasible region, see Figure 5.5(a) for an example).

Formally, the 2D feasible region parameterised by the set of all possible param-



(a) (b)

Figure 5.5: (a) Scatter plot of the obtained 1803 sample parameter settings using (Snoek et al., 2012) with the estimated frontier overlaid. (b) Precision-Recall tuning curves for the proposed system, the SICK LDMRS's native tracking system, and the independent tracking baseline.

eters $\mathcal{P}$ is given by $\mathcal{F} = \{(R(\mathbf{p}), P(\mathbf{p})) : \mathbf{p} \in \mathcal{P}\}$, the frontier parameter set of $\mathcal{F}$ is given by $F = \{\mathbf{q} \in \mathcal{P} : \forall \mathbf{p} \in \mathcal{P},\ R(\mathbf{p}) \leq R(\mathbf{q}) \text{ or } P(\mathbf{p}) \leq P(\mathbf{q})\}$. In other words, a parameter set is in F if and only if it is not possible to achieve both a higher Precision and a higher Recall.

To find this frontier parameter set, we apply a Bayesian parameter tuning algorithm developed by Snoek et al. (2012) to bias the search in the high-dimensional parameter space to look for satisfactory parameter settings, and obtain an approximation to the frontier parameter set by finding the upper part of the convex hull of the obtained (R, P) scatter plot. Figure 5.5(a) shows the obtained 1803 sample parameter settings with the algorithm, and the the blue curve shows the extracted frontier.

Since the SICK LDMRS's native tracking system clusters each incoming scan and keeps track of every cluster, it makes no distinction between static and dynamic objects. To compare the systems under the same setting, we take tracks with estimated speeds higher than a given threshold to be the detected dynamic objects. It would be desirable to be able to fine-tune the parameters of the LDMRS's native

tracking system. However, the most critical parameters are fixed internally to the sensor, and modifications are unfortunately not feasible.

The independent tracking baseline is similar in the respect that it also does not distinguish between static and dynamic objects. Hence thresholding on the object's absolute speed is also used to find dynamic objects to compare with the proposed system. However, in the case of this baseline, we have control over the internal parameters, thus we apply the same Bayesian parameter tuning technique also to the classical independent tracking baseline to find the best internal parameter setups as well as the best speed threshold.

Figure 5.5(b) presents the Precision-Recall curves for the proposed and the two baseline systems for comparison. The curve of the LDMRS's native system is generated by varying the speed threshold as described above, whereas for the classical independent tracking baseline, shown in the plot is the extracted frontier after Bayesian parameter selection including both the internal parameters and the speed threshold. As can be seen, the proposed system outperforms both baseline systems by a significant margin. This is somewhat expected, since both the LDMRS's native system and the independent tracking approach track only the cluster centroids, which are not stable reference points on the objects to track due to occlusions and dependency on the sensor viewpoint. On the other hand, the proposed system enforces each track's frame of reference to be attached rigidly to the object, and dynamic objects are explicitly handled differently to static ones. The somewhat bizarre behaviour of the Precision-Recall curves for the two baseline systems at the low Recall end is an artefact of the fact that at a speed threshold that is too high, all the sensible speed estimates are below the threshold while there exists false positives with some speed higher than the threshold. In this scenario, one obtains zero Precision *and* zero Recall.

The performance of the LDMRS's native detection system according to the

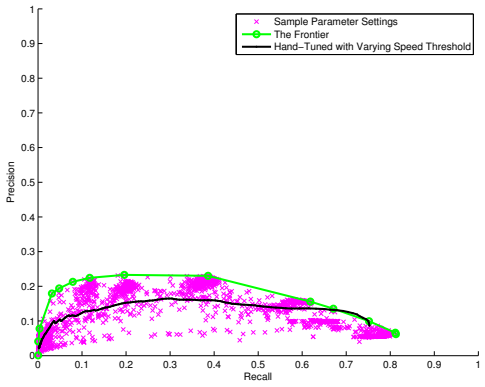


Figure 5.6: An experiment to provide insights to the importance of Bayesian parameter tuning. All results are obtained with the classic independent tracking baseline. Shown on common axes are: a scatter of (R, P) pairs of the 1749 sample parameter settings evaluated by the Bayesian parameter tuning algorithm (Snoek et al., 2012), the extracted frontier based on the sampled parameter settings, and a Precision-Recall curve generated by varying *only* the speed threshold fixing all the internal parameters to a set of hand-tuned values (similar to how the Precision-Recall curve for the LDMRS's native tracking system in Figure 5.5(b) is obtained).

Precision-Recall curves compares inconceivably poorly even against the simple independent tracking baseline. The reason for this apparently poor performance may be explained by the inability to optimise for the system's internal parameters. To see how much difference it makes, Figure 5.6 plots the results for the *classical independent tracking* baseline (whose internal parameters we have total control) in common axes with a Precision-Recall curve for the same baseline generated by fixing the internal parameters to a hand-tuned set of values and only varying the speed threshold, simulating the generation of the Precision-Recall curve in the case of the LDMRS's native system. As can be noted, the Bayesian optimisation algorithm is able to quickly discover different parameter settings that are better than the hand-tuned values, resulting in a much better achievable performance.

	Proposed System	LDMRS Native	Independent Tracking
Precision	**0.45**	0.07	0.14
Recall	0.39	**0.70**	0.23
F_1-measure	**0.42**	0.13	0.18

Table 5.2: Quantitative evaluation of the three systems on the test dataset using the parameters selected with the help of the training set (see text for details). Performance is measured with the standard Precision and Recall metrics, as well as the corresponding F_1-measures for a comparison of overall performance in both Precision and Recall. The best performing system with respect to a specific metric is highlighted in bold.

5.7.3 Test Case Performance : a Quantitative Evaluation

Given a range of operating points along the Precision-Recall curve, we choose empirically a single parameter setting that achieves the best balanced performance from Figure 5.5(b) for each system. Specifically we choose the parameter setting that gives $R = 0.53$ and $P = 0.57$ for the proposed system, the speed threshold that achieves $R = 0.69$ and $P = 0.05$ for the LDMRS's native system, and the parameter setting (including the speed threshold) that gives $R = 0.39$ and $P = 0.23$ for the classical independent tracking baseline. All experiments that follow report metrics evaluated on the *test* dataset using these chosen operating points.

Table 5.2 lists performance metrics evaluated on the test set using the chosen parameter settings for each system. Here in addition to Precision and Recall we also report the standard F_1-measure as an overall performance measure taking account of both Precision and Recall. The trend observed during system training (cf. Figure 5.5(b)) remains also in the test case. Judging by the F_1-measure, the classical independent tracking baseline outperforms the SICK LDMRS's native tracking system, while our proposed system outperforms both by a significant margin. Note however, the absolute figures differ from the training case. In particular, with the selected speed threshold, the LDMRS's native tracking system performs slightly better on the test set than on the training set (a F_1 value on the training set of

0.09 from $R = 0.69$ and $P = 0.05$ compared with 0.13 on the test set), emphasising further the importance of using a different test set for unbiased comparison.

Figure 5.7(a-c) show performance metrics for the three systems on the test dataset as the detection range is varied. This experiment is different from the experiment in Section 4.8.5 (cf. Figure 4.10(c)). Here we are interested in the system's ability to detect objects within a certain range compared with the baseline systems, thus at each range limit, the *ground truth* objects we compare our detections with are also limited to the objects within range. In Figure 4.10(c), we are interested in finding out how system performance degrades when objects above a certain range are ignored by the detector all together, hence in that case, at each range limit, the detections are compared with all *ground truth* objects regardless of range. Therefore in Figure 4.10(c) we expect to see a *decreasing* performance as the range limit reduces, whereas in Figure 5.7(a-c) we expect to see an *increase* in system performance as the range limit reduces, because objects at closer ranges contain more measurements and are usually easier to detect.

From Figure 5.7(a-c), all three systems show a decreasing trend in both Precision and Recall as the detection radius increases. The independent tracking baseline shows an interesting trend in its Recall, which, after an initial drop, increases back up at further distances. This is likely due to the fact that objects far away tend to appear as well-separated clusters of very small sizes – a situation particularly suited to simple tracking techniques that track only the cluster centroids such as the independent tracking approach here.

Figure 5.7(d) places the systems under common axes using the F_1-measure for comparison. From the figure, although the close-range performance of the proposed system and the LDMRS's native tracking system is similar (with the proposed system slightly outperforming), the difference is significant from 20m onwards. Interestingly, the classical independent tracking baseline exhibit close to uniform

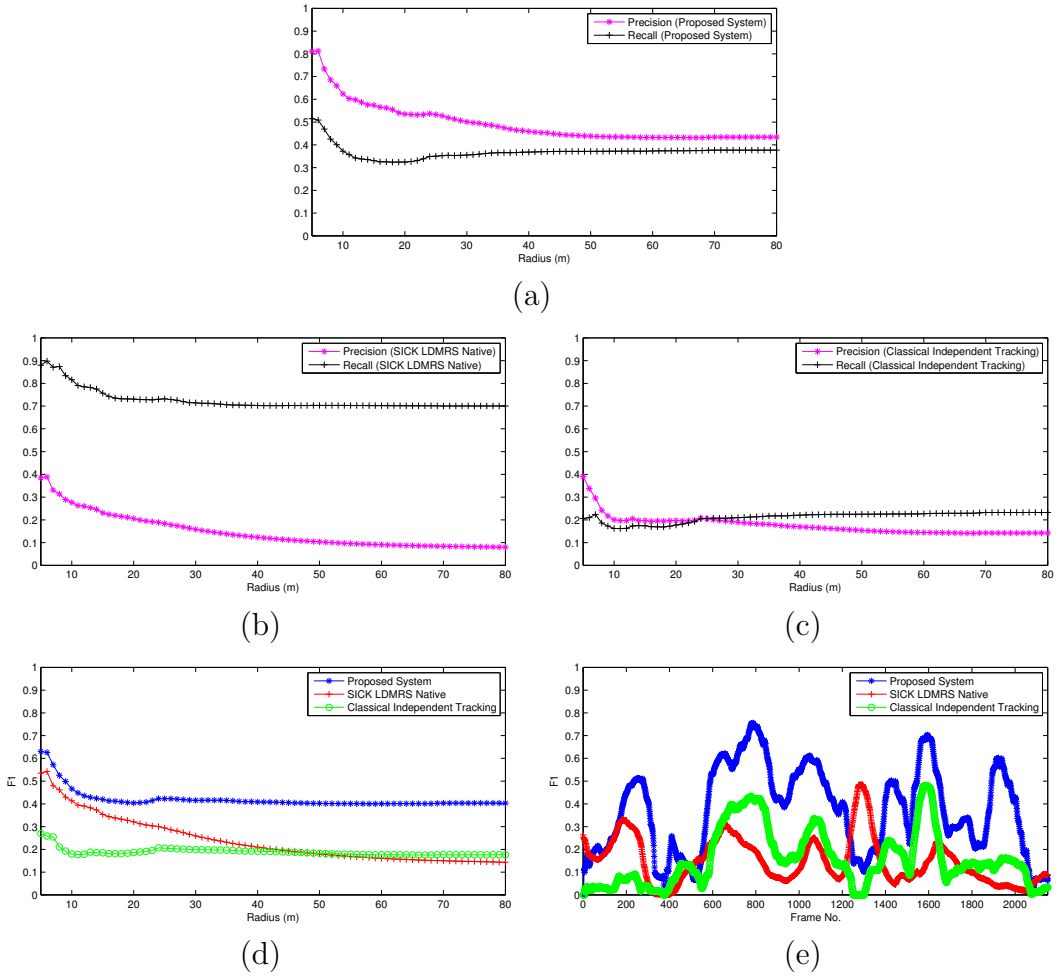


Figure 5.7: (a) Precision and Recall versus operating radius for the proposed system. (b) Precision and Recall versus operating radius for the SICK LDMRS's native system. (b) Precision and Recall versus operating radius for the independent tracking baseline. (d) F_1-measure versus operating radius for all three systems. (e) F_1-measure over past 100 frames versus frame number for the three systems.

performance over different detection ranges. Although performing worse than the LDMRS's native tracking system at close ranges, the independent tracking baseline performs (albeit slightly) better at further ranges, agreeing to the trend observed during training (cf. Figure 5.5(b)).

Figure 5.7(e) compares the instantaneous performance at each frame of the three systems. F_1-measures are evaluated at each frame based on detections of the past 100 frames for each system, and results are plotted against the frame number. While the proposed system outperforms the LDMRS at most frames, there are occasional performance drops. Closer inspection into the dataset reveals that around Frame 400 there exists a period of driving with very few dynamic objects present, hence the apparent low performance from both systems. However, near to Frame 1300, many walking pedestrians close to background clutter are present which are missed out by the proposed system due to segmentation failure. The LDMRS performs better in this scenario but in sacrifice of Precision. The performance of the independent tracking baseline follows approximately the same trend over time as that of the proposed system, albeit at a lower quality, suggesting the performance of any model-free tracking approach is heavily influenced by scene complexity. We take a closer look into the instantaneous performance of the proposed system on the test dataset in the next section, and identify some interesting cases, both the challenging but successful ones, and the common failure modes, together with cases where the performance of the proposed system differs from that of the two baseline systems and discuss about possible reasons for it being so.

5.7.4 Test Case Performance : a Qualitative Evaluation

Continuing with the setup from the previous section, we take a closer look behind the numbers in this section. All qualitative results presented in this section are obtained by running the respective system (our proposed system and the two baselines for

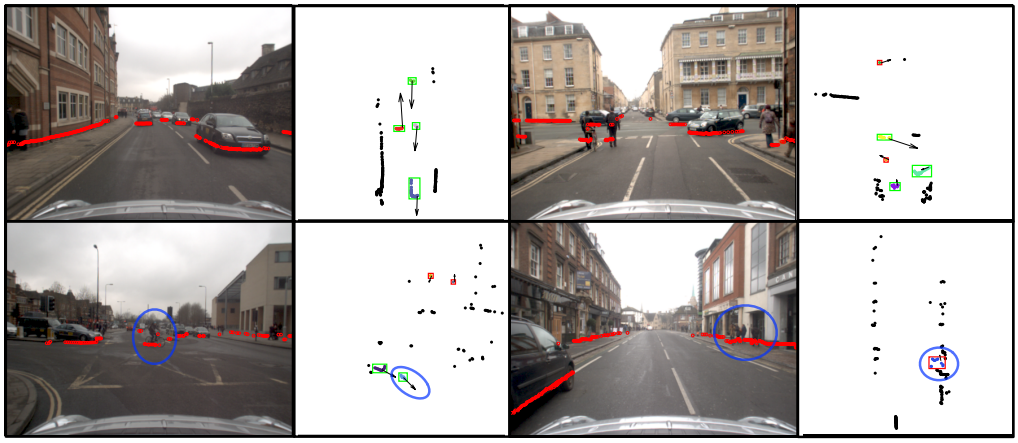
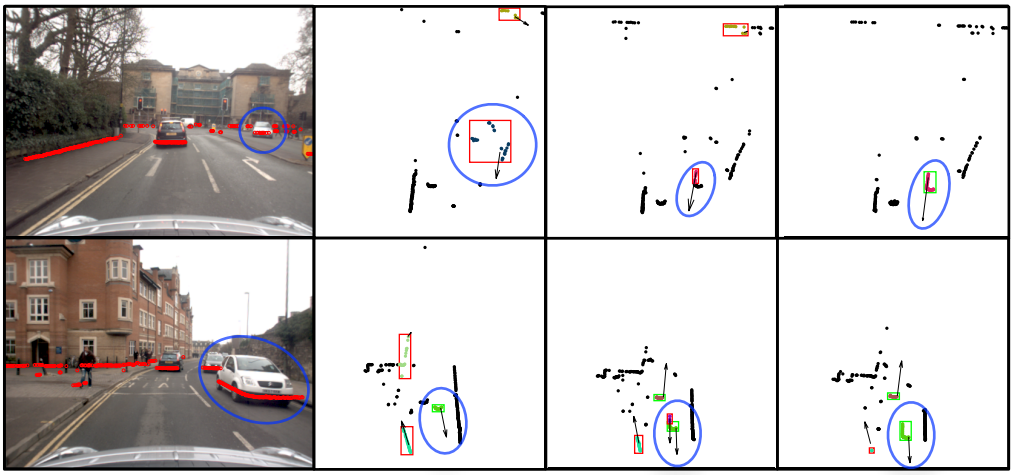


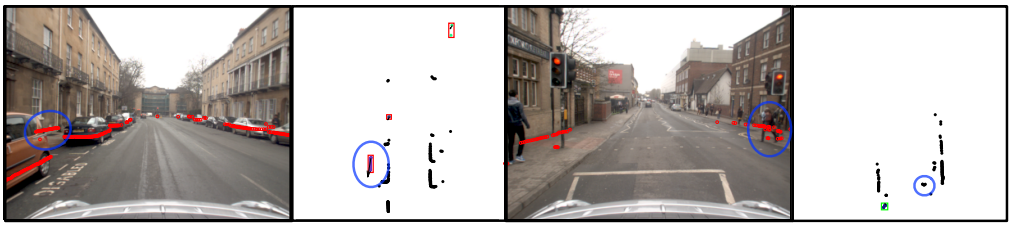
Figure 5.8: Examples of cases where the proposed system is able to successfully detect and track dynamic objects under challenging situations. As in Figure 5.1, green boxes represent true detections according to the ground truth labels, and red boxes denote false alarms. Blue ellipses highlight interesting situations in the scene. Images are provided for visualisation *only*. See text for details.

comparison) on the *test* dataset using the parameter set selected during the training phase (cf. Section 5.7.2) for unbiased evaluation.

Figure 5.8 presents some examples of situations where the proposed system is able to successfully detect and track the dynamic objects in the scene, despite of the complexity of a real driving scenario in a busy town centre. The top left example shows the performance of the proposed system during busy oncoming traffic. All oncoming and leading cars are tracked successfully. Although two pedestrians can be observed on the left, they are not tracked because they are stationary. Here we note again that, in the model-free approach, only *moving* objects are detected and tracked, for example, parked or instantaneously stationary cars are also not included. The top right example gives a complex situation where the system has been successful in tracking a manoeuvring vehicle, two walking pedestrians (one is pushing a bicycle) as a group and another car travelling in the far field. Some pedestrians walking along walls on the right are missed due to segmentation failure (under-segmentation with the wall). In the bottom left example, our vehicle itself is turning giving rise to a large relative motion to the tracked objects. Here a bicyclist



(a)



(b)

Figure 5.9: Common failure modes of the proposed system. (a) The recoverable cases. From left to right: an image with projected laser data to provide some context *only*, and tracking results from the proposed system showing the evolution of the system's perception over time. (b) The unrecoverable cases. In particular, the case on the right is potentially dangerous. In both (a-b), as with other figures, green boxes denote true detections, red boxes denote false alarms. Blue ellipses highlight regions of interest. See text for details.

(highlighted), together with a vehicle behind, are tracked successfully. The far field "false detections", upon closer look, may actually correspond to unlabelled moving pedestrians at a large distance. The bottom right example shows an interesting situation where a group of pedestrians (highlighted) are successfully tracked however the corresponding ground truth label is missing. This suggests that imperfect ground truth labelling may also contribute, to a certain degree, to a possibly underestimated system performance. Note the car on the left in this example is parked.

Figure 5.9 studies some common failure modes of the proposed system. We

divide this study into two different cases: a recoverable case, where despite initial tracking failure, the system subsequently is able to recover from the incorrect states, and an unrecoverable case, where an object is erroneously tracked or missed until it moves out of the view of the sensor.

Figure 5.9(a) gives two common causes of recoverable failure. The top row shows the situation where some measurements on the static background get erroneously associated with a moving vehicle being tracked, due to under-segmentation (left laser scan). However, the hierarchical data association procedure (cf. Section 5.6) means that in the next iteration, this unreasonably large "object" is likely to be associated with more measurements on the static background. This in turn results in its estimated motion to drop, until eventually low enough to be merged with the static background (cf. Section 5.5.4). Then because motion inconsistency, soon a new object track will be initialised on the same vehicle (middle laser scan), and old boundary points that are supposed to belong to the tracked vehicle but falsely merged with the static background previously will expire because they are no longer observed. After that the system successfully recovers and keeps good track of the vehicle (the right scan). The bottom row, on the other hand, shows a case of recoverable failure due to *over*-segmentation. Because of viewpoint changes, a car is first erroneously tracked as two separate segments (the middle scan). But thanks to the merging procedure (cf. Section 5.5.4), the system soon realises these segments should belong to the same object, and the car is subsequently tracked successfully as a single entity (the right scan). Although during a recoverable failure, the system eventually corrects its mistakes, there is nonetheless a transient time during which tracking is incorrect. Most of recoverable failures are due to segmentation errors. Despite our efforts in obtaining perceptually coherent segments in introducing the EMST-EGBIS algorithm in Section 5.6.5, segmentation errors are inevitable. One possible way to improve on this matter is to introduce semantics into the framework,

to actively *look for* certain known types of cluster boundaries. We will return to this point in our discussions in Chapter 6.

Figure 5.9(b) presents two common cases of unrecoverable failure. The inclusion of a static background in our model helps significantly in resolving ambiguities resulting from viewpoint changes or occlusion (this will be demonstrated later qualitatively in this section). However, in regions beyond the current extend of the estimated static background, false detections may still arise when a static structure appears dynamic due to sensor motion, such as the short section of the wall visible only between the two parked cars (highlighted) in the left example in Figure 5.9(b). This false detection may not be as severe because the erroneous "dynamic" object appears to travel parallel to and in reverse direction as our own vehicle. The failure case in the example on the right however, is very *critical*. In this case, it is possible that sometimes motion of slow objects such as pedestrians is below the threshold for detection. When this happens, if the pedestrian remains walking in a low speed, they may be missed entirely, albeit crossing *right in front of the vehicle*. This situation exposes a weak point of purely model-free approaches (such as the proposed system here). Because of a lack of any semantic interpretation, the situation is possible to occur no matter what system parameters are actually used, provided a pedestrian walks *slowly enough*. Again, one possible solution is to bring in model-based elements in aiding difficult situations such as a slow walking pedestrian. We include this possible extension in our discussions in Chapter 6.

Figure 5.10 gives three common cases where the performance of the proposed system tends to differ from that of the two baselines we compared with quantitatively in Section 5.7.3. The first example (the top row), demonstrates the importance of keeping an estimate of a local static background as part of the state. As can be seen from the figure, both baselines give false detections around the section of the wall on the right, due to either uncertain motion estimates (LDMRS native tracking,

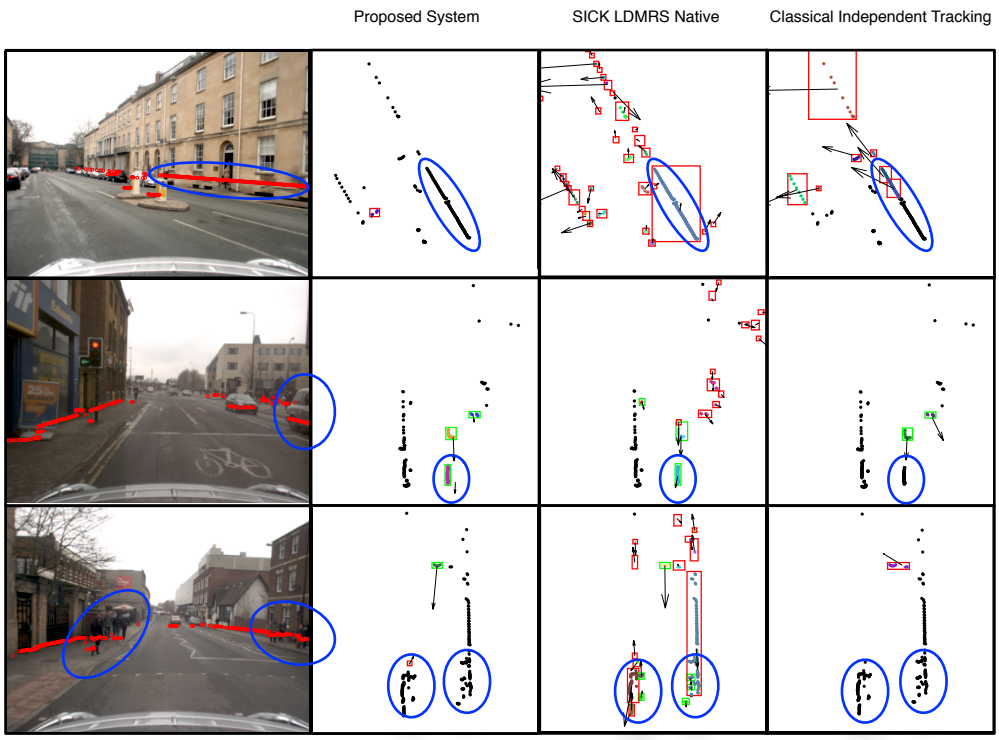


Figure 5.10: Three cases of differing performance over the three systems evaluated quantitatively in Section 5.7.3. Each row is structured from left to right: the camera image for visualisation *only*, the tracking result of the proposed system, the tracking result of the SICK LDMRS's native system, and that of the classical independent tracking baseline. As with other figures, green boxes represent true detections, red boxes indicate false alarms. Interesting areas are highlighted with blue ellipses. See text for details.

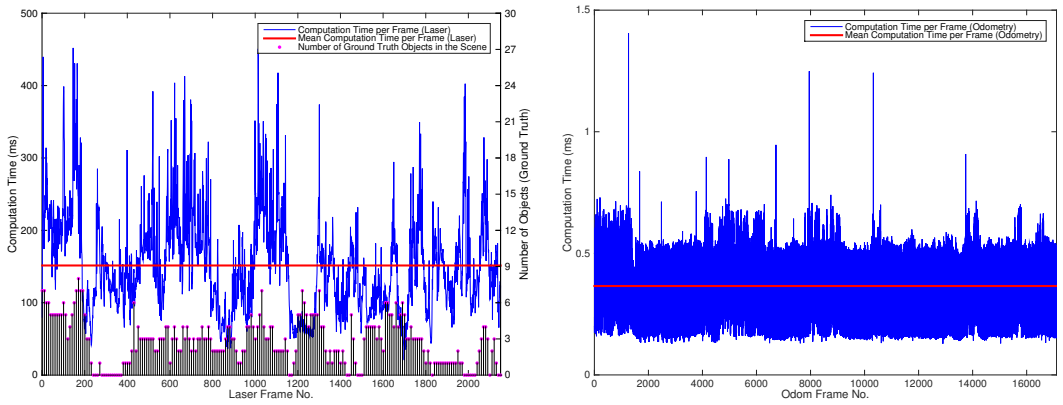
middle scan), or data association failure (independent tracking, right scan). The proposed system, however, suffers none of these problems, because as soon as the measurements belonging to the wall are assigned to the static background, they are *known* to be stationary. The second example (the middle row), shows different system responses in the event of viewpoint changes. A car (highlighted) traveling towards our vehicle in the opposite direction is being tracked successfully until it gradually goes out of view. The appearance of the car thus changes significantly as it moves past the sensor. Both baseline systems, because they all track only the cluster centroids, do not have a stable reference tracking point on the car, thus the motion estimates they produce are incorrect (in the case of the independent tracking, it is believed that the car has stopped). Our proposed system, on the other hand, tracks a fixed reference point rigidly attached to the object due to our special object representation (cf. Section 5.4.2), hence is able to maintain correct tracking on the car (left laser scan). The third example (bottom row) examines in detail the discrepancy in performance of the three systems around Frame 1300 observed in Figure 5.7(e) earlier. As may be noted from the detection results shown, due to noisy sensor measurements and segmentation errors, the two groups of pedestrians (highlighted) on pavements either side of the road are largely missed by both the proposed, and independent tracking systems. The LDMRS's native tracking system, however, is able to detect and track the majority of the pedestrians with success, albeit at the sacrifice of a large number of false positives. The fact that both the proposed system and the independent tracking baseline make the same mistakes in this case may suggest that the error is due to either a common mistake made by the clustering algorithm (both approaches use the EMST-EGBIS clustering algorithm described in Section 5.6.5), or the way multiple scanning layers are fused into a single scan described in Section 5.7.1. The mechanisms the LDMRS's native tracking system relies on to fuse information from multiple scanning layers are not known to

us. Some specifics of the information fusion mechanism may have proven successful in this particular situation.

5.7.5 Timing

In this section, we take an empirical analysis of the computational efficiency of the proposed system. The parameter training procedure followed in Section 5.7.2 is excellent in investigating the full potential of a system in terms of system performance. However, different parameter settings inevitably give rise to different computational requirements. For example, a larger maturity threshold (cf. Section 5.5.4) means a larger number of tentative tracks may be kept around in the system before they are merged with the static background or existing object tracks, increasing the load of the system. The best performing parameter set is not necessarily the most efficient parameter set. As a compromise, we hand-tuned the system parameters by visual inspection using *only* the training dataset for a reasonable performance and at the same time a satisfactory computational time. We made sure during this hand-tuning process the test set is completely hidden away from us so that any performance and timing measures remain unbiased. The resulting parameter set gives a Precision of 0.47 and a Recall of 0.38 evaluated on the *test* dataset, corresponding to an F_1-measure of 0.42. These figures compare well with the values quoted in Table 5.2 using the best performing parameter set selected during the training phase.

Figure 5.11 then shows timing results using this hand-tuned parameter set as plots of processing times for the laser and odometry measurement types respectively. The results are generated using our current prototype implementation in MATLAB on a MacBook Pro equipped with a quad-core 2.8GHz Intel i7 CPU and 16GB of RAM. Figure 5.11(a) shows a plot of the time taken per frame over the entire test sequence (in milliseconds) in the case of laser measurements, together with a rough measure of scene complexity at each time instant. Scene complexity here



(a) (b)

Figure 5.11: (a) Computation time per frame for each laser frame. Also shown on the same graph is a simple measure of scene complexity given by the number of ground truth objects present at each frame. There exists a certain degree of correlation between computation time and scene complexity. (b) Computation time per frame for each odometry measurement frame. In both (a) and (b), the mean computation time per frame is indicated by a horizontal line.

is measured by the number of ground truth objects present at any given instant. Some degree of correlation may be observed between computation time per frame and scene complexity, especially during the first half of the sequence (up to around Frame 1200). Of course, we note here computation time per frame should not completely depend on scene complexity measured by the number of dynamic objects present, because at each laser frame the static background also needs to be updated. The average time taken per frame evaluates to 151ms, corresponding to a frame rate of around 6.6Hz if buffering is used. Figure 5.11(b) shows a similar time plot for odometry measurements. Update times taken on each odometry measurement appear much more stationary (with occasional outliers) compared with those of the laser measurements. This makes intuitive sense because odometry updates do not depend on scene complexity. The average time taken per odometry update evaluates to a mere 0.37ms, confirming the theoretical insight gained in Section 5.5.1.

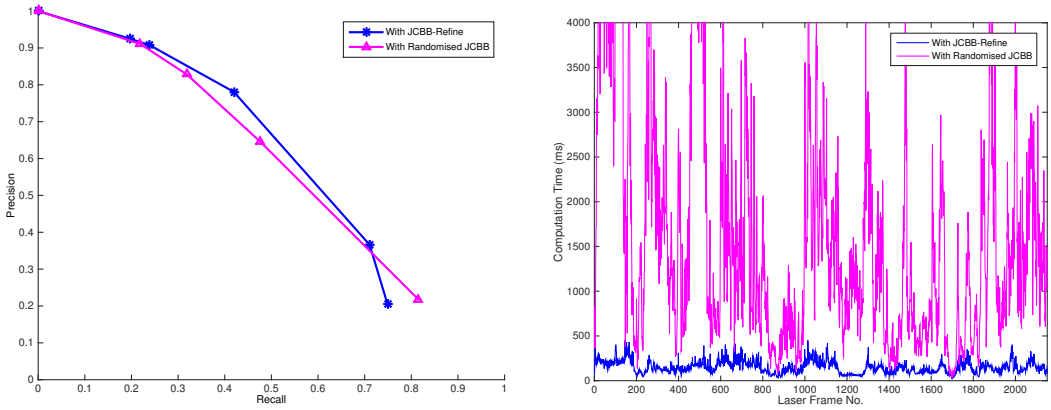
5.7.6 An Evaluation of the JCBB-Refine Algorithm

In this section, we evaluate the efficacy of the proposed JCBB-Refine fine-level data association algorithm (cf. Section 5.6.3). The JCBB-Refine algorithm enables processing of a large number of measurements that is infeasible for the standard JCBB due to its exponential complexity with respect to the number of measurements. An alternative known extension to the JCBB algorithm for associating a large number of measurements is the Randomised JCBB algorithm proposed by Paz et al. (2008).

Instead of refining an initial association like JCBB-Refine, Randomised JCBB does not require an initial association. Instead, Randomised JCBB follows the RANSAC paradigm (Fischler and Bolles, 1981, also see Section 3.2.2). At each iteration, a small number of measurements are selected at random (four measurements are selected in (Paz et al., 2008)). Then the standard JCBB algorithm is applied to optimally assign this small set of measurements, following which the rest of the measurements are assigned in a greedy fashion. The total number of assigned measurements is taken as the number of inliers. This process iterates. In the end, the association with the most number of inliers is returned as the final solution.

Though attractive as a solution in our situation, the RANSAC iterations in Randomised JCBB are likely to introduce additional computational burdens. JCBB-Refine, on the contrary, starts with an initial guess of the association, and reaches a jointly compatible association in a single greedy pass through the measurements.

Figure 5.12(a) compares the performance of the proposed system with a variant using Randomised JCBB replacing JCBB-Refine for fine-level data association. For fair comparison, the variant with Randomised JCBB has also been tuned systematically using the approach detailed in Section 5.7.2 on all tunable system parameters using the training set. There is no notable performance difference between the proposed system using JCBB-Refine for fine-level data association and the variant using Randomised JCBB. However, using JCBB-Refine enables a significantly faster



(a) (b)

Figure 5.12: (a) Comparison of the proposed system using the JCBB-Refine fine-level data association with a variant using Randomised JCBB (Paz et al., 2008) for data association. Shown on common axes are Precision-Recall curves for the respective variants generated after searching for optimal parameter settings with the Bayesian parameter optimisation package (Snoek et al., 2012) using the training set. The curve for the original variant using JCBB-Refine is the same as the curve in Figure 5.5. (b) Computation time per frame for laser measurements versus frame number for the two variants. The curve for the original variant using JCBB-Refine is the same as the curve in Figure 5.11(a). Note the different scale on the y-axis.

computation as can be noted from Figure 5.12(b), confirming our reasoning above. Here the timing plot for the variant using Randomised JCBB was generated following the same procedure as the proposed system (Section 5.7.5) to remove any bias in comparison, i.e. all system parameters were hand-tuned for a good performance as well as a reasonable computational speed using the training data, then timing results were obtained on the test set using this set of parameters.

If a good initial guess is available, such as from the result of ICP in our case, it is more preferable to use JCBB-Refine.

5.8 Conclusions

In this chapter, we presented a unified Bayesian framework for jointly estimating the sensor pose, a local static background and dynamic states of moving objects.

The main focus of our work is on the detection and tracking of moving objects independent of classes and shapes. We described our model-free representation of objects using boundary points initialised with raw laser measurements, and derived their observation models. The dynamics of the moving objects are inferred as hidden variables under a rigid body constraint, making the quality of the data association algorithm critical to the system's correct operation.

Therefore, within the same unified framework, we proposed a novel two-level data association algorithm that takes benefits of both the density of observations and strong correlations between them. A new variant of the JCBB algorithm (Neira and Tardos, 2001) was suggested to tackle with large numbers of measurements, and a solution to numerical stability issues under such scenarios was also presented.

Finally, the proposed system was tuned systematically on real-world data against hand-labelled ground truth, and both quantitative and qualitative evaluations demonstrated the system's superior performance over an existing industry standard also targeted at object tracking for automotive applications and a classical model-free independent tracking approach.

Chapter 6

Conclusions and Discussions

6.1 Summary of Contributions

In this thesis, we presented our contributions to laser-based dynamic object detection and tracking from two different perspectives: the model-based, and model-free perspectives.

6.1.1 Model-Based 3D Object Detection

Model-based approaches focus on the detection of *potentially* moving objects based on the semantics of the objects. We first proposed an end-to-end segmentation-based object detector for the detection of cars, pedestrians and bicyclists in Chapter 3. Then in Chapter 4 we presented one of the key contributions of this thesis – our sliding window approach to object detection in 3D.

6.1.1.1 Segmentation-Based Object Detector

The proposed segmentation-based detector takes as input a *raw* stream of 3D point locations and produces clusters (segments) corresponding to distinctive object entities labelled with the object class. In particular, we tackled the issue of segmenta-

tion errors by first producing an over-segmentation, and then grouping the resulting super-voxels into object clusters once more information has accumulated about their identities. Our key contribution in this work is the postulation that at the level of the over-segmented super-voxels, there is insufficient semantic information to decide the true identity of the segment other than whether it belongs to one of the three object classes of our interest (*car*, *pedestrian* or *bicyclist*), or the background clutter. Based on this *binary* classification, we may collate all *foreground* super-voxels, and group them into holistic object clusters, where there will be more evidence at this level to decide the object's true identity. We verified our postulation using a collection of custom and publicly available data of urban street scenes.

As part of our detection pipeline, we also presented a novel EMST-based clustering algorithm that adapts to the native characteristics of the laser by the use of a RANSAC edge selection criterion. The resulting EMST-RANSAC clustering algorithm is able to successfully segment clusters of points according to their Euclidean distances of separation independent of the number of objects in the scene.

6.1.1.2 Sliding Window Object Detection in 3D

The sliding window approach to object detection, while ubiquitous in the Computer Vision community, is largely neglected in laser-based detection methods, possibly due to its perceived computational inefficiency.

In Chapter 4, we demonstrated that, by taking advantage of the sparsity of 3D laser scans, the processing of a large number of window locations can be avoided. Specifically, we proved mathematically the equivalence between convolution on a sparse feature grid and voting, providing an efficient algorithm to exactly compute the detection scores at all window locations and all orientations using only the bare minimal amount of computation, resulting in an average computation time of less than half a second for a complete 3D scan containing in the order of $100,000$ points.

We demonstrated the superior performance of the proposed detector on the publicly available KITTI dataset, and showed that it achieves commensurable qualitative performance to the best available vision-based detectors on the *car* class. The proposed sliding window detector is both faster and better-performing than the previously proposed segmentation-based detector in Chapter 3.

This work is to our best knowledge the first to apply a sliding window approach to object detection in 3D data.

6.1.2 Model-Free Tracking with 2D Laser

Model-free methods, in contrast to model-based methods, focus on the detection and tracking of *instantaneously* moving objects, independent of their classes and shapes. In Chapter 5, we presented a new approach to model-free tracking of dynamic objects that represents an object by a collection of points on its boundary initialised (and subsequently updated) with raw laser measurements, thus allowing a flexible nonparametric representation. Dealing with raw laser points poses a significant challenge to data association. We proposed a hierarchical approach, and presented a new variant of the well-known Joint Compatibility Branch and Bound (JCBB) algorithm to respect and take advantage of the constraints of the problem introduced through correlations between observations. Finally, we calibrated the system systematically on real world data containing 7.5K labelled object examples and validated on 6K test cases. We demonstrated its performance over an existing industry standard targeted at the same problem domain as well as a classical approach to model-free object tracking.

6.2 Discussions and Future Research

6.2.1 On the Sliding Window 3D Object Detector

Although we focused our evaluation of the sliding window 3D object detector on the car class in this thesis, our formulation is general enough for the detection of *any* object class. Evaluating the effectiveness of the current feature set on the detection of other object classes such as pedestrians and bicyclists is a part of our future work.

Currently, we have chosen six simple features capturing both the shape and appearance aspects of the objects. Although our current choice of the feature set gives superior performance as demonstrated by the evaluations in Chapter 4, designing the best feature set for sliding window detection in 3D is not the main focus of the work presented here. In seeking further improvements to detection performance, we are keen to apply unsupervised feature learning techniques (Bo et al., 2012; Ngiam et al., 2011; Ranzato et al., 2008) to search for better feature representations than the currently hand-picked simple features.

We are also interested in generalising the proposed sliding window approach, in particular the voting paradigm, to other cases where the feature grid is inherently sparse, such as the *2D* laser scan, or the 2.5D scan formed by the four scanning layers of the SICK LDMRS sensor (without projection to a single scanning plane as we have done in the case of the model-free tracker). We wish to investigate the effectiveness of various feature choices (including the possibility of unsupervised feature learning) on these alternative sensor modalities, then evaluate the performance and efficiency of the proposed sliding window framework in these cases. In particular, computation time is expected to reduce significantly due to a much reduced number of points.

6.2.2 On Improving Tracking Performance

There are several advantages in taking a model-free approach over a model-based approach to object tracking. For example, dynamic objects are identified and tracked independent of their shapes and classes, lifting the design efforts required in model-based approaches to treat each object class separately. Also, unexpected object classes are covered in the same framework with no difference to other familiar object classes, thus providing full situational awareness.

Despite these advantages, a purely model-free approach does lack semantic interpretation to the objects it tracks, giving difficulties in many folds, some of which we have already encountered in Section 5.7.4. There is still room for improvements over the model-free tracking framework proposed in Chapter 5. In the future, we aim to extend our model-free framework with model-based elements to bring the best of both worlds. Specifically, model-based elements may be designed as "plug-ins" to the existing model-free framework serving only to improve its performance. Individual object classes (for example, the familiar object classes cars, pedestrians and bicyclists) can be added one at a time or whenever a corresponding "plugin" is available (e.g. by using the sliding window detector proposed in Chapter 4 trained on a specific object class). Each of such modules provides additional semantic interpretation for the objects the model-free tracker is currently tracking, aiding it in many fronts. First, given semantic interpretations, we may bias the output of the EMST-EGBIS algorithm to correctly segment objects that are known to the model-based modules, reducing segmentation errors. Second, once an object of a given class is recognised, it may be flagged as a *potential* dynamic object even if it is not actually moving or moving in a low speed. Finally, additional semantic knowledge will enable us to apply more sophisticated motion models, or even multiple models (e.g. the Interacting Multiple Model (IMM) filter (Zhao and Thorpe, 1998)) to objects whose class is known with confidence to better account for all

complexities of motion the object may exhibit. This may include, for example, in the case of pedestrians, a slow motion model to help with the situation encountered in Section 5.7.4. When an unexpected object is observed, or the object class may not be determined with confidence, the system falls back to model-free tracking, still maintaining a full situational awareness.

6.2.3 On Combining Sensor Modalities

The focus of this thesis has been object detection and tracking with *lasers*. Of course, we do not need to restrict ourselves to using only laser data if other sensors are available such as cameras. The additional information contained in other sensor modalities such as rich appearance information in vision compliments rich shape information from lasers. When fused appropriately detection performance is expected to increase over using either individual sensor modality alone. This idea of combining sensor modalities for object detection has been a focus of active research in recent years (e.g. Spinello et al., 2010; Oliveira and Nunes, 2013; Fotiadis et al., 2013; Premebida et al., 2014).

The approaches proposed in Chapters 3 and 4 can be readily extended to fuse information from a camera. In the simplest case, assuming a good extrinsic calibration is available between the laser and the camera, we can obtain RGB colour information for each laser point by re-projection. After this process, each laser point will contain, in addition to its (x, y, z) coordinates and intensity value, a (r, g, b) colour information. Then additional appearance features based on the RGB values can be added to the 3D feature set to aid detection. In the case of the segmentation-based detector proposed in Chapter 3, we may fuse the information at an even earlier stage of the pipeline – appearance information can be used to define similarities between neighbouring points in addition to surface normal discontinuity to improve pre-segmentation.

Following such a simple approach essentially transfers only a part of the dense appearance information available from an image into the sparse points in 3D. To take advantage of dense appearance information such as texture contained in an image, we may extract dense features on the image such as HOG (Dalal and Triggs, 2005), and transfer the computed dense features, in additional to raw (r, g, b) pixel values to corresponding 3D points. Going further from this idea, we may think of interpolating depth values at all pixel locations given the projected pixel locations from the sparse laser data similar to (Premebida et al., 2014). If another camera is available, for example from a stereo system, the second camera can be used to verify the interpolated pixel depths to remove erroneous interpolated values. After depth interpolation, each pixel in the image can be traced back to 3D to create a dense 3D point cloud. Investigating the feasibility of such a dense 3D reconstruction from stereo aided by a sparse 3D laser, and the applicability and performance of the proposed detection systems (Chapters 3 and 4) on such dense 3D data are all part of our future work.

Appendix A

Preliminaries

A.1 The Support Vector Machine

In this appendix, we briefly describe the Support Vector Machine for classification. While a full exposition of the details of SVM classifiers is beyond the scope of this thesis, we are well served by a synopsis. First consider a set of training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) : i \in \mathbb{N}, i \leq N\}$ where $\mathbf{x}_i$ is a training instance in the feature space and y_i is its label. The SVM classifier is a *binary* classifier, so that the data is to be classified into one of *two* classes only, i.e. $y_i \in \{1, -1\}$. The actual values of the labels are arbitrary, the specific choice is due to compatibility with existing literature and in aid of later formulation of the cost function.

The simplest formulation of SVM assumes $\mathcal{D}$ is linearly separable, that is, there exists a hyperplane in the D-dimensional feature space such that all $\mathbf{x}_i$ with $y_i = 1$ lie on one side of the plane and all $\mathbf{x}_i$ with $y_i = -1$ lie on the other. If the equation of the hyperplane is given by

$$\mathbf{w}^\top \mathbf{x} = -b \ , \tag{A.1}$$

the classifier finds the parameters $\mathbf{w}$ and b for the plane such that the classification margin is maximised, that is, the training examples are as far away from the decision

hyperplane as possible. Because the margin of the decision boundary is inversely proportional to $||\mathbf{w}||$, SVM classifiers minimise a cost function that is proportional to $\mathbf{w}^\top \mathbf{w}$.

In reality strictly linearly separable classes are rare, and this is accommodated by allowing a small number of data points to fall inside the margin, with the formulation

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{N} \xi_i \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \ , \\ & \xi_i \geq 0 \ , \forall (i \in \mathbb{N}, i \leq N) \ , \end{aligned} \tag{A.2}$$

where ξ_i are slack variables that are strictly positive for points penetrating the margins, zero otherwise.

When the feature space is not likely to be linearly separable, it is common to apply so called the "kernel trick", where the original feature space is mapped to a higher dimensional feature space by a nonlinear mapping $\boldsymbol{\phi}(\mathbf{x}_i)$ such that in the new feature space the two classes are linearly separable.

Equation A.2 can be expressed alternatively in a *dual* formulation, such that when $\boldsymbol{\phi}(\mathbf{x}_i)$ is substituted for $\mathbf{x}_i$, it appears only in the form of $k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}^\top(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)$, which is termed the *kernel function.*

Since the nonlinear mapping $\boldsymbol{\phi}$ is never explicitly evaluated in the dual formulation, kernel functions can be constructed that map the original feature space into an *infinite* dimensional feature space, achieving better separability between the classes. One of such kernel functions that are commonly deployed in practice is the Gaussian Radial Basis Function (RBF) kernel defined by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2) \ . \tag{A.3}$$

When the number of classes is more than two, a common practice is to train one-versus-all classifiers for each class, that is, a *binary* SVM classifier for each class against all examples *not* belonging to the class, with the final classification label determined by a greedy winner-takes-all approach – the class whose binary one-versus-all classifier gives the most positive result.

For a detailed exposition to SVM classifiers, the reader is referred to (Bishop, 2006, Chapter 7).

A.2 Precision, Recall and F-measures

Precision and Recall are measures of performance for classification or detection systems defined as

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \ , \tag{A.4}$$

and

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \ , \tag{A.5}$$

where P and R denote Precision and Recall respectively, and TP, FP, FN represent the numbers of *true positive*, *false positive* and *false negative* classifications (or detections depending on what system is being evaluated) respectively.

The F-measures are balanced measures between Precision and Recall, and are defined by

$$F_\beta = \frac{(1 + \beta^2)PR}{(\beta^2 P) + R} \ . \tag{A.6}$$

Here again, P and R denote Precision and Recall respectively, and β is a positive real number that specifies the *weight* given to *Recall*. For example, as $\beta \to 0$, $F_\beta \to P$, whereas as $\beta \to \infty$, $F_\beta \to R$.

Arguably the most useful F-measure is the F_1-measure, where $\beta = 1$:

$$F_1 = 2\frac{PR}{P+R} \ . \tag{A.7}$$

The F_1-measure gives equal weights to Precision and Recall, hence is commonly taken as an overall measure of system performance.

Bibliography

Anguelov, D., Taskarf, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., and Ng, A. (2005). Discriminative learning of Markov random fields for segmentation of 3D scan data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 169 – 176 vol. 2.

Arras, K., Grzonka, S., Luber, M., and Burgard, W. (2008). Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1710–1715.

Arras, K., Mozos, O., and Burgard, W. (2007). Using Boosted Features for the Detection of People in 2D Range Data. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3402–3407.

Asano, T., Bhattacharya, B., Keil, M., and Yao, F. (1988). Clustering algorithms based on minimum and maximum spanning trees. In *Proceedings of the fourth annual symposium on Computational geometry*, SCG '88, pages 252–257, New York, NY, USA. ACM.

Bar-Shalom, Y., Kirubarajan, T., and Li, X.-R. (2002). *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, Inc., New York, NY, USA.

Besl, P. and McKay, N. D. (1992). A method for registration of 3-D shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Biswas, R., Limketkai, B., Sanner, S., and Thrun, S. (2002). Towards object mapping in non-stationary environments with mobile robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 1014–1019 vol.1.

Bo, L., Ren, X., and Fox, D. (2012). Unsupervised Feature Learning for RGB-D Based Object Recognition. In *ISER*.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2:27:1–27:27.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1.

Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., and Frenkel, A. (2011). On the segmentation of 3D LIDAR point clouds. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2798 –2805.

Dubout, C. and Fleuret, F. (2012). Exact Acceleration of Linear Object Detectors.

In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 301–311.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Endres, F., Plagemann, C., Stachniss, C., and Burgard, W. (2009). Unsupervised discovery of object classes from range data using latent Dirichlet allocation. In *Proceedings of Robotics: Science and Systems*, Seattle, USA.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338.

Fairfield, N. and Urmson, C. (2011). Traffic light mapping and detection. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5421 – 5426.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object Detection with Discriminatively Trained Part-Based Models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. *Int. J. Comput. Vision*, 59:167–181.

Fidler, S., Dickinson, S., and Urtasun, R. (2012). 3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model. In Pereira, F., Burges,

C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 611–619. Curran Associates, Inc.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395.

Fotiadis, E. P., Garzón, M., and Barrientos, A. (2013). Human Detection from a Mobile Robot Using Fusion of Laser and Vision Information. *Sensors*, 13(9):11603–11635.

Gavrila, D. M. and Munder, S. (2007). Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle. *Int. J. Comput. Vision*, 73(1):41–59.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237.

Gottschalk, S. A. (2000). *Collision Queries Using Oriented Bounding Boxes*. PhD thesis, The University of North Carolina at Chapel Hill. AAI9993311.

Granström, K. (2012). *Extended target tracking using PHD filters*. PhD thesis, Linköping University, Automatic Control, The Institute of Technology.

Grygorash, O., Zhou, Y., and Jorgensen, Z. (2006). Minimum Spanning Tree Based Clustering Algorithms. In *Tools with Artificial Intelligence, 2006. ICTAI '06. 18th IEEE International Conference on*, pages 73 –81.

Hahnel, D., Triebel, R., Burgard, W., and Thrun, S. (2003). Map building with mobile robots in dynamic environments. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1557–1563 vol.2.

Hedau, V., Hoiem, D., and Forsyth, D. (2010). Thinking Inside the Box: Using Appearance Models and Context Based on Room Geometry. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 224–237, Berlin, Heidelberg. Springer-Verlag.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University.

Huang, A. and Teller, S. (2010). Probabilistic Lane Estimation using Basis Curves. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain.

Isack, H. and Boykov, Y. (2012). Energy-Based Geometric Multi-model Fitting. *International Journal of Computer Vision*, 97(2):123–147.

Jenssen, R., Hild, K.E., I., Erdogmus, D., Principe, J., and Eltoft, T. (2003). Clustering using Renyi's entropy. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 1, pages 523 – 528 vol.1.

Johnson, A. (1997). *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Johnson, A. and Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433 –449.

Klasing, K., Althoff, D., Wollherr, D., and Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3206 –3211.

Klasing, K., Wollherr, D., and Buss, M. (2008). A clustering method for efficient

segmentation of 3D laser data. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 4043 –4048.

Kolmogorov, V. and Zabin, R. (2004). What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159.

Koppula, H. and Saxena, A. (2013). Anticipating Human Activities using Object Affordances for Reactive Robotic Response. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011a). A large-scale hierarchical multi-view RGB-D object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817 –1824.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011b). Sparse distance learning for object recognition combining RGB and depth information. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4007 –4013.

Lehmann, A., Leibe, B., and Van Gool, L. (2011). Fast PRISM: Branch and Bound Hough Transform for Object Class Detection. *International Journal of Computer Vision*, 94(2):175–197.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust Object Detection with Interleaved Categorization and Segmentation. *International Journal of Computer Vision*, 77(1-3):259–289.

Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S.,

and Williams, J. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774.

Levinson, J., Montemerlo, M., and Thrun, S. (2007). Map-Based Precision Vehicle Localization in Urban Environments. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA.

Long, C., Wang, X., Hua, G., Yang, M., and Lin, Y. (2014). Accurate Object Detection with Location Relaxation and Regionlets Relocalization. In *Asian Conference on Computer Vision*.

Luber, M. and Arras, K. (2013). Multi-Hypothesis Social Grouping and Tracking for Mobile Robots. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany.

March, W. B., Ram, P., and Gray, A. G. (2010). Fast euclidean minimum spanning tree: algorithm, analysis, and applications. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 603–612, New York, NY, USA. ACM.

McManus, C., Churchill, W., Napier, A., Davis, B., and Newman, P. (2013). Distraction suppression for vision-based pose estimation at city scales. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3762–3769.

McNaughton, M., Urmson, C., Dolan, J. M., and Lee, J.-W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4889 –4895.

Mertz, C., Navarro-Serment, L. E., MacLachlan, R., Rybski, P., Steinfeld, A., Suppe, A., Urmson, C., Vandapel, N., Hebert, M., Thorpe, C., Duggins, D., and Gowdy, J. (2013). Moving object detection with laser scanners. *Journal of Field Robotics*, 30(1):17–43.

Miyasaka, T., Ohama, Y., and Ninomiya, Y. (2009). Ego-motion estimation and moving object tracking using multi-layer LIDAR. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 151–156.

Montesano, L., Minguez, J., and Montano, L. (2005). Modeling the Static and the Dynamic Parts of the Environment to Improve Sensor-based Navigation. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4556–4562.

Neira, J. and Tardos, J. (2001). Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on*, 17(6):890 –897.

Neubeck, A. and Van Gool, L. (2006). Efficient Non-Maximum Suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855.

Ngiam, J., Chen, Z., Bhaskar, S. A., Koh, P. W., and Ng, A. Y. (2011). Sparse Filtering. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 1125–1133. Curran Associates, Inc.

Oliveira, L. and Nunes, U. (2013). Pedestrian detection based on LIDAR-driven sliding window and relational parts-based detection. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 328–333.

Oliveira, L., Nunes, U., Peixoto, P., Silva, M., and Moita, F. (2010). Semantic fusion of laser and vision in pedestrian detection. *Pattern Recognition*, 43(10):3648 – 3659.

Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2002). Shape distributions. *ACM Trans. Graph.*, 21:807–832.

Paz, L. M., Tardos, J., and Neira, J. (2008). Divide and Conquer: EKF SLAM in O(n). *Robotics, IEEE Transactions on*, 24(5):1107–1120.

Pellegrini, S., Ess, A., Schindler, K., and Van Gool, L. (2009). You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268.

Premebida, C., Carreira, J., Batista, J., and Nunes, U. (2014). Pedestrian Detection Combining RGB and Dense LIDAR Data. In *IROS*.

Quigley, M., Batra, S., Gould, S., Klingbeil, E., Le, Q., Wellman, A., and Ng, A. (2009). High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2816–2822.

Ranzato, M., lan Boureau, Y., and LeCun, Y. (2008). Sparse Feature Learning for Deep Belief Networks. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 1185–1192. Curran Associates, Inc.

Rusu, R. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1 –4.

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2001). Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1665–1670 vol.2.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In *Neural Information Processing Systems*.

Spinello, L., Triebel, R., and Siegwart, R. (2010). Multiclass Multimodal Detection and Tracking in Urban Environments. *The International Journal of Robotics Research*, 29(12):1498–1515.

Stewart, A. and Newman, P. (2012). LAPS - localisation using appearance of prior structure: 6-DoF monocular camera localisation using prior pointclouds. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2625–2632.

Sun, Y., Ren, S., and Lin, Y. (2014). Object-object interaction affordance learning. *Robotics and Autonomous Systems*, 62(4):487 – 496.

Sung, K.-K. and Poggio, T. (1998). Example-Based Learning for View-Based Human Face Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):39–51.

Teichman, A., Levinson, J., and Thrun, S. (2011). Towards 3D object recognition via classification of arbitrary object tracks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4034 –4041.

Teichman, A. and Thrun, S. (2011). Tracking-Based Semi-Supervised Learning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski,

G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692.

Tipaldi, G. and Ramos, F. (2009). Motion clustering and estimation with conditional random fields. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 872–877.

Topp, E. and Christensen, H. (2005). Tracking for following and passing persons. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2321–2327.

Triebel, R., Shin, J., and Siegwart, R. (2010). Segmentation and Unsupervised Part-based Discovery of Repetitive Objects. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz, A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D. (2008). Autonomous driving in urban environments: Boss and the Urban Challenge. *Journal of Field Robotics*, 25(8):425–466.

van de Ven, J., Ramos, F., and Tipaldi, G. (2010). An integrated probabilistic model for scan-matching, moving object detection and motion estimation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 887–894.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of

simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1.

Vu, T.-D. and Aycard, O. (2009). Laser-based detection and tracking moving objects using data-driven Markov chain Monte Carlo. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3800–3806.

Vu, T.-D., Aycard, O., and Appenrodt, N. (2007). Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 190–195.

Wang, C.-C., Thorpe, C., and Thrun, S. (2003). Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan.

Wang, D., Posner, I., and Newman, P. (2012). What Could Move? Finding Cars, Pedestrians and Bicyclists in 3D Laser Data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*.

Wang, D., Posner, I., and Newman, P. (2013a). A New Approach to Model-Free Tracking with 2D Lidar. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, Singapore.

Wang, X., Yang, M., Zhu, S., and Lin, Y. (2013b). Regionlets for Generic Object Detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 17–24.

Westin, C.-F., Peled, S., Gudbjartsson, H., Kikinis, R., and Jolesz, F. A. (1997). Geometrical Diffusion Measures for MRI from Tensor Basis Analysis. In *ISMRM '97*, page 1742, Vancouver Canada.

Williams, S. B. (2001). *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, Australian Centre for Field Robotics, The University of Sydney.

Wolf, D. F. and Sukhatme, G. S. (2005). Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments. *Auton. Robots*, 19(1):53–65.

Yang, S.-W. and Wang, C.-C. (2011). Simultaneous egomotion estimation, segmentation, and moving object detection. *Journal of Field Robotics*, 28(4):565–588.

Yao, B. and Fei-Fei, L. (2012). Recognizing Human-Object Interactions in Still Images by Modeling the Mutual Context of Objects and Human Poses. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1691–1703.

Zahn, C. (1971). Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *Computers, IEEE Transactions on*, C-20(1):68 – 86.

Zhao, L. and Thorpe, C. (1998). Qualitative and Quantitative Car Tracking from a Range Image Sequence. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR '98, pages 496–, Washington, DC, USA. IEEE Computer Society.

Zhou, D. and Wang, J. (2011). Identification of deer in thermal images to avoid deer-vehicle crashes. In *Electronics and Optoelectronics (ICEOE), 2011 International Conference on*, volume 3, pages V3–342–V3–345.