

Experience-Based Object Detection for Robot Perception



Jeffrey Hawke
University College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2017

To Claire

Acknowledgements

I am exceptionally grateful to Prof. Ingmar Posner, my supervisor, for his guidance, academic support, and passion for robotics during my time at Oxford. I also thank Prof. Paul Newman for his input and continual drive to develop robotics at Oxford with Ingmar, growing the Mobile Robotics Group into what the Oxford Robotics Institute is today.

I have been supported by a number of postdoctoral colleagues during my DPhil: Chi Hay Tong, Alex Bewley, and Will Maddern. Research discussions were incredibly fruitful and a key factor in steering my research.

Thanks to my fellow students at MRG (now ORI): Matt Gadd, Geoff Pascoe, Corina Gurau, Chris Linegar, Raja Ouies, Markus Wulfmeier, Julie Dequaire, Ollie Bartlett, Dan Barnes, Paul Amayo, Michael Tanner, Stefan Saftescu, Martin Engelcke, and others. The time spent working together in the lab was a tonne of fun, and I learned from each of you.

Oxford is a very unique place to study, a large part of which is due to the collegiate system, sporting clubs, and cultural societies. I have made numerous friends at Univ, Hertford, UCBC, OUFC, the Christchurch Meadow touch rugby group, and more. Each of you has contributed to my experience at Oxford in your own unique way. In particular, I'd like to thank Eamon Byrne, Mandy Rojek, Graeme MacGilchrist, Cate Laporte-Oshiro, Jonathan Burley, Ali Cigari, Matthias Gerstgrasser, Jamie Bridges, Alex Lynchehaun, Alex Dyzenhaus, and the rest of the Univ MCR. Thanks also to Sir Ivor and Lady Crewe for helping shape Univ into the wonderful academic environment it is - I'm particularly fond of the Martlet Society talks.

A final special thanks to Claire, for your continual friendship, love, and support.

Jeff Hawke
October 2017

Abstract

A fundamental challenge in deploying vision-based object detection on a robotic platform is achieving sufficient perceptual performance for safe and effective operation. While general purpose object detectors have steadily improved in performance, we are still some way from being able to rely solely on purely vision-based robotic perception systems.

This thesis seeks to resolve this problem by exploiting a characteristic of computer vision unique to robotic applications, with a particular focus on pedestrian detection for autonomous driving. Robots - even mobile ones - typically have some inherent structure to the visual data they process. Firstly, they operate in the same environment (for example, regular commuting or errands with autonomous cars), and secondly, they are able to localise within that environment.

Place matters in robotics, as we can exploit this additional context to boost perceptual performance by constructing object detector models fitted to a mobile robot's place of operation. We demonstrate that, in an ideal scenario with ground truth labels, we can significantly improve the detection performance of lightweight object detectors by exploiting place. Our results suggest that a key factor limiting detection performance is the model capacity, suggesting that this approach could equally be applied to higher capacity models as the computational budget dictates.

This local expert detector is developed into a deployable, self-supervised system, using offline image segmentation and spatial heuristics to construct the detector models and a visual localisation system to retrieve them at run time. This approach boosts the perceptual performance of our lightweight object detector models.

Finally, this ensemble approach to local expert object detection is extended further with a neural network trained to generate detector models conditioned on the input image, an approach we refer to as a Dynamic Detection Filter Network. The network learns a representation of the operating environment, generating model parameters based on the input image. This offers a general approach to constructing place specific object detectors independently of localisation, with the potential to operate on larger scales with many different environments.

Contents

List of Figures	xi
1 Introduction	1
1.1 Robot Perception	1
1.2 Object Detection in Mobile Robotics	3
1.3 Exploiting Local Expertise	4
1.4 Contributions	6
1.5 Thesis Structure	8
2 Experience-Based Classification	11
2.1 Introduction	11
2.2 Background	14
2.3 Experience-Based Detection Framework	17
2.3.1 Object Detector Implementation	17
2.3.2 Spatial Scene Filter	25
2.3.3 Model Retraining	33
2.4 Self-Supervised Data Mining	34
2.4.1 Comparison to Hard Negative Mining	34
2.4.2 Fitting to Seasons	35
2.5 Fitting Models to Different Environments	37
2.5.1 Methodology	38
2.5.2 Results	40
2.6 Conclusions	45
3 What Makes a ‘Place’?	49
3.1 Introduction	50
3.2 Background	52
3.3 Bespoke Local Expert Detectors	55
3.3.1 Methodology	56
3.3.2 Object Detector and Training	57
3.3.3 Results	57
3.4 Local Expert Detectors using Image Appearance	67

3.4.1	Methodology	67
3.4.2	Results	68
3.5	Local Expert Detection Discussion	73
3.5.1	Higher Capacity Local Expert Detectors	74
3.6	EBC Deployment: A Local Expert Detection System	76
3.6.1	System Outline	76
3.6.2	Model Initialisation	77
3.6.3	Extending the Scene Filter: Robot Dreaming	79
3.6.4	Results	86
3.6.5	Discussion	94
3.7	Conclusions	95
4	Learning ‘Place’: Dynamic Detection Filter Networks	97
4.1	Introduction	97
4.2	Background	101
4.3	System Overview	104
4.3.1	Object Detector	105
4.3.2	Filter Generating Network	106
4.4	Evaluation Methodology	111
4.5	Results and Discussion	112
4.5.1	Detection Performance	112
4.5.2	Run-time Performance	112
4.5.3	Scene Encoding	114
4.5.4	Learning the static environment	121
4.5.5	Generalisation to Novel Routes	121
4.5.6	Model Sensitivity	126
4.6	Conclusions	126
5	Conclusions	129
5.1	Problem Statement	129
5.2	Experience-Based Classification	130
5.3	Local Expert Detector Models	131
5.4	EBC Deployment: A Self-Supervised Local Expert Detector System	132
5.5	Learning ‘Place’: Dynamic Detection Filter Networks	133
5.6	Summary Remarks	134
	References	137

List of Figures

2.1	The experience-based detector architecture.	18
2.2	A multi-scale sliding window detector pipeline.	19
2.3	Visualisation of the Histogram of Gradients (HOG) feature descriptor.	20
2.4	Visualisation of the Aggregate Channel Features (ACF) descriptor.	21
2.5	Benchmark performance for our <i>SVM+ACF</i> pedestrian detector and a range of publicly available pedestrian detectors on the <i>North Oxford January</i> driving dataset.	26
2.6	An illustration of our geometric scene filter.	27
2.7	Screen capture of the scene filter rejecting candidate detections.	28
2.8	The scale filter estimates detection heights by projecting candidate detections into the scene.	30
2.9	Object height estimates for true positives from a pedestrian detector trained on INRIA data, used to set spatial heuristic parameters.	32
2.10	Precision-Recall curves comparing ground truth HNM to self-supervised EBC.	36
2.11	Average precision for detectors fitted to seasonal data for a given operating route.	38
2.12	The robot vehicle (a) used to gather data along driving routes in Oxford (b).	40
2.13	Performance curves using an EBC framework to fit detector models to an operating route and season.	42
2.14	Aggregate detection performance metrics over successive training cycles on data from different environments.	43
2.15	Aggregate performance metrics while training a detector in a global scenario, training on all data rather than just the operating route.	44
2.16	Scene filter improvements on the initial and final <i>SVM+ACF</i> detector models trained using EBC.	46
3.1	Constructing bespoke environmental detector models.	56
3.2	Performance of an ideal case place dependent detector compared to detectors using generic models.	61

3.2	Performance of an ideal case place dependent detector compared to detectors using generic models.	62
3.3	Illustration: using a simpler (faster) classifier can mean we lack class separability. Using models fitted to subsets of the background class mitigates this limited capacity.	63
3.4	Variation in pose between localised images on subsequent laps will limit the efficacy of models fitted to a single frame.	64
3.5	Aggregate performance metrics for place dependent detector models (both spatial and appearance) as a function of N , the number of frames in the nearest neighbour training swathe.	66
3.6	Image appearance metrics can differ substantially with minor perspective change.	68
3.7	Image similarity metrics (GIST, Mutual Information) correlate with the distance travelled along the robot's trajectory.	69
3.8	Histograms showing image similarity (GIST, Mutual Information) based on distance between frames.	71
3.9	Aggregate performance metrics for bespoke environmental detector models using image appearance and spatiotemporal sequences. . . .	72
3.10	Detection performance improvements on the <i>MSCNN</i> detector models	75
3.11	Local Expert Detector System Architecture	78
3.12	Further scene filtering using offline image segmentation.	82
3.13	Image segmentation filter performance for a pedestrian detector model run on the labelled <i>North Oxford January</i> dataset.	83
3.14	An example of model initialisation data mining using the multi-stage scene filter.	85
3.15	A random sample of key errors made by the spatial heuristic scene filter on pedestrian detections from the <i>North Oxford January</i> labelled dataset.	87
3.16	A random sample of key errors made by the image segmentation scene filter on pedestrian detections from the <i>North Oxford January</i> labelled dataset.	88
3.17	A random sample of key errors made by the combined spatial heuristic & image segmentation scene filter on pedestrian detections from the <i>North Oxford January</i> labelled dataset.	89
3.18	Detector performance curves for the self-supervised experience-based detector.	90
3.19	Top scoring fifty false positives for pedestrian detectors run on the <i>North Oxford January</i> labelled dataset.	92
3.19	Continued: Top scoring fifty false positives for pedestrian detectors run on the <i>North Oxford January</i> labelled dataset.	93

3.20	Histogram showing the fraction of false positive detections belonging to a dynamic object class.	94
4.1	A Dynamic Filter Network. Reproduced from [105].	103
4.2	Dynamic Detection Filter Network architecture.	104
4.3	Performance curves for the baseline <i>SVM+ACF</i> detector implemented in Tensorflow.	106
4.4	Detector Filter Generating Network structure.	108
4.5	Detection performance for the DDFN on a test dataset.	113
4.6	DDFN scene encoding similarity matrices.	116
4.7	Route colour map used in the t-SNE plots.	117
4.8	t-SNE decomposition of the DDFN scene encoding.	118
4.9	A qualitative assessment of t-SNE reduction of the DDFN scene encoding.	120
4.10	Histogram showing the fraction of false positive detections belonging to a dynamic object class.	122
4.11	Oxford evaluation dataset route map.	123
4.12	Detection performance for the DDFN on a test dataset from a novel route.	125
4.13	Detection performance for the DDFN on the test dataset with added noise.	127

1

Introduction

Contents

1.1	Robot Perception	1
1.2	Object Detection in Mobile Robotics	3
1.3	Exploiting Local Expertise	4
1.4	Contributions	6
1.5	Thesis Structure	8

1.1 Robot Perception

Robotic systems are often described as having three key phases of operation: Sense, Plan, Act [1]. Robotic technology has evolved since this general approach was first described (and critiqued [2]), but the core tasks of any robot system architecture broadly encompass one or more of these three areas. A robot must sense the environment (using sensors such as cameras or lasers), perceive the scene from raw sensor data, then form a plan for what it is going to do. Finally the robot acts on this plan, sending the appropriate commands to the underlying control systems. Each of these components must work successfully for the robotic system to function as a whole.

A robot's sensing suite frequently consists of a number of different sensor

modalities, using a collection of passive and active sensors to infer something about the robot's internal state and the external environment surrounding the robot.

A large part of this sensing process (and the focus of a significant body of research) is perception: understanding the robot's environment from the raw sensor data. Robot perception includes tasks such as object detection (finding relevant objects around the object), scene segmentation (dividing the scene into categories to help the robot make better planning decisions), and localisation (recognising where the robot is located).

As autonomous robotic systems are becoming increasingly normalised in society, there is an ever increasing push towards robotic systems which work robustly in real world environments. Robots no longer only exist in research laboratories or large, automated factories. The idea of autonomous systems being responsible for transportation is well accepted [3, 4], with both automotive and technology companies pushing to be the first to deploy autonomous cars in urban environments. Senior automotive executives have publicly stated that this will be available for consumers from 2020 [5], with public policy challenges being a key obstacle facing increased automation in driving, arguably larger than the technological challenge [6]. Pilot studies in robotics for domestic use care are also becoming more common [7], including specialised robots to assist the elderly and people with impairments in daily living tasks [8, 9].

Enabling all these robotic applications (and more) are reliable perception systems. These systems are crucial to a robot's operation: the adage *garbage in, garbage out* holds true in robotic systems as with general computing, with the added complication that robots can move. Mistakes in a robot's perception propagate through the wider system, resulting in poor high level decision making. Mitigating these mistakes requires adding layers of complexity to the higher level motion generation systems, such as in [10].

Fundamentally, the better the perception system performs, the better other system components which depend on perception (and indeed the robot as a whole) will also perform.

1.2 Object Detection in Mobile Robotics

One key pillar of robot perception is object detection, which finds notable objects of various classes in the environment around the robot. These object classes are task dependent. In a warehouse this might be detecting conveyer belts or racking for pallets, whereas for autonomous driving applications this includes pedestrians, cars, buses, traffic lights, and more. Knowing information about key objects in a robot’s workspace is helpful: telling the difference between a pedestrian and a plastic bag blowing across the road will help a robot car’s planning system decide whether to slam on the brakes, or continue driving.

Object detection is commonly performed on laser scans (both 2D and 3D) or images. Pedestrian, cyclist, and car detection have been successfully implemented using 3D laser scanners (also called LIDAR) [11, 12], as well as using camera image data [13, 14]. Visual object detection is popular, a large part of which can be attributed to the low cost and prevalence of cameras compared to LIDAR. However, vision based detection has significant challenges associated with it.

All vision systems are subject to drastic appearance change, including image lighting, seasonal effects, and structural change in the scene. Addressing this challenge of varying image appearance has been a staple of robotic vision research, particularly in visual localisation [15]. Various methods have been proposed, including illumination invariant image representations [16–18], predicting appearance change to generate synthetic images [19], or designing visual localisation systems to store multiple representations of the same place under appearance change [20, 21].

In contrast, active sensors such as LIDAR which ‘touch’ the world do not typically suffer from these issues: a 3D point cloud of a scene is very similar regardless of time of day. They also have the benefit of consistent geometric measurements, a feature which has helped further object detection [22]. Nevertheless, the low cost and simplicity of fitting cameras to robots motivates using computer vision over laser sensing for many applications.

Underpinning all these methods is one critical, absolute requirement: a robot must be able to run this in real time. In this aspect, robotics diverges from computer

vision. Robots must be able to sense and perceive the environment sufficiently quickly to ensure safe operation.

The key problem facing visual object detection in robotics is how to achieve a high level of detection performance across the varied, changing environments a robot operates in, while also meeting this critical run-time performance requirement.

This thesis addresses this problem using an approach unique to robotics, as there are two factors distinguishing visual perception in robotics from general purpose computer vision:

1. Visual data in mobile robotics is temporally and spatially correlated, gathered sequentially as the robotic moves in its environment.
2. Robots - even mobile ones - typically operate in the same environment repeatedly, rather than discovering novel environments. An example of this is an autonomous car performing the same commute or regular trips around a neighbourhood.

These factors provide an opportunity to address both aspects of our key detection problem: mitigating the non-stationarity in visual data, and doing this in a manner which meets the computational efficiency needs of mobile robots.

The core message of this thesis is that we are able to improve detector performance under scene variation using fast, light-weight detector models, exceeding the detection performance of more complex, prohibitively slow state-of-the-art detectors. This is possible by exploiting the fact that robots typically operate in the same environment, building expert detector models fitted to a known operating environment.

1.3 Exploiting Local Expertise

Visual data in mobile robotics changes drastically due to a number of factors:

1. Lighting: shadows, time of day.
2. Season: bare winter trees, green leaves in summer.

3. Weather: overcast or sunny, rain or snow.
4. Scene structure: road works, changing billboards.

All of these collectively influence the performance of our visual perception systems. This non-stationarity (or constant change) is common-place in robotics, and must be managed by robotic systems seeking to operate outside a constrained environment.

In visual localisation, Churchill *et al.* proposed a radical shift in managing this change [20]. Rather than trying to have one model which we use everywhere, we accept that the world appears differently - even in the same spatial location - and we build a system which localises against all the different ways the robot experiences the world. In practice, if the robot can no longer localise to an existing topometric map as it is moving (for instance due to structural scene change from construction work), it begins to form a new branch. In doing so it keeps a separate parallel representation of the same spatial place.

We can draw on this idea for object detection: rather than using one common model everywhere, we use local expert models suited to the robot's operating environment. If we know our robot is driving through London, we need our object detection to perform exceptionally well there. Its detection performance in Tokyo or Berlin (or indeed off-road environments) is irrelevant. This argument is strengthened by the fact that the majority of vehicle trips tend to be repeated: commuting to work, driving to local schools, or running errands to the local shops [23–26]. This repetition means that it is advantageous to optimise all perception systems for the robot's place of operation, rather than use general purpose systems.

Through the addition of navigation data (using localisation systems running in parallel with object detection), we can use better informed visual models which are experts in their place of operation. These expert models can be constructed to suit not only the spatial location, but other environmental conditions specific to the robot's place of operation. Knowing where a robot is important, not only

in generating appropriate motion commands but also in that it can be exploited to improve the performance of other robotic systems.

This ensemble approach relaxes the usual requirement for generality in machine learning, opting instead to boost performance by using many expert detector models. In doing so we hope to improve detection performance in the places the robot operates using computationally lightweight models which satisfy the run-time performance requirements needed in robotics.

New computer vision approaches using deep learning have been particularly successful at object classification [27, 28], and have recently become viable for object detection. These tasks differ subtly: object classification typically assumes that an object of a known class exists in the image, and it occupies a sizeable proportion of the image area. In object detection this is frequently not the case. The object class we are trying to detect may or may not be present, and - particularly in the case of pedestrian detection - the objects may be very small. This has made pedestrian detection difficult for these deep learning methods [29], and it took a number of years before these methods began to top the various canonical benchmark leaderboards [30, 31]. The majority of work in this area has sought to achieve state-of-the-art performance, resulting in large, heavy-weight CNN detectors which achieve benchmark leading performance but often only run at 1 – 2 Hz [31]. In contrast, classical object detection methods (e.g., [32, 33]) have traditionally favoured run-time performance over absolute detection performance.

Our experience-based approach to object detection targets the computational efficiency of classical object detection methods for our local expert detectors, though equally applies to any detection system.

This approach is agnostic to the specific detector model employed, and we expect to be able to boost detection performance with more complex, high capacity CNN models using an experience-based approach.

1.4 Contributions

This thesis presents six contributions to object detection for mobile robotics:

1. We introduce the idea of experience-based object detection: using local expert detectors suited to a mobile robot's operating environment. This theme underpins the entire thesis.
2. We present a method for constructing object detector models using self-supervised data mining on unlabelled data based on spatial heuristics (Chapter 2), and extend this with image segmentation (Chapter 3).
3. We assess the merits of fitting detector models to known categories of operating environment: firstly to seasonal effects along a known route, and secondly to seasonal and spatial effects incorporating both route and season (Chapter 2).
4. We demonstrate an approach to building local expert detection models, boosting performance by relaxing the requirement by strongly fitting highly specialised detectors to a narrow window of operation (Chapter 3). This approach mitigates the impact of limited model capacity for lightweight detector models.
5. We design a system (Chapter 3) to scale this highly specialised expert detector approach to operate with unlabelled data, coupling object detection to an experience-based visual localisation system. Quantitative and qualitative results demonstrate the effectiveness of these models in their operating environment.
6. We propose a more extensible approach which decouples the expert object detector from visual localisation, instead relying on a deep learning approach which conditions the detector model on the input image (Chapter 4).

Some of the work presented here is taken from the following publications:

J. Hawke, C. Gurau, C. H. Tong, and I. Posner, "Wrong Today, Right Tomorrow: Experience-Based Classification for Robot Perception," in *Field and Service Robotics (FSR)*, Toronto, ON, Canada, 2015.

J Hawke, A. Bewley, I. Posner, "What Makes a Place? Building Bespoke Place Dependent Object Detectors for Robotics," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, 2017

C. Gurau, J. Hawke, C. H. Tong, and I. Posner, "Learning on the Job: Improving Robot Perception Through Experience," in *Neural Information Processing Systems (NIPS) Workshop on "Autonomously Learning Robots"*, Montreal, Quebec, Canada, 2014.

1.5 Thesis Structure

This thesis is structured in three main chapters, followed by concluding remarks. Each chapter has its own background section, and relevant literature review for the problem addressed by that chapter.

The theme of this thesis is constructing local expert object detectors suited to the data a robot experiences in its operating environment. Throughout the thesis we use pedestrian detection as our case study in object detection as this is a well researched problem applicable to autonomous robot driving, though none of the methods we use are specific to this task.

Chapter 2 outlines how we can construct these local expert detector models, and presents experiments demonstrating that we can fit detector models to a known route and operating season (categories of operating environment).

Chapter 3 extends this initial work, looking at how the resolution of these models affects performance: how narrow an operating environment should these be fitted to? We extend the initial, controlled experiments into a self-supervised detector system coupled tightly to a visual localisation system, enabling local expert detectors constructed with unlabelled data.

Finally, Chapter 4 presents a more extensible, general approach to experience-based object detection, learning an artificial neural network which conditions the detector model parameters based on input. This decouples the expert detector from a visual localisation system while retaining the same benefits.

This thesis concludes in Chapter 5. Overall, this thesis proposes and investigates how we as roboticists can leverage the knowledge of where the robot is to improve perceptual performance in the robot's operating environment.

2

Experience-Based Classification: Fitting Detectors to the Operating Environment

Contents

2.1	Introduction	11
2.2	Background	14
2.3	Experience-Based Detection Framework	17
2.3.1	Object Detector Implementation	17
2.3.2	Spatial Scene Filter	25
2.3.3	Model Retraining	33
2.4	Self-Supervised Data Mining	34
2.4.1	Comparison to Hard Negative Mining	34
2.4.2	Fitting to Seasons	35
2.5	Fitting Models to Different Environments	37
2.5.1	Methodology	38
2.5.2	Results	40
2.6	Conclusions	45

2.1 Introduction

Object detection forms one of the cornerstones of autonomous operation in complex, dynamic environments. Whether it concerns the detection of assets in an infrastructure survey, the detection of goods and co-workers in logistics, or the detection of other traffic participants in an autonomous driving context, object detectors need

to provide fast, reliable performance across a number of operating environments. This is explicitly encouraged in the machine vision community as witnessed by, for example, the ImageNet Large Scale Visual Recognition Challenge [34].

However, while much progress is being made, error rates of state-of-the-art approaches are still prohibitive, particularly for safety critical applications (e.g. [29, 31, 35] for the case of pedestrian detection). This is often due to a significant amount of variation in the negative class which, in reality, is not captured in the training data [36]. While it is relatively easy to obtain negative samples (selecting images which contain no examples of the target class), computational limits imply that we should only include ones that have a large effect on the decision boundary.

The standard method for obtaining relevant negative samples is known as hard negative mining (HNM) [37, 38], which is commonly used to bootstrap the underlying classifier used in an object detector. HNM is widely considered a mandatory part of detector training, where the classifier is first trained on the original training data and then used to perform object detection on a *labelled* dataset. False positives are identified using the ground truth labels provided and included for classifier retraining. This provides considerable improvement over the original detector, but the data used for negative mining strongly influences the resulting performance due to dataset bias [39, 40]. In addition, HNM does not account for non-stationarity in the data. As the robot moves, the data (and innate bias) changes with the operating environment and image appearance. As we know something about the operating environment (due to running localisation in parallel) we can exploit this bias and non-stationarity in image data with a detector model trained to suit the robot’s operating environment.

In order to improve performance for a particular application, scene context – obtained through online sensing or contained in (semantic) map priors – is commonly leveraged as a filter (e.g. [41, 42]). Typically, this takes the form of discarding detections as spurious if certain validation criteria are not met (e.g. a car needs to be found on or near a road [43]).

In this chapter we cover fitting a detector model to a robot’s operating environment by exploiting scene context to validate the detections obtained. This is a substantially different detector deployment model from the status quo, and provides the tools necessary for self-supervised and environment-dependent performance improvement over the lifetime of the detector. This reflects our desire for lifelong learning systems which excel in a robot’s specific application domain instead of providing mediocre performance everywhere.

This approach is, from one perspective, simple and straightforward. However, it brings significant benefits to our problem of embedding visual object detection in mobile autonomous systems, which is bounded by hard real-time constraints and high performance expectations. Some applications of embedded perception can afford to trade generality for specificity. Robotic agents should adjust to a subset of possible operating environments: the ones they operate in, or ‘experience’, on a daily basis.

Inspired by hard negative mining, we continue to train our detectors in a self-supervised manner by exploiting scene context from the robot’s operating environment. This is achieved by continuously feeding back into the training process of the detector any false positives identified by a validation step throughout the lifetime of the system. We refer to this process as Experience-Based Classification (EBC), where the detector’s classifier responds to the environment a robot experiences.

In effect, EBC automatically trains detectors for specific operating environments. While this may lead to overfitting to the background encountered, we argue that this is exactly what is required in mobile robotics where autonomous agents often traverse the same workspace over and over again. In fact, EBC relies on this behaviour and, inspired by work in the vision community such as [44], exploits similarities in geo-spatially related locations. Furthermore, the self-supervised, operational nature of this approach means that it can incorporate considerably more data in training than conventionally performing HNM on small canonical datasets. This opens up the possibility for life-long learning in robot perception, building up a collection of environment dependent object detectors over the robot’s lifetime.

The robot’s navigation system provides the key tool which enables this: the ability to localise and both build and retrieve experience-based detector models.

EBC is agnostic to the application domain, detection framework and object class considered. However, we frame the discussion in this chapter (and thesis as a whole) in the context of pedestrian detection for autonomous driving. In this chapter we utilise the fact that object detection is commonly performed alongside navigation and that current navigation solutions localise against a previously-acquired map [20, 45, 46]. This provides the scene context for EBC as we are able to draw on additional knowledge of the operating environment at runtime.

This chapter first outlines the basic building blocks of object detection, and describes how we can use scene context in robotics to fit detector models to a known environment using EBC, providing us with experience-based object detection.

With these tools in hand, we then investigate the efficacy of building experience based detector models. We compare the use of this self-supervised model fitting method to conventional Hard Negative Mining with our *SVM+HOG* detector model, then use the same method and model to fit the detector model to different seasons along the same operating route.

Building on this, we consider both the operating route and season when fitting a higher capacity *SVM+ACF* detector model to unlabelled driving data. In doing so, we demonstrate that we can adapt a generic detector model to suit the operating conditions experienced by our mobile robot.

2.2 Background

This thesis focuses on pedestrian detection as the target object class for two reasons. Firstly, pedestrian detection is a mature field of research with established methods. Secondly, as we have an interest in autonomous driving, pedestrian detection is one of the primary object classes our robot must detect during operation.

One significant work on pedestrian detection from monocular imagery proposed using a linear SVM classifier on Histogram of Oriented Gradients (HOG) features

[47]. The use of a linear model permits efficient sliding window computation [48] when a sliding window detector is implemented using this classifier.

More recent work in pedestrian detection improved performance through use of alternative feature types such as Aggregate Channel Features (ACF) [33], or alternative classifiers such as Latent SVMs with deformable parts models [38], and decision trees with Adaboost [32].

These classical sliding window methods have very recently been surpassed in performance by deep learning methods [29, 31], but this performance comes at a large computational cost, potentially prohibitive for many robotics applications. At the time of writing these typically require a dedicated, high end Graphics Processing Unit (GPU) card, and typically process images significantly slower than camera frame rates (2Hz, compared to common 20Hz or higher frame rates).

Various approaches have been investigated to boost the performance of object detectors by additional pre- or post-processing steps. A common approach is to incorporate additional information from the scene.

3D scene information has been primarily used in object detection to generate Regions of Interest (ROIs). For example, a ground plane computed from stereo imagery can provide a search space for detections (e.g. [41, 42]), or enforce scale [49]. Enzweiler et al. [50] extend this idea by maintaining a height-based representation of the local environment to generate ROIs, and Ess et al. [51] jointly infer the depth, ground plane and object detections. Instead of generating ROIs to present to our classifier, we invert the order and apply scene information after we compute detections. While both approaches provide us with a set of valid positive classifications, this ordering also allows us to obtain a set of informative negative data samples that can be used for detector improvement.

As mentioned in this chapter’s introduction, the conventional approach for obtaining these hard negatives when initially training a detector is Hard Negative Mining (HNM), performed on a labelled training dataset. Initially introduced by Sung and Poggio [37] as a bootstrap method for expanding the training set, Felzenszwalb *et al.* [38] tailor it for structural SVMs by defining ‘hard’ negatives as

examples that are incorrectly classified or within the margin of the SVM classifier. HNM has also been used for multiclass object detection [52], where positive samples of other classes can serve as hard negatives.

Instead of HNM, Hariharan et al. [53] suggested training Linear Discriminant Analysis (LDA) classifiers with an extremely large negative class. This was made possible for SVM classifiers by Henriques et al. [54], who used block-circulant decomposition to train with an approximation to the set of all negative samples from a series of images. In effect, training with a vast set of negatives reduces the need to specifically mine for hard negatives. While efficient, the training remains limited by computational resources, and does not escape the core requirement of labelled data.

This prior work on HNM for object detectors is complementary to our work. HNM still forms a critical step when initially training a detector, and we build on this to continue bootstrapping the detector to the robot’s operating environment. In addition, many of these techniques to extend HNM could equally be applied here.

In all these approaches, labelled data are used to identify negative samples. While the labelling effort may be tolerated for individual datasets, real-world operation is subject to variation from seasonal, lighting and environmental changes. This has a significant impact on detection performance, but manually labelling data for all of these scenarios is impractical for life-long learning in robotics. EBC is able to meet these requirements by identifying relevant samples in a self-supervised manner.

We share some similarities with the concept of group induction [55], where self-supervised training is performed by alternating between classifying unlabelled tracks and incorporating the most confident positive classifications in retraining. Our approach differs by the fact that we use an external signal in the form of an environmental prior to provide labels for the whole scene. This allows us to focus only on hard samples and provides a means to automatically train our detectors for specific environments.

2.3 Experience-Based Detection Framework

We want to be able to fit an object detector to suit the operating environment it is deployed in. To do this we need to be able to adapt an object detector model to data drawn from this environment.

This section outlines a general framework for building detector models fitted to the environment the robot experiences (shown in Figure 2.1), and our implementation of each component in the framework.

An experience-based detection framework augments a standard perception pipeline by introducing a self-supervised data mining module after the output of an object detector, gathering data from the environment through spatial heuristics. These data are stored, then used to augment training dataset for the detector. We focus solely on augmenting the training data with negative samples, as these are significantly easier than positive samples to collect using a self-supervised method.

This section first covers our implementation of an object detector in Section 2.3.1, with a specific focus on pedestrian detection. Section 2.3.2 explains the scene filter, and how it is used to perform self-supervised data mining. The final step in the process, the model update, is outlined in Section 2.3.3.

2.3.1 Object Detector Implementation

Approach

An object detector in computer vision processes a stream of images and produces candidate detections with an associated score (or probability measure). These detections are commonly represented as bounding boxes, or the coordinates of a rectangle which bounds the object in the image.

Most object detection methods to date use one of (or a combination of) three key approaches. The first approach finds interest points in the image, then compares patches from these interest points to the training set (e.g., Implicit Shape Models [56]). The second (and most common) approach is sliding window detection, in which a test window is exhaustively moved through the image, evaluating each image patch with a classifier [32, 33, 47]. The final approach uses region proposals: sub-regions

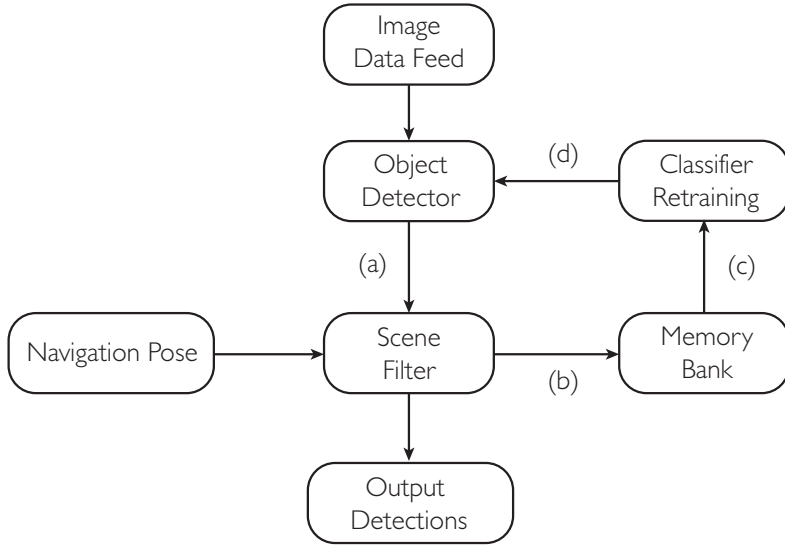


Figure 2.1: The experience-based detector architecture implemented in this work. (a) An object detector provides detections based on the image feed. (b) A scene prior is used to filter out detections that do not touch the ground plane or have an unexpected scale. (c) Rejected samples are stored. (d) The detector is retrained at the end of an experience using additional rejected samples. The wrapped detector improves through successive outings as it automatically adjusts to what it experiences.

of the image (or candidate detections) are presented to a classifier for evaluation [57]. A major benefit of the region proposal approach is reducing the computation time by limiting the number of patches to evaluate, enabling the use of more computationally complex classifiers. There is a secondary benefit: this can improve detector precision by reducing the number of tested image patches to only those which are more likely to contain the object. Recent object detectors using deep learning frequently use a region proposal approach [29–31], as do detectors which combine a sliding window detector with additional scene context to limit the search space [50].

We focus this work on an approach-agnostic framework for building experience-based detector models. As such, we heavily base our object detector models on existing prior work in [33, 58], with minor variations to suit our implementation requirements. For completeness we cover this detector implementation, though all aspects of the work in this thesis can be applied to any object detector.

Our object detector implementation is based on a classical multi-scale sliding window detector, illustrated in Figure 2.2. The input image is resized into an image pyramid based on a predefined set of scales. Each layer of the image pyramid (each

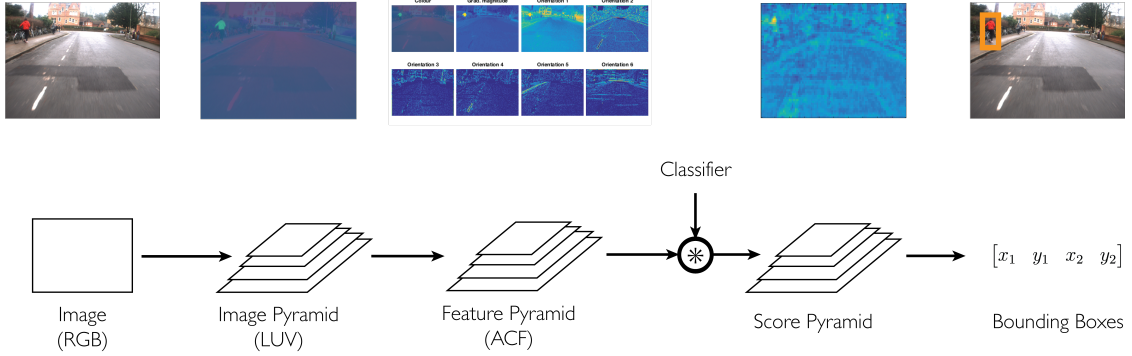


Figure 2.2: A multi-scale sliding window detector pipeline. The input image is first converted to the desired colour space (e.g., LUV [60]), then resized to a number of different scales in an image pyramid. The image pyramid is converted to a feature pyramid, computing a feature descriptor (e.g., ACF [33]) for each image in the pyramid. Each model-sized patch in the feature pyramid is then evaluated by a classifier (e.g., a linear SVM or Adaboost [61]), producing an output score pyramid. Bounding boxes are then produced for high scoring detection locations using the model size. These boxes are then pruned using *non maximum suppression*, favouring the highest scoring overlapping boxes.

a scaled copy of the input image) is processed by a feature computation function, converting the scaled image into an abstracted dense feature descriptor. The feature pyramid is then evaluated by a sliding window detector: we ‘slide’ a window of the right size for our object model around each layer of the feature pyramid, evaluating each patch with a discriminative classifier. For implementations using a linear classifier, this operation is simply the cross-correlation between the linear filter and the feature image. The classifier returns a score for each patch. Candidate detections are created by thresholding each score image to find high scoring image patches, rescaling the detections to map these back to the original image size. Finally, non-maximum suppression is used to filter out overlapping detections across all scales [59]. This is a greedy algorithm which sorts the detections based on scores, sequentially suppressing detections which sufficient overlap any higher scoring bounding boxes.

Feature Descriptor

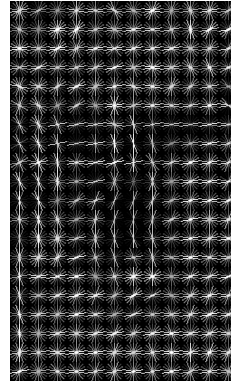
The feature descriptor used is a major influence on the object detector performance, accounting for much of the improvement in performance over the past decade

(Benenson *et al.* [61]). We implemented two dense feature descriptors: Histogram of Gradients (HOG) [47], and Aggregate Channel Features (ACF) [33].

The Histogram of Gradients was first proposed by Dalal and Triggs [47] for use in pedestrian detection in static imagery, though it has been applied to many other tasks since. HOG computes the image gradients using simple $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ filter kernels. The gradients in a group of pixels (commonly a cell of 8x8 pixels) are then binned based on their orientation. These cells can be rectangular or circular, and are evenly spread over a range of $0 - 180^\circ$ (an unsigned descriptor) or $0 - 360^\circ$ (a signed descriptor). We use unsigned rectangular bins, as in the original work for pedestrian detection. The contribution of the gradient for a given pixel to the histogram is linearly weighted between the two closest bins to the pixels gradient. The descriptor is then block normalised: rather than normalise each histogram individually, blocks of 2x2 cells are grouped and stacked into a single vector v , dividing v by its l^2 -norm: $\hat{v} = v / \sqrt{\|v\|_2^2 + \epsilon}$, where ϵ is a very small number to prevent dividing by zero.



(a) RGB image.

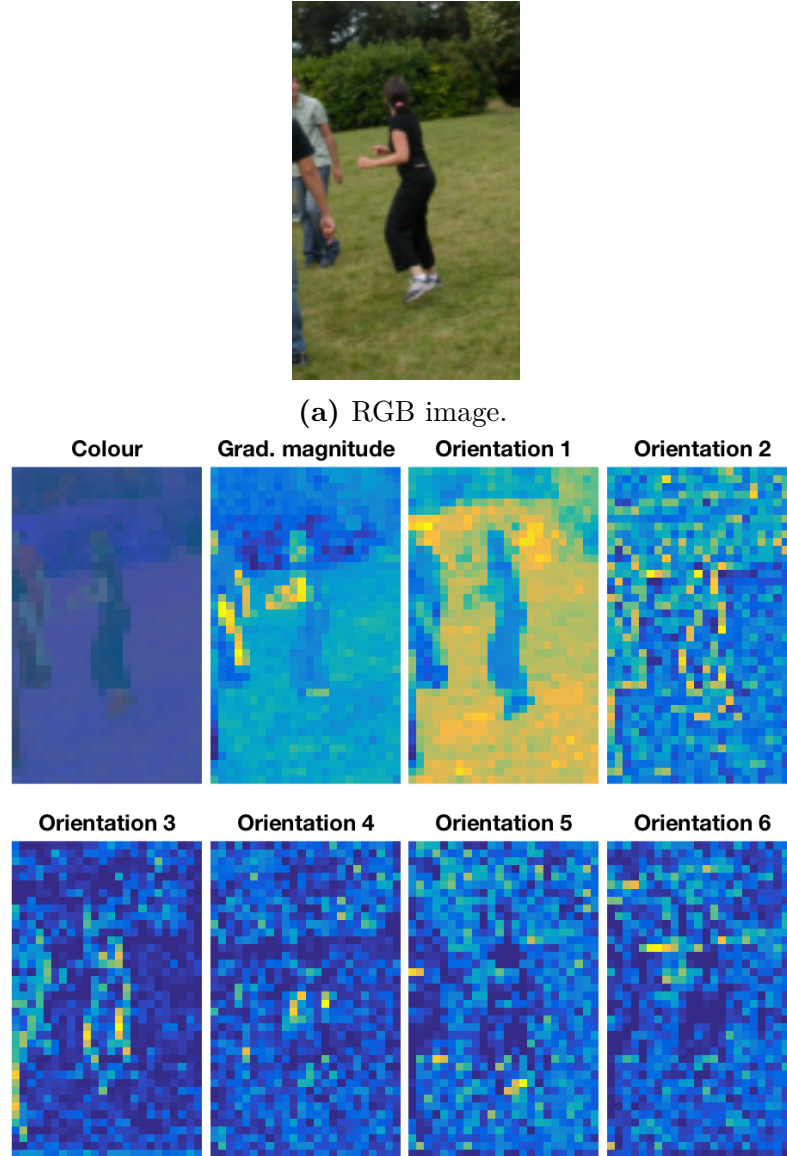


(b) HOG descriptor visualisation.

Figure 2.3: Visualisation of the Histogram of Gradients (HOG) feature descriptor for one of the training samples from the INRIA pedestrian dataset [47]. The intensity of the greyscale lines in each cell indicate the strength of response for that orientation.

Aggregate Channel Features are based on the channel features [62], consisting of a number of colour channels, gradient magnitude, and gradient orientation (illustrated in Figure 2.4). The gradient orientation channels are computed similarly to HOG, but channel features have fewer orientation bins. There have been many variants of

channel features (e.g., [32, 63]), with subtle variations in which colour channels are used and the way the features are grouped. ACF aggregates the channel features over a small cell window. As with HOG, this was introduced for the problem of pedestrian detection but has been applied to other object detection tasks successfully [64, 65].



(b) ACF descriptor visualisation.

Figure 2.4: Aggregate Channel Features (ACF) visualisation for one of the training samples from the INRIA pedestrian dataset [47]. The colour channels here are CIELUV.

Classifier

The classifier used in a sliding window detector can vary, though commonly this is either a linear Support Vector Machine (SVM) or boosted decision tree (e.g., Adaboost). Wojek *et al.* argued that there is little difference between these two common classifiers provided the dimensionality of the feature space is great enough [66]. Linear approximations to nonlinear SVMs have been tested [67], though this typically comes at a very large computational cost with only a small improvement in performance. Our detector uses a linear SVM as the underlying classifier, primarily due to the ease of implementation and run-time performance on general purpose hardware.

Benchmark Performance

We compare this detector on a 2km urban driving dataset gathered in Oxford, along with three publicly available pedestrian detectors in Figure 2.5. This labelled dataset - referred to as *North Oxford January* - consists of two laps of the same route and is used throughout this thesis as a benchmark. The objective here is to verify our implementation, and show commensurate performance with existing similar approaches. The first detector compared is *DPM* from Felzenswalb *et al.* [38] is the original deformable parts model detector, using HOG features with a Latent SVM trained on INRIA pedestrian data [47].

Our detector (*SVM+ACF*, as in Chapter 3) performs comparably to Dollar’s *FPDW* detector [33], using the same ACF feature descriptor but a linear SVM classifier rather than *FPDW*’s boosted decision tree. Both detectors are fast, able to process images at the camera frame rate of 20Hz given a reasonable selection of image scales. We use the author provided model trained on INRIA data.

Finally, we also assess our object detector against the KITTI benchmark [68] leader at the time of writing: *MSCNN* [31]. This multi-scale convolutional neural network object detector was trained on the Caltech pedestrian dataset [63] as INRIA is a cropped dataset which can’t be used for training this type of detector. This is an extremely high capacity neural network detector, with over 80 million

parameters. As a result it requires a high performance NVIDIA GTX graphics card to run, and can only run at around 2Hz.

We acknowledge that these benchmark results will be influenced by the different data used in training *MSCNN*. INRIA is widely recognised as a diverse training dataset which offers slightly better performance under domain transfer than other common training datasets such as Caltech or KITTI [36]. Unfortunately, there is no way to avoid this as *MSCNN* and similar detectors require whole images in training rather than cropped patches.

Run-time performance requirements will always be task dependent for robotics. For the mobile robotics applications we are interested in (e.g., pedestrian detection for urban driving) we would ideally like our object detectors to be able to run in excess of 20Hz. This is largely based on the speed of the robots: a car moving at $50kmh^{-1}$ will cover almost 14 metres in a second. A 2Hz frame rate would mean the robot will move 7 metres between captured images - far too significant a risk for robot safety. Our requirement for such a high run-time performance from the detector means that operational performance is then limited by the frame rate of the cameras used. Our platforms use Pointgrey Bumblebee2 or Bumblebee3 cameras, which record images at 20 and 16 Hz respectively.

Throughout this thesis we compare detection performance using Precision - Recall and Miss Rate - False Positives Per Image plots. We also use a number of aggregate metrics which help quantify the difference between detector models. These metrics are: average precision, log-average miss rate, and the maximum F1 score. These metrics for the benchmark detectors are listed in Table 2.1.

To generate these metrics, we perform data association between ground truth annotations and detections in the same manner as [69], where we check if the area of overlap between the ground truth bounding box BB_{gt} and a detection BB_{dt} is greater than a threshold $\psi \in [0,1]$. The area of overlap is defined by calculating the *intersection-over-union* of the two bounding boxes. Commonly, ψ is set to 0.5. Detection bounding boxes with an area of overlap greater than ψ to any ground truth boxes are given a positive label, with all others a negative label.

$$a_{overlap} = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} \geq \psi \quad (2.1)$$

With the candidate detections labelled, we can then compute a number of performance measures from binary classification. Each of these is based on the number of true (and false) positive (and negative) classifications. These detection counts are referred to as TP (true positive count), FP (false positive count), TN (true negative count), and FN (false negative count).

Precision measures how often a detector is correct when it returns a positive candidate detection.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

Recall, or the *true positive rate*, measures how many of the true detections were picked up by the detector.

$$TPR = \frac{TP}{TP + FN} \quad (2.3)$$

Together, these are used to plot a Precision - Recall curve from a list of candidate detections as the detection score used to assign positive and negative labels varies.

The first aggregate metric, Average Precision (AP), is simply the area under a Precision - Recall curve, defined as:

$$AP = \int_0^1 p(r) dr \quad (2.4)$$

where r is the recall and $p(r)$ the precision at a given recall value.

The *log-average miss rate* is a similar metric to average precision, proposed initially by Dollar *et al.* in [69]. Object detection is a variation on binary classification and using per-image metrics rather than per-window can give a better understanding of performance. This metric is computed from a Miss Rate - False Positives Per Image plot. The *False Positives Per Image* (or FPPI) used as the independent variable is self-descriptive: the average number of false positives per image. The miss rate (or *False Negative Rate*) is defined as:

$$FNR = \frac{FN}{FN + TP} \quad (2.5)$$

$$= 1 - TPR \quad (2.6)$$

This *log-average miss rate* metric computes the average miss rate at nine FPPI rates, evenly spaced in log-space in the range 10^{-2} to 10^0 .

Finally, the F-score is the weighted harmonic mean of precision and recall, adjusted by the parameter β . For the F-score (formally F1-score), $\beta = 1$, favouring precision and recall equally. We report the maximum F-score for a detector as an aggregate metric, which corresponds to the knee of a Precision - Recall curve.

$$F_{\beta} = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}} \quad (2.7)$$

Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
DPM [38]	0.201	0.698	0.285
FPDW [33]	0.481	0.440	0.491
MSCNN [31]	0.588	0.371	0.579
SVM+ACF	0.483	0.485	0.473

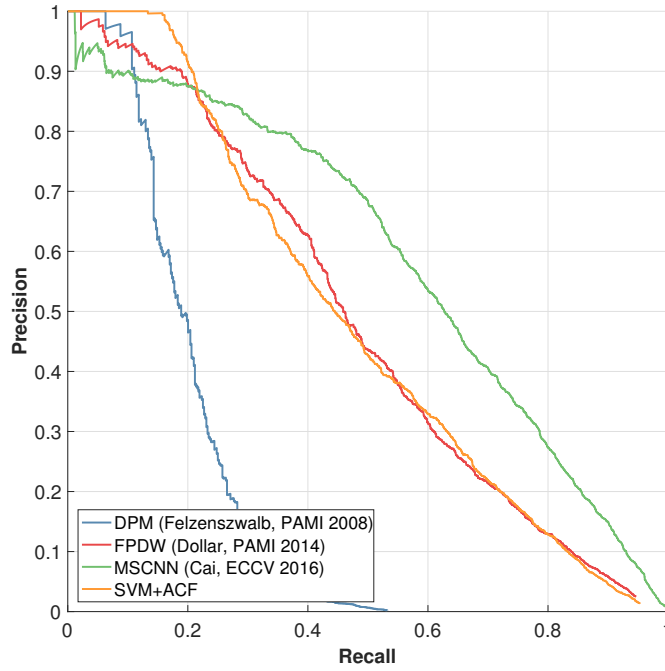
Table 2.1: Aggregate performance metrics for detection benchmarks. A higher number is better for Average Precision and F1 Scores, lower is better for Log-Average Miss Rate.

2.3.2 Spatial Scene Filter

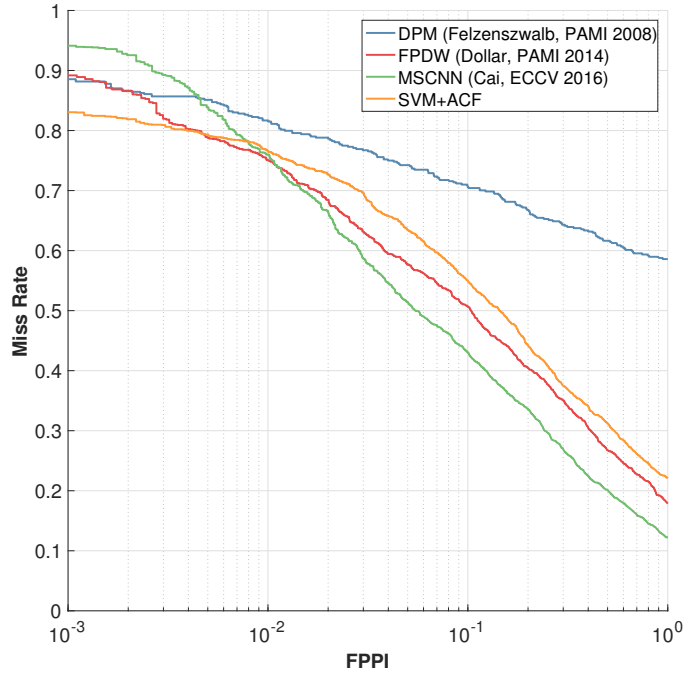
Candidate detections from this object detector are then processed by the scene filter in the second stage of this self-supervised model fitting framework, outlined earlier in Figure 2.1. Our approach to this filter uses spatial heuristics, as it is possible to define an explicit set of heuristics for our desired object class (pedestrians).

Given a set of detections, the scene filter employs local context to filter out false positives according to strong heuristics. Accepted detections are passed through the filter to the object detection pipeline’s output, while rejected detections are filtered and stored. Since the rejected samples are detections that scored highly in the previous step these are by definition ‘hard’ negatives.

Using localisation information and a 3D scene prior, we first look up the local ground plane for our current location, then project this ground plane into the image, illustrated in Figure 2.6. This is used by a first filter, which rejects detections that lie off the ground plane (above the horizon) for the current navigation frame.



(a) Precision - Recall



(b) Miss Rate - False Positives Per Image

Figure 2.5: Benchmark performance for our *SVM+ACF* pedestrian detector and a range of publicly available pedestrian detectors on our labelled *North Oxford January* driving dataset, consisting of two *1km* loops through North Oxford. We use this dataset throughout this thesis. This detector is very similar in operation to FPDW, and achieves similar performance.

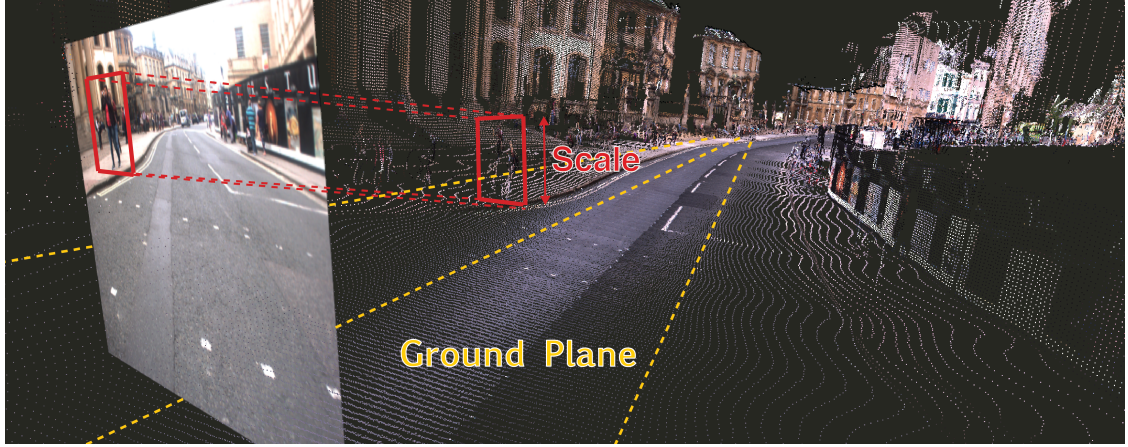


Figure 2.6: An illustration of our geometric scene filter. Given localisation information and a 3D scene prior, we first project the local ground plane into the image. This is used by first filter, which rejects detections that lie outside of a ground polygon. Our second filter then projects each remaining bounding box into the 3D scene to ensure detections are of viable heights.

Our second filter then projects each remaining bounding box into the 3D scene to ensure detections are of a viable scale. An example of the resulting output of these filters is shown in Figure 2.7.

Ground Plane Estimation

Mobile robots frequently have a map of their operating environment which they can localise against. We use a 3D scene prior, a pointcloud collected by driving a planar laser scanner through the environment. An example of this pointcloud can be seen in Figure 2.6.

From this scene prior we estimate the ground plane for short (50m) segments of the map using *Maximum Likelihood Estimation Sampling Snd Consensus* (MLESC) [70]. This variant of the *Random Sampling and Consensus* algorithm (RANSAC) maximises the likelihood of the data fitting the model, rather than simply the number of inliers.

Each ground plane is defined in Hessian normal form in the robot frame, where $\hat{\mathbf{n}}$ is the plane’s unit normal and p is the distance to the plane along the normal from the origin. The vector $\mathbf{x}$ represents a valid point on the plane, where $\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}$.



Figure 2.7: Screen capture of the scene filter rejecting candidate detections. Orange boxes are rejected by the horizon filter, whereas purple boxes are rejected by the scale filter.

$$\hat{\mathbf{n}} \cdot \mathbf{x} = -p \quad (2.8)$$

Horizon filter

The horizon is computed by picking two points h_1 and h_2 far ahead of the robot on the ground plane, then projecting these back into the image. The line defined by these two points is used as the horizon estimate.

We select the two horizon points h_1 and h_2 by choosing $x = 1km$ (forward direction in the robot frame) with $y = \pm 50m$ (left/right from the robot's perspective), solving for z using the plane equation.

These are then projected from the robot's coordinate frame to camera uv coordinates using the intrinsic calibration matrix K and a pinhole model for the camera.

The intrinsic matrix K is parameterised by Hartley and Zisserman [71] as:

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.9)$$

where f_x and f_y are the focal length (identical with a pinhole camera model), s is the axis skew, and x_0 and y_0 are the principal point offset.

Using K and the extrinsic transform G_{rc} (from robot frame to camera frame) we can transform the horizon from robot frame $\begin{bmatrix} x & y & z \end{bmatrix}$ to camera frame $\begin{bmatrix} x_c & y_c & z_c \end{bmatrix}$, then project these into the image $\begin{bmatrix} u & v \end{bmatrix}$.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad (2.10)$$

The horizon is then computed as the line formed between the two points in image coordinates, parameterised by the gradient m_h and intercept c_h .

$$m_h = \frac{v_{h2} - v_{h1}}{u_{h2} - u_{h1}} \quad (2.11)$$

$$c_h = v_{h2} - m_h u_{h2} \quad (2.12)$$

The horizon filter is then simply a binary test using the centre $\begin{bmatrix} u_c & v_c \end{bmatrix}^T$ of the lower edge of the candidate detection (where a detection is defined as a bounding box $\begin{bmatrix} u_1 & v_1 & u_2 & v_2 \end{bmatrix}$): if this is above the horizon, the detection is rejected. The detection location $\begin{bmatrix} u_c & v_c \end{bmatrix}^T$ is defined by:

$$u_c = \frac{u_2 - u_1}{2} \quad (2.13)$$

$$v_c = v_2 \quad (2.14)$$

Therefore the filter is computed as:

$$f_{horizon} = \begin{cases} 0 & \text{if } m_h u_c + c_h > v_c \\ 1 & \text{if } m_h u_c + c_h \leq v_c \end{cases} \quad (2.15)$$

Scale filter

To estimate an object's height, the second stage of the scene filter makes the assumption that the object is located on the ground plane and is oriented orthogonally to the ground plane. For most of the object detection tasks we are interested in for

mobile robotics (pedestrians, cars, etc.), this is a reasonable assertion. For other classes (and robot tasks) other heuristics may be required.

For each candidate object detection we define the detection plane: a plane orthogonal to the ground plane defined by the projection of the two lower corners of the candidate bounding box onto the ground plane and the ground plane's normal.

We can then estimate the height of the candidate detection by projecting the centre of the bounding box's upper and lower edges onto this detection plane, taking the l^2 -norm of the difference vector between these projected points as the estimate of our candidate object's height. This process is illustrated in Figure 2.8.

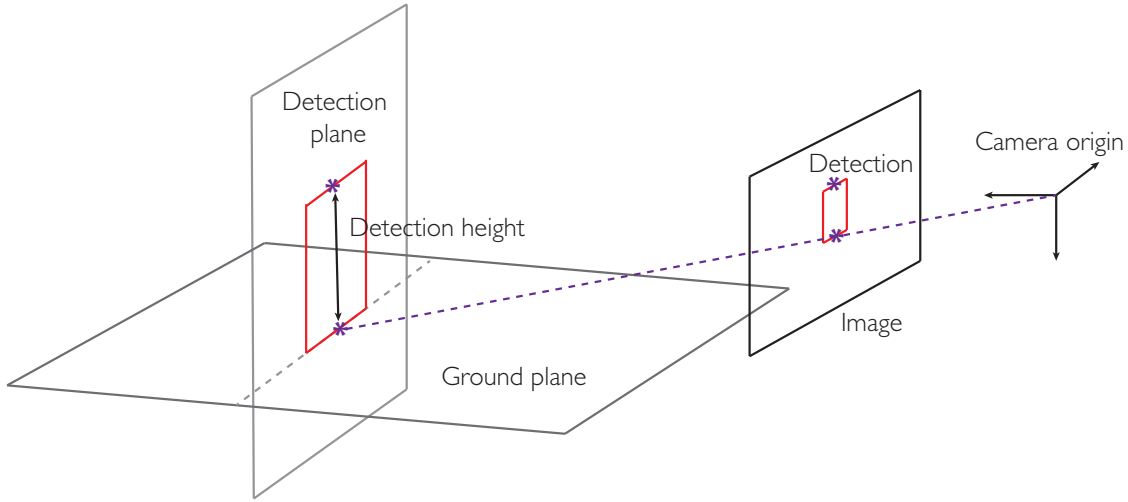


Figure 2.8: The scale filter estimates detection heights by projecting candidate detections into the scene.

Firstly, the object intersection with the ground plane is assumed to be located at the lower edge of the bounding box.

Using the pinhole camera model in 2.10 we find the equation of the line L_c from the camera origin to a point in the image $\begin{bmatrix} u & v \end{bmatrix}$.

$$L_c = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (2.16)$$

We can now find the intersection between this line and the ground plane, after first converting it from the camera frame c to the robot frame r using the transform matrix G_{rc} .

A line is parameterised by a direction vector $L = [x \ y \ z]^T$ and a point on the line $L_0 = [x_0 \ y_0 \ z_0]^T$. A point P on the line L is defined by the equation

$$P = dL + L_0 \quad (2.17)$$

where d is a scalar distance along the line.

The ground plane is defined in Hessian normal form 2.8, comprised of the unit normal vector $\hat{\mathbf{n}}$ and a scalar distance p . We pick an point P_0 on the plane. This can be any valid point on the plane; we select one by following the z axis.

The intersection of a line with a plane can be calculated by substituting the equation for P into the plane definition $n \cdot (P - P_0)$ and solving for d , where n is the plane normal and P_0 the plane origin.

$$d = \frac{(P_0 - L_0) \cdot \hat{\mathbf{n}}}{L \cdot \hat{\mathbf{n}}} \quad (2.18)$$

With d we now have the intersection of the line and the plane by substituting this back into the original line equation 2.17.

We can compute two points on the ground plane corresponding to the lower corners of the detection bounding box. The detection plane is defined by these two points, and a third point by stepping along the ground plane normal $\hat{\mathbf{n}}$. This detection plane is orthogonal to the ground plane.

The detection height is then estimated by computing the intersection of detection plane and the vectors to the centre of the upper and lower edges of the bounding box. This yields two points d_1 and d_2 in 3D space, and the detection height estimate is then simply the l^2 -norm of the difference between these points.

$$h = \sqrt{\|d_1 - d_2\|_2^2} \quad (2.19)$$

The scene filtering step should be conservative to avoid rejecting a valid detection (true positive), which may lead to semantic drift [72]. The goal of this filtering component is to reduce the number of false positives while not introducing false negatives.

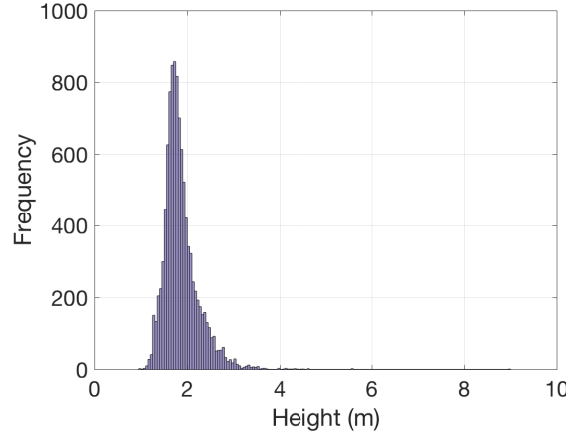


Figure 2.9: Object height estimates for all true positives from a pedestrian detector trained on INRIA data. We use the 1st and 99th percentile values from this to set a reasonable acceptance range for the scale filter (1.3m through 3.1m). This approach encapsulates the many sources of error which influence the height estimate: detection localisation, box padding, natural variance in the object height, and noise in the ground plane estimate.

The filter heuristics have only two parameters: the lower and upper bounds for the expected object height in the second scale step. Rather than try to estimate these based on real-world properties (e.g., population statistics for pedestrians) and task specific parameters (e.g., noise in ground plane estimates, poorly localised bounding boxes, training image patches being larger than the actual object), we instead consider the distribution of estimated object height for a large number of ground truth labels for our desired task. This data centric approach encapsulates all these sources of variance without trying to explicitly model or separate them. We collated all true positive detections from our labelled *North Oxford* data for a pedestrian detector trained on INRIA, and set the scale filter bounds to the 1st (minimum) and 99th (maximum) percentiles of the estimated object height. For this pedestrian detector model, the acceptable range of height estimates is to 1.3 m and 3.1 m (shown in Figure 2.9). This coarse range is very loose, but ensures that the filter is overly conservative and accounts for the various sources of error.

The second filtering stage (scale) is simply a check that the estimated detection height h_{det} lies in these bounds.

$$f_{scale} = \begin{cases} 0 & \text{if } h_{det} < h_{min} \\ 1 & \text{if } h_{min} \leq h_{det} \leq h_{max} \\ 0 & \text{if } h_{max} < h_{det} \end{cases} \quad (2.20)$$

Combined filter

We then combine the two filters, requiring a detection d to pass both spatial heuristic checks to be considered *validated*.

$$f_{scene}(d) = f_{horizon}(d) \wedge f_{scale}(d) \quad (2.21)$$

2.3.3 Model Retraining

The final component of this self-supervised detector fitting framework constructs the experience-based detection model by augmenting the original training set with the rejected environmental data from the scene filter then retraining the classifier model. Since these additional negatives are obtained during operation, each subsequent training cycle further adapts the classifier to the specific operating environment.

The training method for the augmented dataset is identical to the original training pass used for the detector model. We select the same hyperparameters (a single optimisation parameter C for linear SVMs, initially set by cross validation on the general training set), and we ensure class balance during training by weighting the negative data:

$$w_p = 1 \quad (2.22)$$

$$w_n = \frac{N_p}{N_p + N_n} \quad (2.23)$$

The optimisation parameters are set initially through cross-validation on the training set, selecting appropriate values to ensure the model generalises to the variance of the training set. We maintain this value as we assume that the variance of the augmented training set will not be substantially different to the non-augmented training set. We also ensure that class balance is maintained as we wish to prioritise type I and II errors equally, even as we expand the data in the negative training set.

2.4 Self-Supervised Data Mining

We evaluate the capability of this self-supervised data mining method to fit a classifier model to data, firstly directly comparing this method to HNM on a single labelled dataset, then applying this same method to fit detector models to unlabelled data from different seasons on a known route.

The object detector used in this section’s experiments is the detector in Section 2.3.1 using HOG features. The detector’s linear SVM classifier was trained using LIBLINEAR [73] and OpenCV [74] on the Daimler Pedestrian Detection Benchmark Dataset [75]. We performed ten-fold cross validation for parameter selection on the training set, consisting of 52,112 positive and 32,465 negative cropped samples. The base detector model has no hard negative mining performed on it as we wish to assess data mining images from the robot’s operating environment.

To evaluate our self-supervised data mining approach we require data from multiple traverses of the same operating environment. We also require labelled data from this operating environment, hence we must use our own datasets rather than canonical computer vision datasets. We used nine different datasets gathered with a Bumblebee2 stereo camera mounted on our vehicle driving in an urban environment. These datasets consisted of multiple runs of the same *North Oxford* route split into four runs per route for training, and one for testing. They are referred to in this section as *Urban Driving January* and *Urban Driving May*. The labelled test set is a lap of this route driven in January, referred to as *North Oxford January*. In both datasets, we used only the left stereo image with a capture rate of 20Hz. This provided a total of 14km of unlabelled training data and 2km of labelled test data. The datasets are summarised in Table 2.2.

2.4.1 Comparison to Hard Negative Mining

The model retraining procedure in Section 2.3.3 is analogous to typical Hard Negative Mining (HNM), where a detector model iteratively mines high scoring false positives from a set of labelled negative images. HNM improves a detector’s performance by improving precision, retraining the model on an augmented training

set consisting of the original training data plus the high scoring false positives. Low scoring ‘easy’ negatives can be dropped from the training set over successive HNM cycles, though we do not do this.

We first compare our self-supervised method to HNM, to see if we can achieve comparable improvement without the need for labelled data. Since HNM requires ground truth labels, we split the labelled *Urban Driving January* test set into two parts, with 70% allocated for training and 30% for testing. Figure 2.10 shows the six training iterations performed for each method and the raw detector performance without scene filtering.

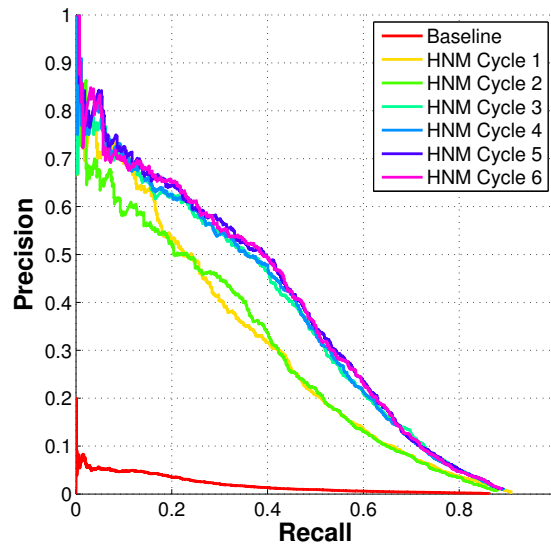
While the classifier performs well on the training set, we see that the baseline detector performance in Figure 2.10 using this classifier is poor. A large part of this is due to the negative training set not being sufficiently representative of the background class in our test data. Sliding window detectors present tens of thousands of windows to the classifier per image, with the majority being negative. In effect, the training set is not representative of the test data. This issue is well known in object detection and is typically addressed through Hard Negative Mining (HNM).

As expected, we see a dramatic improvement in performance through successive HNM training cycles. We also see that self-supervised data mining using EBC yields similar performance to HNM. This demonstrates that using scene context is a useful practical alternative to using manually labelled data when training an object detector.

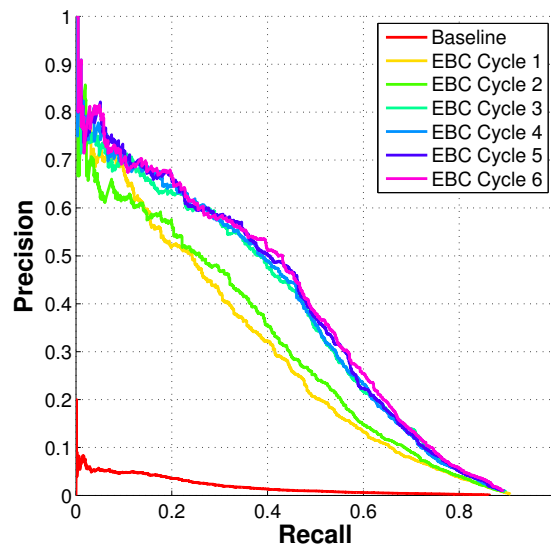
2.4.2 Fitting to Seasons

Changes in image appearance can drastically affect the performance of computer vision algorithms, and addressing this has been heavily researched in robotics, with a particular focus on visual localisation [16, 76, 77]. We expect the appearance change due to seasonal effects to impact an object detector’s performance accordingly.

We therefore seek to investigate whether object detector models fitted to a given operating route are able to capture this seasonal variation.



(a) Hard Negative Mining (using ground truth).



(b) Self-supervised data mining (using spatial heuristics).

Figure 2.10: Precision-Recall curves comparing HNM to EBC training on a segment of the Urban Driving January dataset. There is a dramatic improvement after the first iteration, and similar performance from both approaches. This demonstrates the value of environment-specific tuning and that EBC is a practical alternative to HNM.

We use the same detector as earlier in Section 2.4, using the scene filter to mine data from a common urban driving route. The unlabelled training data is organised into two seasonal categories: *January* (Winter) and *May* (Spring) in Table 2.2. Each training category consists of four repeats of the same driving route, and we perform four passes of self-supervised training on each seasonal category. We evaluate the performance of this detector after each training cycle on the labelled training dataset from *January*.

	Urban Driving January			Urban Driving May		
	Average per training cycle	Total training	Total testing	Average per training cycle	Total training	Total testing
Kilometers	2.11	8.45	1.99	1.35	5.40	-
Images	9651	38605	9155	5680	22720	-
Minutes	8.05	32.2	7.63	4.73	18.9	-

Table 2.2: A summary of the *North Oxford* route datasets used for fitting detector models to seasons. We use one fully annotated *North Oxford January* dataset for evaluation.

Figure 2.11 shows the detector performance as training progresses. We see a large performance gain which is common to both training datasets, indicating that there is some shared task or route bias to the data which is mutually captured.

We see a further boost in performance when training on unlabelled data from the same season as the test dataset. This demonstrates that we can indeed increase an object detector’s performance by capturing seasonal effects on image appearance.

2.5 Fitting Models to Different Environments

Section 2.4 demonstrated the capability of self supervised data mining as a tool for building object detectors fitted to a known environment. We also see that it is possible to improve performance by fitting the model to unlabelled data drawn from the same season as the test environment. However this raises further questions: can we fit to a given route, rather than just appearance change along a route?

This section expands the work in the previous section on fitting object detectors to an operating season, by fitting to different operating routes in addition to season.

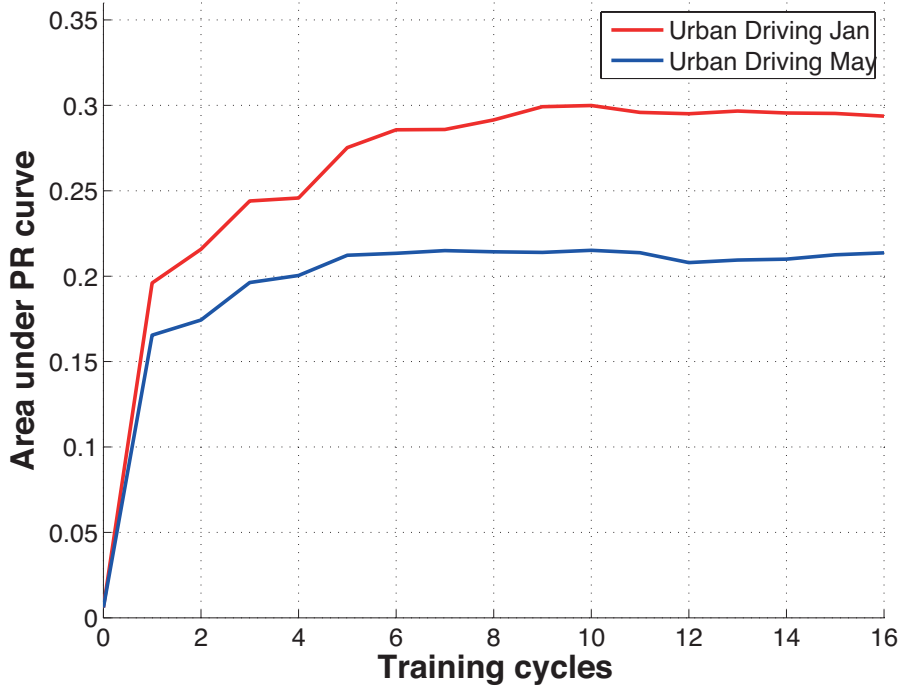


Figure 2.11: The average precision (area under PR curve) for EBC classifiers trained on unlabelled seasonal data (January, May) from a common operating route, tested on labelled data from January. This demonstrates that we can fit models to our driving route in a self-supervised manner. We also note that there is a seasonal effect to the performance of our detector after training. Both models have been trained on scene data from the same route, however, the one trained on driving data from the same season as our test dataset performs better. This implies an underlying difference in the image data in driving this route at different times of year.

2.5.1 Methodology

We seek to evaluate the implications of an object detector learning from the environment it experiences. We do this by taking a common baseline detector model, then use this to train separate detectors for different classes of data, comparing their performance on test datasets from these same classes. To do this, we put a baseline pedestrian detector through successive training cycles on urban driving datasets gathered from two different routes in Oxford at different times of year. We anticipate that the detector which has learned from operating data which most closely matches the test data (place and season) will perform the best, as the detector becomes fitted to the operating environment.

A single experiment consisted of the baseline detector being presented with driving data from successive days, with a detector retraining step between each dataset. This process follows the EBC system architecture diagram outlined earlier in this Chapter in Figure 2.1. For a given detector model, the image data from the single specified urban driving dataset was processed to compute detections. The detections were then processed by the scene filter to validate or reject the samples according to spatial heuristics. The resulting negative samples were then sampled (taking the top 10 false positives from a random frame in every second of time), aggregated with prior rejected negative data samples, then used to retrain the detector along with the original training data. The negative data were weighted to ensure the class balance from the original training data was maintained.

Each experimental run was evaluated against a manually labelled test dataset which shared the same location and environmental conditions as one of the training categories.

For all experiments in this section we use the object detector in 2.3.1, but with ACF features rather than HOG (as in the previous section). ACF has demonstrably better performance than HOG for object detection [33]. We train the linear SVM classifier using ACF features from the INRIA pedestrian dataset [47], selecting the SVM optimisation parameter C as 2^{-6} through ten fold cross validation. This parameter selection through cross validation aims to ensure the model generalises across the training data. We also perform hard negative mining on the INRIA negative image set, maintaining class balance by weighting the positive and negative classes as in Section 2.3.3. This improved feature set and hard negative mining provides a much stronger baseline classifier than the starting model in Section 2.4.2, one we would reasonably expect to use as-is in a pedestrian detector.

To show the impact of environmental variation and to evaluate our self-supervised learning approach, we used twelve different urban driving training datasets gathered with a Bumblebee2 stereo camera mounted on our Wildcat vehicle (Figure 2.12a) driving around Oxford. These unlabelled datasets were gathered from two different routes from successive outings at different times of year. We allocated these datasets



Figure 2.12: The Wildcat vehicle (a) used to gather the image data, and a map (b) depicting the routes for our datasets, where we gathered images in January, May, and August. The difference in time of year provided seasonal variation, which affects the visual appearance of the scene. The two North Oxford routes, illustrated in blue, provided variation in season, and the Central Oxford route, depicted in red, provided a difference in location.

to three categories based on the route and time of year (season), with 4 training datasets per category. These categories are referred to in this section as *North Oxford January*, *North Oxford May*, and *Central Oxford August*. A map of the routes is provided in Figure 2.12b, with no overlap between the North Oxford and Central Oxford routes. For all datasets, we used only the left stereo image with a capture rate of 20Hz.

For evaluation, we used an additional manually labelled test dataset from the *North Oxford January* category. This provided a total of 40km of unlabelled training data and 2km of labelled test data. The datasets are summarised in Table 2.3.

2.5.2 Results

We trained three different detectors from the same starting base detector, one per category. These detectors are referred to by their category name: *North Oxford January* and *North Oxford May*, and *Central Oxford August*. Each detector was evaluated against a separate test dataset also derived from the North Oxford route during January.

Route	Metric	Train 1	Train 2	Train 3	Train 4	Train Total	Test
North	Distance (km)	2.60	2.01	1.92	1.92	8.45	1.99
Oxford	Image frames	12782	9436	8172	8215	38605	9154
January	Time (min)	10.7	7.86	6.81	6.84	32.2	7.63
North	Distance (km)	1.43	1.95	1.01	1.01	5.40	-
Oxford	Image frames	6066	8676	4001	3977	22720	-
May	Time (min)	5.06	7.23	3.33	3.31	18.9	-
Central	Distance (km)	6.91	6.79	6.68	6.56	26.94	-
Oxford	Image frames	36472	27720	27607	23463	115262	-
August	Time (min)	30.4	23.1	23.0	19.6	96.1	-

Table 2.3: A summary of the datasets used for training and evaluation. We use 4 runs through each operating environment for training. Two the operating environments are driving the same route at different times of year, whereas the third is a different route entirely.

Performance Gains Using Experience-Based Detectors

Firstly, the results in Figure 2.13 show that we are able to improve the perceptual performance of a detector by training it on data gathered from the environment it operates in. However place is clearly important in Figure 2.14. We see that both detectors trained in the same place (North Oxford) improve notably in performance, whereas the detector trained in a different place (Central Oxford) does not.

Secondly, the same figure shows that there is a seasonal effect in addition to the spatial similarities. There are clear perceptual differences between the two North Oxford seasons, and a detector trained on a driving route in January has improved performance when operating in January compared to a detector which learned from the same driving route in May.

Failure Cases: Learning the Global Environment

These results support our argument for experience-specific classifiers. However, while it is clear that a detector trained for its operating environment is better than the general baseline detector, this raises questions around the necessary spatial and temporal resolution for these experiences in robot perception.

To confirm the value of experience-specific classifiers, we also investigated the effects of simply amalgamating all the training data into one classifier, with all

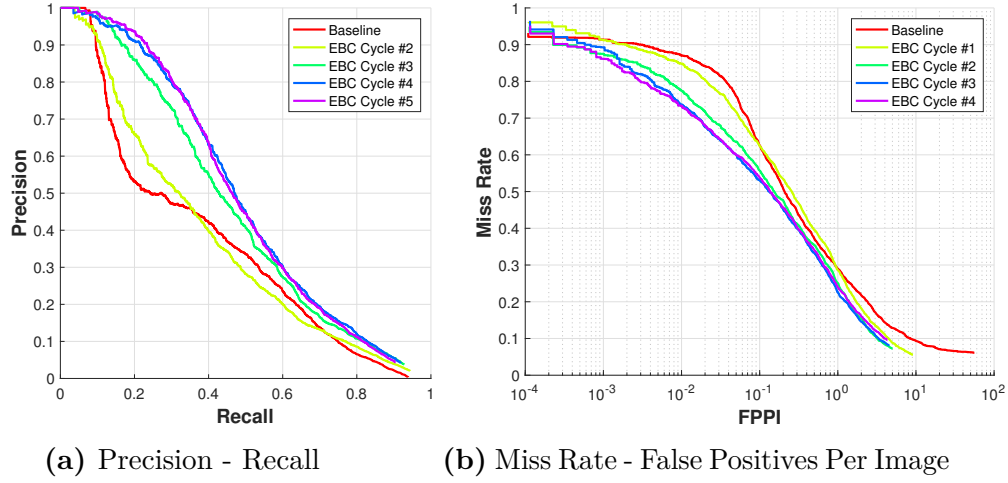


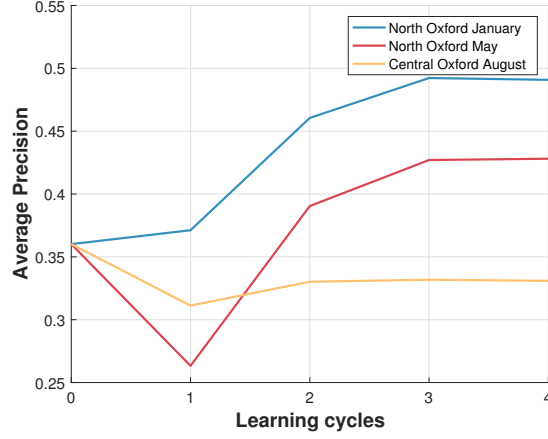
Figure 2.13: The precision-recall (a) and miss rate-false positives per image (b) performance of the detector during self-supervised learning while driving the North Oxford route in January, tested an additional labelled dataset from the same route and season. The PR curve performance increases with the first three training routes, moving to the top right corner of the graph and converging by the fourth repetition of the same operating route. The same trend is visible in the MR-FPPI graph with the curves moving to the lower left corner.

twelve datasets processed in temporal order (January through August). The results in Figure 2.15 show that the global detector’s performance is significantly worse compared to the detector trained solely on data from the same place and season as the evaluation data, with performance degrading substantially as dissimilar data are observed and learned.

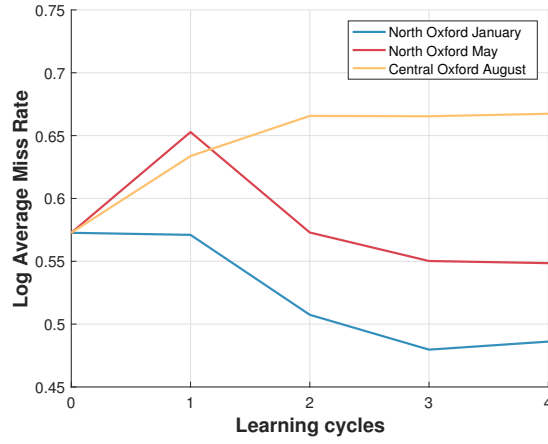
This result adds further weight to the argument for training place dependent classifiers, and emphasises the need for research into what defines a ‘place’. Our trials considered a small set of possible places and conditions, and it is likely that experiences in robot perception will be influenced by more than simply season and route. These factors could include weather, lighting, and additional environmental changes such as traffic.

Experience-based Detectors and Scene Filtering

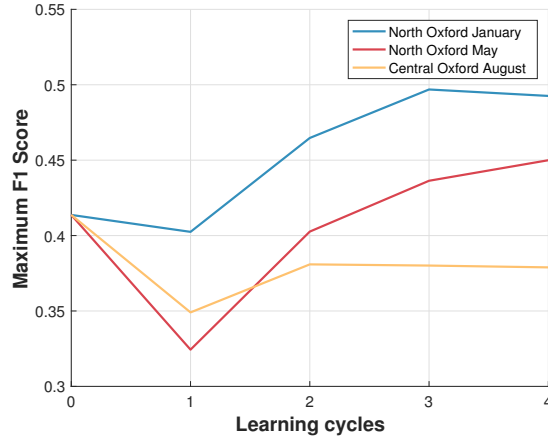
Finally, we note that we have only focussed on the raw detector performance in our experimental trials, rather than the detector and scene filter as a whole. Since the scene filter is already incorporated into the EBC framework, we can also validate



(a) Average Precision

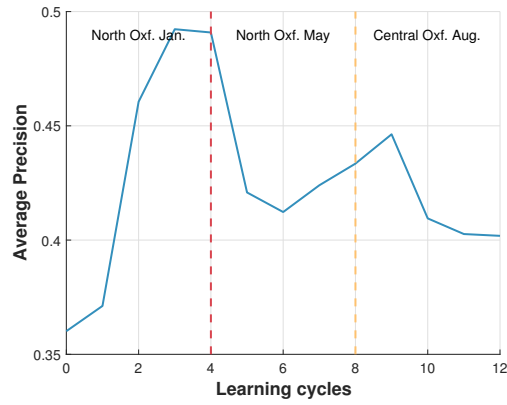


(b) Log Average Miss Rate

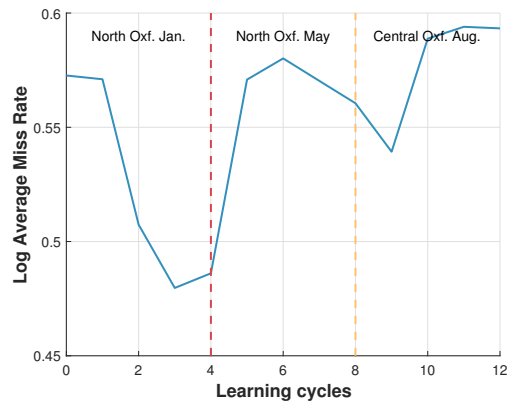


(c) Maximum F1 Score

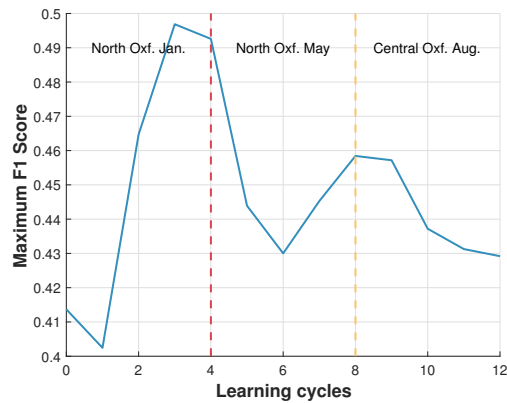
Figure 2.14: Aggregate detection performance metrics for three detectors trained using EBC, evaluated on the North Oxford January test dataset. Learning from operating on the same route (North Oxford) improves performance over the baseline detector, with the detector shown data from the same season as the evaluation set performing the best (January). The detector which learned from operation on a different route and time of year (Central Oxford August) does not improve performance.



(a) Average Precision



(b) Log Average Miss Rate



(c) Maximum F1 Score

Figure 2.15: The average precision (obtained by computing the area under a PR curve) for a detector trained using EBC on all datasets (both routes), evaluated on the North Oxford January test dataset. The detector shows a performance improvement with January data, but drops as it incorporates data from dissimilar environments, changing seasons to May, then changing route and season to August.

our detections while running online if a 3D scene prior and localisation information are available. The performance increase from the scene filter on the detector’s output decreases over successive training cycles, with a large initial improvement tapering off to a very small difference by the end of our trials in Figure 2.16. The small difference at the end may be attributed to the fact that the *SVM+ACF* model is sufficiently expressive to cover what the scene filter is able to invalidate. A more comprehensive scene filter will be needed to further improve the filter results, potentially including computationally expensive offline processing.

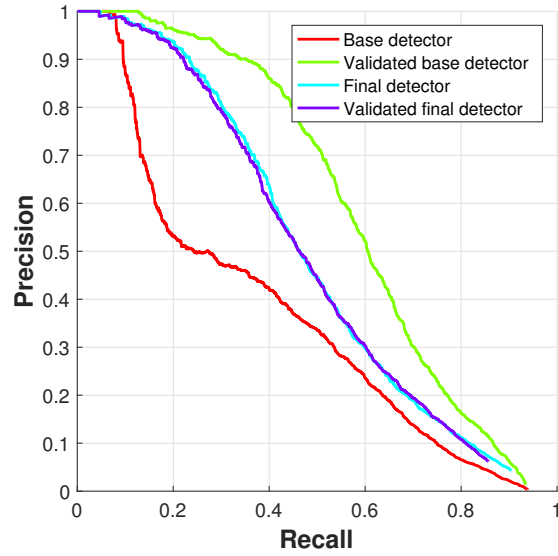
Detector Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
Baseline	0.360	0.573	0.414
Filtered Baseline	0.600	0.383	0.592
Experience-based	0.491	0.486	0.493
Filtered Experience-based	0.485	0.499	0.485

Table 2.4: Aggregate performance metrics for the base and experience-based detector models in Figure 2.16, including the validated output from the scene filter.

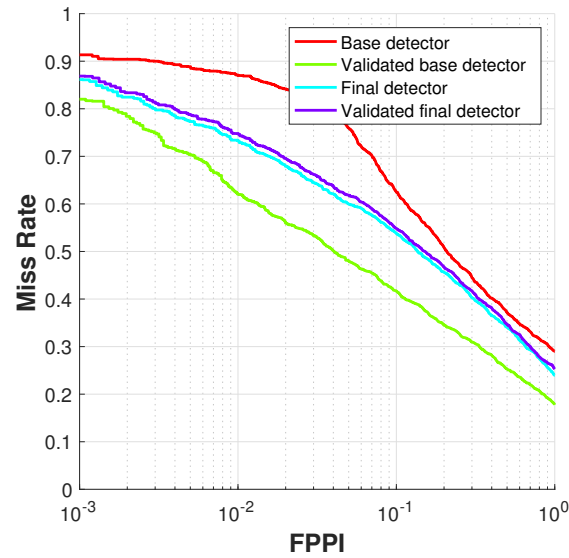
2.6 Conclusions

Though general object detection remains the ultimate goal, applications in robotics tend to be constrained to particular operating environments. We can exploit this fact to obtain practical systems which excel in a specific application domain. This is a major step towards reliable performance for real-world safety-critical systems. In particular, we make use of scene context to validate detections, and feed the rejected samples back to retrain the detector. This augmentation to the standard perception pipeline provides self-supervised environment-dependent improvement over the lifetime of the system. We call this process Experience-Based Classification (EBC).

Using urban driving data, we demonstrate that EBC provides a means to improve a general baseline object detector beyond what conventional negative data mining on a training dataset achieves. This suggests great utility in training experience-specific classifiers, potentially leading to life-long learning in robot perception without the need for human assistance. Perceptual systems benefit from being



(a) Precision - Recall



(b) Miss Rate - False Positives Per Image

Figure 2.16: Performance increase provided by the scene filter (referred to as ‘validating’ a classifier) when applied to both the base classifier and the final EBC detector on the North Oxford January test dataset. The invalidated data from the scene filter facilitates learning from the environment. Table 2.4 outlines the aggregate performance metrics for these plots.

trained to suit the local environment and their performance varies as the robot experiences different environments.

Our experimental results show that environment-specific tuning provides benefits in performance at the cost of generality, but the results raise a number of research questions, primarily around what defines a robot’s perceptual experience.

While we manually divided the datasets here (based on known seasons and routes), we require an automated method to determine when to train new classifiers. This could be achieved through localisation, with a new detector model for every small map segment or GPS location.

However a purely spatial approach would not accommodate normal variation in weather, lighting, and seasons. We believe that this could be accommodated by using experience-based visual localisation (rather than simply GPS), where we rely on the localisation system to maintain multiple representations of the same spatial location under appearance change. An alternative approach could fit models purely based on image appearance, relying on image similarity measure or clustering to group similar data.

The granularity of the experience we fit a model to is also likely to be a major influence on performance: we saw in these results that fitting a model to every operating condition degrades performance. Should we therefore go as far as possible to the other extreme, fitting models to as little as a single image? Other issues which we need to be aware of are semantic drift (the potential for the target class to shift as more data are mined, minimised here by only fitting to the background class), and the limits of model capacity (likely to be coupled with model granularity).

We continue this line of investigation in Chapter 3, looking at what should define the ‘place’ we fit detector models to, comparing models created via spatial proximity and image appearance.

3

What Makes a ‘Place’? Building Local Expert Detectors

Contents

3.1	Introduction	50
3.2	Background	52
3.3	Bespoke Local Expert Detectors	55
3.3.1	Methodology	56
3.3.2	Object Detector and Training	57
3.3.3	Results	57
3.4	Local Expert Detectors using Image Appearance . . .	67
3.4.1	Methodology	67
3.4.2	Results	68
3.5	Local Expert Detection Discussion	73
3.5.1	Higher Capacity Local Expert Detectors	74
3.6	EBC Deployment: A Local Expert Detection System .	76
3.6.1	System Outline	76
3.6.2	Model Initialisation	77
3.6.3	Extending the Scene Filter: Robot Dreaming	79
3.6.4	Results	86
3.6.5	Discussion	94
3.7	Conclusions	95

3.1 Introduction

Place matters in robotics. Mobile robots frequently traverse the same operating environment over and over, such as an autonomous car performing the same weekday commute, or an autonomous forklift moving goods around the same warehouse. In object detection for robotics, we care about achieving the best possible performance on the data the robot observes in its operating environment. We propose that relaxing the need for detector generalisation in favour of fitting to the environment the robot experiences leads to better perception performance using a computationally simpler detector.

Our initial work in this area (Chapter 2) demonstrated that it is possible to fit a detector model to a given operating environment, biasing the detector to favour a mobile robot’s route and the season [78]. Similar ideas have motivated other work in object detection [79], terrain assessment [80], and visual localisation [21, 81]. However, these results also indicated that simply learning a ‘global’ model from multiple routes does not offer this boost to detection performance.

A key question in considering environment specific models is therefore: what defines the ‘place’ we fit if we want to improve a perception system’s performance?

This chapter investigates the effect of ‘place’ granularity or resolution on performance, seeking to understand the key factors linked to model performance as this resolution changes.

Some of the variability in object detector performance will be spatially dependent, with the general scene structure and image texture remaining the same between images taken in the same location.

In addition, we have many temporal sources of environmental variation which will affect image appearance, and therefore detector performance. This variation includes appearance change from seasonal change (summer to autumn), lighting change (day to dusk), and other dynamic objects operating in the same environment.

We also expect the defined operating environment for a local expert detector to play a large part in determining detector performance. The smaller the model’s defined environment the more specialised the model will be, meaning that we can

achieve class separability with limited model capacity. Conversely, if we fit a model to a large category of environmental data, we will likely have to ensure the model generalises somewhat (as in the approach in Chapter 2).

Our goal here is to maximise the performance of a predetermined perception system within the constraints of a robot’s operating environment.

Our results demonstrate that overcoming limited model capacity (the ability of a classifier to separate the data classes) is the key benefit of using local expert detectors, enabling a lightweight detector to achieve a substantial performance increase over our baseline pedestrian detector. We are able to do this by limiting the variability of the background (negative) class to local regions of the robot’s operational envelope.

To conclude the chapter, we show that the approach derived from constrained experiments can be extended to operate on unlabelled data, using a visual localisation against a topometric map and scene segmentation as a supervisory signal in training models.

Outline

This chapter is structured in two parts. Firstly we seek to find the upper bound of what we can achieve with bespoke environmental detector models in an ideal, best-case scenario in which we have ground truth labels and the test and training data are taken from consecutive laps of the same route. These constraints ensure that we minimise performance variation due to appearance change (with test and training data derived from the route and time), and we maximise the ability of the detector to fit the exact background class for the operating environment (with ground truth labels). In doing so we strongly overfit the model to the constrained environment to maximise the potential performance.

We expect data taken from the same spatial location and time to have similar structure, enabling us to construct bespoke detector models on sequential images. Section 3.3 considers a scenario using this strict spatiotemporal definition of a place. We use an ensemble approach for building bespoke place dependent models based

on nearest neighbour methods, creating models based on the nearest frames to each mapping frame, and retrieving these models at run time using localisation.

However, we also expect structure in imagery based on appearance, not just the spatial location. Section 3.4 extends this bespoke model methodology to use the nearest neighbouring frames based on image appearance rather than spatial location.

Our results demonstrate that circumventing limited model capacity is the key benefit of using bespoke place dependent detector models, enabling a lightweight, fast detector to achieve a substantial performance increase over our reference baseline, bringing the performance of this lightweight model in line with a much more complex neural network detector. We are able to do this because we are limiting the variability of the background set (or negative class) to data from the robot’s operational envelope.

The second half of this chapter (Section 3.6) takes the best approach in the ideal scenario and builds on this with a self-supervised local expert detector system.

This system couples object detection with visual localisation, where each key frame in an experience-based visual map has an associated object detector model. The detector models are constructed using the approach from the first part of this chapter, using self-supervised scene filtering to build a model for a sequence of unlabelled images centred around the map key frame.

We see a healthy improvement in detector performance using this system, surpassing a generic base detector model. As the system is self-supervised, this boost in performance is not as large as the one we achieved in the first half of this chapter, when using ground-truth object annotations. Analysis of the detector models indicates the models fit the background class better than a generic model, demonstrated by the local expert models’ errors on other dynamic objects in the scene rather than the static background.

3.2 Background

Pedestrian detection has an extensive body of literature. Much of this work can be summarised as focussing on the development of progressively better engineered

features [33, 82], different classifiers [66], deformable parts models [83], or adding additional data or context [49, 50, 84, 85].

Recent developments in deep learning in the broader computer vision community have lead to convolutional neural networks being applied to the problem [29, 31, 86]. These approaches are now achieving state-of-the-art performance but tend to be slower at run-time, which makes these ill-suited for use in resource constrained robotic applications.

Zhang *et al.* [36] presented a comprehensive failure analysis of state-of-the-art pedestrian detectors on the Caltech pedestrian dataset [33]. They hypothesised that many of the false negatives (i.e., cases where the detector fails to detect a pedestrian even though the pedestrian is large enough in the image) are due to dataset bias in training, which can be solved by augmenting the training dataset. Their analysis of common sources of false positives highlighted two categories of error: localisation errors where the false detection overlaps with a ground truth bounding box, and background errors where there is zero overlap with ground truth. These background errors were the most common error type, spuriously detecting parts of the background as pedestrians.

In the previous chapter [78] we focussed on addressing these background errors in object detection by applying the concept of robot ‘experiences’ from localisation and navigation [20, 87–89] to object detection. In an experience-based approach we accept that the world has many different forms of appearance and fit local experts to suit each experience. For perception tasks such as object detection, this means training our detector model to suit the class of data (e.g., season, spatial location) the robot currently experiences. We exploited the fact that mobile robots tend to be constrained to a particular operating environment to boost test performance by fitting models to data captured at a similar time of the year for the same traversed route. Notably, this experience-based detector outperformed a global model which has been trained on data across multiple experiences, highlighting the importance of defining the environment to fit to.

Geospatial bias in imagery has been highlighted by Doersch et al. [44], who showed that it is possible to automatically find distinctive visual elements for a certain geo-spatial area, comparing data between major cities (for example, Paris and London) and data between districts of a city (Paris). We also know that dataset bias [39] is an influence on the performance of our object detectors, with the training data used being a systematic factor influencing performance across different test datasets [61]. This data bias, however, provides a potential tool to be exploited.

Malisiewicz et al. addressed bias in intra-class variability (such as different types of buses) in general purpose image recognition with Exemplar SVMs [90], an ensemble approach which uses a nearest neighbour method to train a unique model for each positive training datum, fitted to a single positive sample and many thousands of negatives.

Various approaches have been taken to leverage data bias to form scene specific classifiers, largely motivated by visual surveillance tasks. Hattori *et al.* took the concept of scene specific classifiers to one extreme, training a unique detector model for a specific image location in a static scene using synthetic data [91]. Wang *et al.* used a transfer learning framework to adjust a generic pedestrian detector to a scene specific detector [92]. They used visual cues to generate scene specific data in a self-supervised manner, weighting training data samples based on similarity to the observed test data using an affinity graph.

In the domain of mobile robotics, Suzuo *et al.* divided a driven route into a fixed number of scenes using visual bag-of-words, then trained models using negative data gathered from each scene [93]. Bewley and Upcroft presented similar evidence for the need to adapt object detectors to their operating environment in [94], augmenting a generalised object detector by validating each detection against a background model trained on environment specific data.

Berczi *et al.* [80] created bespoke terrain assessment classifier models for finite patches of a route driven with visual teach and repeat. These models detect when the path in front of a mobile robot has changed from the initial teach pass and previous repeat runs. They demonstrated a marked performance improvement over

place independent methods, comparing their bespoke place dependent models to a place independent learned classifier and a traditional heuristic based method.

Similarly, McManus *et al.* [81] and Linegar *et al.* [21] both proposed using distinct scene specific patches to weakly localise across large appearance change.

These methods all have similar motivation: solving a simpler problem than a generic model which works well everywhere. However, why do these work so well? Why can’t we simply use a generic model? Here we present evidence for *why* bespoke, place dependent models work well for solving these sorts of problems in robotics, and offer insight into how we should approach the construction of bespoke expert detectors.

3.3 Bespoke Local Expert Detectors

We evaluate how to build bespoke place dependent models by firstly adopting a fine-grained, strict spatiotemporal definition of a place, where we construct a place dependent model for every single frame in a known route. We then immediately repeat the route, localise each image to our mapping run, then retrieve and apply the detector model associated with the localised map frame.

This is the best possible scenario we could hope for when fitting models to similar data. By using a strict spatiotemporal prior we are able to control most sources of variation in scene content (as the images are from the same spatial location) and appearance (as the images are from effectively the same time). In addition, we use a very fine grained approach, associating a model with every frame, rather than select key frames.

In practice, this will form the upper bound of what we could hope a place dependent method to achieve with this model (classifier and feature type), though we expect to be able to get close to this using experience-based visual localisation [20] with models associated with mapping key frames.

This approach is very efficient at mapping scales we operate at today: simply retrieving a small detector model (approximately 10kB) from a database indexed by a key-frame identifier. However, this approach will unnecessarily duplicate

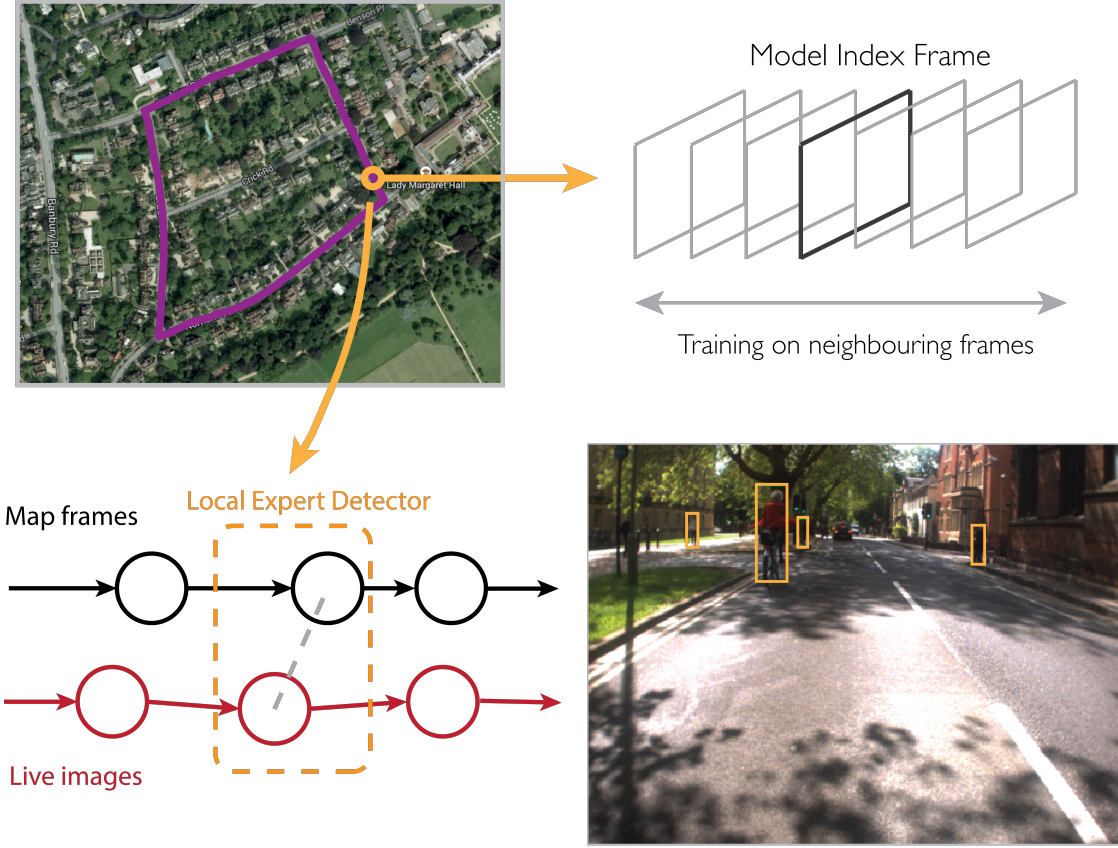


Figure 3.1: We can use the knowledge of where our robot is to improve object detection performance using a local expert model fitted to that specific place. Using lightweight, fast detector models with limited model capacity we can boost performance beyond more computationally complex state-of-the-art detectors. We achieve this by limiting the variability of the negative (background) class our model is trained on, mitigating the model capacity limits of these lightweight detector models.

models where they may not strictly be required, and potentially suffer from scale as the map sizes increase. We seek to address this limitation later in Chapter 4 though learning, though there may be other possible approaches for managing model growth and duplication.

3.3.1 Methodology

We select an urban driving dataset consisting of two consecutive loops of the same route in the Oxford driving dataset [95]. This dataset is the same one used for evaluation in Chapter 2, referred to as *North Oxford January*. Each loop is approximately 1km in length and we have fully annotated this with pedestrian labels. For each image (or model index frame) in this dataset, we fit a model to the closest

N images in time to the model index frame (including the index frame), illustrated in Figure 3.1. At run time, we localise each image to the alternate lap and retrieve that model, using the models built on the second lap during the first and vice versa.

3.3.2 Object Detector and Training

The object detector here is the same as in Chapter 2: a lightweight sliding window implementation using Aggregate Channel Features (ACF) on LUV images [33], with a linear Support Vector Machine (SVM) as the primary classifier. We refer to this detector as *SVM+ACF*. This is a general purpose object detector which is extremely lightweight, able to run at frame rate on general purpose hardware. Variations on sliding window detectors using the ACF feature type have been used for detection tasks such as traffic lights [64] and road signs [65] in addition to the original purpose of pedestrian detection. Further details on this are in Chapter 2.3.1.

We fit detector models to the nearest N frames in time by performing hard negative mining (HNM) against our ground truth labels until convergence (or a maximum number of 20 iterations), using the INRIA pedestrian dataset [47] for positive samples and an initial seed negative set. Unlike earlier in Chapter 2, here we heavily penalise misclassification of training data by setting the SVM optimization parameter to a very large number ($C = 100$), forcing the model to strongly fit to the training data presented. The training data is weighted based on the number of samples to ensure class balance, as in Chapter 2.3.3.

3.3.3 Results

To assess the merits of a precise spatiotemporal definition of a place, we compare our place-fitted detector to a generic model for our pedestrian detector trained only on a canonical dataset. This generic model is trained on the same initial training data as the place dependent models, but with HNM performed on the INRIA negative training data rather than data from the robot’s environment. We refer to this baseline trained on generic data as the *SVM+ACF model*. Table 3.1 outlines the performance metrics on our test dataset.

Detector Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
Baseline:	0.483	0.485	0.473
SVM+ACF			
Baseline: FPDW [33]	0.481	0.440	0.491
Baseline: MSCNN [31]	0.588	0.371	0.579
PD Model: N=1	0.622	0.354	0.592
PD Model: N=10	0.689	0.292	0.651
PD Model: N=100	0.689	0.288	0.672
PD Model: N=1000	0.450	0.564	0.521
PD Model: N= $\sim$ 4100 (full lap)	0.305	0.712	0.405

Table 3.1: Aggregate performance metrics comparing our bespoke place dependent (PD) detector models to a number of generic baselines, where N is the length of image sequence used to train a model for a given frame. A higher number is better for Average Precision and F1 Scores, whereas a lower number is better for Log-Average Miss Rate.

Benchmarks

Results from two publicly available detectors are included to provide a baseline from which to assess potential performance gains. These are *FPDW* [33], and *MSCNN* [31]. We use models provided by the respective authors, trained on INRIA (for *FPDW*) and Caltech pedestrian datasets [33] (for *MSCNN*). There will be some difference in performance attributable to different training data, particularly for the higher capacity neural network. The influence training data has on performance is known, but has not been extensively researched as most computer vision research tends to be constrained to improving performance on canonical datasets due to the difficulty in creating new datasets. INRIA has been shown to generalise well across a diverse range of benchmark datasets [61], as fewer, diverse pedestrian data is more useful than many samples of the same (or similar) pedestrians. Unfortunately, as INRIA is a cropped dataset, it is only applicable to sliding window approaches

making it unsuitable for training detectors like *MSCNN*. This issue around selection of training data will always be present in robotics as our robots will inevitably operate in a different domain than the training data. We therefore include these benchmarks as reference points, but draw our conclusions on the performance change with respect to the *SVM+ACF* reference model trained on the same data.

FPDW is very similar to our *SVM+ACF* generic model, using ACF features but with a boosted decision tree classifier (AdaBoost) rather than an SVM. These two detectors are largely equivalent in performance and speed, able to run at camera frame rate (20Hz) for a reasonable range of scales using only a moderate CPU. *MSCNN*, a multi-scale convolutional neural network detector, is currently one of the best performing object detectors. However, it does require significant GPU computational power and runs at approximately 2Hz on the current top-end NVIDIA GTX graphics card. The authors state a run-time performance of 0.4s per frame on KITTI object detection images (1242×375) [68]. For many robotics applications, this level of computation (and power) is not feasible, or the application requires a much higher frame rate. An example of this type of application is pedestrian detection for autonomous driving, where a car travelling at highway speeds of 100kmh^{-1} covers nearly 30 metres per second.

Detector Performance

The bespoke place-fitted detector significantly outperforms the baseline generic model, with an absolute twenty percentage point improvement over the reference model of the same type (*SVM+ACF*). Figure 3.2 shows the characteristic performance curves for these models as the training sequence length varies.

The dotted lines on each plot are the performance of the model on the training data, providing us with an understanding of when our problem suffers from limited model capacity. Model capacity refers to the ability of a model to accurately fit the data: a low capacity model being unable to separate the two classes (object and background). This model capacity is influenced by both the features and model used, and the environment itself. We would expect a high capacity classifier or a

more informative feature definition to increase this. Similarly, we would expect a more uniform environment to enable the use of a lower capacity model without impacting detector performance.

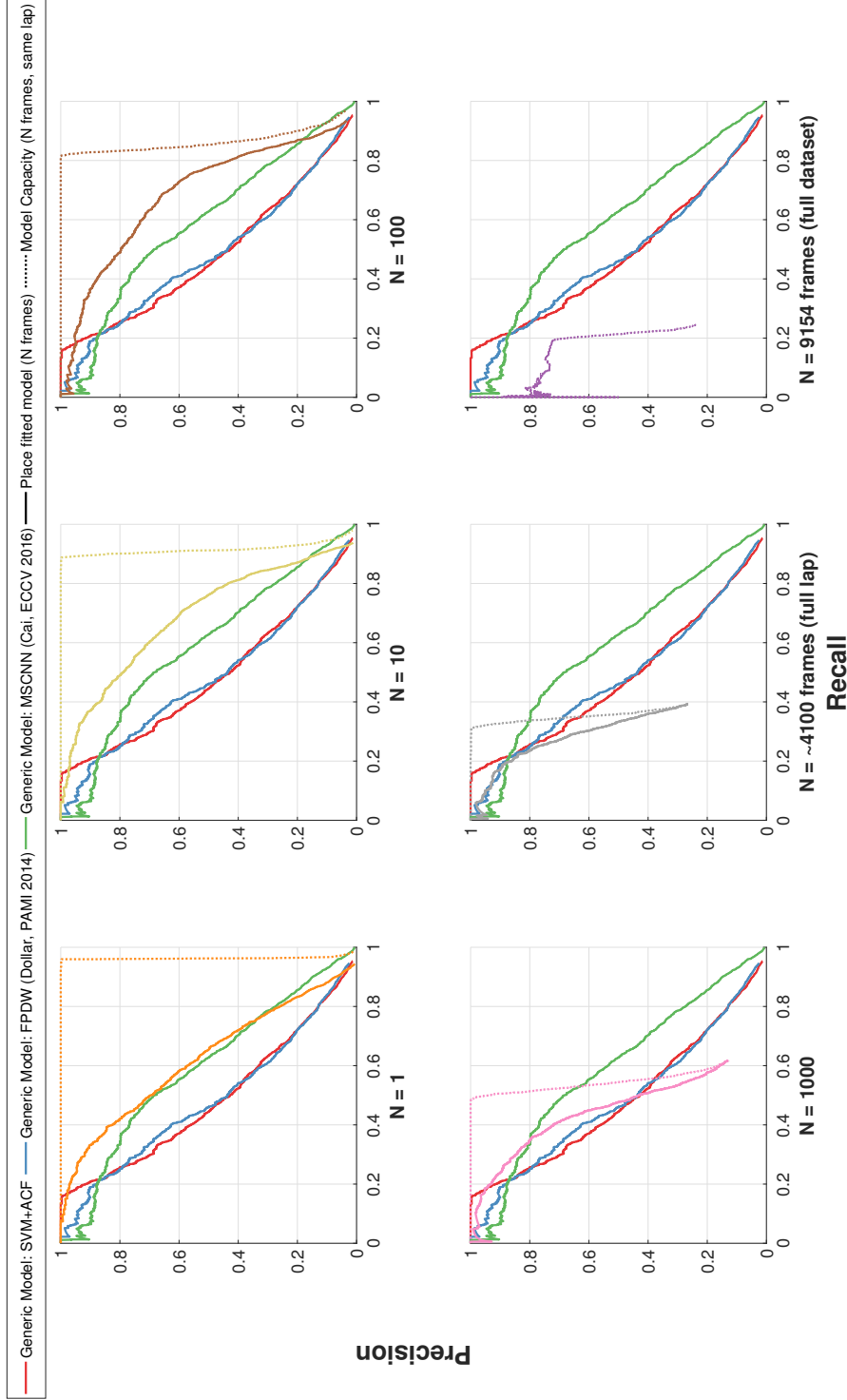
From the results in Figure 3.2 we see that this proxy for model capacity reduces as we increase the size of the training swathe. This logically holds: we have fixed, finite model capacity for a given classifier and feature type, hence as we increase the extent (variability and size) of the background class we reduce the performance of the detector model on the training set. We discuss the relationship between performance and training swathe size further in the subsequent section, as it relates to model capacity.

Training Swathe Size and Model Capacity

The detection performance results in Figure 3.2 highlight a clear relationship between detection performance the size of training swathe used for each model. We attribute this to the increase in extent (variability) of the background class with finite model capacity.

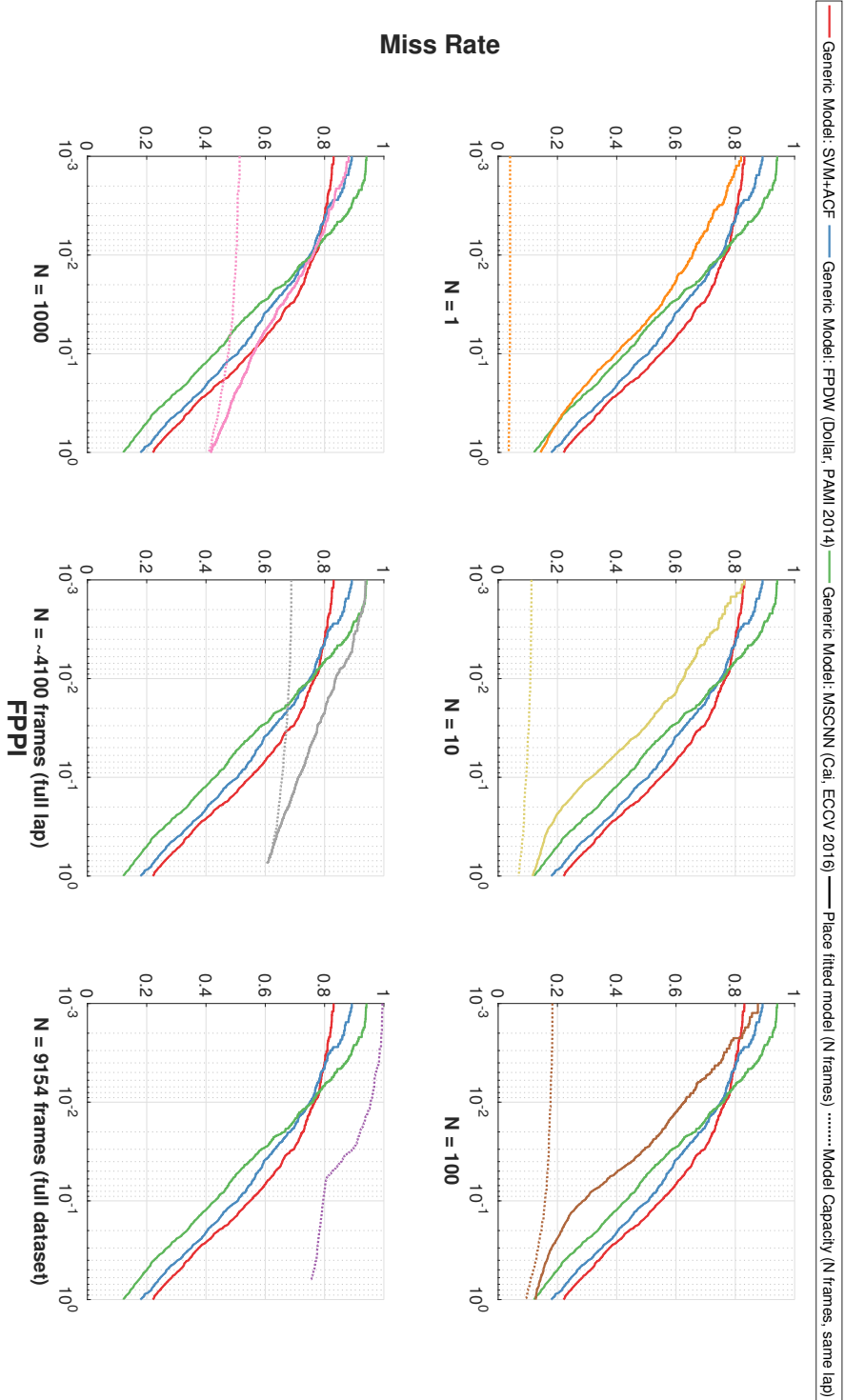
Model capacity refers to the ability of a machine learning model to fit data. This capacity will be dependent on the feature type, classifier type, and the data distribution. In a binary classification scenario like this, a useful proxy for capacity is the performance (for example, accuracy) of the classifier model on the training set. For the generic baseline model used in this section, the accuracy on the INRIA training set is 77%. This fact that this metric is less than 100% on the training data indicates that the model has insufficient capacity to fully represent the classes given the classifier and feature type. In these circumstances (when we want a generic model) we must ensure the model generalises between the classes, by selecting appropriate training parameters through cross validation.

Given we have finite model capacity, the variability (and thus separability) of the two classes will affect detection performance. If we can reduce the variability of one of the classes (the background, in this case), performance improves due to greater separability of the smaller classes. Figure 3.3 illustrates this conceptually.



(a) Precision - Recall (higher is better)

Figure 3.2: Performance of an ideal case place dependent detector compared to detectors using generic models. The place fitted detector uses models fitted on a frame-by-frame basis to a sequence of N images centred on the closest matching frame from a second lap (the solid coloured lines). The dotted line on each plot represents an proxy for the model capacity of the place fitted models, i.e., the performance when applying models to the same index frame they were trained on. This highlights the necessary trade-off between the extent (variability and size) of negative class we fit to and the point at which we need the model to generalise.



(b) Miss Rate - False Positives Per Image (lower is better)

Figure 3.2: [Continued] Performance of an ideal case place detector compared to detectors using generic models. The place fitted detector uses models fitted on a frame-by-frame basis to a sequence of N images centred on the closest matching frame from a second lap (the solid coloured lines). The dotted line on each plot represents an proxy for the model capacity of the place fitted models, i.e., the performance when applying models to the same index frame they were trained on. This highlights the necessary trade-off between the extent (variability and size) of negative class we fit to and the point at which we need the model to generalise.

The positive class is represented by red, and the background of the two operating environments is represented by the orange and green points. A single model (represented by the dashed red line) is unable to separate the complete positive (red) and negative (orange plus green) classes, making misclassification errors. When using a unique model for each operating environment (the orange and green dashed lines), the detection performance is boosted with fewer misclassification errors.

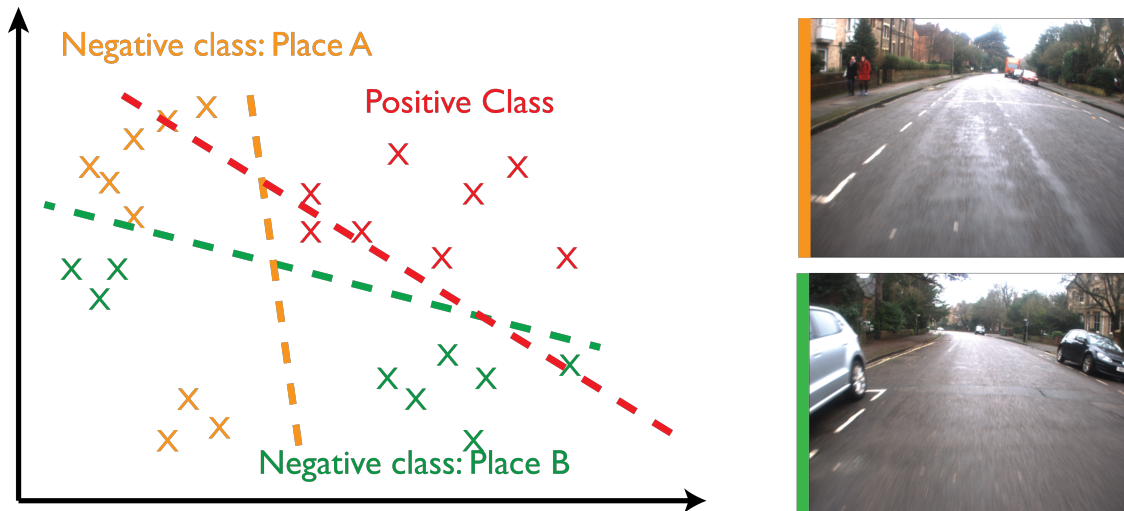
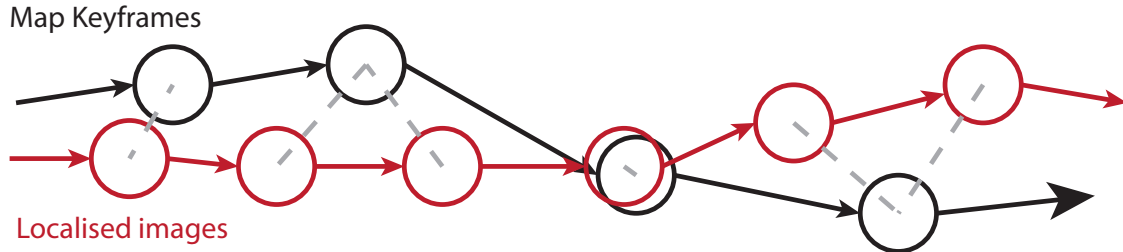
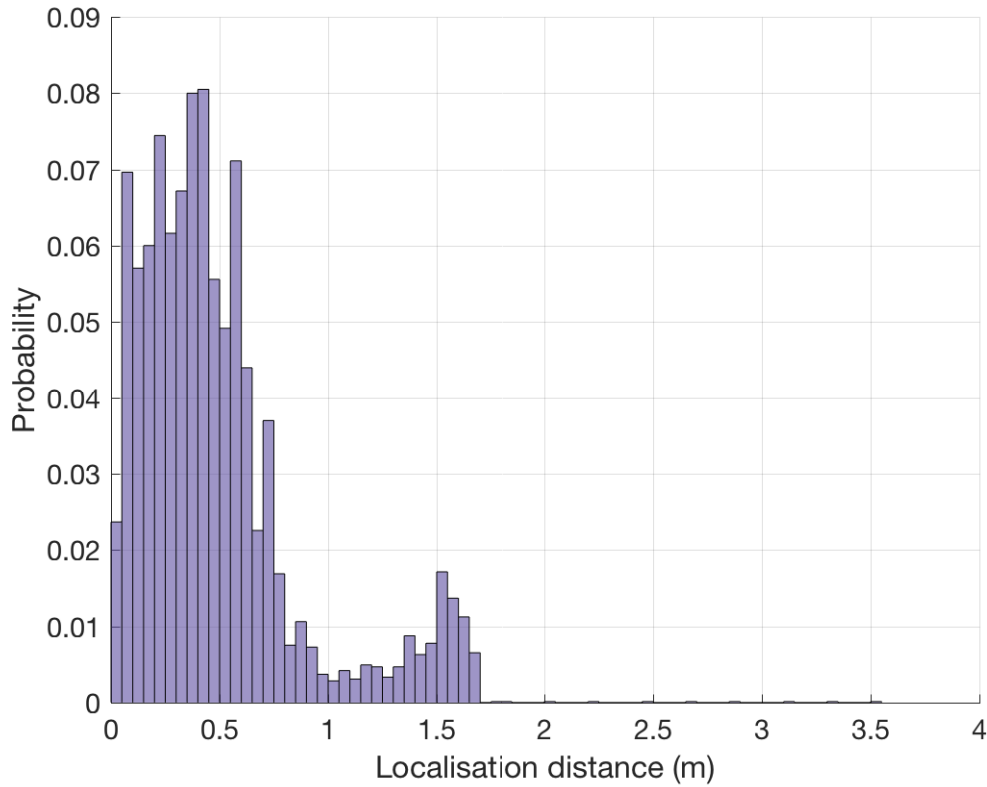


Figure 3.3: This illustration outlines a simple example of where using a simpler (faster) classifier can mean that it is impossible to separate the positive and negative class perfectly. We can reduce the variability of the background (negative) class to boost performance by constructing models associated with a certain place, allowing a lower capacity model to capture the reduced negative set without misclassification errors.

From the detection performance curves in Figure 3.2, we firstly note that using an extremely tightly fitted model (spatially, where the model is fitted to the single closest frame) offers a good improvement, but this is outperformed by models which are fitted to a slightly longer swathe ($N = 10 - 100$). There are a number of factors which will affect this. Firstly, longer training swathes will inevitably add more training data, which will generally help performance. In addition, these longer swathes will also help to account for the differences in pose between the image and the model index frame, illustrated in Figure 3.4a. This pose error is small (as per Figure 3.4b), but will limit the efficacy of models fitted to a single frame due to variation in perspective and scene content.



(a) Localised images will not necessarily align due to differences in the mapping and localisation trajectories. This variation makes fitting models to a single frame undesirable.



(b) In our best-case scenario fitting models to sequences of images, we see a median distance in pose of 0.40m between the map and localised model frames.

Figure 3.4: Variation in pose between localised images on subsequent laps will limit the efficacy of models fitted to a single frame.

On the other hand, using a large sequence of frames comes with a penalty. As we are forcing the model to fit the training data firmly, we begin to see a drop in performance due to model capacity once the negative set is sufficiently large (dotted lines on each plot in Figure 3.2). In this case with our classifier and feature type, this occurs at a training swathe of between 100 and 1000 frames. Beyond 1000 frames we suffer dramatically from limited model capacity, where the linear SVM classifier is unable to separate the positive and negative training data with no errors and fails to generalise (illustrated in Figure 3.3). As we weight the training data for class balance, this lack of class separability results in the hard negative mining reaching the maximum number of cycles rather than converging.

This performance drop due to model capacity is even clearer in the global model case, where we fit the model to the entire dataset. In this case, we only have a model capacity plot as we train the model on every image ($N = 9154$). Even though we are training the global model on the exact test data (with a high penalty for misclassification), it is unable to separate the positive and negative class, requiring the model to generalise.

Summary

Using a simple, lightweight pedestrian detector we are able to boost performance using models which have a very specific notion of a place. We should avoid using models fitted to as small a window as a single frame due to pose differences between training and testing data, but we should also ensure that we use a somewhat narrow definition of a place and thus a relatively small background class. In doing so, we can fit models to this narrow background class while avoiding issues of class separability and model capacity. For our classifier, features, and environment this means using a sequence of around 10 – 100 frames.

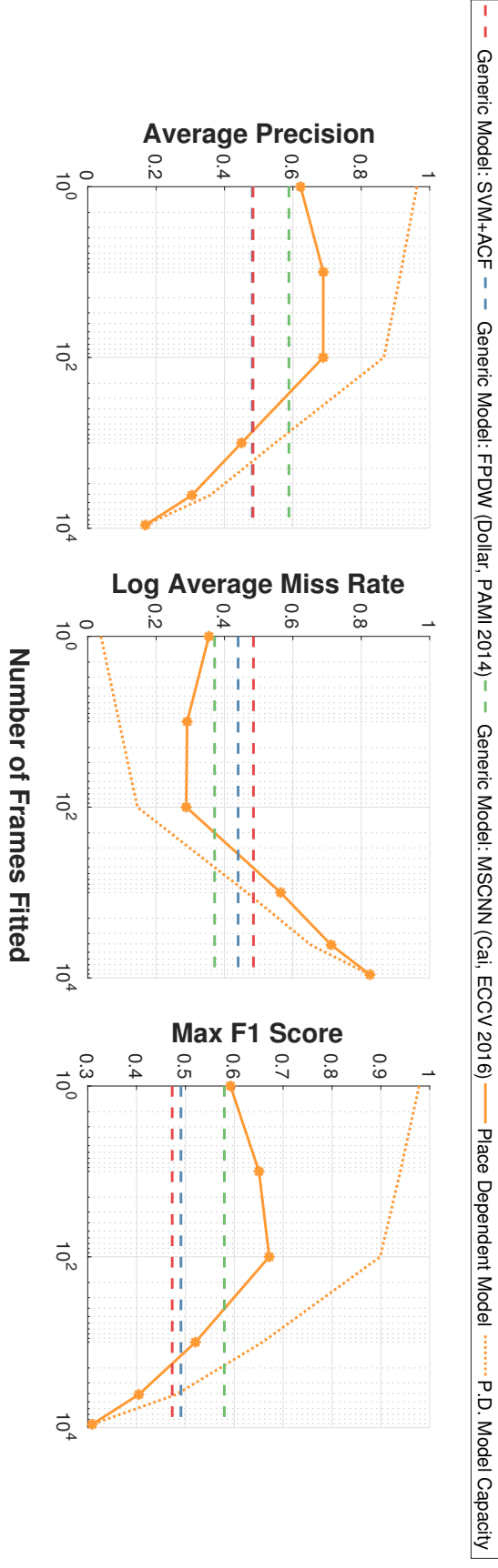


Figure 3.5: Aggregate performance metrics for place dependent detector models as a function of N , the number of frames in the spatial sequence training swathe. The dotted line represents training performance, interpreted as a proxy for model capacity. Note that our generic model ($SVM+ACF$) and $FPDW$ score almost identically in average precision.

3.4 Local Expert Detectors using Image Appearance

Spatial location is important, but we also expect to see significant image appearance change over time: seasons, lighting, and other moving objects. From visual localisation literature we know that image appearance change is significant, and has a dramatic effect on the performance of perception systems [20].

As we are attempting to boost performance by reducing the extent of the background class our model fits to, it logically holds that we should build models on images with similar appearance. In many cases this will have a spatial component, as shown in Figure 3.7. However this is frequently not the case. A key example of this is a collection of images taken while driving around a corner: these will be dissimilar but are spatially very close (Figure 3.6).

3.4.1 Methodology

We adopt the same experimental scenario as before in Section 3.3, where we have two consecutive laps of an urban driving route, along with ground truth localisation and pedestrian annotations. For every image, we localise to the closest frame in the alternate lap, retrieving the detector model associated with the map frame and apply it. The object detector setup and training is as per Section 3.3.2.

However, to build the detector models we now use the N nearest frames based on image similarity metrics, rather than N nearest temporal frames. We use two similarity measures: the l^2 -norm between two GIST global image descriptors [96], and mutual information [97] between two intensity images. We selected these due to their success in other image matching tasks: image retrieval and scene matching with GIST [98], and image registration with mutual information [99].

Our results in Section 3.3 indicate that our detector model was able to fit well for a place definition of between 10 – 100 frames, with 1000 frame detector models suffering in performance due to model capacity (Figure 3.5). This result agrees with the data we see in Figure 3.7, which indicates that the correlation between image appearance and distance no longer applies beyond 100 frames at

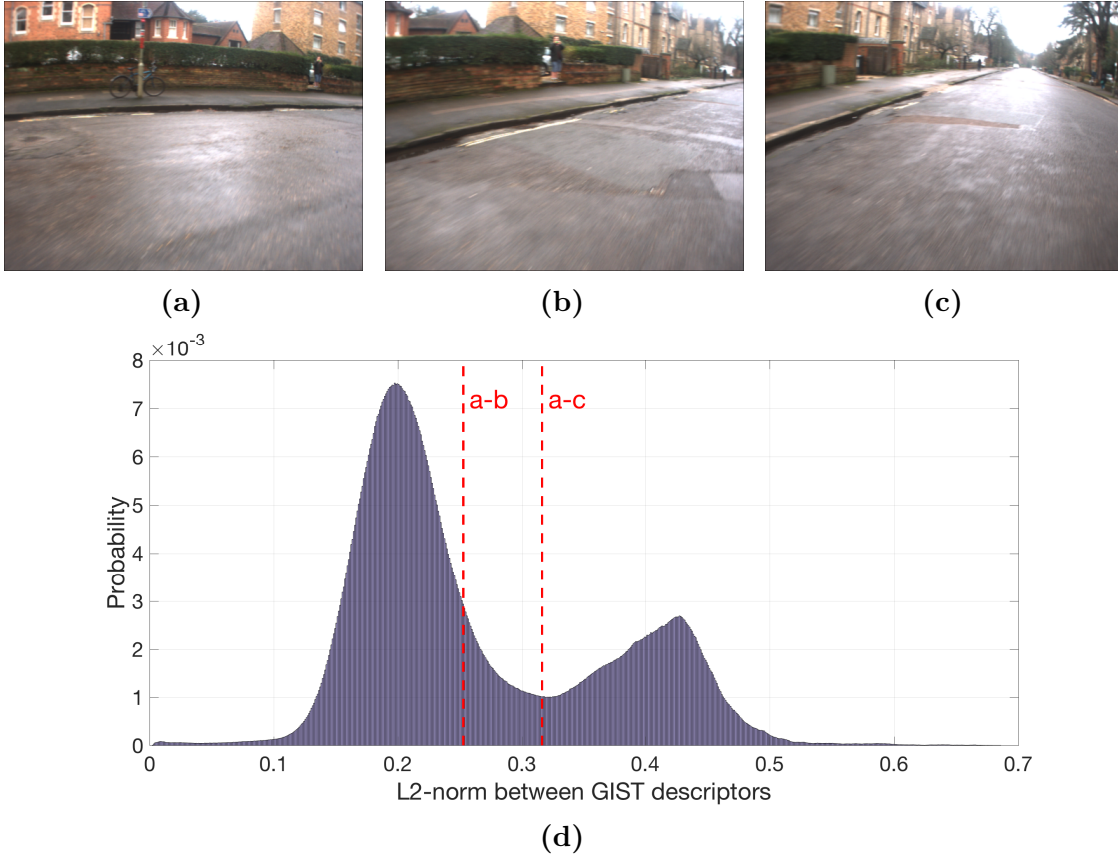


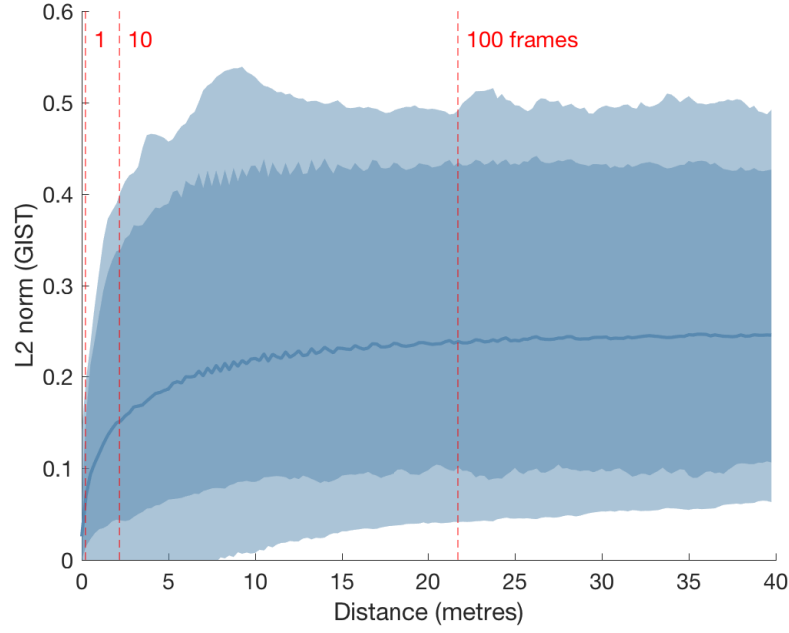
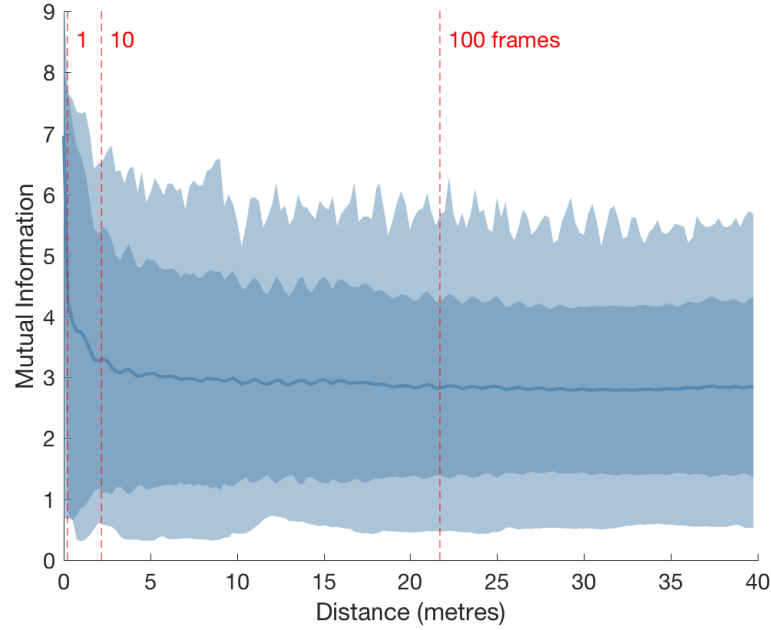
Figure 3.6: Images only one second apart (**a-c**, approximately 3 metres of travel each step) can be substantially different by image similarity metrics (l^2 -norm between GIST descriptors) due to the perspective changing while cornering. Figure 3.6d shows the full distribution of pair-wise image similarities for this test dataset, where the similarity between the first image in the pictured sequence (**a**) and the subsequent two (**b,c**) lie on the 61st and 71st percentiles respectively.

the average vehicle speed. We focussed our experiments on models built using the same number of frames as before ($N = 1 - 100$), starting from a single frame and increasing by a decade at each step.

3.4.2 Results

To assess the merits of using detector models based on image similarity (rather than spatial location), we use the same experimental methodology as in Section 3.3, using localisation to retrieve the model associated with the map frame closest to the current image.

In Figure 3.9 and Table 3.2 we see a performance boost over the reference

(a) l^2 -norm between GIST

(b) Mutual Information

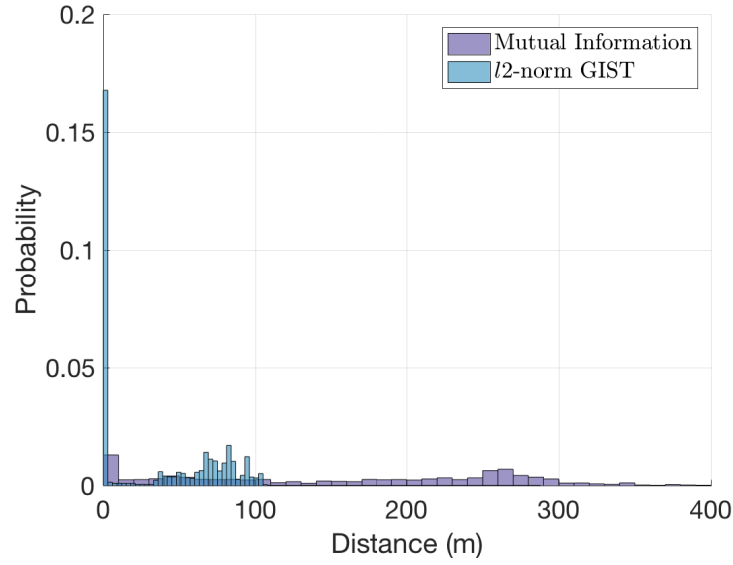
Figure 3.7: Image similarity metrics correlate with the distance travelled along the robot’s trajectory, highlighting the spatial structure present in image data. It is this structure which we exploit in Section 3.3 through spatiotemporal models, and in this section (Section 3.4) through models constructed using image appearance. The mean similarity is shown by the solid line, and the filled polygons represent the 2σ and maximum extent. The red dashed lines indicate the distance travelled within $N = 1, 10, 100$ frames during this test route at the vehicle’s average speed of 4.3 ms^{-1} .

baseline detectors. Using the nearest images based on GIST yields more or less equivalent performance to the spatiotemporal models.

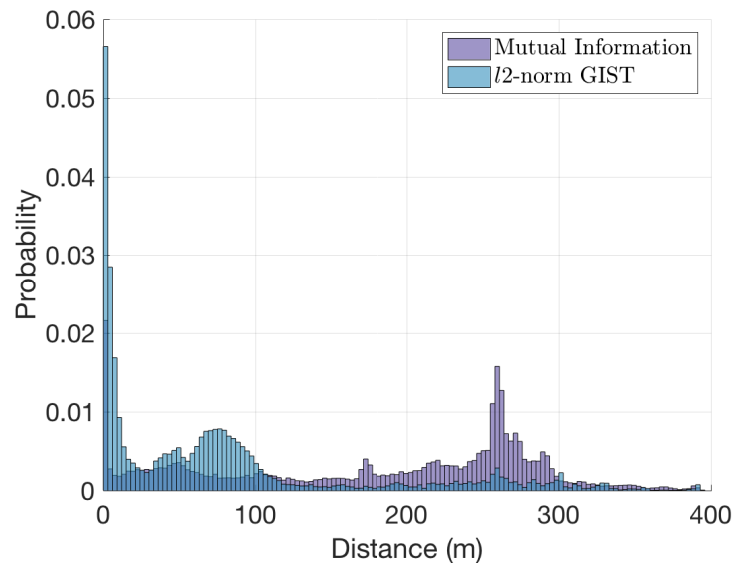
This logically holds: for our dataset we expect the temporally adjacent frames to be very similar in appearance, with the exception of turning corners. Using mutual information is less beneficial: this can boost performance slightly over a generic detector, but less than the gains when using spatial sequences or GIST to group the images used to create a model. This can be attributed to the efficacy of the similarity metric, indicated by the strength of correlation between spatial distance and image similarity. In Figure 3.8b we note that the correlation between mutual information and distance plateaus at 5 metres, unlike with GIST. The mutual information models are also trained on data which is much further from the source frame than GIST, demonstrated in Figure 3.8. We expect that a better metric would be using mutual information between each colour channel rather than the intensity image, though this is unlikely to offer any gains over a temporal sequence or the simpler GIST comparison.

Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
Baseline: SVM+ACF	0.483	0.485	0.473
Baseline: FPDW [33]	0.481	0.440	0.491
Baseline: MSCNN [31]	0.588	0.371	0.579
PD Spatial: N=1	0.622	0.354	0.592
PD Spatial: N=10	0.689	0.292	0.651
PD Spatial: N=100	0.689	0.288	0.672
PD l^2 GIST: N=10	0.678	0.303	0.652
PD l^2 GIST: N=100	0.689	0.300	0.666
PD Mut. Inf.: N=10	0.594	0.382	0.584
PD Mut. Inf.: N=100	0.656	0.347	0.640

Table 3.2: Aggregate performance metrics comparing our bespoke place dependent (PD) detector models to a number of generic baselines. A higher number is better for Average Precision and F1 Scores, lower is better for Log-Average Miss Rate. The $N = 1$ case is the same for all methods as this is simply the model trained on the closest localised frame.



(a) 10 frame models



(b) 100 frame models

Figure 3.8: These histograms show the spatial distance from each frame used during model construction to the model’s index frame. This highlights the differences in models’ training data when using different image similarity metrics. When using the nearest spatial frames, a 100 frame model will on average have been trained on data up to 22 metres from the index frame (Figure 3.7). When using both GIST and mutual information this can be up to 400 metres.

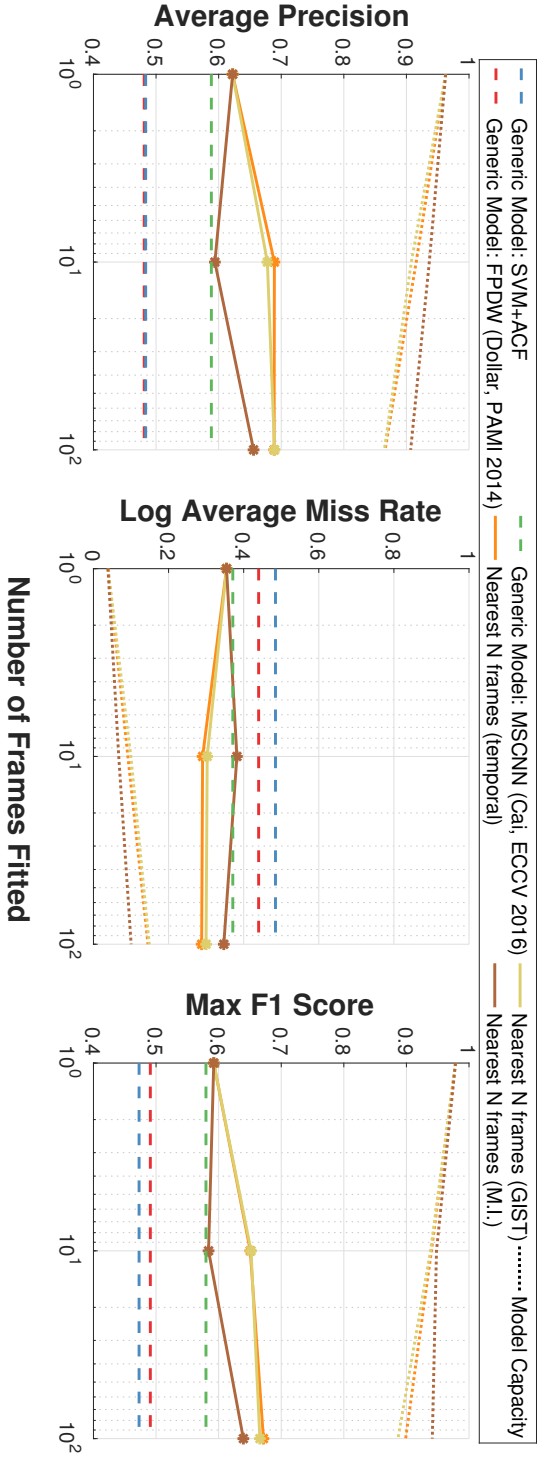


Figure 3.9: Aggregate performance metrics for fitting models to the nearest N frames based on image similarity metrics and spatiotemporal sequences. We can improve over the generic baseline model, with similar performance when using detector models created by grouping images by the l^2 -norm between GIST descriptors and spatial sequences. Note that our generic model (SVM+ACF) and FPDW score almost identically in average precision, and the dotted lines represent performance on the training data as a proxy for model capacity.

3.5 Local Expert Detection Discussion

In robotics we have the opportunity to exploit knowledge of the world around our robot to improve performance of vision systems. Here we demonstrate how prior knowledge about what a robot will see (by repeated operation along the same route) can be used to boost performance beyond what a more complex state-of-the-art object detector can achieve. However, defining the place of operation is not trivial for mobile robots. We present an assessment of approaches to building bespoke place dependent detector models, investigating which factors influence performance.

Our results indicate that we can heavily leverage a strong spatiotemporal prior, tightly fitting models to a narrow window of operation. Relaxing the need for generality aids performance in bespoke place-fitted methods as we can ensure our models fit a smaller negative background class, hence avoiding the trade-off between model capacity and generalisation. When using appearance to build models, we are able to achieve effectively the same performance on groups of similar images.

From a practical view, we suggest using local expert detector models built using spatial sequences of images. Localisation provides a useful proxy for image appearance, and as roboticists we have relatively robust, well developed methods for localisation. We did not observe any significant improvement in using image appearance metrics directly, hence we prefer the simpler approach with no further computation required.

We see significant potential for this approach to be coupled to experience-based visual localisation systems [20, 87–89], using their ability to localise to many parallel ‘experiences’ of the same location. These bespoke detector models are very sensitive to the data they are applied to at test time: if the scene content changes substantially from the environment they were trained on, it is likely that the detector would generate false positive errors due to the change in background class. This brittleness could be addressed by using models constructed from appearance rather than spatiotemporal sequences, or more simply by using the ability of experience-based localisation to localise across different appearances of the same place, with detector models associated with each experience of the same spatial location.

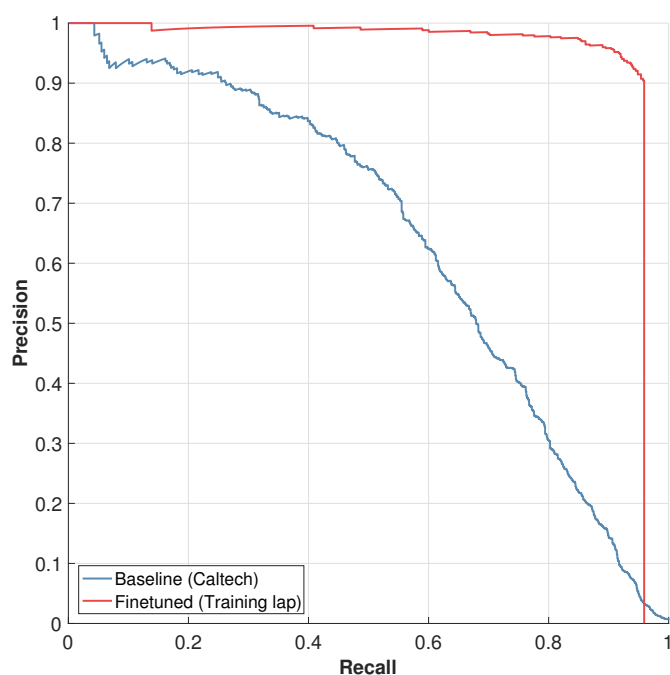
3.5.1 Higher Capacity Local Expert Detectors

In order to validate the influence of model capacity when using local expert detectors, we repeat the experiments in 3.3 using a higher capacity pedestrian detector. The method and data are as per Section 3.3.1, training a model on ground-truth labelled data from a given driving route and evaluating the model on a second instance of that same driving route. For this experiment we selected *MSCNN* [31], a high capacity convolutional neural network detector considered state-of-the-art. The initial *MSCNN* model was trained on the Caltech dataset [62], provided by the authors. *MSCNN* has a two-stage training procedure. The first phase randomly samples negative patches from the training images, whereas the second performs hard negative mining (HNM). This is split as the authors state that the model can fail to converge if HNM is performed from the outset. To construct a local expert detector using *MSCNN* we repeat the second phase of training using data from the robot’s operating route (i.e., the same route as the test dataset).

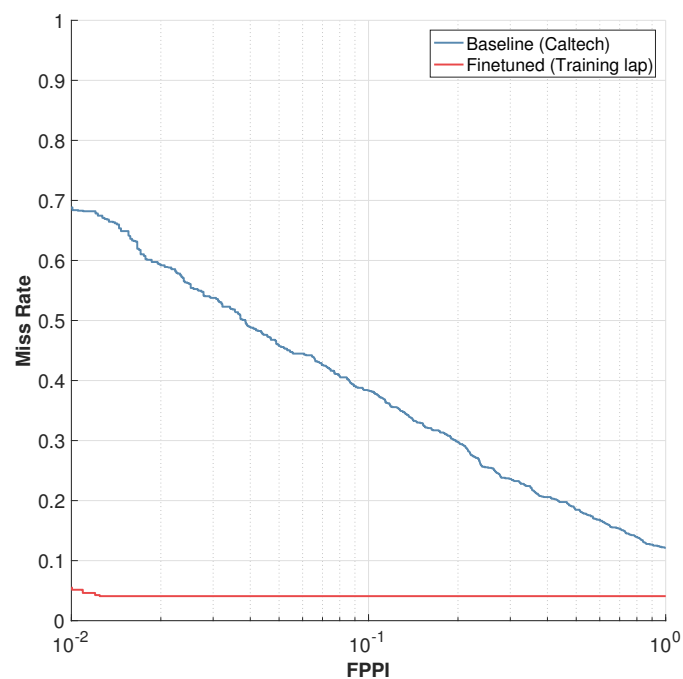
Figure 3.10 shows the performance improvement of the baseline *MSCNN* detector once fitted to the local operating route, showing a substantial increase fitting to data derived from the full driving route. This detector demonstrates the ability to fit to the entire driving route without loss of performance, unlike the much lower capacity SVM detector (which has $3K$ parameters compared to *MSCNN*’s $300M$).

Adapting a neural network to a different target domain is referred to as ‘fine-tuning’ a model [100]. For example, when taking a model trained for a task on one dataset and applying it to the same task on a different dataset, we may ‘fine-tune’ the model to improve performance by training it on a training subset of this second dataset.

Creating local expert models is fundamentally this same process: creating a detector model fitted to operation in a given target domain. Here, the target domain is the robot’s operating environment.



(a) Precision - Recall



(b) Miss Rate - False Positives Per Image

Figure 3.10: Detection performance improvements on the *MSCNN* detector models: the general baseline model, and a model fine-tuned to create a local expert detector.

3.6 EBC Deployment: A Local Expert Detection System

The preceding sections of this chapter showed that we can leverage localisation knowledge to improve detection performance, circumventing limited model capacity by using bespoke models for the operating environment. This approach mitigates the impact of model capacity by fitting to a smaller background (negative) set with less variability.

The experiments in Section 3.3 and 3.4 sought to identify the upper bound for expected performance, with many variables being constrained to ensure near optimal results. These used ground truth labels, and the training data differed from the test data by less than five minutes in time on the same driving route. In reality we cannot expect to have all these factors working in our favour.

We want to move from an idealised scenario where these variables are controlled to a deployable, self supervised system, coupling place dependent detector models to a experienced-based visual localisation system.

Experience-based visual localisation systems such as [20, 21, 88, 89] construct locally consistent topometric maps, where geometric transforms are defined between key frames in a map graph. These systems create new vertices in the graph if they cannot localise to the existing map, joining back to the original graph when successful localised. This approach has the benefit of being able to handle multiple parallel experiences of the same spatial location, accommodating changes in image appearance which would otherwise degrade localisation performance.

We integrate the object detector outlined in Chapter 2.3.1 with the visual localisation system in [21]. This section outlines the system structure, and presents results on the earlier scenario in Section 3.3 using this self-supervised approach.

3.6.1 System Outline

Our local expert object detector tightly couples object detection to a visual localisation system [21], illustrated in the diagram in Figure 3.11.

This visual localisation system has a topometric map, structured as a graph where each vertex is a key frame. Each edge between two vertices has an associated geometric transform, relating the pose of the two adjacent key frames. The map is implemented as a database using the general purpose database interface in [101], with a common software interface for both PostgreSQL or SQLite database backends. SQLite is used for performance reasons.

We extend this topometric map using this same database interface as this allows arbitrary data structures to be associated with vertices and edges in a graph. For practical reasons associated with read-write permissions, the object detector model database is structured as a clone of the SQLite map, rather than additional entries in the same database.

Each key frame (vertex in the map graph) has a model associated with it. The creation of these models is important, and is discussed later in Section 3.6.2.

When processing an image at runtime, the expert detector accepts localisation information from the localisation system running simultaneously. This provides a reference to a key frame in the map. The detector queries the model database for this key frame identifier and retrieves the model for this locale, applying it to the current frame using the multi-scale sliding window detector from Section 2.3.1. In the event that localisation fails (or the key frame has no associated model), the detector defaults to a generic model.

3.6.2 Model Initialisation

Model initialisation is a purely offline process, constructing local expert object detector models suited to the robot’s operating environment in a self-supervised manner.

We adopt a similar approach as before when creating local expert models; the key difference here is that we do not have ground truth labels. We saw in Section 3.3 that using models fitted to image sequences of 10-100 frames offered the greatest performance boost, alleviating issues of model capacity while minimising misclassifying the background class.

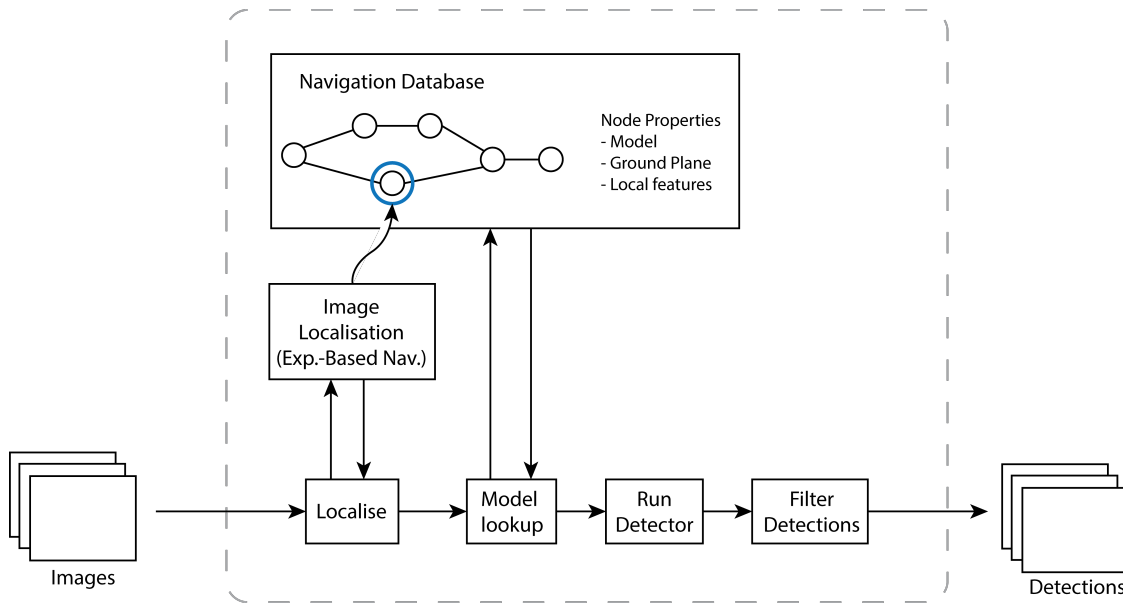


Figure 3.11: The Local Expert Detector System Architecture. A standard object detector as in Chapter 2.3.1 is tightly coupled with a visual localisation system. Each image is localised, and the model associated with the localised key frame is retrieved from the database and applied to the input image. In the event that localisation fails, the system defaults to a generic object detection model rather than a local expert.

The model initialisation therefore uses the same approach, fitting detector models to a sequence of 80 images centred on the map key frame. This corresponds to a four second collection of images at a camera frame rate of 20 Hz. Each model is created using a cropped training dataset for the desired class, training on the positive samples and a small seed negative set. For a pedestrian detector, we use the INRIA pedestrian dataset [47], with 2416 positive samples and 10 negative samples randomly drawn from the each negative image. This seed set of negatives increases the speed of model training as it reduces the number of negative mining iterations required before convergence.

We then perform experience-based classification (the self-supervised hard negative mining method from Chapter 2) using a scene filter to select high scoring negative samples to augment the training dataset. The model is then retrained using the augmented training data. This method is very similar to Chapter 2.3, with the key difference here being the way we manage data from the negative class. The training process runs to a maximum of 20 iterations, or until 3 passes

of training occur with no further negative data mined from the training image set (i.e., convergence). Typically, we found that the models train in eight to ten passes of training for a sequence of 80 frames.

This process is heavily parallelised, as the model for each map frame can be generated independently of the key frames. This is implemented as multithreaded application with a variable sized pool of worker threads, able to scale with the computational resources available. On a 16 core server, the map used in the experiments later in this chapter will be fully processed overnight, generating a model for each of the 1818 map key-frames. This process runs offline, hence this level of pre-processed computational time is not prohibitive.

3.6.3 Extending the Scene Filter: Robot Dreaming

Earlier in Section 2.3.2, we performed model initialisation using the scene filter based solely on spatial heuristics. However, this earlier work started with a baseline negative training dataset (usually derived through HNM), whereas this approach here relies on a very targeted negative training dataset derived solely from the operating environment.

Spatial heuristics alone are unable to filter out false positives of plausible size for the desired object class. This means that the model initialisation process will run to convergence on the map image data, yet will generate many false positives on the original training images as the model has not captured the full extent of the background class for those images. An example of this could be a pedestrian detector which returns a positive detection on a postbox: this is approximately the right scale as a potential pedestrian, thus we cannot discount this through simple heuristics alone. We need additional information to aid this process beyond geometric plausibility.

Model initialisation is an important process, but one which does not generally need to be performed online or in real time. This opens up new possibilities for our mobile robots to generate models when idle, for example, when an autonomous

car is parked up overnight with spare time and computational resources. We refer to this offline model generation approach as *robot dreaming*.

As this model initialisation is performed when the robot is idle, we can utilise computationally expensive algorithms. We need a method for self-supervised data mining of image data, one which can augment the simple spatial heuristics.

We opted to use image segmentation as a supervisory signal for the filter. We selected *Full Resolution Residual Network* (FRRN) segmentation [102], the best performing image segmentation algorithm on the Cityscapes segmentation dataset [103] at the time of writing. This is implemented in Python with Theano, and requires a high performance NVIDIA GTX Titan graphics processor to run. The per-frame computation time is in the order of seconds; too slow for mobile robot operation in real-time but perfectly adequate when the robot is idle, post-processing prior data.

FRRN generates a segmented image, where each pixel is allocated one of thirty discrete classes defined in the Cityscapes dataset. These classes include a variety static objects (fence, sidewalk, road, etc) and dynamic classes (pedestrian, car, truck, bus, etc.). For our work on pedestrian detection we consider the *person* and *rider* (cyclist) classes, as this matches the cropped INRIA training data.

We add a third scene filter criteria to the geometric filters in Section 2.3.2. This filter looks at the fraction of the candidate detection segmented as the desired object class.

We first get the cropped patch $S(d)$ from the segmented image S defined by the candidate detection $d = [x_1 \ y_1 \ x_2 \ y_2]$. This patch is of size $u_p \times v_p$, where $u_p = x_2 - x_1$, $v_p = y_2 - y_1$. We then define the set of classes we are interested in, C . The filter is controlled by a threshold parameter $\gamma \in [0, 1]$: if less than this fraction of area of the detection belongs to the classes in C , we reject the candidate detection.

$$A_{patch} = u_p v_p \quad (3.1)$$

$$A_{class} = \sum_{v=1}^{v_p} \sum_{u=1}^{u_p} (S(d)_{uv} \in C) \quad (3.2)$$

$$f_{segmentation} = \begin{cases} 0 & \text{if } A_{class}/A_{patch} < \gamma \\ 1 & \text{if } A_{class}/A_{patch} \geq \gamma \end{cases} \quad (3.3)$$

The image segmentation filter performance will vary depending on the choice of γ . Figure 3.13 shows this performance on candidate detections generated by the generic SVM+ACF detector model from 3.3.2 on the labelled *North Oxford January* dataset. If we were able to run this filter live for real-time operation, we might choose the knee of the curve in Figure 3.13b, maximising the filter efficacy.

However, we want the filter to be as conservative as possible as we are it as the supervisor signal for negative data mining. We set $\gamma = 0.0$, meaning that if even one detection window is segmented as one of the object classes we are interested in, it is accepted as valid by the filter. This parameter selection incorrectly filters 6% of true positives for correctly filtering 64% of the false positives. Qualitatively, the filtered true positives are commonly in images with motion blur, or where the target object is small in size.

We combine this filter with the previous two in Section 2.3.2, requiring a candidate detection d to pass all three filters before being considered a valid detection. In practice this is implemented sequentially: we only compute the image segmentation filter results on candidate detections which have passed the two preceding filters.

$$f_{scene}(d) = f_{horizon}(d) \wedge f_{scale}(d) \wedge f_{segmentation}(d) \quad (3.4)$$

Combined, this multi-stage filter offers sufficiently high performance to be used as a self-supervisory training signal for object detector models.

To compare the performance of various filter stages we compare a number of metrics, each based on the assignment of positive and negative labels to the candidate detections.

1. True Positive Rate (TPR, or recall), $TPR = \frac{TP}{TP+FN}$
2. True Negative Rate (TNR, or specificity), $TNR = \frac{TN}{TN+FP}$
3. Precision, $Precision = \frac{TP}{TP+FP}$
4. False Negative Rate (FNR, or Miss Rate), $FNR = \frac{FN}{FN+TP}$

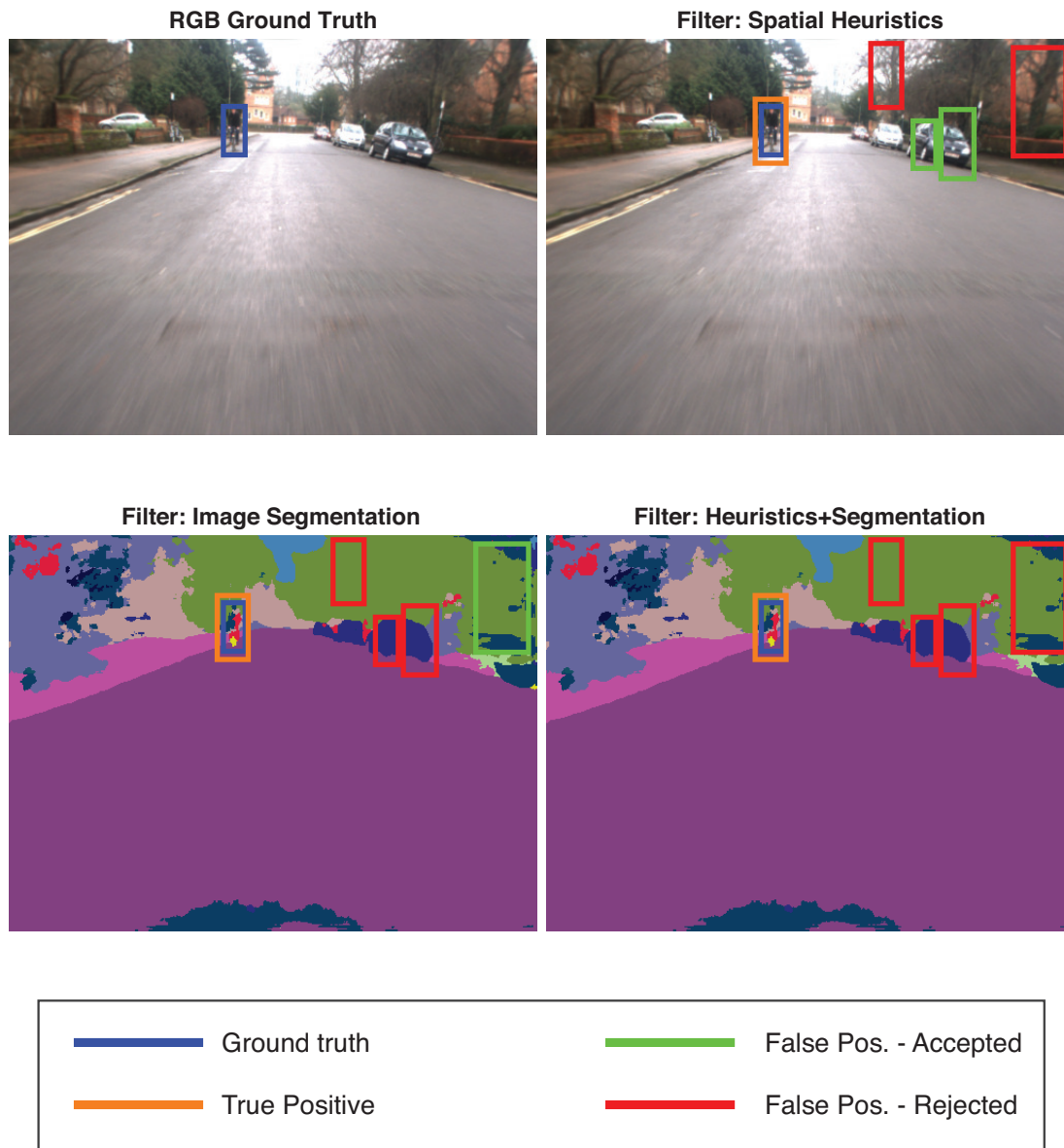


Figure 3.12: Further scene filtering using offline image segmentation. This provides complementary information about patches mined from the scene, enabling us to filter patches which pass geometric checks.

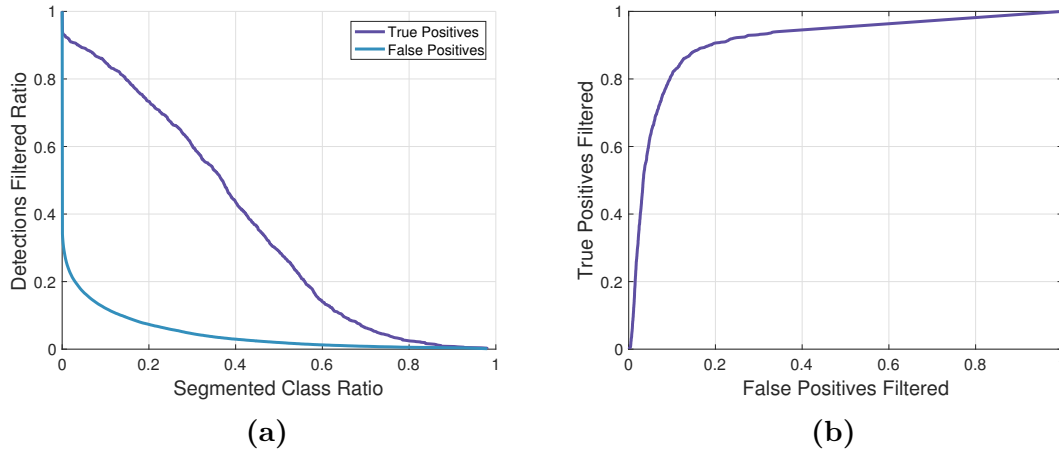


Figure 3.13: Image segmentation filter performance for a pedestrian detector model run on the labelled *North Oxford January* dataset. Detections are filtered based on the fraction of the window corresponding to the object class (pedestrian, in this case). Figure 3.13a shows the filter performance on true and false positives from a detector model. Figure 3.13b shows the same data in a different format, plotting the True Positive Rate against False Positive Rate as the filter parameter varies. If we were to select the optimal filter performance, this would correspond to the knee of the curve. However, we prefer the filter to be overly conservative to mistakenly filter the minimum number of true positives. Hence we choose set this parameter to 0, corresponding to the y-axis in Figure 3.13a.

5. Accuracy, $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

6. F-measure, $F_1 = \frac{2TP}{2TP+FP+FN}$

We want our filters to have extremely high recall (TPR), permitting every potential positive detection through the filter while rejecting as many false positives as possible, maximising the number of background patches rejected by the filter - measured by the true negative rate (TNR). This is a conservative approach (hence the filter output has extremely low precision), but necessary as we wish to avoid any potential semantic label drift.

Table 3.3 outlines these performance metrics on an INRIA-trained pedestrian detector model from earlier in Chapter 3 on the *North Oxford January* dataset. All candidate detections (those with a score over 0.0) are passed through three filters: the simple spatial heuristics used in Section 2.3.2, the image segmentation filter, and the combined spatial heuristics and image segmentation. The combined filter offers a healthy improvement to filter performance with minimal errors.

Metric	Raw Detections	Spatial Heuristics	Image Segmentation	Combined SH+IS
TPR	1.0	0.857	0.940	0.859
TNR	0.0	0.543	0.640	0.714
Precision	0.024	0.044	0.060	0.069
FNR	0.0	0.143	0.060	0.141
Accuracy	0.024	0.551	0.647	0.717
F-measure	0.047	0.084	0.114	0.127

Table 3.3: Filter performance on detections from the generic INRIA pedestrian detector from earlier chapters applied to the *North Oxford January* dataset, with 1030 true positive detections and 41853 false positives at the decision boundary, where the SVM score threshold is 0.0. Here we want our filters to have extremely high recall (TPR) and true negative rate (TNR), permitting instance of the positive class through the filter while rejecting as many background patches as possible.

Figure 3.14 shows an example of the model initialisation process in Section 3.6.2 using the expanded scene filter, combining spatial heuristics and image segmentation to achieve more informed data mining. The example model initially performs poorly as it has a very small negative training set. We use this scene filter to reject data which is either in an inappropriate part of the scene (above the horizon), an inappropriate scale (too large or small for the expected class), or doesn't match the class allocated by preprocessed image segmentation. After five passes of data mining a given image sequence, the model has converged, generating no detections the filter is able to reject.

Figures 3.16 through 3.17 shows a random sample of the errors made by each filter stage, providing a qualitative understanding of the key failure modes of each filter, and insight into how each filter might be improved.

The true negative errors (true candidate detections rejected by the filter) introduced by the spatial heuristics filter are largely attributable to errors in the object scale estimate, frequently due to poorly localised bounding boxes. A small change in the detection localisation (how well the box is centred on the true location) can result in a comparatively large change in the scale estimate, depending on where the object is in the scene. The dominant error mode for false positives (false candidate detections permitted by the filter) is dictated by the very conservative nature of the filter: all detections of a plausible scale are permitted.

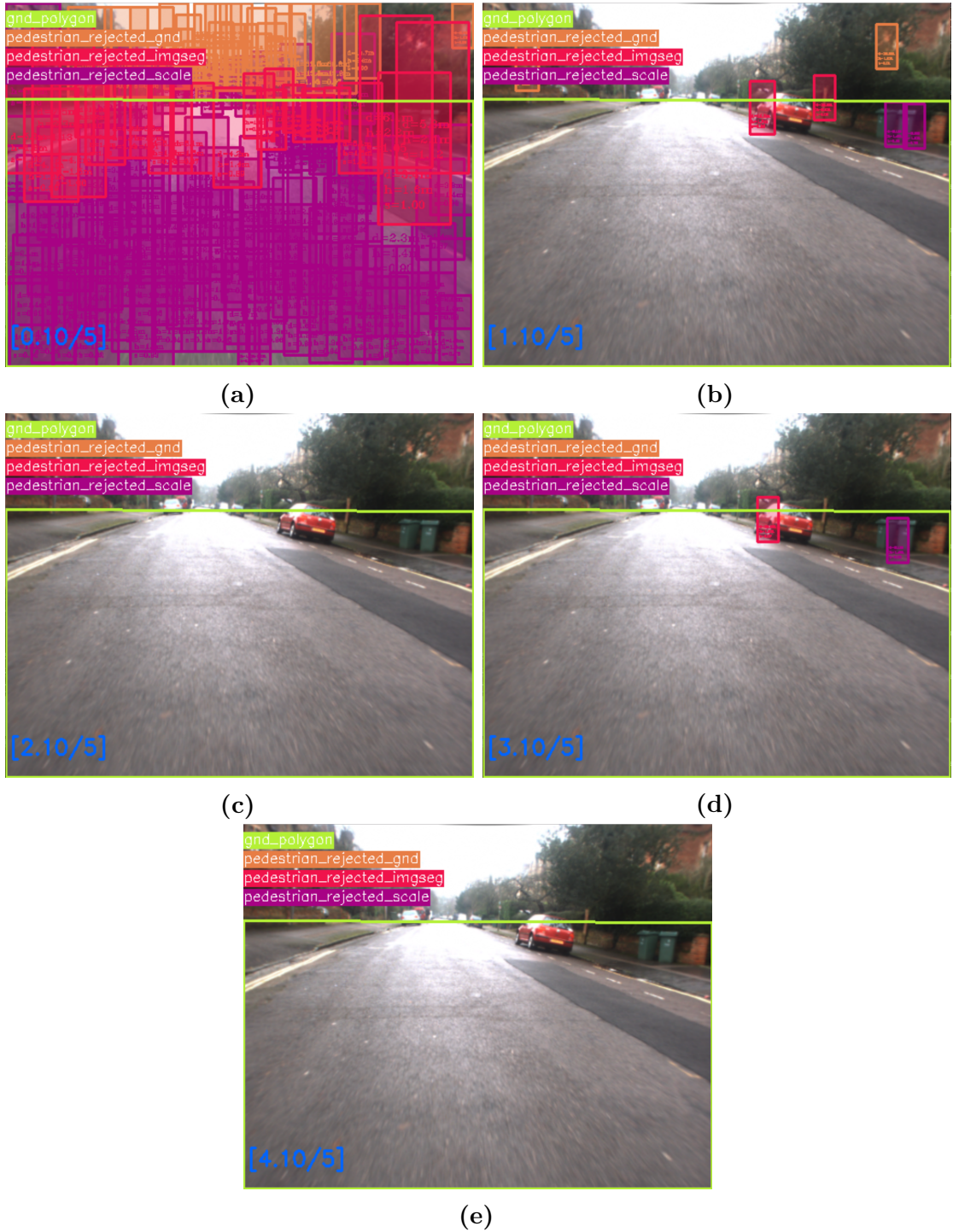


Figure 3.14: An example of model initialisation data mining using the multi-stage scene filter, where mined negative samples are coloured by the filter selecting them. The horizon filter is in orange, the scale filter is in purple, and the image segmentation filter is in red. The first model in (a) has only been trained on the positive training set and a small seed negative set, resulting in many rejected false positives. These are aggregated and used to augment the training dataset. The model is retrained and the process repeats. After five passes of data mining we see the model converge for this example in (e).

We see similar key error modes in the image segmentation filter. False positives can be grouped into two distinct categories: firstly, poorly localised detections (patches partly containing the target object class), and secondly, false patches with image segmentation noise (patches not containing the target class). The inverse is true for true negative errors, where image segmentation consistently misses the same individual across multiple frames.

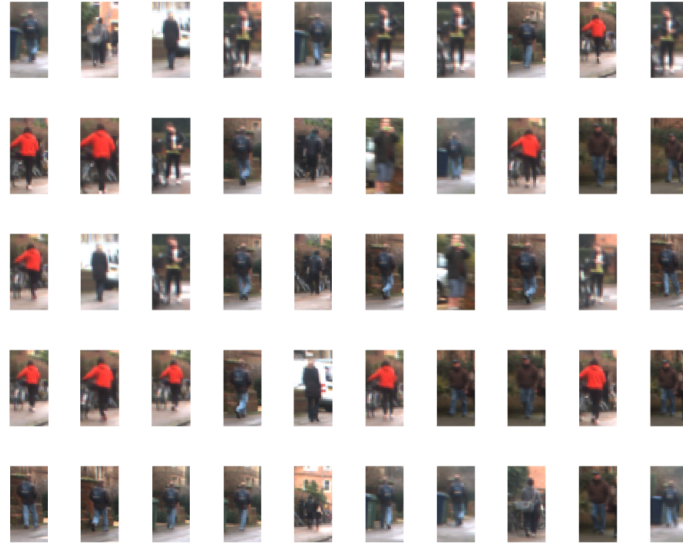
It is likely that these errors could be addressed in future work, though the filter accuracy in Table 3.3 is likely to be sufficient for initial self-supervised training. The true negative errors could be addressed by adding a further offline filtering stage, using a computationally complex object detector (rather than image segmentation) to evaluate candidate detection patches rejected by the filter. This could be implemented using the detection stage of many modern neural network detectors, such as Faster RCNN [30]. We could address false positive errors due to image segmentation noise through a better understanding of uncertainty in [104]. We could also further improve the spatial heuristic filter, varying the permissible scale estimate depending on the location of the object in the scene (for example, distance from the camera).

3.6.4 Results

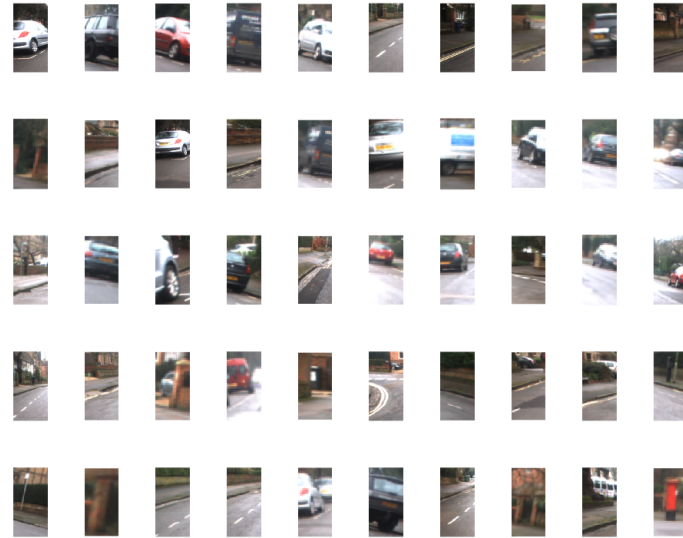
We compare the performance of the detector built from unlabelled data drawn from the same route as the *North Oxford January* test dataset to the ‘ideal’ detector model described earlier in Section 3.3.

The performance curves in Figure 3.18 and aggregate metrics in Table 3.4 show that this self-supervised system is still able to offer a substantial improvement over the generic detector, though naturally not to the same extent as the ‘ideal’ scenario using ground truth labels to hard negative mine models for every single frame in the mapping traverse.

Notably, the place dependent detector does not perform as well as the generic model for the most high scoring detections. This corresponds to the upper left corner of the Precision-Recall curve.

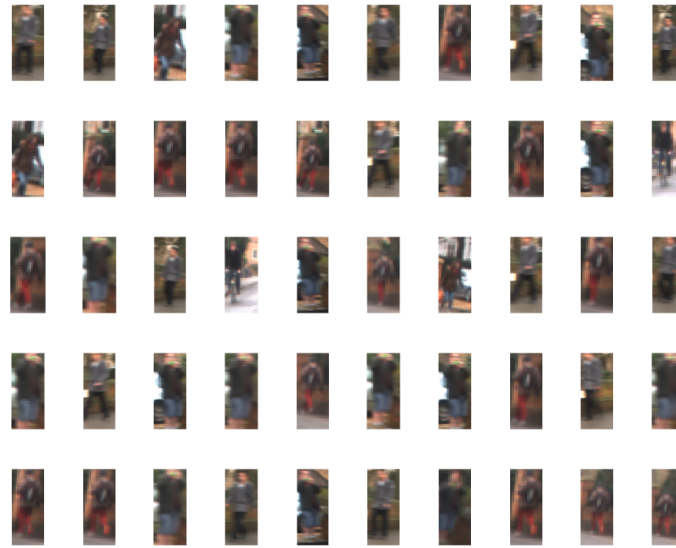


(a) True negative errors

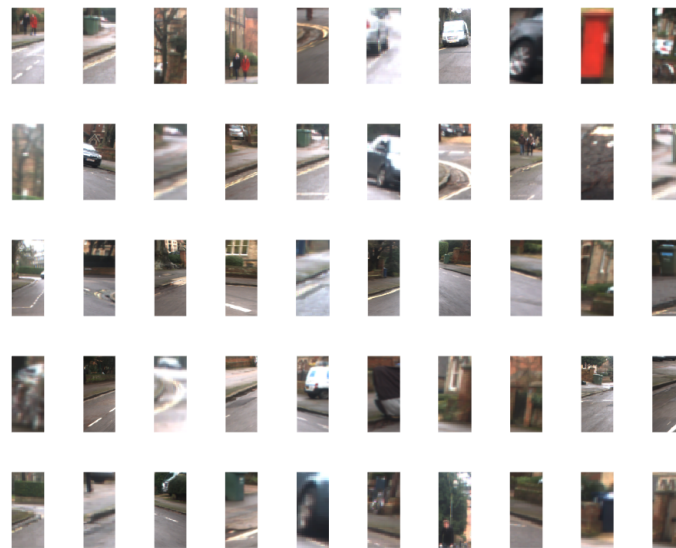


(b) False positive errors

Figure 3.15: A random sample of key errors made by the spatial heuristic scene filter on pedestrian detections from the *North Oxford January* labelled dataset. The true negative errors introduced by the spatial heuristics filter are largely attributable to errors in the object scale estimate. This is often affected by bounding box localisation, where a relatively small offset in the image frame can result in a comparatively large change in the scale estimate. The dominant error mode for false positives (false candidate detections permitted by the filter) is dictated by the very conservative nature of the filter: all detections of a plausible scale are permitted.

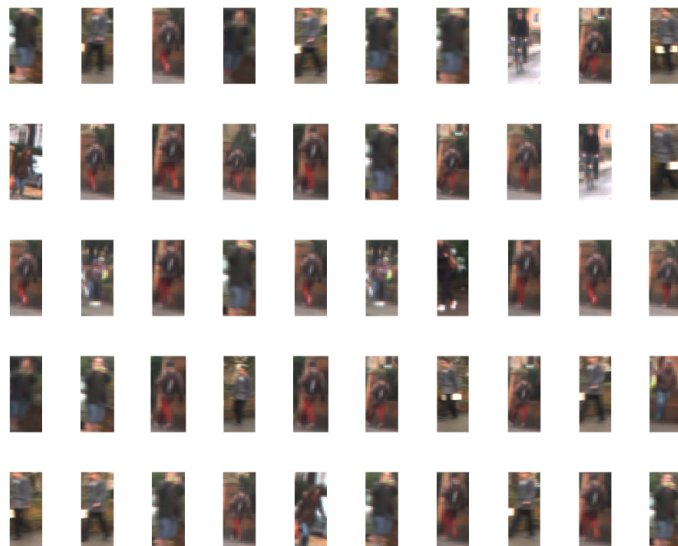


(a) True negative errors



(b) False positive errors

Figure 3.16: A random sample of key errors made by the image segmentation scene filter on pedestrian detections from the *North Oxford January* labelled dataset. False positive errors can be grouped into two distinct categories: firstly, poorly localised detections (patches partly containing the target object class), and secondly, false patches with image segmentation noise (patches not containing the target class). The inverse is true for true negative errors, where image segmentation consistently misses the same individual across multiple frames.



(a) True negative errors

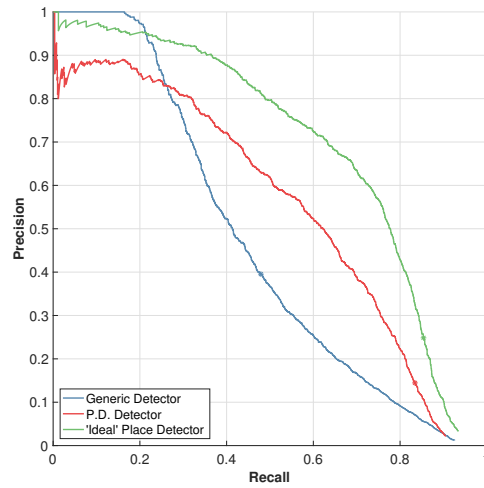


(b) False positive errors

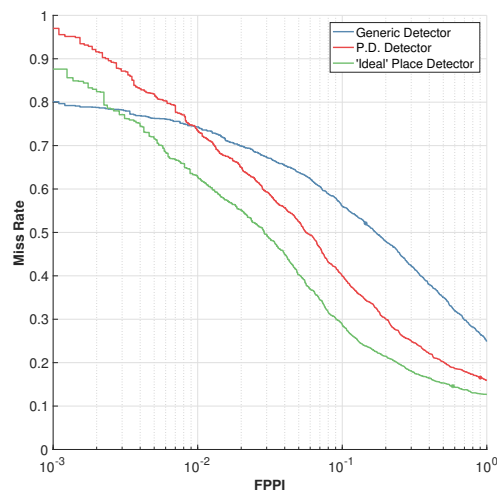
Figure 3.17: A random sample of key errors made by the combined spatial heuristic & image segmentation scene filter on pedestrian detections from the *North Oxford January* labelled dataset. This has less obvious structure than the simpler filters, but can be largely attributed to noise in the image segmentation, or bounding box localisation causing scale estimate errors.

Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
Baseline: SVM+ACF	0.483	0.485	0.473
‘Ideal’ PD Detector	0.689	0.292	0.651
Self-supervised PD Detector	0.547	0.372	0.567

Table 3.4: Aggregate performance metrics comparing our self-supervised place dependent detector to a generic detector model, and the ‘ideal’ place dependent (PD) detector for the same dataset (using ground truth labels to build models for every image, rather than key frames).



(a) Precision - Recall curve: higher is better



(b) Miss Rate - False Positives Per Image (FFPI) curve: lower is better

Figure 3.18: Detector performance curves for the self-supervised experience-based detector.

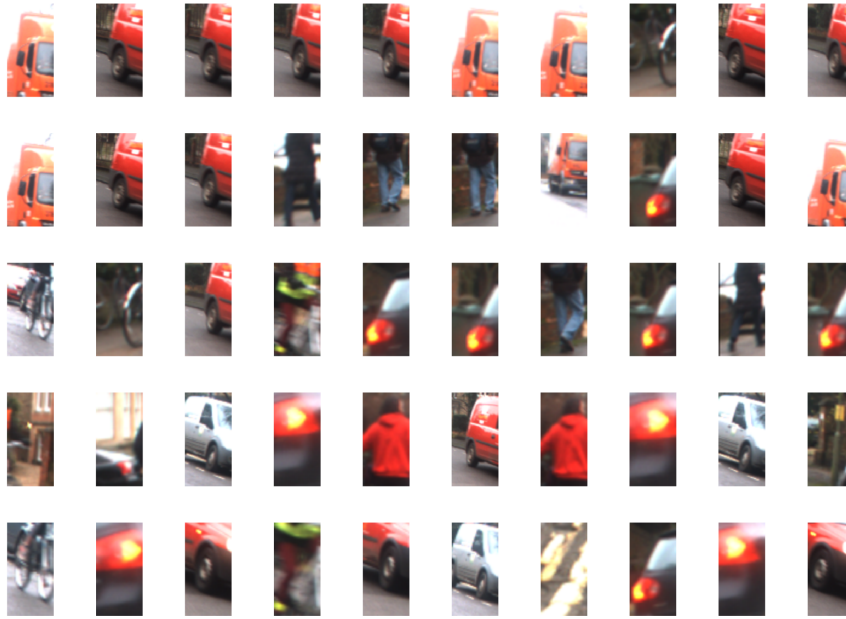
This provides some intuition for how brittle these place dependent models are. The models are sensitive to changes in the background data, and are liable to generate false positives when presented with data which deviates from the scenes they were trained for.

A key example of this is dynamic objects in the robot’s operating environment. A qualitative analysis of the top scoring false positives from the test dataset (Figure 3.19) shows that the top scoring false positives for the place dependent detector are largely other dynamic objects (cars, trucks) or poorly localised detections (partially overlapping with a pedestrian). In contrast, the generic model’s top scoring false positives on the same test dataset consist of a mix of static patches (buildings and windows) and dynamic objects (cars).

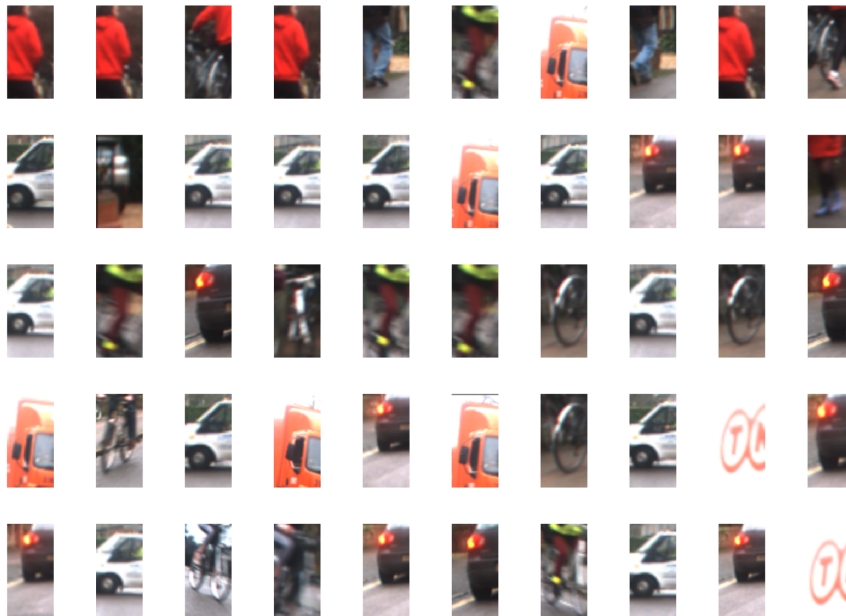
The influence of dynamic objects on false positive detections is further verified using image segmentation. We investigate this effect by computing the fraction of detection area occupied allocated to a dynamic object class. This is the same process as with the image segmentation filter, but with an expanded list of object classes.

We compute the dynamic class areas for false positive detections from three pedestrian detectors on the same dataset: the generic INRIA pedestrian detector used previously, the ideal place-dependent detector trained in Chapter 3.3, and the self-supervised place-dependent detector trained here in Chapter 3.6.2. We construct a histogram from these dynamic class areas, shown in Figure 3.20. Table 3.5 outlines the percentage of false positives generated from the static environment (defined as $< 5\%$ detection area from dynamic objects). The results show a substantial reduction in the error rate on the static environment, from 85.3% with the generic detector to 49.0% with the self-supervised expert place-dependent detector.

This confirms the qualitative results, indicating that the local-expert detector models’ errors are influenced by the presence of other dynamic objects sharing their operating environment. This intuitively makes sense: they have been trained on data drawn from this environment hence the static background class will be very similar, changing with appearance rather than appearance and structure.

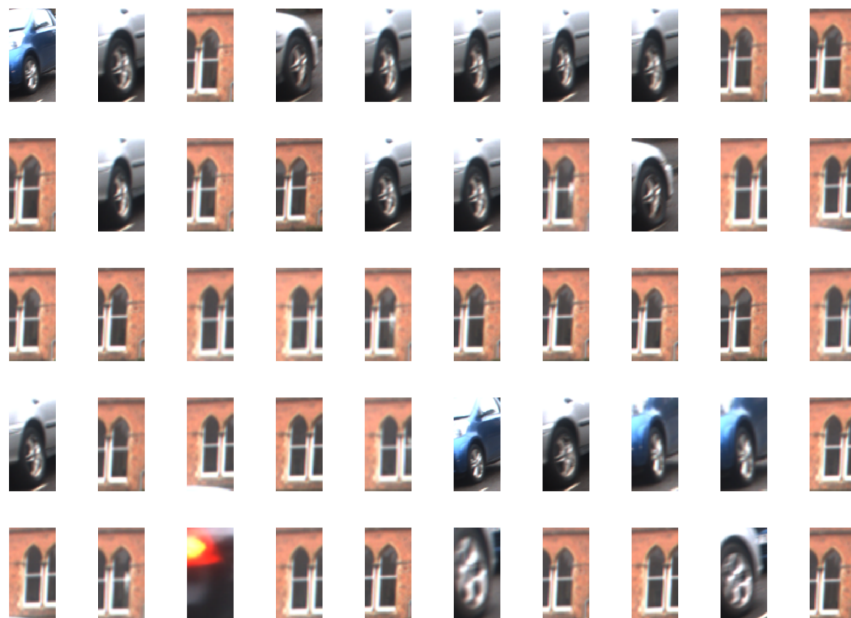


(a) Self-supervised Experience-based Detector Model



(b) 'Ideal' Experience-based Detector Model

Figure 3.19: Top scoring fifty false positives for pedestrian detectors run on the *North Oxford January* labelled dataset. The experience-based detector models' errors (both 'ideal' ground-truth and self-supervised) tend to be dynamic objects or poorly localised pedestrians (by definition also dynamic objects), whereas the generic model's mistakes include patches from the static background as well as objects moving through the scene.



(c) Generic Detector Model

Figure 3.19: Continued: Top scoring fifty false positives for pedestrian detectors run on the *North Oxford January* labelled dataset. The experience-based detector models’ errors (both ‘ideal’ ground-truth and self-supervised) tend to be dynamic objects or poorly localised pedestrians (by definition also dynamic objects), whereas the generic model’s mistakes include patches from the static background as well as objects moving through the scene.

Model	False Positives from the static environment (%)
Generic SVM+ACF	85.3
‘Ideal’ Expert Detector	58.5
Self-supervised Expert Detector	49.0

Table 3.5: The percentage of false positive detections classed as static, where the fraction of the detection patch that belongs to dynamic classes in image segmentation is less than 5%. These data points correspond to the first (left-most) bar in the histogram in Figure 3.20.

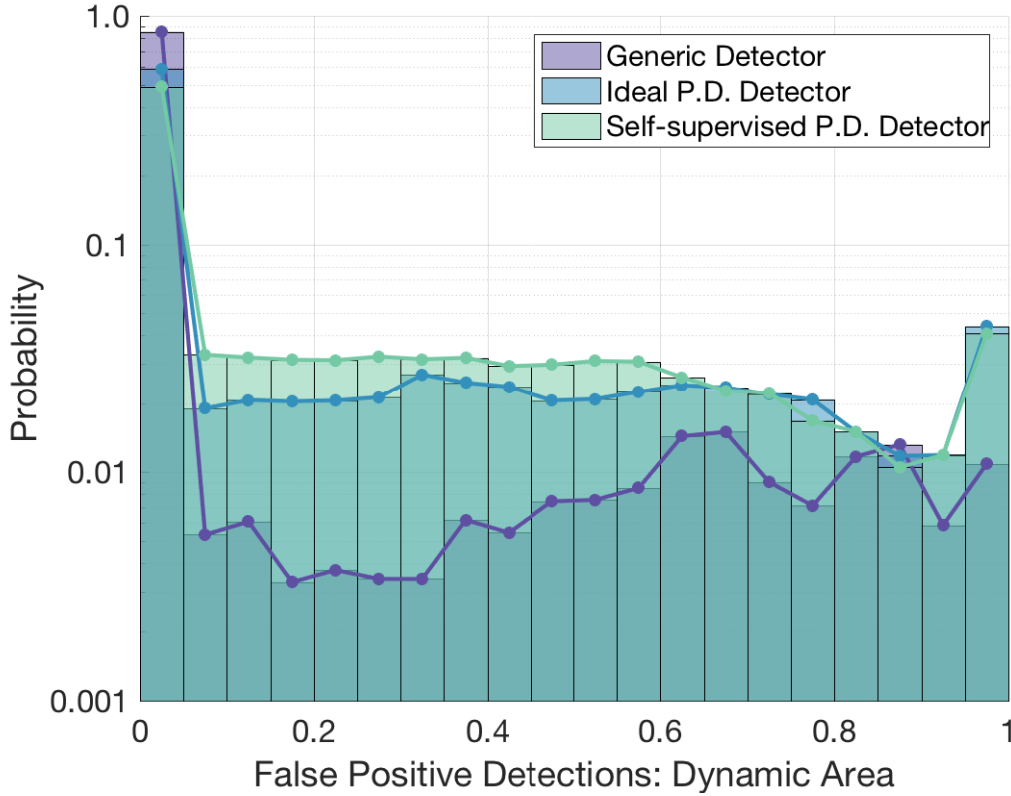


Figure 3.20: Histogram showing the ratio of detection area belonging to a dynamic object class for false positive detections on evaluation dataset. The place dependent models (blue, green) demonstrate a lower rate of false positives on the static environment (the left-most bar), and a higher rate of false positives on the dynamic environment (the remainder of the graph). The rate of static false positives is outlined in Table 3.5. These local expert models are sensitive to changes in the background class which deviate from the training scene, hence yielding a higher incidence of false positives on dynamic objects compared to the generic model.

3.6.5 Discussion

The results validate the self-supervised approach to building place dependent models, demonstrating an improvement in detection performance. This approach couples an experience-based visual localisation system with fitted object detector models. Using spatial heuristics and pre-processed image segmentation, we construct models for each key frame in a topometric map by data mining a short sequence of images centred on the key frame. At run time, localisation to the map is used to retrieve the model associated with the localised key frame, applying the detector model to the image.

This system offers a substantial improvement over a generic model in our test cases, though naturally not as substantial gains as the ideal, supervised scenario. The self-supervised models are effective in the environment they were trained for but can be brittle. We rely on the experience-based localisation system to accommodate changes in appearance, though significantly more labelled data is required to comprehensively test the capacity of the detector models (and localisation) to handle appearance change.

We see that the place dependent models are susceptible to generating false positives on other dynamic objects in the environment. Logically, this makes sense: the model has not been trained to generalise and has not seen data like this before. This is likely to mean that performance will degrade in highly dynamic environments, though this could potentially be mitigated by separating the negative training data into static and dynamic classes.

3.7 Conclusions

Fitting an object detector to a robot’s place of operation is a useful way to boost detection performance. The method presented here trains detector models using positive training data from a publicly available dataset, and negative training data derived from a short sequence of images from initial mapping data. We use visual localisation to retrieve the appropriate model for each frame during operation.

We evaluated this approach in an ideal scenario, where we trained a model for every frame in the mapping route, using annotated ground truth labels to construct detector models with Hard Negative Mining on the map image sequence. This demonstrated that fitting models to a narrow background class is an effective mechanism for mitigating the effects of having limited model capacity. For our problem and model (the underlying classifier and feature types), we found that using shorter sequences of 10-100 frames was appropriate in constructing these models. Increasing the number of frames used for model construction beyond this range degraded performance due to overfitting a limited capacity model.

This approach provided an informative upper bound for what could hope to achieve: we can use a computationally lightweight detector to achieve better performance than a much more computationally complex (and often prohibitively slow) state of the art object detector [31].

Deploying this method on operating robots requires removing a number of constraints we imposed for this ideal scenario: we lack ground truth annotations, and we use key frames in topometric maps rather than maintaining every frame.

Our self-supervised system addresses these by constructing models using spatial heuristics and image segmentation to generate ‘hard’ negatives for further training. Each key frame (vertex) in a topometric visual map has a detector model associated with it, where this model is trained on a short sequence of frames surrounding the key frame. At run-time, the detector queries a model database to retrieve the model associated with the currently localised key frame. In the event that localisation fails, the detector defaults seamlessly to a generic model.

This detection system offers a substantial performance improvement over using a general model, though we do not see the same level of improvement as in our ‘ideal’ scenario. This is to be expected for a self-supervised system.

The experience-based detector models are fitted to their operating environment, hence substantial deviation from the background class of the training scene is likely to result in spurious self positives. We see some evidence for this in our results: the dominant mode of type I errors (false positives) in our experiments was due to dynamic objects in the environment, representing image data the detector has not been trained for. In contrast, the generic model type I errors were - as expected - mixed in the image content.

Experience-based object detectors offer a mechanism to boost an object detector’s performance substantially, mitigating limited model capacity. This is ideally suited to mobile robotics applications, where we repeatedly operate (and localise) in the same environment.

4

Learning ‘Place’: Dynamic Detection Filter Networks

Contents

4.1	Introduction	97
4.2	Background	101
4.3	System Overview	104
4.3.1	Object Detector	105
4.3.2	Filter Generating Network	106
4.4	Evaluation Methodology	111
4.5	Results and Discussion	112
4.5.1	Detection Performance	112
4.5.2	Run-time Performance	112
4.5.3	Scene Encoding	114
4.5.4	Learning the static environment	121
4.5.5	Generalisation to Novel Routes	121
4.5.6	Model Sensitivity	126
4.6	Conclusions	126

4.1 Introduction

Our work in Chapter 2 demonstrated that we are able to fit detector models to broad categories of operating conditions: seasonal and route effects. Following on from this, Chapter 3 constructed a series of very local expert detection models:

rather than trying to fit to a large, pre-defined category of operating conditions, the detector models were highly specialised, tightly fitted to data from a known place in a visual map, such as an experience-based navigation system.

A key factor in these approaches is picking the correct operating resolution for the task. If we try to fit to a large set of operating conditions we see some performance gains, but this is still limited by the detector model capacity as we must have some degree of generalisation, demonstrated in Chapter 3. We see similar patterns of behaviour for models fitted to predetermined categories of data (e.g., seasons for a given route as in Chapter 2).

This presents us with a dilemma. How do we manage picking the appropriate level of model resolution? We could avoid this question by limiting models to known locations in a experience-based visual map (as in Chapter 3), as we rely on the localisation system to finely discretise the world - both spatially and over appearance change.

However, this tightly coupled approach duplicates effort, training many more detector models than we might need, and fundamentally requires an active localisation system. It is likely that we could merge data from adjacent map key frames to construct a bespoke model which works along a sequence of the map graph vertices rather than a single node. Conversely, in other areas we are likely to find that we require a highly specialised model for just one node. This problem is difficult, as it is dependent on both the model capacity we have available and the variability in appearance of image data the model is trained on.

This approach also assumes that the visual localisation system is able to manage appearance change in a way that benefits both localisation *and* the associated detector models. While this is broadly true (and they both operate on the same image data), these are different tasks and a single experience representation which suits localisation may not be adequate for object detection on the same data.

One potential solution would be to organise a robot's visual memory into clusters, building models on each group. This approach would be heavily dependent on the efficacy of grouping similar images, either with a global feature descriptor [96] or

bag-of-features image retrieval approaches [98]. Models could then be retrieved by associating the target image with one of these models. This is similar to our approach in Section 3.4 using image appearance, which offered comparable performance gains to spatial grouping of image data.

At its core this is essentially a more generic variant of our localisation approach. A visual localisation system is a highly specialised image retrieval system designed to scale well for real-time robot operation, providing us with a robust spatial prior for image data in the observed scene. A clustering approach would lose this knowledge, and clustering doesn’t really address the core issue: we still have to pick clusters, and how do we know what the right degree of clustering is for our detector model and operating environment?

Rather, we propose adopting a machine learning approach. We want to use detector models which are suited to a specific operating environment. This operating environment is heavily influenced by the location as well as image appearance, which are closely correlated. We know model capacity is a limiting factor in the specialised detector models’ performance (thus well constrained ‘places’ are helpful), but we lack the means to determine the point this becomes a limiting factor as this will be both task and model dependent.

Our solution to this is to have a neural network learn this structure for us, relying on their powerful representational capacity to generate the detector models used. We propose a system we refer to as a Dynamic Detection Filter Network, or DDFN.

The DDFN provides a more flexible mechanism for local expert object detection as it removes the dependency on localisation by generating models conditioned on the input image. This facilitates using a local expert model even when we fail to localise. Potentially, an approach based on this could also be used to scale local expert detection to a diverse range of operating conditions and routes using the self-supervised data mining methods discussed earlier in Chapters 2 and 3.

To evaluate the DDFN as a potential method, we structure experiments around the prior work in Chapter 3.3, wrapping a neural network around a well tested object detection pipeline. Specifically, we seek to replicate the same local expert model

behaviour as the experiments in 3.3.1 (where we have a constrained experiment consisting of two annotated laps of the same operating environment), generating local expert models conditioned on the input image, rather than using localisation as a tool to retrieve local expert models from a database.

The standard object detector pipeline described in Section 2.3.1 is augmented with a filter generating network, which conditions the detector model based on the input image. The structure is inspired by the Dynamic Filter Network proposed by De Brabandere *et al.* [105].

Outline

This chapter first discusses the background and implementation of a Dynamic Detection Filter Network, outlining how this approach can be merged with a classical object detection pipeline and how it fits in the wider literature of deep learning in computer vision.

We then discuss our implementation of the detector framework in Section 2.3.1 in Tensorflow [106] for learning. We train a generic detector model using the same methodology throughout this thesis, using this model as a baseline reference point between experiments.

We present results on experiments similar to those in Chapter 3, demonstrating that we are able to substantially improve the performance of an object detector operating in a known environment. The key difference is that, in this case, we no longer require the tight coupling to a visual localisation system, relying purely on the input image.

Finally we present a short qualitative assessment of the trained filter network, assessing how the network’s feature representation of the scene varies along the driving route. We finish with some concluding remarks, including how this approach could be applied to modern deep learning perception systems where the necessary specialist hardware is available.

4.2 Background

Over the past five years we have seen substantial advances in deep learning, to the point that this is arguably now the dominant approach to many machine learning problems. These neural networks have enjoyed remarkable success in object classification and recognition [27, 28], various games [107], speech recognition [108], and more.

In computer vision, the *AlexNet* object recognition system introduced convolutional neural networks (CNNs) to the object recognition literature, following its success in the ImageNet Large-Scale Visual Recognition Challenge - a popular object recognition challenge - in 2012. This adopted a simple (by current standards) network architecture consisting of five convolutional layers with pooling, and three fully connected layers.

This was quickly followed by rapid development in the field, with Zeiler and Fergus [109] providing further insight into the inner workings of CNNs in their work on *ZFNet*. Simoyan and Zisserman took the concept of deep (i.e., layered) neural networks for object recognition further in *VGGNet* in 2014 [28]. They expand the number of layers to nineteen and reduce the stride and size of the filters compared with *AlexNet*, arguing for a deeper network of simpler filters. This development was further expanded on, rapidly increasing the size of networks used in computer vision. The 22 layer *GoogLeNet* [110] was followed by the 152 layer *ResNet*, which further improved object recognition performance and introduced the idea of a residual block [111].

In addition to object recognition tasks, CNNs have been applied to object detection. The first substantive results with a CNN came from *R-CNN* (Regions with CNN features) by Girshick *et al.* [112]. As with most object detection approaches, this is structured as a region proposal system coupled to an object classifier. The first stage of the network provides a candidate detection region, which the classification stage then evaluates, assigning a probability for the target object class. This was subsequently followed by *Fast-RCNN* [113] and *Faster-RCNN* [30], offering improvements in run-time and detection performance.

This issue of scale has been identified as a major limitation of neural network object detectors, particularly in the case of pedestrian detection. These approaches have enjoyed substantial success with other object classes, such as cars, but have struggled with pedestrians. Zhang *et al.* presented a systematic analysis of modern approaches to pedestrian detection, demonstrating that *VGGNet* and *AlexNet* in an *R-CNN* framework offer competitive performance, but generate a large number of high scoring false positives near the true positives [36]. They hypothesise that this is an intrinsic limit to the architectures, and is partly a spatial limitation. These networks are often heavily based on those from object recognition, including using pre-trained weights from ImageNet. In detection rather than recognition tasks, the object scales tend to vary more significantly.

Subsequently *MSCNN*, a multi-scale convolutional neural network, suggested an architecture which generates region proposals at multiple scales [31], rather than relying on anchors at different scales as in *R-CNN*. This architecture is one of the leading pedestrian detectors on the KITTI benchmark [68] at the time of writing.

One of the biggest disadvantages to these object detector approaches is their run-time performance. *MSCNN* has of the order of 217 million parameters for a pedestrian model, as an example. While often heavily optimised, these neural network detectors are still slow, offering around 2Hz runtime performance on the best available dedicated graphics cards.

A different approach was proposed by Redmon *et al.*, favouring a unified architecture in *YOLO* which has a single neural network to predict bounding boxes and object probabilities in one evaluation [35]. This is much faster (45Hz) but tends to make more detection localisation errors than competing systems, particularly when changing operating domains. Further refinements of this method in *YOLO9000* improved detection performance [114]. This method is appealing to roboticists due to meeting our run-time performance requirements, but to date offers little detection performance improvement over classical pedestrian detection methods.

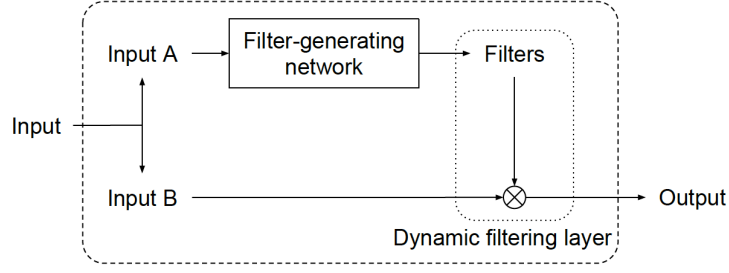


Figure 4.1: A Dynamic Filter Network. The filter (both linear kernel and bias) in a convolutional layer is conditioned on an input. This may be the input applied to the convolution, but does not strictly need to be. Reproduced from [105].

Our goal in this chapter is to arrive at an object detection framework which can learn the structure present in robot’s operating environment, and generate detector models based on the input image being processed.

De Brabandere *et al.* recently proposed Dynamic Filter Networks (DFN), a replacement for the typical convolutional layer in which the filters used in the convolutional layer are dynamically conditioned on the input [105]. This network structure offers increased flexibility as the convolutional filters can adapt, without a substantial increase in the number of parameters. Figure 4.1 outlines the structure of a DFN layer.

There are strong structural similarities between the DFN architecture in Figure 4.1 and an object detector using a linear SVM. An object detector is essentially a number of fixed transformations from the original image to a predetermined feature descriptor, which is then convolved with a linear filter, much like a DFN.

We apply this Dynamic Filter Network approach to our problem, using the representational capacity of a neural network to learn object detection filters for the robot’s operating environment. This approach offers the benefits of building bespoke local expert detectors without the strict coupling to a visual localisation system. We draw on much of the lessons learned here in the literature on object recognition and detection with neural networks when designing and training the network.

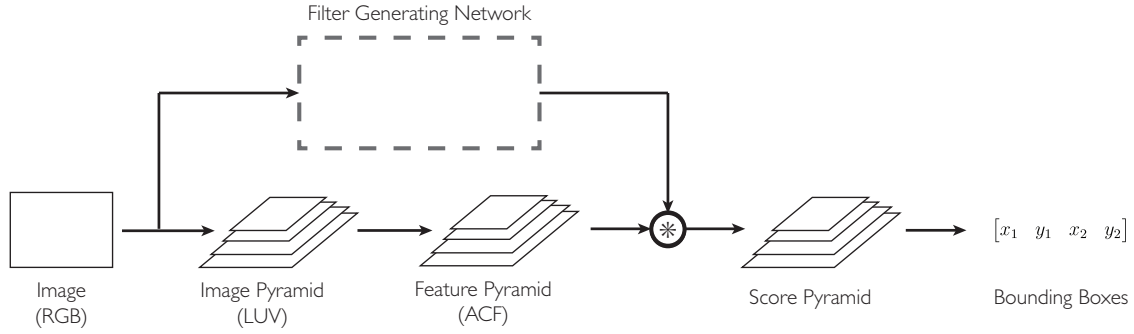


Figure 4.2: The Dynamic Detection Filter Network architecture (DDFN): applying the DFN network to a classical multi-scale object detection pipeline as in Section 2.3.1. The image is scaled, converted to a predetermined feature descriptor, convolved with a linear filter (and bias), and the result scores thresholded and converted into bounding boxes via non-maximum suppression. In a standard implementation this linear filter is fixed, here it is generated by the filter generating network.

4.3 System Overview

We augment the sliding window object detector pipeline in Section 2.3.1 to use a dynamic classifier model. Rather than use a static linear classifier model retrieved from a database, we generate the linear filter using a Dynamic Detection Filter Network, or DDFN. This architecture is illustrated in Figure 4.2.

The standard DFN module is intended for implementation with a single convolutional layer as part of a larger neural network, but the structure translates directly to a sliding window object detector. A sliding window object detector using a linear SVM is fundamentally the convolution (strictly the cross-correlation) of a linear filter (and bias) on a number of scaled feature descriptors. These feature descriptors are simply a fixed transformation of the input image data.

The filter generating network we use here is illustrated in Figure 4.4, and discussed in further detail below in Section 4.3.2. Fundamentally, the network generates a dynamic detector model $\psi(\mathcal{D})$ conditioned on the input image data $\mathcal{D}$.

This section discusses the architecture of the system, loss function, and training methodology used for the DDFN.

4.3.1 Object Detector

We implement the multi-scale sliding window detector architecture in Section 2.3.1 in Tensorflow [106]. The input image is scaled to an *image pyramid*, and subsequently converted into a *feature pyramid*. Each layer in the feature pyramid is the Aggregate Channel Features (ACF) feature descriptor [33] computed on the corresponding resized image in the image pyramid. The feature pyramid is convolved with a linear filter (plus a bias term), producing a *score pyramid*. All score locations above a given operating threshold are returned as detections in the form of bounding boxes. Re-implementing this detection pipeline in Tensorflow is a necessary step as we need to be able to train the filter generating network through back propagation of gradient from losses derived from the score pyramid.

Benchmark Model Performance

We benchmark this detector implementation to use as a reference point to previous work. Rather than use the filter generating network, we instead apply a static filter kernel from a linear SVM: replicating the classical detector architecture.

We retrained the linear *SVM+ACF* model on INRIA data, instead using ACF features computed using the Tensorflow implementation. As before, we performed three passes of exhaustive hard negative mining on the INRIA negative set and used an optimisation parameter of $C = 2^{-6}$ selected by cross validation on the training set. Figure 4.3 shows the detection performance we use as our reference point for this chapter.

We note that this implementation of a sliding window detector reduces the performance on our test dataset when compared to the optimised C++ implementation used in earlier chapters, from an average precision of 0.483 to 0.372. This is due to differences in implementation, primarily a different selection of scales due to memory limitations in Tensorflow (the C++ implementation is optimised, permitting a better multi-scale detector). When using the linear model trained on C++ ACF features in this detection pipeline we see comparable performance to the model trained on Tensorflow ACF features, differing only by 0.003 in average

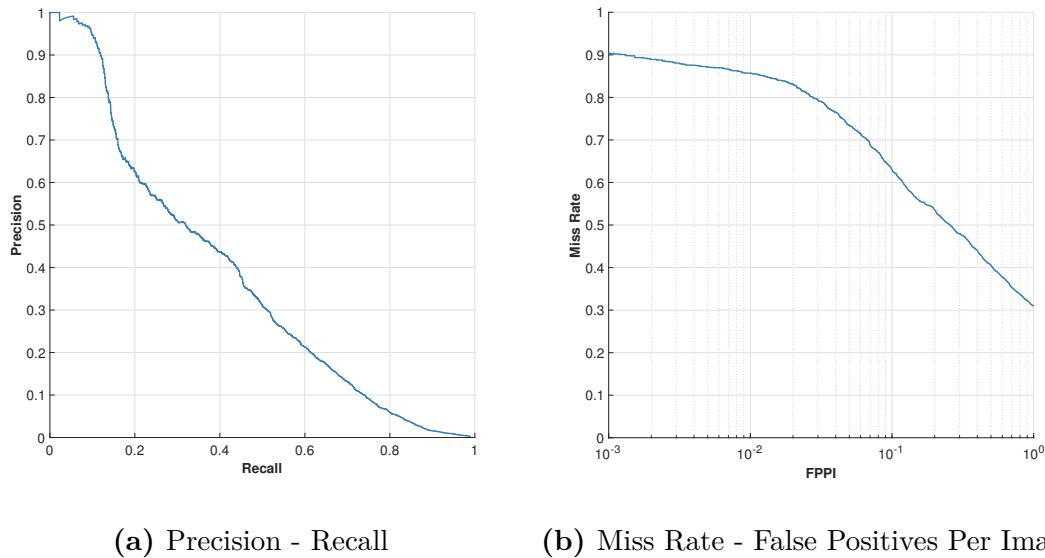


Figure 4.3: Performance curves for the generic baseline *SVM+ACF* pedestrian detector implemented in Tensorflow. The model here has been trained on the INRIA pedestrian dataset, with hard negative mining on INRIA negative images. This is the same process as the generic baseline model used in earlier chapters, using the Tensorflow ACF features.

precision. This demonstrates that the feature implementation is comparable with only minor implementation differences. We therefore attribute the difference in performance to the detection pipeline implementation (e.g., scales, some edge effects due to padding differences), rather than differences in feature set or linear classifier.

4.3.2 Filter Generating Network

We generate the object detector’s linear model from the input image, where the filter generating network learns what aspects of the image data influence detection performance. This section outlines the network architecture we used, the design choices we made, and training methodology required.

Network Architecture

We structure the filter generating network as a feature encoder (formed of a number of convolutional and pooling layers), followed by global average pooling, and finally a small fully connected layer mapping this internal representation to the desired model size. This architecture is shown in Figure 4.4.

We selected a canonical feature encoder in Computer Vision as a de-facto standard for object recognition tasks: the *AlexNet* [27]. Other commonly used options for this type of task include *ResNet*, or *VGG*. We elected to use the *conv1-3* layers (i.e., the first three convolutional layers, with maximum pooling between layers), as wider work in transfer learning for computer vision (e.g., [100]) indicates that these lower layers tend to be less specialised than the higher layers, encoding features which transfer well between computer vision tasks. Transfer learning applied to existing networks tends to benefit primarily from fine-tuning the higher order layers [112]. Examples of this transfer learning approach being used for novel vision problems include [115] and [116].

The global average pooling layer [117] spatially averages each channel in the feature vector for the final convolutional layer. We wish to learn a spatially invariant feature representation of the scene, since the detector model generated by this network is applied globally to the input image. Global average pooling provides this spatial invariance across the image.

We finally use two fully connected layers to map these features to a linear filter (and bias term) of the correct size for our desired object class. The first fully connected layer uses an *exponential linear unit* (*ELU*) transfer function, and the output layer uses a *linear* transfer function in generating the output for this regression task. The *ELU* transfer function was selected for the FC layer due to its improved learning characteristics compared to other activation functions, such as *ReLU*s [118].

The FC layer’s dimensionality was set to 100, as this corresponds to the number of models used for this experiment in the system in Section 3.6. The system configuration resulted in approximately 100 models per lap of the driving route, where each model was trained on a two second image sequence for 3 – 4 minutes of driving. As we wish to benchmark this learned approach as an alternative to using the system approach, we select a comparable number of nodes in the fully connected layer. We expect that this network dimensionality should be able to replicate the system approach, since - at worst - the fully connected layer could

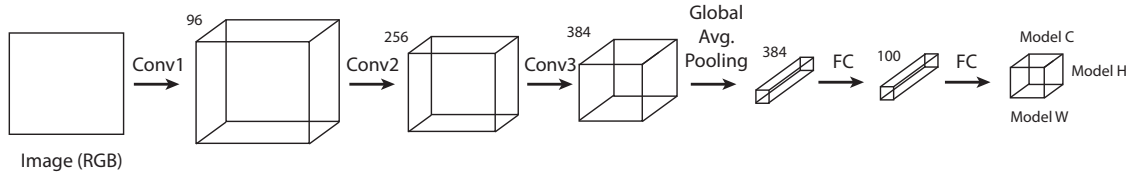


Figure 4.4: Detector Filter Generating Network architecture. This network consists of three components: a feature generating hierarchy based on *Alexnet* [27] (three convolutional layers with maximum pooling), a global average pooling layer on this feature representation, and finally a small fully connected layer mapping this representation to the desired model dimensions.

mimic a multiplexer, selecting models by activating a single node. In practice, this may be able to be reduced in dimensionality, but this is likely to depend on the route data distribution (specifically, its variance).

The final (linear) fully connected layer size corresponds to the dimensionality of the required model. As per earlier chapters, the dimensionality of the pedestrian model is $24 \times 12 \times 10$ plus a bias channel, for a total of 2881 nodes.

The filter generating network is relatively lightweight for a neural network used in object detection, with 1.5M parameters. For reference, *MSCNN* has 217M parameters for a pedestrian detection model.

Loss Function

The dynamic detection filter network needs to solve two competing problems. Firstly, it needs to generate a detection model which is able to generalise across the target positive class (in our case, pedestrians). In addition to this, it must generate a detection model which adequately captures the background class of the input image.

We therefore use a loss function $\mathcal{L}_{DDFN}$ which captures these two requirements in the loss term $\mathcal{L}(\mathcal{T}, \mathcal{D}, \theta)$, in addition to a regularisation term $\mathcal{R}(\mathcal{D}, \theta)$ and adaptive hyperparameter regularisation $\mathcal{R}(\lambda)$.

$$\mathcal{L}_{DDFN}(\mathcal{D}, \theta) = \lambda \mathcal{L}(\mathcal{T}, \mathcal{D}, \theta) + \beta \mathcal{R}(\mathcal{D}, \theta) + \mathcal{R}(\lambda) \quad (4.1)$$

Classification Loss Term

The main loss term $\mathcal{L}$ is computed as the hinge loss of the generated model on an external training dataset $\mathcal{T}$ and the highest scoring negative regions in the input image batch $\mathcal{D}$.

The external training data $\mathcal{T}$ provides a supervisory signal to ensure that the generated models capture the target positive class. This is an unusual approach to training neural networks, but external data is necessary as we condition the generated model on the input image data: the models will be trained to solely detect the instance of the target class in that image rather than the larger target class. For example, the model will be conditioned to detect the single pedestrian visible in that scene (e.g., a pedestrian in red) rather than the set of all potential pedestrians (e.g., pedestrians wearing blue). Similarly, if the input image in training has no instances of the target class, the conditioned models are likely to yield false negatives on instances of the target class in images of the same scene.

The input image data $\mathcal{D}$ provides the target scenes we wish to condition the models on, representing the background class the filter should respond to. The scene loss component is structured similarly to traditional hard negative mining: selecting the highest scoring N regions (we use $N = 100$) of the feature pyramid not associated with a positive label. These high scoring negative regions have associated scores y_D , each with a negative label $t_D = -1$.

Collectively, this loss term is actually very similar to the loss function used in *MSCNN* or *Faster-RCNN*, in that it is a weighted sum of losses from a select number of regions of the image. The only points of difference are that 1) we use hinge loss rather than log loss (though these are very similar), and 2) we use the loss on positive class data from an external training dataset rather than instances from the image being trained on. *MSCNN*’s bootstrapping loss function is fundamentally the same as hard negative mining here, using a loss derived from the highest scoring (i.e., most incorrect) negative regions.

As with the detector models in Chapters 2 and 3, we use the INRIA pedestrian dataset as the external training dataset $\mathcal{T}$. Applying the model to each patch generates a list of classifier scores y_T , corresponding to labels $t_T = \pm 1$.

These scores y_T and y_D are concatenated together, as are the associated labels t_T and t_D . This forms a set of predictions $\mathbf{y}$ with labels $\mathbf{t}$. Each score is assigned a weight in the vector $\mathbf{w}$ to ensure class balance, where each w_i is set to 1.0 for positive labels and the class ratio N_P/N_N for negative labels.

The loss $\mathcal{L}$ is then defined as the hinge loss on these predictions, defined as:

$$\mathcal{L}(\mathcal{T}, \mathcal{D}, \theta) = \max(0, 1 - \mathbf{t} \cdot \mathbf{y}) \cdot \mathbf{w} \quad (4.2)$$

Regularisation Terms

In addition to this primary loss term, we add two further regularisation terms. We include a regularisation term on the generated model to mitigate network overfitting using l^2 loss on the network parameters θ . In addition to this weight regularisation term, we add a second regularisation term to promote model smoothness within a training batch of contiguous images. This smoothness term is computed as the l^2 -norm between each detector model generated by the network $\psi(\mathcal{D}) = \{\psi_1, \dots, \psi_n\}$ and the mean model for each training batch $\bar{\psi}$.

$$\mathcal{R}(\mathcal{D}, \theta) = \|\theta\|_2^2 + \sum_i \|\psi_i(D_i) - \bar{\psi}\|_2^2 \quad (4.3)$$

The loss function is formed by the weighted sum of these three terms, controlled by the set of hyperparameters $\boldsymbol{\lambda} = \{\lambda\}$. We follow Kendall *et al.* [104] and learn the weights $\boldsymbol{\lambda}$ by $R(\boldsymbol{\lambda}) = -\sum_i \log(\lambda_i^{-1})$. Initially, this adaptive weight is set to 1, with the β regularisation hyperparameter set to a constant value of 10^{-4} .

Optimisation

We use the Adam optimiser [119], a variant of Stochastic Gradient Descent, to solve for the parameters in the filter generating network. A brief exploration of parameter range indicated that a learning rate in the range $[10^{-3} \ 10^{-5}]$ was suitable, with

higher learning rates being unstable and lower rates being impracticably slow in training. We therefore used a learning rate of 10^{-4} for all training.

Training Methodology

The DDFN has parameters trained in three stages. Firstly, we initialise the lower feature computation layers (*conv1-3*) to parameters from *AlexNet* pretrained on ImageNet [27] for object classification. This provides a useful starting point for the network parameters.

Secondly, we pretrain the full DDFN network by regressing from an input image to the corresponding model trained on a sequence of 10 frames from the experiments in Chapter 3.3. We use a mean-squared-error (MSE) loss function to train the network to regress from the input image to the output model. These models have been trained on a different ACF implementation, but this provides a useful initialisation set of parameters for the network.

Finally, we fine-tune the network by training it on randomised contiguous batches of images from a training dataset, referred to as $\mathcal{D}$. The size of these batches (and hence the models used for pretraining) is set to 10 based on the results in Chapter 3, where linear detector models trained on 10 – 100 frame sequences demonstrated the best performance for the model capacity of a linear SVM on this feature type and object class. Each training batch consists of sequential frames as we expect models on adjacent frames to share some similarity, enforced by the model smoothness regularisation term.

4.4 Evaluation Methodology

We assess the ability of the DDFN to generate detector models conditioned on the input image by performing the same experiment as in Chapter 3. In this experiment we seek to optimise object detection for our known operating route, using a lightweight linear object detector.

We require a dataset which is both suitable for localisation and has object annotations, hence we use our own dataset for this evaluation. We select the same

2km urban driving dataset outlined earlier in Section 3.3.1, consisting of two laps of the same route fully labelled with pedestrian annotations.

We train a DDFN on the mapping lap, then evaluate it on the live images from the alternate lap. To increase the quantity of data available for testing, we train a separate DDFN for each lap, then test these on the alternate lap not seen during training. These networks had 20 epochs of training, where one epoch consists of a training with a contiguous batch centred on each image in the training set (hence each image is processed 200 times with a 10 image batch size).

4.5 Results and Discussion

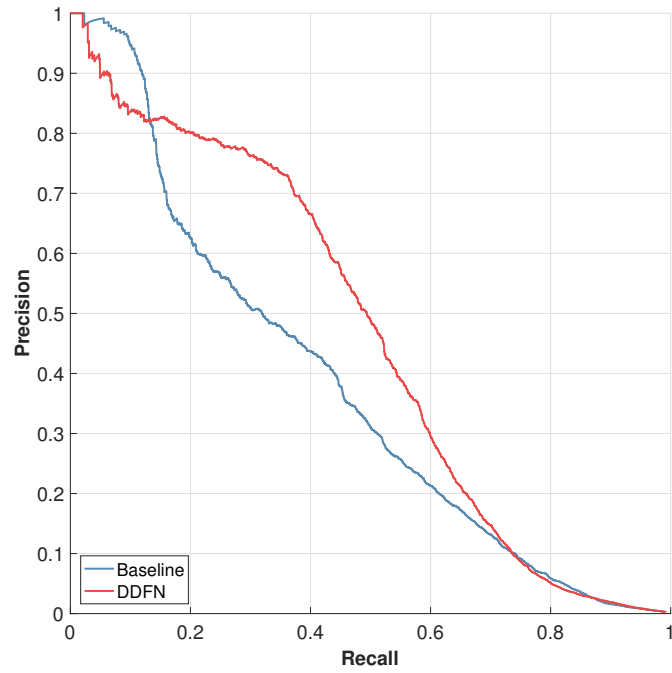
4.5.1 Detection Performance

We compare the performance of the DDFN to our baseline *SVM+ACF* model on the labelled 2km urban driving dataset, shown in Figure 4.5. The DDFN demonstrates a performance gain over the generic baseline on both the Precision - Recall and Miss Rate - FFPI curves. We see improvement on each aggregate metric in Table 4.1, with around an 8 percentage point improvement on all metrics.

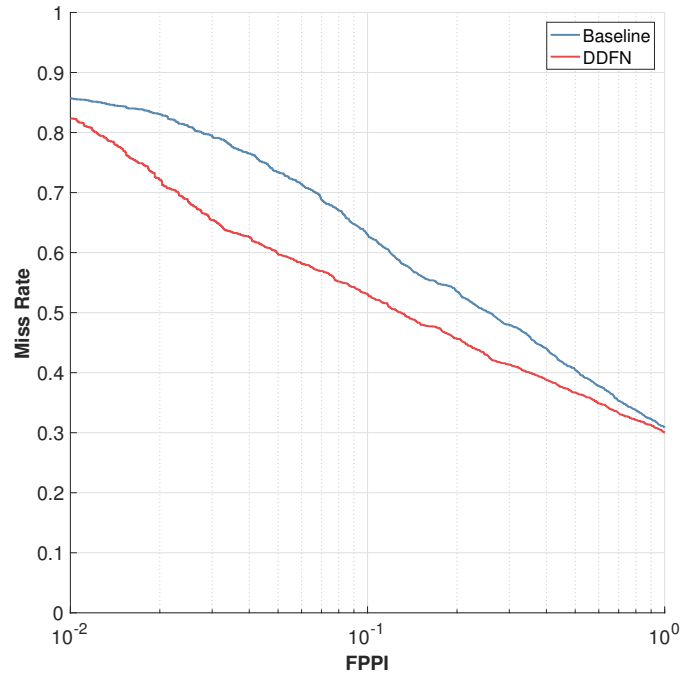
This is a good improvement in detection performance, though not as great a gain as we saw with the system in Chapter 3. This is plausible: an accurate experience-based localisation system provides us with a strong prior on the visual data we present to the detector. Here the network learns an internal representation based on appearance of the scene. It is possible that a higher capacity network with more parameters could potentially achieve better results, though it is unlikely that we would be able to exceed the performance figures of the local expert detection system in Chapter 3, as this is constrained by the limit of what we can achieve with a finite capacity linear model.

4.5.2 Run-time Performance

These performance results are positive, demonstrating that we can construct a local expert detector model without being strictly bound to localisation. However this



(a) Precision - Recall



(b) Miss Rate - False Positives Per Image

Figure 4.5: DDFN pedestrian detection performance on the 2km *North Oxford January* urban driving test dataset. The DDFN detector models demonstrate a performance boost in this known operating environment. The baseline model is a static *SVM+ACF* model trained on the same dataset (INRIA), with hard negative mining performed INRIA negative images rather than the robot’s operating environment.

Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
Generic SVM+ACF	0.372	0.584	0.423
DDFN SVM+ACF	0.456	0.516	0.506

Table 4.1: Aggregate performance metrics for the DDFN and generic baseline detectors along the known route, *North Oxford January*. By all metrics, the DDFN boosts performance over the generic model. The generic model is a static *SVM+ACF* model trained on the same dataset (INRIA), with hard negative mining performed INRIA negative images rather than the robot’s operating environment. A higher number is better for Average Precision and F1 Scores, lower is better for Log-Average Miss Rate.

approach does require additional compute time, compared to the trivial time taken to retrieve a small model through a database query in the system in Section 3.6.

The DDFN network here can generate models at approximately 20 Hz on general purpose CPU hardware, which is suitable for deployment in most robotic applications (our cameras here operate at 20 Hz). Table 4.2 outlines summary statistics for the run-time performance, showing a mean compute time of 0.051 seconds with a standard deviation of 8 milliseconds. The network itself will be able to evaluated in a deterministic constant time (as it is a fixed number of floating point operations) - the variability seen here is due to the operating system.

Metric	Model Generation Time (s)
Mean	0.051
Median	0.051
99 th percentile	0.056
Standard deviation	0.008

Table 4.2: Run-time performance metrics for the filter generating DDFN on the test dataset (9154 frames at 512×384 pixels), running on general purpose computing hardware (i.e., a CPU). This network is able to generate detector models at 20 Hz, which matches our camera frame rate.

4.5.3 Scene Encoding

The DDFN learns an internal representation of the scene, based on the input image. This scene representation is the output of the global average pooling layer: a spatially invariant feature descriptor representing the image.

This scene encoding is of interest to roboticists, as it encapsulates key information about the image, mapped to a detector model by the upper fully connected layers. We present a short assessment of the scene encoding learned by the DDFN.

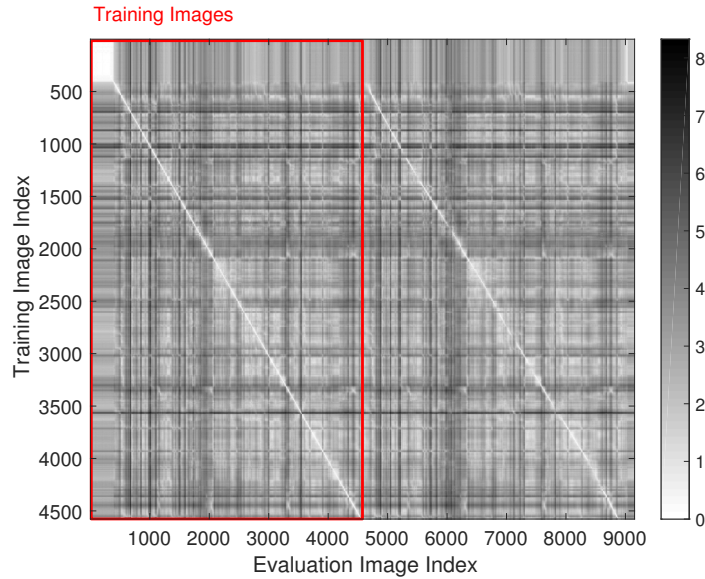
Firstly, we compare the scene encoding of images on our *North Oxford* driving route. We take a DDFN network trained on each lap of this driving route, and compute the scene descriptor for each image. We compare the descriptors between training and test images by taking the l^2 -norm of the difference between these. These differences are plotted in Figure 4.6, visualising this as a similarity matrix for each network. The section of each similarity matrix occupied by images used in training is outlined in red.

We see strong spatial structure in this, much like the sort of structure commonly seen in visual similarity matrices in localisation literature [120]. A strong diagonal is visible in both similarity matrices, indicating that the DDFN has learned an internal scene representation which is able to recognise images taken from a similar viewpoint on the same driving route: an implicit representation of place.

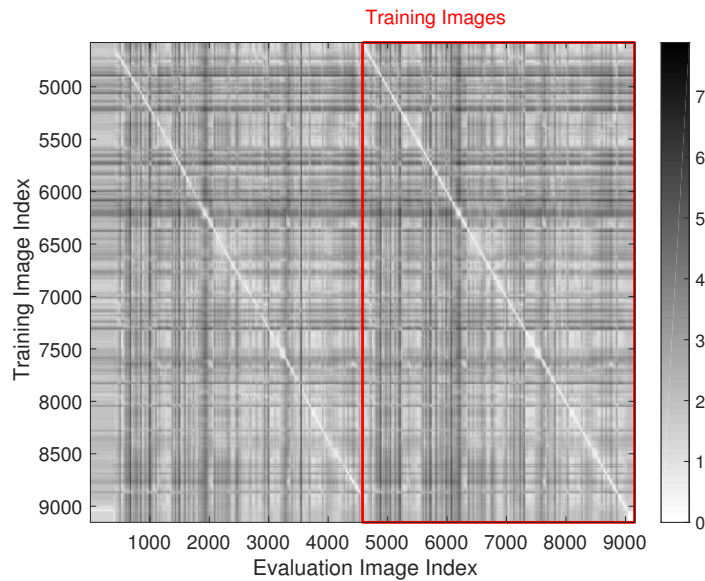
We investigate this implicit localisation representation further by performing a dimensionality reduction on the 384 dimensional scene encoding from scene descriptors taken from a DDFN network trained on the first driving lap, i.e., the data used in Figure 4.6a. This enables us to visualise and better understand these descriptors. For this we used t-SNE dimensionality reduction [121], a non-linear dimensionality reduction method commonly used for embedding high dimensional data in two (or three) dimensions for visualisation.

We first reduce scene descriptor dimensionality to 50 using Principal Component Analysis (PCA) [122] (the t-SNE authors recommend this preprocessing step for higher dimensional data), then apply the Barnes-Hut t-SNE approximation to this reduced descriptor with a perplexity parameter of 30, reducing the dimensionality further to just two dimensions.

To aid in interpreting this reduced data, each point is assigned a colour based on the spatial location along the route. This spatial colour mapping along the route is shown in Figure 4.7.



(a) DDFN Lap 1



(b) DDFN Lap 2

Figure 4.6: The scene encoding is generated by the global average pooling layer, which outputs a single vector describing the input image. We compare the scene encoding between training images and the scene encoding generated by images from a second lap of the same route. The similarity between these features is computed as the l^2 -norm between the two descriptors. DDFNs trained on both laps demonstrate a similar pattern: a strong diagonal on the evaluation images closely matching the structure of the training lap.



Figure 4.7: We allocate a colour map to the driving route, assigning a colour to each location along the route and using this colour in the t-SNE plots in Figure 4.8 to aid visualisation of the various groupings.

With the points coloured according to their position along the route, the t-SNE embedding has very clear spatial structure in Figure 4.8. The two graphs have the reduced scene encoding of both driving laps plotted, with the alternate lap plotted with partly transparent points to highlight the differences between laps.

Using this visual comparison between laps, we can see the cases where the evaluation lap scenes closely match the training scenes. This is the case for the majority of the plots. We also see the cases where these diverge, for example, the green coloured ‘Y’ shape in the lower right corner of the plots. We assess three key cases to gain some intuition for the scene encoding.

Figure 4.9 presents this qualitative assessment of three key cases visible in the t-SNE embedding. The diagram compares an image extracted from the training data (opaque points on the plot) with one from the evaluation data (transparent

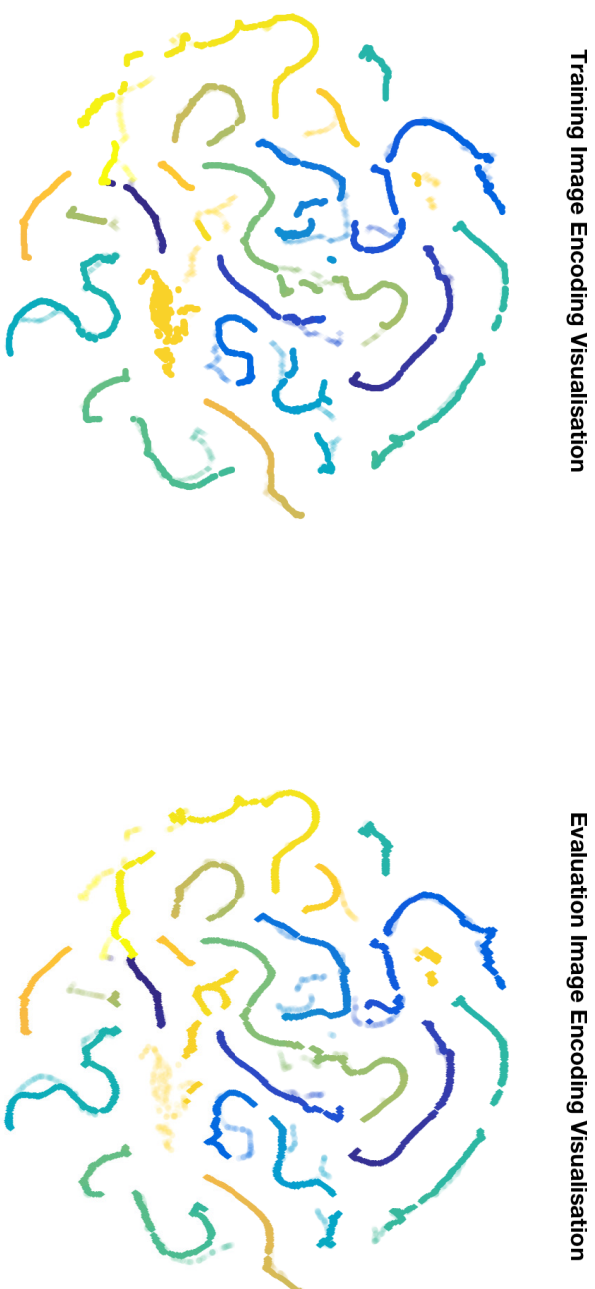


Figure 4.8: We perform a t-SNE [121] dimensionality reduction on the scene encoding of a DDFN trained on the first lap of our route, defined as the output of the global average pooling layer. We colour each data point based on spatial location, to help visually understand the various groupings in the plot. The left plot shows the scene encoding datapoints for the training images, and the right highlights the scene encoding of the second evaluation lap. The transparent points on each plot are the scene encoding for images from the alternate lap (i.e., evaluation scenes plotted on the training scenes graph on the left).

points on the plot).

Firstly, we see cases where a contiguous section of the driving route is split, indicated in the plot by separate groups of points with similar colour gradients. We selected the example of the mid-blue section of the route, which skips from an ‘S’ shaped cluster in the middle of the plot to a similar shaped cluster in the upper left. Selecting an image from the training lap and evaluation lap (images A and B in the Figure) help interpret why a contiguous section of the driving route is split in this way: this corresponds with turning the corner (Figure 4.7).

Secondly, we see many cases where the evaluation scene encoding strongly matches the encoding of training images. This is the case for the majority of data. Images C and D from the teal group in the lower right demonstrate this. Both images result in very similar descriptors, and from a visual inspection this is plausible: they capture the same scene from almost the exact same viewpoint.

Finally, we note that there are a number of cases where a contiguous spatial group differs slightly between the training scene descriptor and evaluation scene descriptor. One example of this is the green ‘Y’ shaped cluster in the lower right, where the evaluation descriptor diverges from the training scenes. Figures E and F provide an insight as to why this might be. Both scenes contain instances of positive classes. Visually, the only notable change between these two images is the change of dynamic objects in the scene: a bicycle parked against a pole has changed orientation, and there are different pedestrians present on the footpath. This change in the positive class present in the scene is sufficient to cause a small change in the network’s encoding of that scene.

This provides some interesting insights into the capabilities of neural networks to learn a global descriptor for the observed scene. The scene descriptor varies (to some extent) with dynamic scene content, which is apparent when two scene descriptors differ by the presence of dynamic objects. However, in this case the DDFN’s scene encoding clearly has a very strong spatial component, apparent in both the similarity matrices and t-SNE embedding presented here. Another way of interpreting this is that the DDFN has learned an implicit representation of

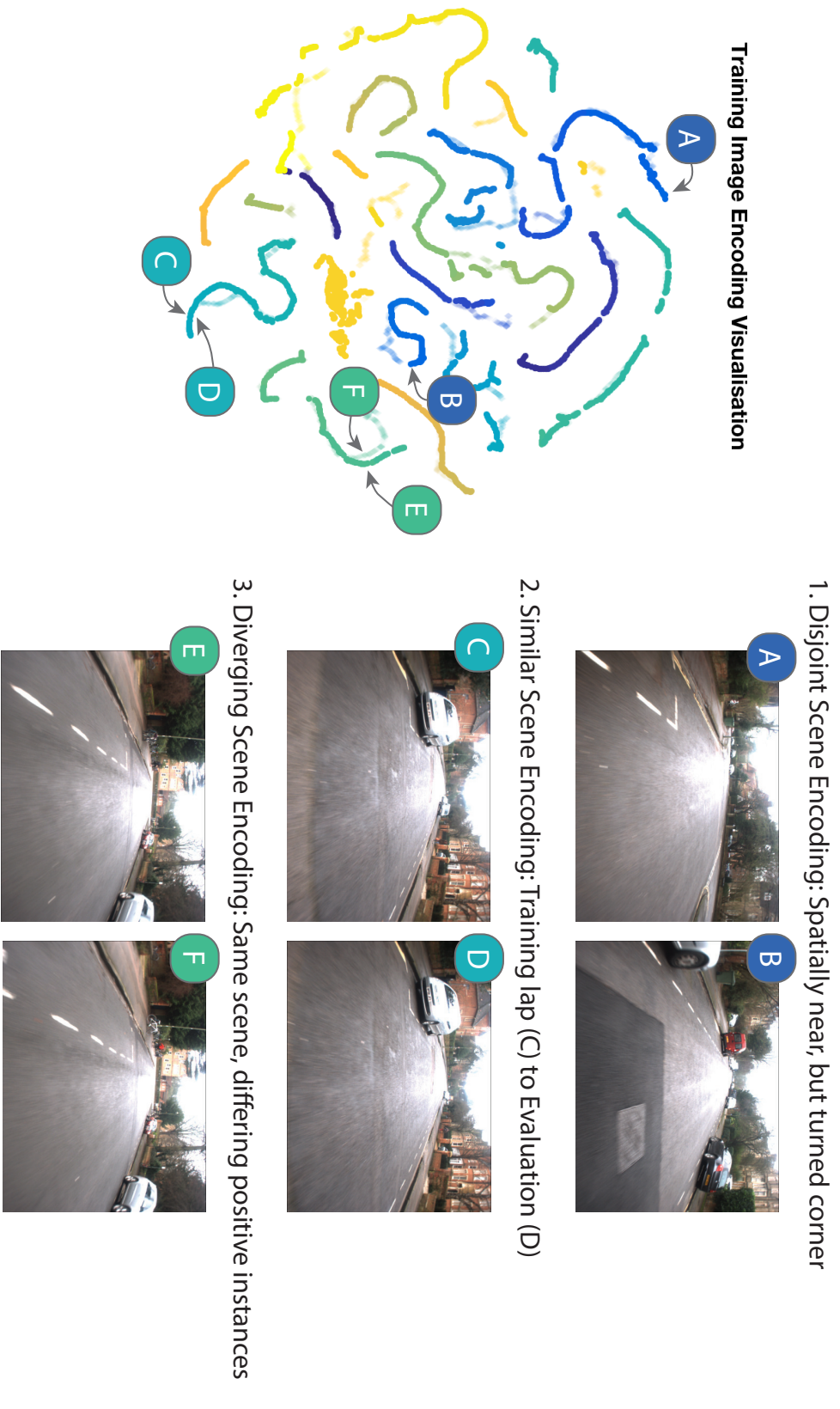


Figure 4.9: We perform a t-SNE [121] dimensionality reduction on the scene encoding of a DDFN trained on the first lap of our route, defined as the output of the global average pooling layer, colouring each data point based on spatial location. We qualitatively assess the structure of the reduced 2D structure by considering three key cases: 1) where the sequence of a spatial group (colour gradient) is separated by t-SNE, 2) where a spatial group is consistent, and 3) where a spatial group internally diverges. In all cases we compare an image from the training lap (A, C, E) to an image from the evaluation lap (B, D, F) of our driving route.

place, mimicking localisation. In other words, the DDFN has learned an encoding of the local environment, or background class.

4.5.4 Learning the static environment

The analysis in Section 4.5.3 suggests that the DDFN learns an implicit representation of the static environment in each scene. To further assess this, we evaluate the error rate of the DDFN on the static environment, as compared to the baseline generic detector.

This effect of learning the scene’s background class is shown in Figure 4.10 and Table 4.3, showing the reduction in errors on the static background class present in the dataset. This histogram is generated in the same manner as Section 3.6.4, using a state-of-the-art image segmentation algorithm to generate what is assumed to be ground-truth object classes.

The DDFN exhibits very similar behaviour to the systems approach to place dependent detection (shown earlier in Figure 3.20), with a significant drop in errors on the static background class (i.e., the local environment or ‘place’) coupled with an increase in errors on other dynamic objects in the scene.

This further supports the assertion that the DDFN learns a representation for place dependent detector models which are tightly coupled to the positive class and the local background class.

Model	False Positives from the static environment (%)
Generic SVM+ACF	76.3
DDFN	35.0

Table 4.3: The percentage of false positive detections classed as static, where the fraction of the detection patch that belongs to dynamic classes in image segmentation is less than 5%. These data points correspond to the first (left-most) bar in the histogram in Figure 4.10.

4.5.5 Generalisation to Novel Routes

We would ultimately like a general purpose DDFN which has captured a wide variety of scenes across many different operating environments with large appearance change.

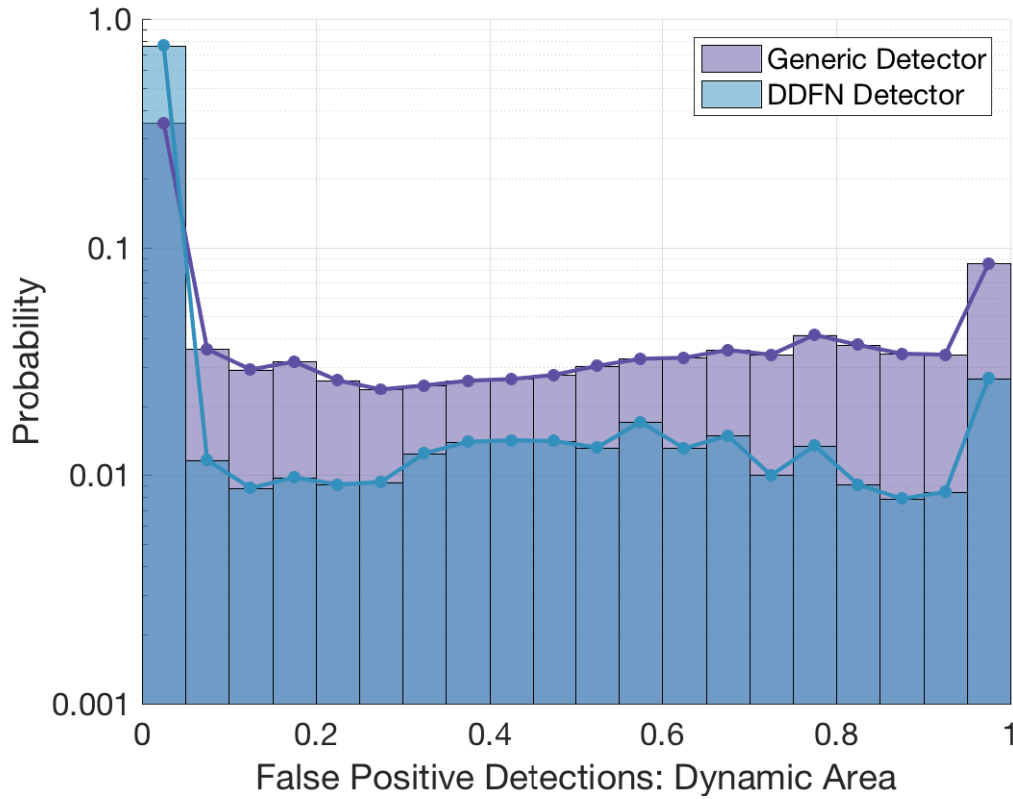


Figure 4.10: Histogram showing the ratio of detection area belonging to a dynamic object class for false positive detections on evaluation dataset. The place dependent models (blue, green) demonstrate a lower rate of false positives on the static environment (the left-most bar), and a higher rate of false positives on the dynamic environment (the remainder of the graph). The rate of static false positives is outlined in Table 4.3. These local expert models are sensitive to changes in the background class which deviate from the training scene, hence yielding a higher incidence of false positives on dynamic objects compared to the generic model.

However, this is likely to require a large quantity of training data to achieve if we want to see the same performance gains from known operating routes seen earlier in Figure 4.5. Here we have trained our DDFN network for a single operating route, proving the benefits of models conditioned for a previously observed route. Based on the analyses in Sections ??, we would expect the DDFN to generalise poorly to a new operating environment, drive as will generate models conditioned on this novel environment.

We assess the ability of this DDFN to generalise by evaluating this on data drawn from a different operating route, *Central Oxford*. This dataset was captured by the same autonomous vehicle platform and Bumblebee camera, and is fully

annotated with pedestrian labels. Table 4.4 outlines key statistics for the datasets, and Figure 4.11 shows the difference in driving routes between *North Oxford* and *Central Oxford*. Central Oxford is a longer, more diverse route which forms part of the publicly available Oxford driving dataset [95], with pedestrian labels for a single pass of this route.

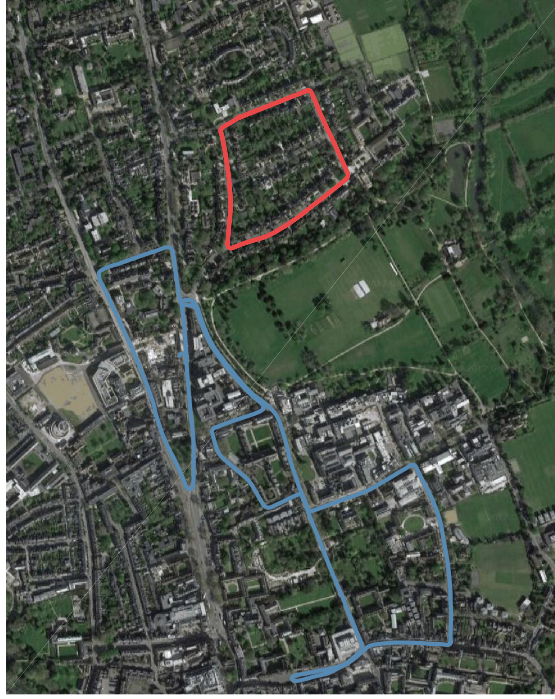


Figure 4.11: We assess the performance of a DDFN trained on *North Oxford* (in red) on a labelled dataset from *Central Oxford* (in blue), evaluating the ability of the DDFN to generalise to a novel route not seen in training. Table 4.4 contains key statistics about both datasets.

Dataset Metric	North Oxford January	Central Oxford May
Distance (km)	1.99	4.19
Frames	9154	18868
Time (min)	7.63	15.72
Pedestrian Labels	4467	41134

Table 4.4: We assess the ability of a DDFN trained on one lap of the *North Oxford* route to generalise to a novel route, *Central Oxford*. This is a longer route through Oxford with busy pedestrian scenes through the centre of the city.

The models generated by the DDFN trained on our *North Oxford* route generalise poorly to this new route, confirming our expectations, as the scene content differs

substantially to the scenes the network was trained on. Figure 4.12 shows the difference in detection performance between the general baseline model and the DDFN, with aggregate metrics summarised in Table 4.5.

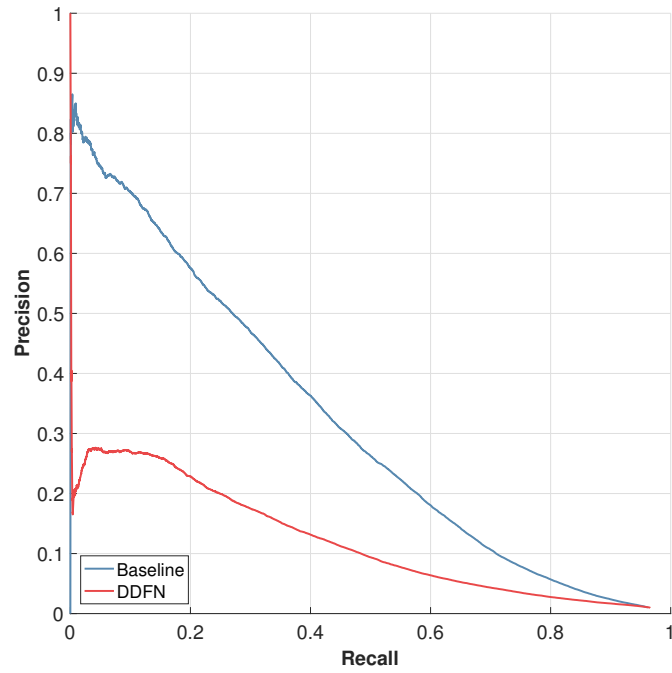
Model	Average Precision	Log-Average Miss Rate	Maximum F1 Score
Generic SVM+ACF	0.313	0.792	0.381
DDFN SVM+ACF	0.118	0.920	0.223

Table 4.5: Aggregate performance metrics for the DDFN and generic baseline detectors on a novel test route in *Central Oxford* which differs from the *North Oxford* training route. By all metrics, the DDFN generalises poorly, as the network is conditioning the models on scenes unrelated to its training data. A higher number is better for Average Precision and F1 Scores, lower is better for Log-Average Miss Rate.

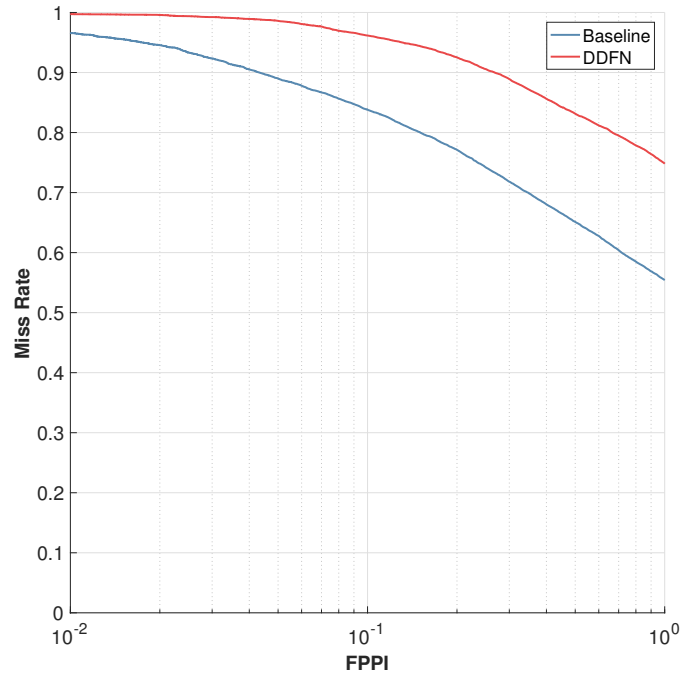
These results on the unseen and seen driving routes highlight a key point. Our DDFN can generate effective detector models for prior routes (i.e., same operating environment or ‘places’) without the need for localisation input, however the network trained here cannot transfer to novel places. This is likely to be a function of training data. We see two potential avenues for future improvement, potentially enabling a DDFN to generalise well to novel scenes.

Firstly, we could train the DDFN on a broad variety of driving datasets to learn a generalised scene representation. Examples of these include KITTI [68], or the CityPersons annotations [123] on the very diverse Cityscapes [103] dataset, which consist of many discrete images (rather than repeated routes) from potential driving scenes. The downside with this approach, however, is that it does not account for the fact that training on data derived from our robot’s operating environment is extremely powerful. These general datasets lack this structure.

An alternative approach is to extend the self-supervised data mining used to build local expert models in Chapter 3, using spatial heuristics and an external supervisory signal from a computationally complex segmentation algorithm to identify the background class for images along the robot’s operating route. We believe that this is likely to be the best approach, benefiting from training the network on data most closely fitting the data observed at run time.



(a) Precision - Recall



(b) Miss Rate - False Positives Per Image

Figure 4.12: DDFN pedestrian detection performance on a novel 5km *Central Oxford May* urban driving test dataset. This is captured using the same robotic platform and camera, but shares no overlap with the *North Oxford* training route. The models generated by the DDFN do not generalise well as it has not been trained on scenes from this route, underperforming compared to a general baseline model.

4.5.6 Model Sensitivity

The results in Section 4.5.5 indicate that the DDFN generalises poorly to novel driving routes which differ from the training route. A further question is how the DDFN responds to changes in appearance when the test route is the same as the operating route the DDFN was trained for.

As a proxy for change in image appearance, we assess the DDFN’s performance against the generic baseline detector when processing images with additive Gaussian noise, increasing the standard deviation of the noise σ at each step.

Figure 4.13 shows the aggregate detection performance metrics (average precision and log-average miss rate) on the test dataset with increasing noise. As the magnitude of the noise increases, the performance gains from the DDFN drop, degrading to a point below the generic baseline.

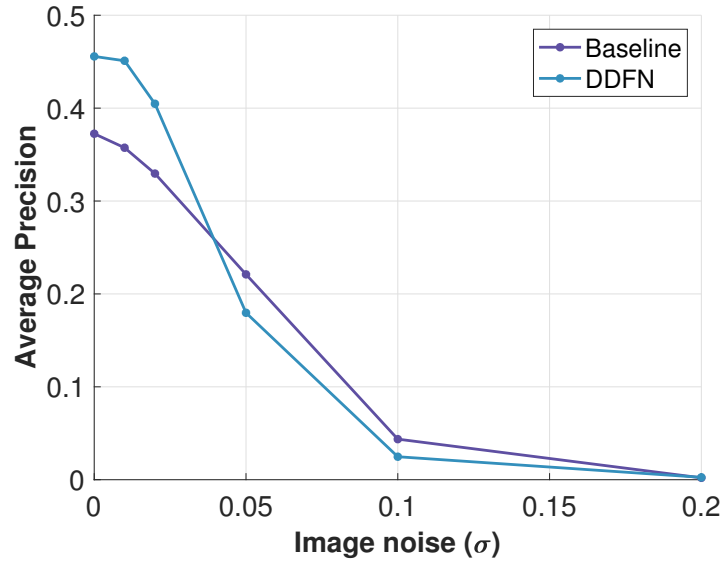
Based on these results and the route generalisation results in Section 4.5.5, we expect that changes in image appearance (e.g., due to lighting or seasonal changes) could have a significant effect on the DDFN performance. This sensitivity is a consequence of fitting the detector models to the operating environment.

A potential strategy for mitigating this effect could be include a diverse range of training data gathered from the operating environment. As data annotation is difficult and time consuming, we see merit in pursuing the self-supervised data mining methods used earlier in Chapters 2 and 3.

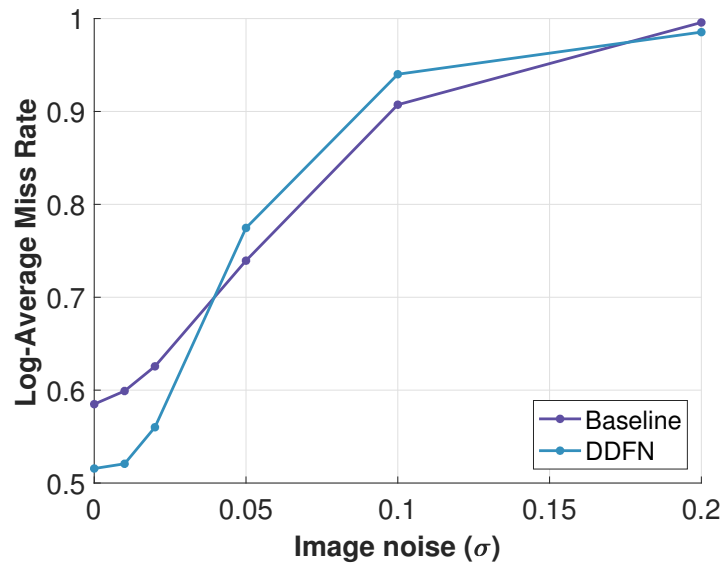
4.6 Conclusions

Conditioning detector models on the input image shows significant promise: we are able to generate models fitted to places our robot has operated before, without the need for explicit localisation. This is a powerful tool for local expert object detection, and means that - when we are operating on a prior route - we are able to boost object detection performance using local expert models without explicitly localising.

The DDFN framework provides an elegant way to define these models, as the network encodes the place dependent nature of these models. This has the



(a) Average Precision



(b) Log-Average Miss Rate

Figure 4.13: DDFN pedestrian detection performance on the *North Oxford January* urban driving test dataset, the same driving route as the test dataset. Gaussian noise is added to the image to act as a proxy for changes in appearance, demonstrating the sensitivity of the DDFN to changes in image appearance. The DDFN offers a performance boost when the image appearance is close to the training data (i.e., zero noise added), but drops below the generic baseline detector in the presence of higher noise.

potential to provide greater extensibility as we train models on further operating environments, allowing the network to capture significant aspects of the scene rather than pre-defined model extents. One interesting observation was that the DDFN network learns an implicit representation of place.

We see great potential for this approach to be applied to current deep learning approaches in perception: both object detection and image segmentation. Conditioning the model parameters on the input image allows these perceptual systems to maximise the performance of these models for the given operating environment. It is likely that higher capacity detectors (such as *MSCNN* or *Faster-RCNN*) would equally benefit from this approach, particularly as the results in Chapter 3 indicated that model capacity was a key factor in determining the extent of the ‘place’ for a local expert detector.

A particular challenge which will need to be addressed is the self-supervision of such systems. Neural networks have immense representational capacity, but have a tendency to overfit to data, requiring significant quantities of data in training. This is both good and bad: we want the network to fit places we have observed before (as Chapters 2 and 3 demonstrate), but this does mean it is likely to perform poorly on novel places.

We believe the conservative approach to self-supervised data mining outlined in Section 3.6.3 (also used in the self-supervised system in 3.6.1) offers a potential avenue to address this, and provide the volume of data required to effectively train such networks.

This self-supervised training provides one possible avenue for overcoming one of the disadvantages of these place dependent methods: their detection performance in novel places. Using self-supervised training, the DDFN could learn novel places as the robot encounters new environments and different driving routes.

5

Conclusions

Contents

5.1	Problem Statement	129
5.2	Experience-Based Classification	130
5.3	Local Expert Detector Models	131
5.4	EBC Deployment: A Self-Supervised Local Expert Detector System	132
5.5	Learning ‘Place’: Dynamic Detection Filter Networks	133
5.6	Summary Remarks	134

5.1 Problem Statement

This thesis seeks to address the fundamental question faced when deploying vision-based object detection on robots: how can we maximise our perception system’s performance in the robot’s operating environment?

We approach this problem using a characteristic unique to robotic systems. Robots, even mobile ones, typically operate in the same place repeatedly, rather than exploring new environments. At one extreme, static robots (such as a robot manipulator in a factory) have an incredibly constrained environment: images are taken from the same perspective of the same environment, likely with constant lighting. In mobile robotic applications the operating environment is much less

constrained. Nevertheless, there are still substantial constraints on the operating environment for mobile robots, particularly in the context of autonomous driving.

Taking autonomous driving as our ideal use case for mobile robotics, the majority of operating time is spent performing regular trips, such as regular commuting, trips to the supermarket or local schools [23–26]. In addition, and perhaps more importantly, autonomous cars are constrained to fixed routes with little variation in perspective. These qualities means that, even if the route is novel to a certain robot car, the route will have been driven by many other autonomous cars previously. Finally, we also have the ability to localise along these routes, giving a mobile robot vital additional information about what to expect at that point of the route.

The constraints on operating environment and the additional information from localisation are a powerful tool which we exploit to boost the perceptual performance of object detectors for mobile robots. This thesis investigates potential methods for this.

5.2 Experience-Based Classification: Fitting Detectors to an Operating Environment

We began this work in Chapter 2 by evaluating a self-supervised detector training process we refer to as *Experience-Based Classification*, or EBC. EBC provides a framework to adapt detector models to a given operating environment by exploiting scene context to validate detections, retraining the detector model on an augmented training dataset with the additional rejected samples.

EBC improves a baseline object detector beyond the performance level achieved through hard negative mining on a generic training dataset. This method facilitates life-long learning in robot perception, learning from the environment the robot experiences during operation.

Our experiments demonstrate that adapting a detector to the operating environment boosts performance at the cost of generality, fitting the detector model to a known route or season. However, this raises a number of research questions, centred around what defines a robot’s perceptual experience.

The results indicate that the granularity of operating environment we fit is likely to be a major influence on performance. This is highlighted by the fact that training a global model across all operating environments did not boost performance, as the model was forced to generalise across a highly variable background data. We investigated this question of environment resolution further in the subsequent chapter.

5.3 Local Expert Detector Models

We have the ability to fit a model to a given operating environment, though we saw that fitting a model to a number of operating environments (a ‘global’ model) did not offer performance gains over a general model. This indicates that the resolution of the operating environment we fit to is important.

Chapter 3 investigated this question. We performed experiments which considered an ideal case scenario, where we constrained the environment to a single, known driving route for which we have ground truth object labels. We construct a sequence of local expert object detector models for each frame in the mapping route. Each detector model is built through hard negative mining a short sequence of images centred on each mapping frame, fitting the model to a short segment of the route. At run-time we use localisation to retrieve the model which most closely matches the current live frame, applying that detector model to the image. As we are interested in the effect of model resolution, we varied the number of frames used to construct each local expert model, from a single frame to the entire map route.

The results indicated that the granularity of the environment captured by each model is a significant factor on detection performance, limited by finite model capacity. Model capacity refers to the ability of a given classifier to fit the data. Single class object detection is fundamentally a binary classification problem, hence the model capacity is dictated by whether the classifier can correctly separate the positive and negative object classes. Model capacity will vary with the classifier type, feature type, and the data itself (which will vary with operating environment, i.e., the background class, and the target object class).

The local expert detector is capable of exceeding the detection performance of the state-of-the-art detector for our target object class (pedestrians), for a range of model resolutions. The ideal range for our problem is using models built using sequences of 10-100 frames. Models with too fine a resolution - for example, a single frame - don't offer the highest performance possible as the background class is extremely small. This makes it more sensitive to slight variations in perspective and image appearance than models trained with a larger negative class. Conversely, if we train models for larger operating environments (defined as a longer sequence of frames), the impact of finite model capacity becomes apparent. The detector's classifier cannot separate the two classes, hence it must generalise, degrading performance.

This results indicate that this approach offers the means to maximise detector performance for a given operating environment. The key issues arising from this are how to extend this idealised scenario (with perfect localisation, models for every mapping frame, and ground truth labels). We focus on addressing these deployment issues in a self-supervised manner.

5.4 EBC Deployment: A Self-Supervised Local Expert Detector System

A key issue facing deployment of a local expert detector system is the absence of ground truth object labels. Localisation error and models attached to key frames are likely to only have a small effect on performance as these sources of error are comparable to our ideal use case experiment. We need a mechanism for data mining images from the robot's environment in self-supervised manner, as labelling these manually is infeasible at scale.

We have the beginnings of a self-supervised model construction method in the work from Chapter 2, fitting an existing model a given environment by using scene context to filter candidate detections based on scale estimates. We adopt a very similar approach, but a scale estimate alone is insufficient here, as we do not have the large generic negative training set used earlier. We augment this approach using a further filtering stage using a state-of-the-art image segmentation

algorithm, FRRN. This offers a reasonably good self-supervisory signal, particularly when combined with the additional scale estimate. The computational time of FRRN is prohibitive for real-time operation, though we do not need real-time performance for offline model creation.

Using this two-stage filter for data mining, we compare the self-supervised local expert detector system to the ideal use case where models are created using ground truth labels. The system offers a performance improvement over our baseline general detector, constructing local expert models without the need for labelled data. The self-supervised system does not achieve the same performance gains as the ideal use case, though this is to be expected as there will be some error introduced from performing data mining without ground truth annotations.

From an analysis of the detector output, we see that the local expert detector models are brittle: sensitive to dynamic objects in their environment. This is a logical outcome, as the models have been trained on a snapshot of one pass through that environment. A quantitative analysis of the detections confirms this, with the local expert detectors (both ideal and self-supervised) generating substantially fewer false positives on the static environment than the comparable generic model.

5.5 Learning ‘Place’: Dynamic Detection Filter Networks

Our deployable local expert detection system works well, but has a number of shortcomings. The system depends on a visual map of the robot’s operating environment, relying on the localisation system to discretise the world - both spatially and across appearance change. For our test scenario with a single experience, this works well, however we do not have labelled data to exhaustively test this across appearance change. Our system assumes that a experience in navigation also suits an equivalent experience for perception: this is likely in many instances, but may not always be the case. We would prefer to use models which vary with image appearance, rather than using localisation as a proxy.

We propose a Dynamic Detection Filter Network as a potential solution to this problem. Here we train a neural network to condition the parameters of a linear classifier model on the input image they are applied to, learning an implicit representation of place. This approach demonstrates a similar improvement on the robot’s operating environment.

An assessment of the DDFN’s scene encoding indicates that the network learns an implicit representation of place, demonstrated by comparing the similarity between scene descriptors of a training lap and evaluation lap of the same driving route. We also see a strong spatial structure in the dimensionality reduction of this scene encoding.

This method demonstrates the capacity of neural networks to learn a representation of place. We believe that this approach is likely to equally benefit higher capacity modern neural network detectors, conditioning the network’s convolutional filters on the input scene. A key challenge for this will be addressing the quantity of training data needed for these networks, though we believe an approach based on the conservative self-supervised data mining methods in Chapter 3 could address this.

5.6 Summary Remarks

This thesis investigated exploiting a known operating environment to boost perceptual performance in object detectors for mobile robotics. It presents six contributions to object detection for mobile robotics:

1. We introduce the idea of experience-based object detection: using local expert detectors suited to a mobile robot’s operating environment. This theme underpins the entire thesis.
2. We present a method for constructing object detector models using self-supervised data mining on unlabelled data based on spatial heuristics (Chapter 2), and extend this with image segmentation (Chapter 3).

3. We assess the merits of fitting detector models to known categories of operating environment: firstly to seasonal effects along a known route, and secondly to seasonal and spatial effects incorporating both route and season (Chapter 2).
4. We demonstrate an approach to building local expert detection models, boosting performance by relaxing the requirement by strongly fitting highly specialised detectors to a narrow window of operation (Chapter 3). This approach mitigates the impact of limited model capacity for lightweight detector models.
5. We design a system (Chapter 3) to scale this highly specialised expert detector approach to operate with unlabelled data, coupling object detection to an experience-based visual localisation system. Quantitative and qualitative results demonstrate the effectiveness of these models in their operating environment.
6. We propose a more extensible approach which decouples the expert object detector from visual localisation, instead relying on a deep learning approach which conditions the detector model on the input image (Chapter 4).

In conclusion, place is important in mobile robotics as it provides us with knowledge about the structure of data the robot must interpret. This is a powerful benefit, and one which exclusively applies to robotic problems. We have demonstrated that we can exploit this additional context in object detection systems, enabling a lightweight, fast object detector to be competitive with a state-of-the-art object detector trained solely on a general purposes dataset. We verified this approach in the context of pedestrian detection for autonomous driving, though we expect this to apply equally to other target object classes and robotic applications. As computation resources available to mobile robots improves, we expect to be able to run better, higher capacity detectors in real-time on mobile platforms. We expect this approach to similarly improve performance for these higher capacity detectors, as our results indicate that model capacity is a clear limitation on the size

of operating environment fitted by a detector model. Fundamentally, this approach to object detection allows us to maximise the performance of our object detection systems in robotics, boosting the performance of an object detector constrained by the computational needs of mobile robots.

References

- [1] David Kortenkamp, Reid Simmons, and Davide Brugali. “Robotic Systems Architectures and Programming”. en. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. DOI: 10.1007/978-3-319-32552-1_12. Springer International Publishing, 2016, pp. 283–306. URL: http://link.springer.com/chapter/10.1007/978-3-319-32552-1_12 (visited on 04/14/2017).
- [2] Rodney A. Brooks. “Elephants don’t play chess”. In: *Robotics and Autonomous Systems*. Designing Autonomous Agents 6.1 (June 1990), pp. 3–15. URL: <http://www.sciencedirect.com/science/article/pii/S0921889005800259> (visited on 04/14/2017).
- [3] Brandon Schoettle and Michael Sivak. “A survey of public opinion about autonomous and self-driving vehicles in the US, the UK, and Australia”. In: (2014).
- [4] M. Kyriakidis, R. Happee, and J. C. F. de Winter. “Public opinion on automated driving: Results of an international questionnaire among 5000 respondents”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 32 (July 2015), pp. 127–140. URL: <http://www.sciencedirect.com/science/article/pii/S1369847815000777> (visited on 04/17/2017).
- [5] R. Okuda, Y. Kajiwara, and K. Terashima. “A survey of technical trend of ADAS and autonomous driving”. In: *Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*. Apr. 2014, pp. 1–4.
- [6] Ata M. Khan, Ataur Bacchus, and Stephen Erwin. “Policy challenges of increasing automation in driving”. In: *IATSS Research* 35.2 (Mar. 2012), pp. 79–89. URL: <http://www.sciencedirect.com/science/article/pii/S0386111212000039> (visited on 04/14/2017).
- [7] M. M. A. de Graaf, S. Ben Allouch, and J. A. G. M. van Dijk. “Long-term acceptance of social robots in domestic environments: Insights from a user’s perspective”. In: *2016 AAAI Spring Symposium: technical report*. Palo Alto: AAAI, 2016, pp. 96–103. URL: <http://doc.utwente.nl/101086/> (visited on 04/17/2017).
- [8] E. Broadbent, R. Stafford, and B. MacDonald. “Acceptance of Healthcare Robots for the Older Population: Review and Future Directions”. en. In: *International Journal of Social Robotics* 1.4 (Nov. 2009), p. 319. URL: <https://link.springer.com/article/10.1007/s12369-009-0030-6> (visited on 04/14/2017).

- [9] T. L. Chen et al. “Robots for humanity: using assistive robotics to empower people with disabilities”. In: *IEEE Robotics Automation Magazine* 20.1 (Mar. 2013), pp. 30–39.
- [10] Daniel Kappler et al. “Real-time Perception meets Reactive Motion Generation”. In: *arXiv:1703.03512 [cs]* (Mar. 2017). arXiv: 1703.03512. URL: <http://arxiv.org/abs/1703.03512> (visited on 04/17/2017).
- [11] Dominic Zeng Wang and Ingmar Posner. “Voting for Voting in Online Point Cloud Object Detection”. In: *Proceedings of Robotics: Science and Systems*. Rome, Italy, July 2015.
- [12] Martin Engelcke et al. “Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks”. In: *arXiv preprint arXiv:1609.06666* (2016).
- [13] M. Enzweiler and D. M. Gavrila. “Monocular Pedestrian Detection: Survey and Experiments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (Dec. 2009), pp. 2179–2195.
- [14] S. Sivaraman and M. M. Trivedi. “Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis”. In: *IEEE Transactions on Intelligent Transportation Systems* 14.4 (Dec. 2013), pp. 1773–1795.
- [15] S. Lowry et al. “Visual Place Recognition: A Survey”. In: *IEEE Transactions on Robotics* 32.1 (Feb. 2016), pp. 1–19.
- [16] P. Corke et al. “Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013, pp. 2085–2092.
- [17] Graham D. Finlayson, Mark S. Drew, and Cheng Lu. “Entropy Minimization for Shadow Removal”. en. In: *Int J Comput Vis* 85.1 (May 2009), pp. 35–57. URL: <http://link.springer.com/article/10.1007/s11263-009-0243-z> (visited on 02/29/2016).
- [18] S. Ratnasingam and T. M. McGinnity. “Chromaticity Space for Illuminant Invariant Recognition”. In: *IEEE Transactions on Image Processing* 21.8 (Aug. 2012), pp. 3612–3623.
- [19] P. Neubert, N. Sunderhauf, and P. Protzel. “Appearance change prediction for long-term navigation across seasons”. In: *2013 European Conference on Mobile Robots (ECMR)*. Sept. 2013, pp. 198–203.
- [20] Winston Churchill and Paul Newman. “Experience-based navigation for long-term localisation”. en. In: *The International Journal of Robotics Research* 32.14 (Dec. 2013), pp. 1645–1661. URL: <http://journals.sagepub.com/doi/abs/10.1177/0278364913499193> (visited on 03/01/2017).
- [21] Chris Linegar and Paul Newman. “Made to Measure: Bespoke Landmarks for 24-Hour, All-Weather Localisation with a Camera”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweeden, May 2016.

- [22] Radu Bogdan Rusu et al. “Towards 3D Point cloud based object maps for household environments”. In: *Robotics and Autonomous Systems*. Semantic Knowledge in Robotics 56.11 (Nov. 2008), pp. 927–941. URL: <http://www.sciencedirect.com/science/article/pii/S0921889008001140> (visited on 04/14/2017).
- [23] Rob Van Haaren. “Assessment of electric cars’ range requirements and usage patterns based on driving behavior recorded in the National Household Travel Survey of 2009”. In: *Earth and Environmental Engineering Department, Columbia University, Fu Foundation School of Engineering and Applied Science, New York* 51 (2011), p. 53.
- [24] Kevin J. Krizek. “Neighborhood services, trip purpose, and tour-based travel”. en. In: *Transportation* 30.4 (Nov. 2003), pp. 387–410. URL: <https://link.springer.com/article/10.1023/A:1024768007730> (visited on 08/06/2017).
- [25] Cecilia Jakobsson, Satoshi Fujii, and Tommy Gaerling. “Effects of economic disincentives on private car use”. en. In: *Transportation* 29.4 (Nov. 2002), pp. 349–370. URL: <https://link.springer.com/article/10.1023/A:1016334411457> (visited on 08/06/2017).
- [26] Roger L. Mackett. “Why do people use their cars for short trips?” en. In: *Transportation* 30.3 (Aug. 2003), pp. 329–349. URL: <https://link.springer.com/article/10.1023/A:1023987812020> (visited on 08/06/2017).
- [27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf> (visited on 04/17/2017).
- [28] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv:1409.1556 [cs]* (Sept. 2014). arXiv: 1409.1556. URL: <http://arxiv.org/abs/1409.1556> (visited on 04/17/2017).
- [29] J. Hosang et al. “Taking a deeper look at pedestrians”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 4073–4082.
- [30] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems* 28. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 91–99. URL: <http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf> (visited on 08/19/2016).
- [31] Zhaowei Cai et al. “A unified multi-scale deep convolutional neural network for fast object detection”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 354–370.

- [32] R. Benenson et al. "Pedestrian detection at 100 frames per second". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2012, pp. 2903–2910.
- [33] P. Dollar et al. "Fast Feature Pyramids for Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8 (Aug. 2014), pp. 1532–1545.
- [34] J Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Miami, Florida, USA, 20-25 June 2009, pp. 248–255.
- [35] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.
- [36] Shanshan Zhang et al. "How Far Are We From Solving Pedestrian Detection?" In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.
- [37] Kah-Kay Sung and T. Poggio. "Example-based learning for view-based human face detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.1 (Jan. 1998), pp. 39–51.
- [38] P.F. Felzenszwalb et al. "Object Detection with Discriminatively Trained Part-Based Models". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645.
- [39] A. Torralba and A. A. Efros. "Unbiased look at dataset bias". In: *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2011, pp. 1521–1528.
- [40] Aditya Khosla et al. "Undoing the damage of dataset bias". In: *Computer Vision–ECCV 2012*. Springer, 2012, pp. 158–171.
- [41] D. M. Gavrila and S. Munder. "Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle". en. In: *Int J Comput Vision* 73.1 (June 2007), pp. 41–59. URL: <http://link.springer.com/article/10.1007/s11263-006-9038-7> (visited on 01/15/2014).
- [42] Patrick Sudowe and Bastian Leibe. "Efficient Use of Geometric Constraints for Sliding-Window Object Detection in Video". In: *Proceedings of the International Conference on Computer Vision Systems (ICVS)*. Sophia Antipolis, France, Sept. 2011.
- [43] Anna Petrovskaya and Sebastian Thrun. "Model based vehicle detection and tracking for autonomous urban driving". In: *Autonomous Robots* 26.2-3 (2009), pp. 123–139.
- [44] Carl Doersch et al. "What Makes Paris Look Like Paris?" In: *ACM Trans. Graph.* 31.4 (July 2012), 101:1–101:9. (Visited on 08/26/2014).
- [45] Paul Furgale and Timothy D. Barfoot. "Visual teach and repeat for long-range rover autonomy". en. In: *J. Field Robotics* 27.5 (Sept. 2010), pp. 534–560. URL: <http://onlinelibrary.wiley.com/doi/10.1002/rob.20342/abstract> (visited on 09/01/2014).

- [46] Alex Stewart and Paul Newman. “LAPS - Localisation using Appearance of Prior Structure: 6-DOF Monocular Camera Localisation using Prior Pointclouds”. In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Minnesota, USA, May 2012.
- [47] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*. Vol. 1. 2005, 886–893 vol. 1.
- [48] Charles Dubout and François Fleuret. “Exact Acceleration of Linear Object Detectors”. In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Lecture Notes in Computer Science 7574. Springer Berlin Heidelberg, Jan. 2012, pp. 301–311. URL: http://link.springer.com/chapter/10.1007/978-3-642-33712-3_22 (visited on 10/29/2013).
- [49] David Geronimo et al. “2D–3D-based on-board pedestrian detection system”. In: *Computer Vision and Image Understanding* 114.5 (May 2010), pp. 583–595. (Visited on 01/20/2014).
- [50] M. Enzweiler et al. “Efficient Stixel-based object recognition”. In: *2012 IEEE Intelligent Vehicles Symposium (IV)*. June 2012, pp. 1066–1071.
- [51] Andreas Ess, Bastian Leibe, and Luc Van Gool. “Depth and Appearance for Mobile Scene Analysis”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Oct. 2007.
- [52] Asako Kanezaki et al. “Hard Negative Classes for Multiple Object Detection”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. May 2014.
- [53] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. “Discriminative decorrelation for clustering and classification”. In: *European Conference on Computer Vision* (2012).
- [54] J.F. Henriques et al. “Beyond Hard Negative Mining: Efficient Detector Learning via Block-Circulant Decomposition”. In: *2013 IEEE International Conference on Computer Vision (ICCV)*. Dec. 2013, pp. 2760–2767.
- [55] Alex Teichman and Sebastian Thrun. “Group Induction”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, Mar. 2013, pp. 2757–2763.
- [56] Bastian Leibe, Ales Leonardis, and Bernt Schiele. “Robust Object Detection with Interleaved Categorization and Segmentation”. en. In: *International Journal of Computer Vision* 77.1-3 (May 2008), pp. 259–289. URL: <https://link.springer.com/article/10.1007/s11263-007-0095-3> (visited on 04/03/2017).
- [57] Stephen Gould, Tianshi Gao, and Daphne Koller. “Region-based Segmentation and Object Detection”. In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio et al. Curran Associates, Inc., 2009, pp. 655–663. URL: <http://papers.nips.cc/paper/3766-region-based-segmentation-and-object-detection.pdf> (visited on 04/03/2017).
- [58] P. Dollár, S. Belongie, and P. Perona. “The Fastest Pedestrian Detector in the West”. In: *BMVC*. 2010.

- [59] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. “Non-maximum Suppression for Object Detection by Passing Messages Between Windows”. en. In: *Computer Vision – ACCV 2014*. Springer, Cham, Nov. 2014, pp. 290–306. URL: https://link.springer.com/chapter/10.1007/978-3-319-16865-4_19 (visited on 04/08/2017).
- [60] Jim Schwiegerling. “Field guide to visual and ophthalmic optics”. In: Spie, 2004.
- [61] R. Benenson et al. “Ten years of pedestrian detection, what have we learned?”. In: *ECCV, CVRSUAD workshop*. 2014.
- [62] Piotr Dollar et al. “Integral Channel Features”. en. In: British Machine Vision Association, 2009, pp. 91.1–91.11. URL: <http://www.bmva.org/bmvc/2009/Papers/Paper244/Paper244.html> (visited on 04/03/2017).
- [63] P. Dollár et al. “Pedestrian detection: A benchmark”. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*. 2009, pp. 304–311.
- [64] Mark Philip Philipson et al. “Traffic light detection: a learning algorithm and evaluations on challenging dataset”. In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE. 2015, pp. 2341–2345.
- [65] Andreas Møgelmoose, Dongran Liu, and Mohan Manubhai Trivedi. “Detection of US traffic signs”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3116–3125.
- [66] Christian Wojek and Bernt Schiele. “A Performance Evaluation of Single and Multi-feature People Detection”. In: *Proceedings of the 30th DAGM Symposium on Pattern Recognition*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 82–91. (Visited on 08/19/2016).
- [67] S. Maji, A.C. Berg, and J. Malik. “Classification using intersection kernel support vector machines is efficient”. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*. 2008, pp. 1–8.
- [68] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.
- [69] P. Dollár et al. “Pedestrian Detection: An Evaluation of the State of the Art”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (Apr. 2012), pp. 743–761.
- [70] P.H.S. Torr and A. Zisserman. “MLESAC: A New Robust Estimator with Application to Estimating Image Geometry”. In: *Computer Vision and Image Understanding* 78.1 (Apr. 2000), pp. 138–156. URL: <http://www.sciencedirect.com/science/article/pii/S1077314299908329> (visited on 12/11/2013).
- [71] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, ISBN: 0521540518, 2004.
- [72] James R. Curran, Tara Murphy, and Bernhard Scholz. “Minimising semantic drift with Mutual Exclusion Bootstrapping”. In: *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*. Sept. 2007, pp. 172–180.

- [73] Rong-En Fan et al. “LIBLINEAR: A Library for Large Linear Classification”. In: *Journal of Machine Learning Research (JMLR)* 9 (Aug. 2008), pp. 1871–1874.
- [74] *The OpenCV Library*. URL: <http://opencv.org/>.
- [75] M. Enzweiler et al. “Multi-cue pedestrian classification with partial occlusion handling”. In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2010, pp. 990–997.
- [76] Will Maddern et al. “Illumination Invariant Imaging: Applications in Robust Vision-based Localisation, Mapping and Classification for Autonomous Vehicles”. In: *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, May 2014.
- [77] C. McManus et al. “Shady dealings: Robust, long-term visual localisation using illumination invariance”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 901–906.
- [78] Jeffrey Hawke et al. “Wrong Today, Right Tomorrow: Experience-Based Classification for Robot Perception”. In: *Field and Service Robotics (FSR)*. Toronto, ON, Canada, June 2015.
- [79] Alex Bewley and Ben Upcroft. “From ImageNet to Mining: Adapting Visual Object Detection with Minimal Supervision”. In: *Field and Service Robotics*. Springer. 2016, pp. 501–514.
- [80] Laszlo-Peter Berczi and Timothy D Barfoot. “It’s like Déjà Vu all over again: Learning place-dependent terrain assessment for visual teach and repeat”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 3973–3980.
- [81] Colin McManus, Ben Upcroft, and Paul Newman. “Scene Signatures: Localised and Point-less Features for Localisation”. In: *Proceedings of Robotics: Science and Systems*. Berkeley, USA, July 2014.
- [82] S. Zhang, C. Bauckhage, and A. B. Cremers. “Informed Haar-Like Features Improve Pedestrian Detection”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. June 2014, pp. 947–954.
- [83] A Prioletti et al. “Part-Based Pedestrian Detection and Feature-Based Tracking for Driver Assistance: Real-Time, Robust Algorithms, and Evaluation”. In: *IEEE Transactions on Intelligent Transportation Systems* 14.3 (Sept. 2013), pp. 1346–1359.
- [84] S. Walk et al. “New features and insights for pedestrian detection”. In: *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2010, pp. 1030–1037.
- [85] W. Ouyang and X. Wang. “Single-Pedestrian Detection Aided by Multi-pedestrian Detection”. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013, pp. 3198–3205.
- [86] Fan Yang, Wongun Choi, and Yuanqing Lin. “Exploit All the Layers: Fast and Accurate CNN Object Detector With Scale Dependent Pooling and Cascaded Rejection Classifiers”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

- [87] Chris Linegar, Winston Churchill, and Paul Newman. “Work Smart, Not Hard: Recalling Relevant Experiences for Vast-Scale but Time-Constrained Localisation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA, May 2015.
- [88] Matthew Gadd and Paul Newman. “Checkout My Map: Version Control for Fleetwide Visual Localisation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea, Oct. 2016.
- [89] M. Paton et al. “Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016, pp. 1918–1925.
- [90] T. Malisiewicz, A Gupta, and AA Efros. “Ensemble of exemplar-SVMs for object detection and beyond”. In: *2011 IEEE International Conference on Computer Vision (ICCV)*. Nov. 2011, pp. 89–96.
- [91] H. Hattori et al. “Learning scene-specific pedestrian detectors without real data”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015, pp. 3819–3827.
- [92] Xiaogang Wang, Meng Wang, and Wei Li. “Scene-Specific Pedestrian Detection for Static Video Surveillance”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.2 (Feb. 2014), pp. 361–374.
- [93] Daichi Suzuo et al. “Environment adaptive pedestrian detection using in-vehicle camera and GPS”. In: *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*. Vol. 2. Jan. 2014, pp. 354–361.
- [94] Alex Bewley and Ben Upcroft. “Background Appearance Modeling with Applications to Visual Object Detection in an Open-Pit Mine”. In: *Journal of Field Robotics* (2016), n/a–n/a. URL: <http://dx.doi.org/10.1002/rob.21667>.
- [95] Will Maddern et al. “1 Year, 1000km: The Oxford RobotCar Dataset”. In: *The International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15. eprint: <http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+html>. URL: <http://dx.doi.org/10.1177/0278364916679498>.
- [96] Aude Oliva and Antonio Torralba. “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope”. en. In: *International Journal of Computer Vision* 42.3 (May 2001), pp. 145–175. URL: <https://link.springer.com/article/10.1023/A:1011139631724> (visited on 04/16/2017).
- [97] Paul Viola and William M Wells III. “Alignment by maximization of mutual information”. In: *International journal of computer vision* 24.2 (1997), pp. 137–154.
- [98] Matthijs Douze et al. “Evaluation of GIST Descriptors for Web-scale Image Search”. In: *Proceedings of the ACM International Conference on Image and Video Retrieval*. CIVR ’09. New York, NY, USA: ACM, 2009, 19:1–19:8. (Visited on 07/04/2016).
- [99] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. “Mutual-information-based registration of medical images: a survey”. In: *IEEE Transactions on Medical Imaging* 22.8 (Aug. 2003), pp. 986–1004.

- [100] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 3320–3328. URL: <http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf>.
- [101] Peter Nelson, Chris Linegar, and Paul Newman. “Building, Curating, and Querying Large-scale Data Repositories for Field Robotics Applications”. In: *International Conference on Field and Service Robotics (FSR)*. Toronto, ON, Canada, June 2015.
- [102] Tobias Pohlen et al. “Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes”. In: *CoRR* abs/1611.08323 (2016). URL: <http://arxiv.org/abs/1611.08323>.
- [103] Marius Cordts et al. “The Cityscapes Dataset for Semantic Urban Scene Understanding”. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.
- [104] Alex Kendall and Yarin Gal. “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *arXiv:1703.04977 [cs]* (Mar. 2017). arXiv: 1703.04977. URL: <http://arxiv.org/abs/1703.04977>.
- [105] Xu Jia et al. “Dynamic Filter Networks”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 667–675. URL: <http://papers.nips.cc/paper/6578-dynamic-filter-networks.pdf> (visited on 04/16/2017).
- [106] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [107] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. en. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. URL: <http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html> (visited on 04/17/2017).
- [108] Hasim Sak, Andrew Senior, and Francoise Beaufays. “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition”. In: *arXiv:1402.1128 [cs, stat]* (Feb. 2014). arXiv: 1402.1128. URL: <http://arxiv.org/abs/1402.1128> (visited on 04/17/2017).
- [109] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. en. In: *Computer Vision - ECCV 2014*. Springer, Cham, Sept. 2014, pp. 818–833. URL: https://link.springer.com/chapter/10.1007/978-3-319-10590-1_53 (visited on 04/27/2017).
- [110] Christian Szegedy et al. “Going Deeper With Convolutions”. In: 2015, pp. 1–9. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html (visited on 04/27/2017).

- [111] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 2016, pp. 770–778. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (visited on 04/27/2017).
- [112] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: 2014, pp. 580–587. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html (visited on 04/27/2017).
- [113] Ross Girshick. “Fast R-CNN”. In: 2015, pp. 1440–1448. URL: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Girshick_Fast_R-CNN_ICCV_2015_paper.html (visited on 04/27/2017).
- [114] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *arXiv preprint arXiv:1612.08242* (2016).
- [115] Jeff Donahue et al. “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”. In: *CoRR* abs/1310.1531 (2013). arXiv: 1310.1531. URL: <http://arxiv.org/abs/1310.1531>.
- [116] Adam R Kosiorek, Alex Bewley, and Ingmar Posner. “Hierarchical Attentive Recurrent Tracking”. In: *Neural Information Processing Systems*. Dec. 2017. URL: http://www.robots.ox.ac.uk/~mobile/Papers/2017NIPS_AdamKosiorek.pdf.
- [117] B. Zhou et al. “Learning Deep Features for Discriminative Localization”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 2921–2929.
- [118] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *CoRR* abs/1511.07289 (2015). arXiv: 1511.07289. URL: <http://arxiv.org/abs/1511.07289>.
- [119] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv:1412.6980 [cs]* (Dec. 2014). arXiv: 1412.6980. URL: <http://arxiv.org/abs/1412.6980> (visited on 04/25/2017).
- [120] Kin Leong Ho and Paul Newman. “Detecting Loop Closure with Scene Sequences”. In: *Int. J. Comput. Vision* 74.3 (Sept. 2007), pp. 261–286. URL: <http://dx.doi.org/10.1007/s11263-006-0020-1>.
- [121] Laurens van der Maaten. “Accelerating t-SNE using Tree-Based Algorithms”. In: *Journal of Machine Learning Research* 15 (2014), pp. 3221–3245. URL: <http://jmlr.org/papers/v15/vandermaaten14a.html>.
- [122] Jonathon Shlens. “A Tutorial on Principal Component Analysis”. In: *CoRR* abs/1404.1100 (2014). URL: <http://arxiv.org/abs/1404.1100>.
- [123] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. “CityPersons: A Diverse Dataset for Pedestrian Detection”. In: *arXiv preprint arXiv:1702.05693* (2017).