THE UNIVERSITY OF
SYDNEY

Multi-target tracking using appearance models for identity maintenance

Peter Morton

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

THE UNIVERSITY OF
SYDNEY

Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

June 2014

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Peter Morton

June 15$^{\text{th}}$, 2014

Abstract

Peter Morton
The University of Sydney

Doctor of Philosophy
June 2014

Multi-target tracking using appearance models for identity maintenance

This thesis considers perception systems for urban environments. It focuses on the task of tracking dynamic objects and in particular on methods that can maintain the identities of targets through periods of ambiguity. Examples of such ambiguous situations occur when targets interact with each other, or when they are occluded by other objects or the environment.

With the development of self driving cars, the push for autonomous delivery of packages, and an increasing reliance on technology for security, surveillance and public-safety applications, robust perception in crowded urban spaces is more important than ever before. A critical part of perception systems is the ability to understand the motion of objects in a scene. Tracking strategies that merge closely-spaced targets together into groups have been shown to offer improved robustness, but in doing so sacrifice the concept of target identity. Additionally, the primary sensor used for the tracking task may not provide the type of information required to reason about the identity of individual objects.

There are three primary contributions in this work. The first is the development of 3D lidar tracking methods with improved ability to track closely-spaced targets and that can determine when target identities have become ambiguous. Secondly, this thesis defines appearance models suitable for the task of determining the identities of previously-observed targets, which may include the use of data from additional sensing modalities. The final contribution of this work is the combination of lidar tracking and appearance modelling, to enable the clarification of target identities in the presence of ambiguities caused by scene complexity.

The algorithms presented in this work are validated using established metrics from the literature, and also compared to an additional measure adapted from clustering analysis; V-measure is shown in this thesis to better reflect identity tracking performance. Evaluation is performed on both carefully controlled and unconstrained datasets, comprising 19 labelled scenarios with 89717 observations of 78 targets over a total of 25048 lidar scans, and an unconstrained urban dataset featuring a further 27375 scans. The experiments show that the proposed methods achieve significant improvements in tracking performance, in complex dynamic scenes with interacting targets, such as those commonly found in an urban environment.

Acknowledgements

My heartfelt thanks to my family, friends and colleagues who have made this thesis possible.

To my supervisor, James Underwood. Thank you for your dedication and integrity as a researcher and as a mentor. Your willingness to discuss all the aspects of my project often and in depth has been invaluable to me. Thank you for teaching me the joys (or at least the merits) of technical writing. The detailed feedback you have provided on not only this thesis, but all of my writing, is truly appreciated. Thank you, also, for giving me the freedom to pursue my technical interests. Whilst the results of my "20% time" don't make much of an appearance in this document, the skills I learnt will play a big role in my next steps from here.

To my associate supervisor, Bertrand Douillard. Thank you for the time spent discussing ideas, working through problems and scribbling diagrams. Thanks for your close collaboration on code and publications and for your unwavering confidence that deadlines are "tight, but feasible."

James and Bertrand, the model of collaboration you set up in the CIMS perception team made my time at ACFR so much more productive and enjoyable. Alastair, thank you for shared code and shared ideas. I think our "pyception" development strategy was right on the money.

Thank you to all my friends at ACFR. Thanks to the lunch time crew for the debates about text editors, operating systems and programming languages. Without your vim tips I would never have finished this thesis on time. Thanks to the down stairs gang for the adventures, road trips and general shenanigans. Thank you Chris, Roman and Marcos for the fun at Marulan and helping gather much of the data used in this thesis.

To my family, thank you for your love and support. Although each step on my journey seems to take me further from Melbourne, it is always wonderful to return home. Thanks, finally, to Nicole for all the patience, encouragement and love I could ever ask for.

Contents

Chapter 1

Introduction

This thesis considers perception systems for urban environments. It focuses on the task of tracking dynamic objects and in particular on methods that can recover the identities of targets after periods of ambiguity. Such ambiguous situations may occur when targets interact with each other, or when they are occluded by other objects or the environment.

The principal contribution of this thesis is the development of tracking methods that use a combination of sensing modalities to track the locations and identities of interacting targets. These methods can be broken down into three main parts:

- a lidar-based tracking system, specifically designed to handle closely-spaced targets and determine when target identities have become ambiguous;
- appearance models, extracted from lidar or camera data, that can be used to re-establish target identity after periods of ambiguity; and
- a framework that combines lidar-based tracking with appearance models to allow the extraction of complete object trajectories even in the presence of ambiguous interactions.

In this chapter we present the motivation for developing systems that can maintain object identity, define the problem to be solved specifically, and describe the princi-

pal contributions of this thesis. An overview of the thesis structure is presented in Section 1.4.

1.1 Motivation

Perception is one of the fundamental challenges of robotics. For a robot to operate in an uncontrolled environment it must be able to observe its surroundings and take them into account when making decisions. The role of a perception system is to take raw sensor data and convert it into information useful to higher-level applications.

Urban environments impose certain requirements on and challenges for perception systems. Urban scenes are both geometrically complex and highly dynamic, and operation in such conditions requires that the robot's internal model of the environment is constantly updated and its consistency maintained.

The temporal evolution of a scene is a key component that perception algorithms must address explicitly. In its simplest form, this might involve the classification of the scene into dynamic and static components, however there are many applications for which higher-level information is required. A service robot following its owner through a crowd, monitoring livestock behaviour, tracing the path of a suspected criminal through a closed circuit television (CCTV) network; for all of these applications, the concept of *who* is just as important as *where*. We refer to this as "identity tracking".

Dynamic urban environments present sources of ambiguity that make the identity tracking problem difficult. Complex scene geometry creates occlusions, regions within which a target is not observed. Combined with multiple moving targets, this leads to situations where a tracking system loses the ability to tell which target is which. That is, their identities become ambiguous. Similarly, objects moving in proximity to one another generate additional occlusions or become close enough together such that a sensor can no longer distinguish the individual targets.

Figure 1.1 shows an example of an "interaction" event that causes identity uncertainty. Two targets, initially observed independently, move close together and cannot be

separated by the sensor. When they move apart again, the positions of the targets can be determined, but their identities are ambiguous.

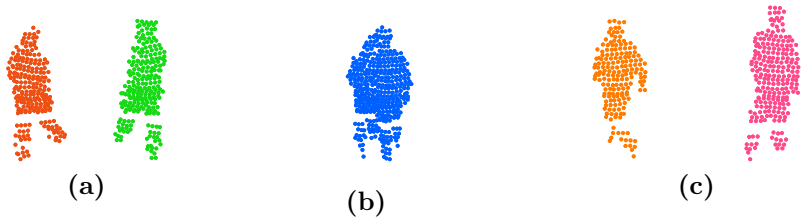


Figure 1.1 – Example of an "interaction" between two pedestrians, as observed using a 3D lidar sensor. The targets are observed apart (a), then come together (b) before moving apart again (c). Whilst the positions of the targets are clear, their identities have are ambiguous. Did the targets cross paths, or move back in the direction whence they came?

Ambiguous interactions are common in urban scenes, such as that shown in Figure 1.2. This image of pedestrians crossing the road at an intersection is taken from a dataset recorded in the city of Sydney, Australia. It shows two different types of interactions, pedestrians crossing paths, and pedestrians walking in the same direction in close proximity to one another. There are also many occlusions; at the instant shown, four pedestrians are in a line from the sensor.

Figure 1.2 – Camera data from a pedestrian crossing in an urban environment

Applications that require information about the identities of tracked targets dictate the need for tracking algorithms that can reason about ambiguous situations and provide the ability to recover identity information.

The literature surrounding this topic is divided into two main areas. Perception systems for self-driving cars provide robust multi-target tracking, but for such a task it is sufficient to know the locations and velocities of objects. The concept of target identity is not important. In the field of computer vision, tracking occurs in the image frame, and most commonly from static sensors. Whilst maintaining target identity is an active area of research, the 3D geometry of a scene is not generally considered. In this thesis, we consider cases that require both accurate estimates of 3D target location and maintenance of target identities.

1.2 Problem Statement

The ability of tracking systems to maintain target identity is constrained by a number of factors. In an ideal scenario, perfect sensor coverage with no occlusions, sufficient resolution to separate all targets, and a frame rate fast enough to observe all dynamics would prevent target identities from ever become ambiguous, but this thesis considers identity tracking from a single sensor vehicle using data that is subject to the limitations of current sensing technologies. Furthermore, no a-priori knowledge of the environment is assumed, and tracking does not require participation on the part of the targets, either in the form of cooperative motion (following predefined paths), or instrumentation (such as transponders).

Given these limitations, target identities will at times become ambiguous. A tracking system must recognise these situations and be able to recover from them. To do so, the system needs to use the observable data to learn target representations that allow them to be recognised at a later point in time. Referring to Figure 1.1, the system is unable to distinguish the targets in frame (b). However, information extracted from the targets in (a), perhaps from a complementary sensor, could be used to solve their identities in (c).

Learning models that allow identities to be determined uniquely within the set of all tracked targets can prove difficult. Tracking systems can reduce the complexity of the problem by explicitly reasoning about which target identities are ambiguous. Then,

the target representations need only be powerful enough to discriminate amongst these candidate identities.

In this thesis we develop approaches that allow object identities to become ambiguous, but do so in such a way that targets can be recognised and their identity recovered at a later point in time.

1.3 Thesis Contributions

This thesis proposes methods that improve the ability of tracking systems to maintain object identity. By fusing data from multiple sensors and explicitly identifying ambiguous situations, the methods achieve robust positional and identity tracking and can maintain an internal understanding of identity uncertainty. A snapshot of the tracking output from the methods developed in this thesis is shown in Figure 1.3.

Figure 1.3 – Sample tracking output obtained by applying the methods developed in this thesis to the data shown in Figure 1.2.

The contributions of this thesis can be broken down into three main areas: lidar-based tracking, appearance modelling and identity reasoning. The specific contributions in these areas are detailed below.

Lidar-based tracking

- Proposal of a method for lidar-based tracking with improved ability to track closely-spaced targets, whilst preserving track homogeneity.
- Adaptation of the event graph approach from [106] to lidar-based tracking, enabling explicit determination and representation of identity ambiguity.
- Detailed experimental evaluation of tracking methods, on a variety of interaction scenarios involving pedestrians and cyclists.

Appearance Modelling

- Definition of appearance models, and how they relate to the identity tracking problem.
- Analysis of methods for fusing lidar and camera data to build appearance models, including techniques to select image regions based on lidar data that was sampled at a lower resolution.
- Experimental comparison of appearance models on real data, using a testing scheme that simulates interactions between objects.

Identity reasoning

- Application of the "hypothesis graph" identity-reasoning framework to the event graphs constructed in this work
- An extension to a Bayesian network based identity reasoning method that increases the range of event graphs to which the method can be applied
- Evaluation of appearance models and tracking techniques over a wide variety of interaction scenarios, including a manually labelled dataset containing 89717 observations of 78 targets over a total of 25048 lidar scans.

- An analysis of identity uncertainty in terms of graph entropy.
- Evaluation of the tracking methods on an unconstrained urban dataset, comprising 27375 scans.

1.4 Thesis Structure

Chapter 2 presents the background of this thesis. A general overview of the field of target tracking is presented. We provide a breakdown of basic tracking methods, describe the "identity management" problem and discuss sensors and their applicability to tracking in urban environments. In Section 2.4, we describe the metrics that will be used in the rest of the thesis to evaluate multi-target tracking, and present the motivation for our choice of V-measure as an identity tracking performance measure.

Chapter 3 examines methods for tracking interacting objects using lidar data. It describes the pre-processing steps required to convert lidar data into "observations", which form the input to the tracking system. The chapter focuses on the key problems of segmentation and data-association, proposes a novel method that addresses some of these issues, and motivates the need to switch to "group tracking" approaches in complex scenes. An "event graph" approach is adopted from the computer vision literature as a way to maintain a record of the relationships between tracks, and determine sources of identity ambiguity.

Whilst tracking the position of objects can be largely solved by the methods of Chapter 3, tracking identities of targets requires additional information. **Chapter 4** examines how to build "appearance models", which improve tracking performance by allowing the re-identification of targets whose identities have become ambiguous due to phenomena such as an interaction with another target or period of non-observation. The appearance modelling parameters that are relevant to a system designer are described and evaluated in extensive experimentation, using real-world data in a testing scheme that simulates arbitrary object interactions.

Chapter 5 combines these appearance models with the event graphs constructed in Chapter 3 to solve ambiguous tracking situations. Two frameworks for achieving this are discussed in detail, and the abilities of these methods compared. An analysis of tracking performance, both in terms of metric accuracy and identity maintenance is presented, and the ability of one of these frameworks to measure its own reduction in identity uncertainty is explored. The results of a large scale experiment, performed in an unconstrained urban environment, are discussed.

Conclusions and a statement of future work are presented in **Chapter 6**.

Chapter 2

Background

Tracking is "the estimation of the state of a moving object based on remote measurements" [6]. Systems which automate this process date back to World War II, when their introduction revolutionised fire-control, the task of aiming and firing naval guns. From 1940 to 1946 the use of radar systems progressed from simple range finders to automated systems that could track a target and predict its future position, determining range, elevation and bearing and enabling accurate fire-control in all visibility conditions [22]. Whilst the field of tracking has greatly expanded over the intervening years and now encompasses applications as diverse as robotic surgery [26], sports broadcasting [17] and self-driving cars [56, 67, 110], the development of radar tracking during World War II, as described by Coales et al. [22], illustrates many of the aspects of the tracking problem that remain active topics of research today.

Consider the task of tracking aircraft in the example above. An automated tracking system must first determine if there are any targets of interest within its field of view (FOV). Once a target is acquired, the range, bearing and elevation to the target can be determined from a single observation. Then, to determine the target's velocity the system must measure the location of the same target at different times. Finally, the system must also be able to predict the future location of a target, such that it can compensate for the time delays associated with loading and firing the gun and also the time taken for the shell to reach the target. That all of these aspects could

be achieved in the 1940s using a combination of mechanical systems and analogue electronics is an impressive feat of engineering.

When designing perception systems for urban environments, aspects of the tracking problem that were not relevant to Coales et al. become important. Foremost amongst these is the concept of tracking multiple targets simultaneously. In the case of the radar fire-control, each radar tracked a single target, however, in an urban environment one sensor system might be used to observe an entire scene and track all the objects present. A second aspect is that the entire state history of a target may be relevant; surveillance and monitoring systems, for example, are concerned with not just the present and future location of target but the whole trajectory.

This chapter describes the fundamental concepts of object tracking, with more specific background material introduced in the subsequent chapters. In Section 2.1, we describe the two broad classes of tracking algorithms, which we refer to as "tracking by detection" and "tracking by registration". Section 2.2 introduces the concept of "target identity" and in Section 2.3 we discuss the sensor technologies applicable to the task of tracking dynamic objects from a moving sensor platform in an urban environment. In Section 2.4, we introduce the metrics that will be used throughout this work to evaluate tracking performance.

2.1 Tracking approaches

In the robotics and computer vision literature there are two broad classes of tracking algorithms, which we will refer to as 'tracking by detection' and 'tracking by registration'. In practise, the line between the two is blurry and many systems make use of elements of each method. We review some specific instances of these hybrid approaches in Section 2.1.3.

In the sections that follow, we use the term 'frame' to refer to sensor data from one sampling period. For a camera, a 'frame' is a single image, whilst for a rotating 3D lidar sensor it encompasses the data from one sensor revolution, which we also refer

to as a 'scan'. Tracking algorithms tend to operate on discrete time steps, one frame at a time.

2.1.1 Tracking by detection

In 'tracking by detection', a data stream is separated into sets of observations through the use of a detector. A detector might, for example, be tuned to look for pedestrians, motion capture markers, or items on a conveyor belt. Objects detected in each frame (*observations*) are matched with observations in subsequent frames to form a track. An overview of the approach as applied to 3D lidar data is shown in Figure 2.1.

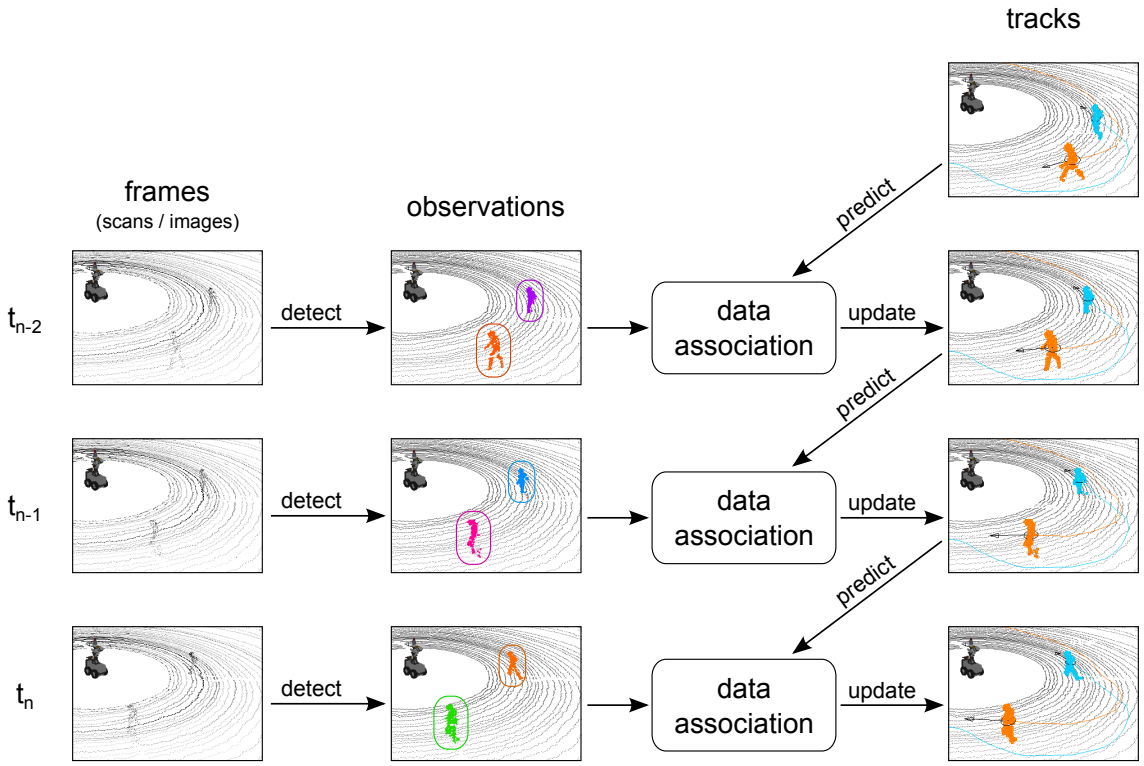


Figure 2.1 – Tracking by detection. At each time step, a data frame is transformed into observations through the use of a detector. Observations are then used to update existing tracks, or create new ones.

Detection is performed on a frame-by-frame basis, independent of previous detection results. The detector finds, for example, pedestrians in a scene, but these detections do not necessarily correspond to the same objects as were detected in a previous

frame. At the same time, the *state* (position, velocity and/or perhaps other properties) of each existing track is used to *predict* where the tracked objects are now. The correspondence of observations to tracks is determined in a process called *data association*. When an observation is assigned to a track, the track's state is updated. New tracks are created for unmatched observations, and old, unobserved tracks expire.

Each part of the algorithm leads to a number of design choices, which will be expanded upon in the sections that follow. We first discuss methods for object detection via classification and background subtraction, followed by the prediction, data association and track management steps.

2.1.1.1 Detection via classification

A common way to find objects of interest in frames of a data stream is to use a class-specific model. A binary classifier that is sensitive to a particular type of object (e.g. car, tree, pedestrian) is either manually specified or learnt from data and evaluated on sub-regions of each frame. Regions in which the classifier returns a positive result form the set of observations. Perhaps the most well-known object-specific detection framework is the "Viola-Jones" method [113], which uses a cascade of AdaBoost [34] classifiers to enable real-time detection of faces in video data. Detection via classification is most commonly used with camera data, and other well-known examples include the use of the histograms of oriented gradients (HOG) descriptor of Dalal and Triggs [23].

Class-specific models for 3D lidar data are yet to achieve the same performance as their camera-based counterparts, and are less commonly used. Recent approaches, however, have shown promising results, including a support vector machine (SVM) based method for detecting pedestrians in 3D lidar data, which was proposed in [93] and combined with tracking in [94]; a model for separating dynamic and static objects [48]; and a method for detecting cars, pedestrians and bicyclists [115].

2.1.1.2 Detection via background subtraction

The previous section described an approach for object detection that sought to model the object of interest. An alternative strategy is to model the background of the scene instead. Subtracting the background then gives the foreground objects, which are then separated from each other to form the set of observations.

For lidar data in urban environments, this is the most common detection approach [28, 43, 69, 76], and is most readily understood by considering the 3D structure of human environments. In street scenes in particular, the ground surface acts as the scene background and once removed, only objects remain. An additional helpful property of street scenes is that dynamic objects (pedestrians, cyclists, vehicles) tend to maintain their separation from other objects. Therefore, non-ground points can be readily clustered into objects. Figure 2.2 shows an example of object detection by background subtraction and subsequent segmentation. The ground points, shown in grey, are detected and removed, and then non-ground points are clustered into observations. A weakness of this approach is that objects in close proximity to one another may be clustered together. This is referred to as under-segmentation.

An alternative form of background subtraction, used for both 3D lidar and camera data is change detection. Rather than modelling the background as a ground surface, the background is learnt from the regions of the scene that do not change from frame to frame. Figure 2.3 shows the result of change detection from [109] applied to the same data as in Figure 2.2. A disadvantage of this method is that objects that might move, but are not presently moving, are ignored. In an autonomous driving application, for instance, pedestrians waiting at a crossing would not be detected until they start to move. By contrast, the ground detection and subtraction approach would allow the pedestrians to be detected and the car to slow down or give way.

For vision data, the colour of the static background can be modelled. Figure 2.4 shows an example of object detection via background subtraction. Even though dynamic objects are present in each input frame, median filtering allows the non-changing parts of the scene to be extracted – notice across the middle row how the dynamic

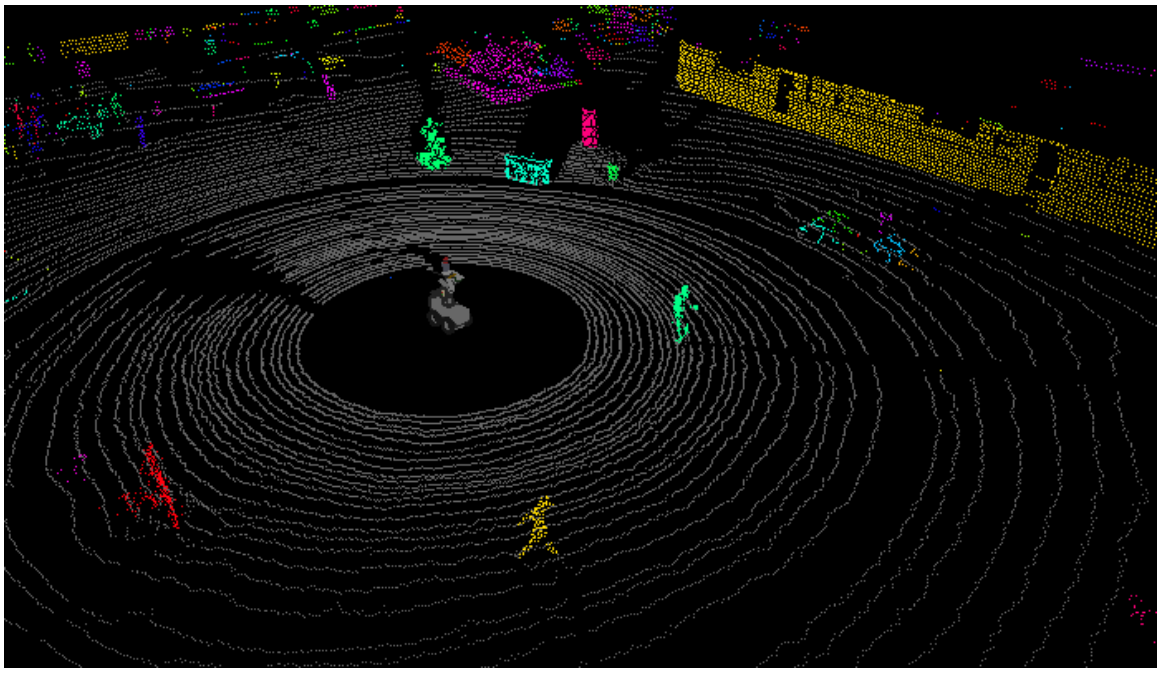


Figure 2.2 – Segmentation of Velodyne data, from a scene containing two walking pedestrians, a wall (top right) and a number of trees. The sensor vehicle is also shown. Ground points are detected and shown in grey. The remaining non-ground points are clustered into observations (each shown in a different colour).

objects disappear as more data is included in the filtering process. The final image in the middle row is then used as a background model, and by subtracting it from each image, the dynamic parts of the scene can be found. The bottom row of Figure 2.4 shows the thresholded difference between images and the final background model, as well as the objects found by blob detection. Whilst this approach works well in this toy example, it is not robust against slight variations in the background (notice how the leaves of the trees show up as changes in the bottom row). Therefore, more robust techniques such as Gaussian mixture models (GMMs) are usually used to model the background [95].

This method is only practical for fixed cameras in scenes with no fast changes in illumination. Additionally, since this method operates in (2D) image coordinates, it is much more prone to merging observations of multiple objects as can be seen in the right hand column of Figure 2.4. Even though the two objects are separated in 3D space, they overlap in image coordinates and are observed as one object. For these

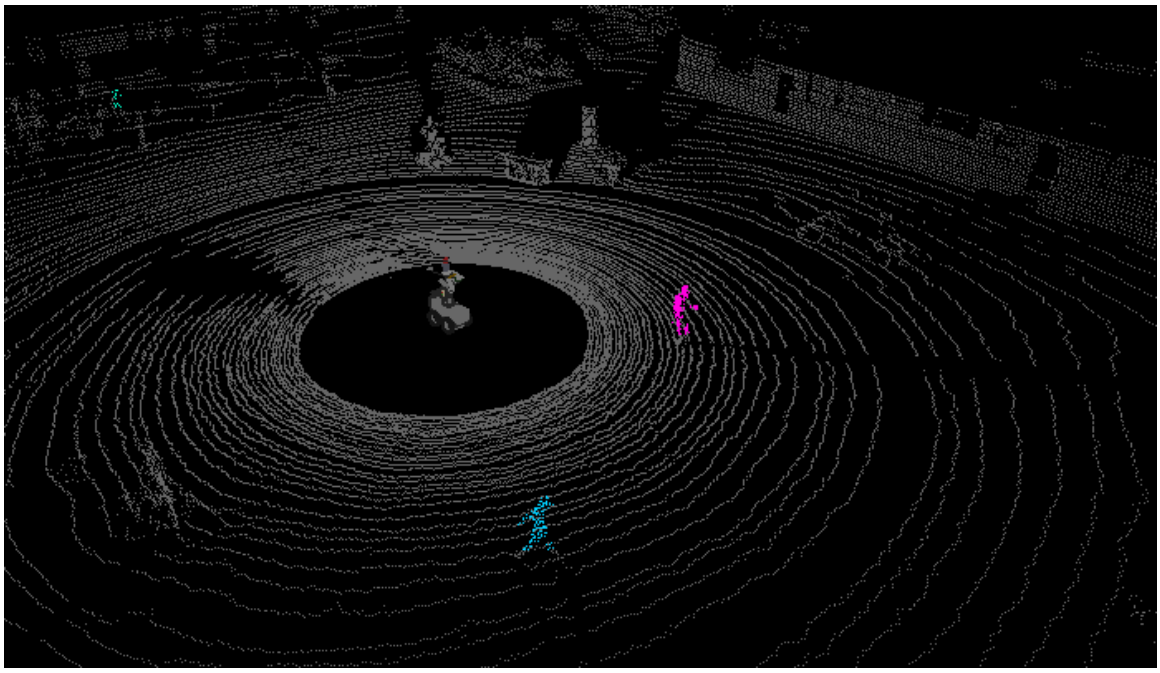


Figure 2.3 – Velodyne change detection example. Image shows the same data as Figure 2.2, but this time change detection [109] was used to find the foreground objects. Buildings, trees, etc. are background and shown in grey. Only three moving objects were detected, and are shown in colour.

reasons, moving cameras in urban environments more commonly use class-specific detectors (e.g. [35]).

2.1.1.3 Prediction

Predicting the near-future location of tracked dynamic targets is an important element of tracking, because it allows noisy observations to be smoothed by a model of possible target motion, and it provides additional constraints for data association that assist when periods of occlusion make observation impossible.

As part of the tracking by detection framework we maintain an estimate for the state of each tracked object. Before performing data association we predict the present state of the object x_k, given the estimate of the state at $k-1$.

$$x_{k|k-1} = f(x_{k-1})$$

Figure 2.4 – Object detection using a background model. Top row: input image sequence. Middle row: background images obtained via median filtering (image in n^{th} column is result of filtering input images 1 to n). After only four images, the foreground objects have been completely removed. Bottom row: Objects detected by comparing each input image to the final background image.

This function, f, is called the motion or process model and represents the dynamics of object motion. This prediction step is particularly important when the speed of the target is high with respect to the sampling rate of the sensor. Without prediction, a target could move a significant distance from its expected location and frame-to-frame data association would fail. However, especially in a multi-target tracking scenario, with limited information about the objects being tracked, the motion model is only an approximation and prediction increases the uncertainty in the state.

Aside from the benefit of improving frame-to-frame data association, prediction is also used to cope with occluded targets and missed observations. If an object is not observed for several frames, predicting the target location successfully can allow the target to be associated to the correct track when it reappears. A better motion model, which matches the object dynamics and has slower uncertainty growth, will enable tracking to continue through longer periods of no observations.

Specifying a motion model for pedestrians is a difficult problem, as humans can transition quickly from stationary to walking, jogging or running and change direction with ease (as compared to cyclists or vehicles) [114]. Knowledge of the environment and prior tracking data can be used to learn goal locations and predict a pedestrian's destination, leading to better motion models [16]. Alternatively, fluid-dynamic models have been used to simulate crowd behaviour [45].

The most relevant work in this field is by Luber et al. [64], which uses the social force model by Helbing et al. [40]. They demonstrate a significant reduction in data association errors as a result of the model, which assumes pedestrians have an intended velocity and direction and adapt their paths based on 'repulsive effects' from other pedestrians and the environment.

In this work we do not differentiate between tracking cars, pedestrians, cyclists, or other moving objects, and so we use a constant velocity model with built-in noise terms. This causes the uncertainty of unobserved tracks to grow quickly, but ensures that few tracks are lost due to unexpected dynamics.

The tracking methods in this thesis are therefore driven by observation. Occlusions, which prevent observations, cause ambiguous tracking situations to arise. Rather than rely on more tightly defined motion models, which are fragile in the presence of unexpected dynamics, the methods developed in this thesis allow for these ambiguous situations to occur, and use observed data to resolve target identities.

2.1.1.4 Data Association

The detection step yields a set of observations, which are matched with the appropriate tracks by the process of *data association*. Ideally, there is a one-to-one correspondence between tracks and observations – each tracked object is observed once, and there are no spurious observations. The data shown in Figure 2.1 is an example of this idealised case.

If the source of each observation (i.e. which target generated it) is known, then data association is trivial. For example, aircraft fitted with an automatic dependent

surveillance-broadcast (ADS-B) transponder [32] provide information about their positions (obtained via the global positioning system (GPS)) to air traffic controllers and other aircraft. Tracking systems using this information do not need to determine which position measurement came from which aircraft, as it is provided in the data.

In general though, observations are considered "anonymous", in that we don't know which observation came from which target and data association needs to be solved using a bipartite matching algorithm such as the "Hungarian" or "Kuhn-Munkres" algorithm [53].

This matching is often based on the assumption that each target generates exactly one observation, however in practice a number of situations arise that make data association more difficult. Targets may not generate observations due to occlusions, targets might generate multiple observations due to over-segmentation or clutter, or multiple observations might only generate one observation because of under-segmentation.

Many different data association strategies have been proposed in the literature, some of the most well known being the probabilistic data association filter (PDAF) (which updates tracks with the sum of multiple observations, weighted by their association probabilities) and multi-hypothesis tracker (MHT) (which maintains a tree of hypotheses over data associations). We examine the effect of data association on tracking performance in more detail in Chapter 3.

2.1.1.5 Track Management

Track management encompasses many of the practical implementation details required to turn a theoretical framework into a working system. Whilst a tracking system is "just" a set of independent filters, each maintaining the state of an object in the scene, engineering design decisions with respect to track management are critical to the robustness of the overall system. The two main management tasks that are not directly covered by data association and track state update are track creation and track deletion.

In an idealised description of a tracking system, a new track is created for every observation that does not get associated to an existing one. In practise though, false positive and fragmented observations occur, and a decision must be made whether to accept or reject an observation. A common approach is to require a minimum amount of "evidence" (measured as number of observations, or duration of continuous observation, for example) before marking a track as confirmed. Frameworks such as MHT use terms such as probability of detection and false target density [83] to calculate the likelihood of an object existing.

Once a track is confirmed, prediction can be used to estimate its state, even when the track receives no observations due to occlusion or sensor errors. After some time without observations, though, the tracked object can be assumed to have left the scene and the track should be deleted. This is generally achieved by monitoring the track uncertainty (as measured by the filter). Once the uncertainty grows above a certain threshold, the track is deleted.

2.1.2 Tracking by registration

Whilst tracking by detection finds objects of interest in each frame of a data stream then connects the observations together to form tracks, tracking by registration starts with a template object and searches for the most likely position and orientation of that object in each frame. This is most easily understood by taking an example from the visual effects (VFX) industry.

Consider a film-making scenario where a scene has been recorded with a moving camera. In the background of the shot there is a billboard showing some advertising that the director wants to remove. If the position of the four corners of the billboard in each frame were known, and no part of it is occluded by other scene elements, then a new image could be warped to fit and replace the billboard.

Manually labelling the corners of the billboard would be very labour intensive, so VFX software makes use of 2D "tracking by registration" algorithms such as the Kanade-Lucas-Tomasi (KLT) tracker [92, 105]. Typically, an operator will select a

key point (in this case a corner of the billboard) and the software will determine the location of this point in subsequent frames of the video. If the software 'looses track' of the point, the user can manually adjust it and tracking continues. In this case, tracking the corners of the billboard would be sufficient for the image replacement task, but if more points are tracked then the complete motion of the camera can be determined. This process is called match moving and is a common VFX technique.

Figure 2.5 shows a screen shot of the open source software "Blender" [102] being used to track the position of a robot in a video sequence. The orientation of the robot changed significantly during tracking, and although the KLT tracker copes with this to some extent, the template had to be manually reselected and tracking restarted several times.

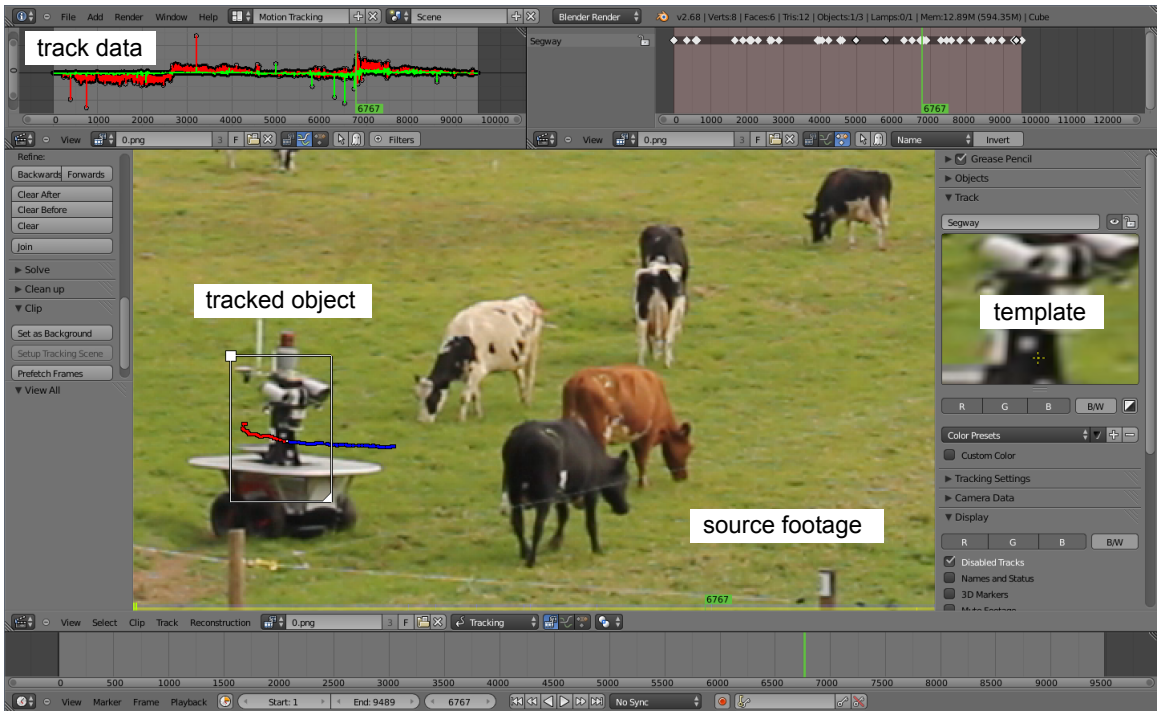


Figure 2.5 – "Blender" [102] is an example of VFX software that implements "tracking by registration".

Some tracking by registration approaches provide robustness against changes in object appearance by adapting the template over time, however these methods still require some way to specify the original data that forms the object template. In the cases

above, this was through human intervention, but could be automated through the use of a detector. This combination of detection and registration leads to hybrid trackers, as discussed in the following section.

2.1.3 Hybrid and alternative approaches

Whilst we describe "tracking by detection" and "tracking by registration" as the two broad classes of object trackers, there are frameworks that exploit elements of both detection and registration, or use feedback between the tracker and detector.

The tracking-learning-detection (TLD) framework by Kalal et al. [49] is an example of a hybrid approach that combines both detection and registration. Starting from an object marked by a bounding box in one frame of a video, the system uses tracking by registration to find the object in subsequent frames. At the same time it uses the tracking results to train an object detector. A feedback scheme uses the detector to ensure that the tracker does not drift, and uses the tracker to improve the accuracy of the detector and adapt it as the object and background appearance changes. Figure 2.6 shows a screenshot from a TLD demo example video.

Other methods use tracking-by-detection but more tightly couple the detection and data association steps to reduce false positives and improve occlusion handling. One such example is described as "people-tracking-by-detection and people-detection-by-tracking" by Andriluka et al. [2].

Another tracking approach that doesn't fit into the two broad categories as discussed above is the probability hypothesis densitiy (PHD) filter. Whilst the detection step of PHD is the same as in "tracking by detection", PHD-based trackers differ in the way they handle tracks and data association. Rather than maintaining independent filters for each tracked object, and explicitly assigning observations to tracks, PHD filters treat the collection of targets as a *set-valued state*, and the collection of observations as a *set-valued observation*. The PHD filter is an approximation to the more general random finite set (RFS) formulation [66], and propagates the first-order statistical moment (intensity) of the state. The integral of this intensity over any region gives

Figure 2.6 – Example of the TLD framework [49] being used to track a car from helicopter footage. Images on the right hand side of the frame are positive samples used to train the detector, whilst on the left hand side are the negative training examples. Image is from `http://www.youtube.com/watch?v=Smh-HwtDHI8`.

the expected number of targets. In general, the PHD filter has no closed-form solution, and so approximations such as the Gaussian mixture probability hypothesis density (GM-PHD) filter have been developed.

In estimating the density of targets over regions, these filters do not represent individual target state. In regions containing multiple targets, therefore, the identities of individual targets are not maintained and the trajectories of such targets cannot be determined [79]. To solve this problem, various extensions have been proposed [61, 79]. In the presence of interacting targets these methods are subject to the same kind of failure modes as tracking by detection approaches.

2.2 Target Identity

An important aspect of multi-target tracking is the "identity management problem" [38]. A target's identity is a somewhat abstract concept in that there is no one definitive label for an object – a person who is tracking objects in a scene may label

the objects as “Alice” and “Bob” or “the red car” and “the blue car”, whilst an automated system might use some internal representation such as “Object-1” and “Object-2”. The exact labels don’t matter though, the important part of identity management is just that each label corresponds to exactly one physical object in the scene.

The previous section described two broad approaches to object tracking. In tracking by registration, target identity is implicit in the tracking process. The system finds an object in one frame, then searches for that very same object in the next. As such, no explicit “identity management” is required.

Tracking by detection systems generally represent observations by features such as their 3D position and the data association process is controlled only by the track to observation distance. So long as objects remain separated from each other, then observations are uniquely associated to tracks, and target identity is preserved. However, when objects come close together, positional information might not be sufficient to determine the correct mapping from observation to track, and identities may become confused. An example of such an interaction, observed using a 3D lidar, is shown in Figure 2.7.

In this situation the lidar is providing what we term “anonymous” observations, the observations tell us where targets are, but not which object is which. The opposite to this is an “identified” observation that is associated with a specific target. Examples of identified observations include the ADS-B messages for aircraft tracking (mentioned previously) and radio-frequency identification (RFID) tags.

Most sensors provide data that falls somewhere in between the ‘anonymous’ and ‘identified’ extremes mentioned above. In the absence of per-target infrastructure, a remote-sensing system can not obtain a target’s identity directly. Instead, target appearance is used as a proxy for identity and appearance information can be used to determine which target is which.

Figure 2.8 shows the same interaction as Figure 2.7, but this time the data is taken from a colour camera. The appearance information in the images enables the viewer

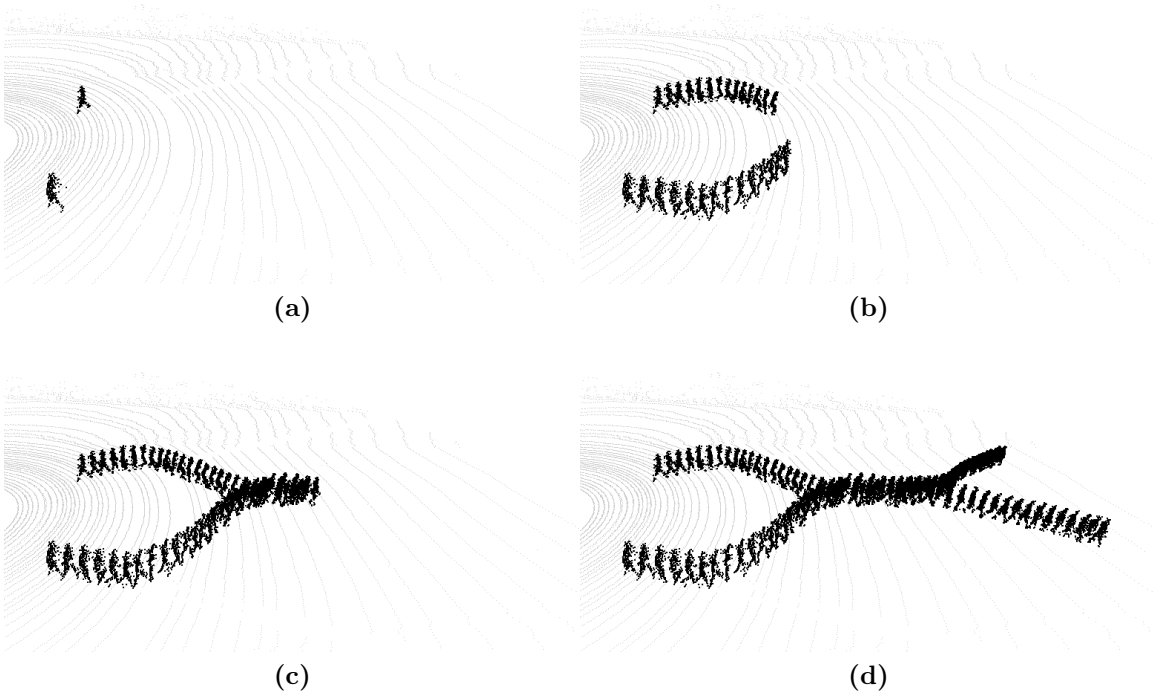


Figure 2.7 – An example of a situation that results in identity ambiguity. Two pedestrian targets are observed using 3D lidar data (a). Whilst the objects remain separated, tracking proceeds by associating observations to the appropriate tracks (b) (in this example only every 10^{th} observation is shown for clarity). When the objects move close together, the sensor is no longer able to resolve individual targets (c). When the targets separate, it is unclear which target moved in which direction (d).

to be quite certain that the person that started on the left of the image in Figure 2.8a is also on the left in Figure 2.8c.

Many sensors obtain some amount of appearance information and how useful this information is depends on the processing techniques employed, and on the differentiability of the appearance properties of the targets. In the following section we discuss sensors for tracking and give a brief overview of how appearance information can be extracted. A more detailed discussion of learning models to represent target appearance is presented in Chapter 4.

(a) (b) (c)

Figure 2.8 – The same interaction as Figure 2.7, seen with a colour camera. This time, there is no ambiguity about which target is which after the interaction.

2.3 Sensors for tracking in urban environments

In this section we describe the sensor technologies most commonly used for task of perception in urban environments, with a particular emphasis on their utility for different parts of the tracking systems that concern this thesis. We are primarily concerned with sensors mounted on a moving platform (such as a self-driving car, service robot, or surveillance platform) rather than an instrumented environment (for example, a CCTV network).

2.3.1 Radar

As described in the introduction to this chapter, radar was one of the first sensors to be integrated into an automated target tracking system. Radar systems operate at a wide range of frequencies, which affects sensing resolution and range. The most common type of radar for automotive applications is the millimetre wave radar, which is used for adaptive cruise control (ACC) [14]. The narrow FOV of millimetre wave radars mean that for autonomous driving applications, a number of units are used together to increase sensor coverage. For example, the Stanford entry in the DARPA Urban Challenge (DUC) used 5 millimetre wave radars whilst the MIT team used 15 [56, 67].

Whilst both teams successfully used radar to detect and track moving vehicles, radars suitable for automotive applications are used to determine position and velocity only

and do not provide good information about the size and shape of the target. Referring the radar systems used on the MIT DUC vehicle, Leonard et al. note that "unfortunately, the radars cannot easily distinguish between small, innocuous objects (like a bolt lying on the ground, or a sewer grate) and large objects (like cars)" [56]. Given this difficulty in telling the difference between objects of different shapes and sizes, determining the identities of individual pedestrians using automotive radar data would be a very difficult task.

2.3.2 2D Lidar

A 2D lidar such as the SICK LMS-291 (pictured in Figure 2.9) uses a single beam laser range finder and a rotating mirror to measure range in a planar "slice" of the environment. Its size, scan rate, accuracy and ease of use has made it a very common sensor for robotics applications. 2D lidar sensors have been used for pedestrian tracking since around 2001 [51, 62, 87], and they remain a popular sensor for tracking to this day. Recently, 2D lidar data has been used to track groups of pedestrians [3] and learn the socio-spatial relations between people [63].

(a) **(b)**

Figure 2.9 – (a) Sick LMS-291 lidar sensor. (b) Two 2D lidars mounted with scanning planes perpendicular to each other on the experimental platform "Shrimp".

The planar nature of the scan means that occlusions are common, and tracking works best in flat environments, where the scanner can remain parallel to the ground. Pedestrian observations from 2D lidar data are typically line segments, extracted from torsos or legs, and as such it is difficult to extract appearance information sufficient to determine target identities. In fact 2D lidar observations don't give much class information either – imaged at the right height, the corner of a car looks the same as the corner of a building, and the legs of a pedestrian may appear exactly the same as the trunk of a tree.

2.3.3 3D Lidar

The introduction of purpose-built 3D lidar sensors has had a large impact on the design of perception systems. Whilst 3D lidar data can be obtained by sweeping or 'nodding' a 2D scanner, the development of specialised sensors with video-like frame rates has made dynamic target tracking feasible in 3D environments. Additionally, the increase in data coverage and density reduces instances of occlusion and the increased number of measurements returned from a given target allows target size and 3D shape to be measured.

For robotics applications, the most well-known 3D lidar sensor is the Velodyne HDL-64E, pictured in Figure 2.10. After its success in the 2007 Defense Advanced Research Projects Agency (DARPA) Urban Challenge, the sensor has been widely adopted amongst the autonomous driving research community. Topics of interest include segmentation [28, 43, 69, 76], classification [29, 68, 101], simultaneous localisation and mapping (SLAM) [57] and tracking [70, 72].

The amount of geometric information captured in a 3D lidar point cloud means that not only can we determine objects classes (pedestrian, car, bicycle), but also recognise different instances of the same class. This is examined in more depth in Chapter 4.

(a) (b)

Figure 2.10 – (a) The Velodyne HDL-64E lidar sensor. (b) The sensor as mounted on the ACFR experimental platform "Shrimp".

2.3.4 Camera

Camera-based tracking has a wide variety of applications, ranging from industrial pick-and-place, to motion capture for the VFX industry and graphical overlays during sports broadcasting. Pedestrian tracking is common in the domain of surveillance, where off-the-shelf units can add automated tracking capabilities to CCTV systems [100].

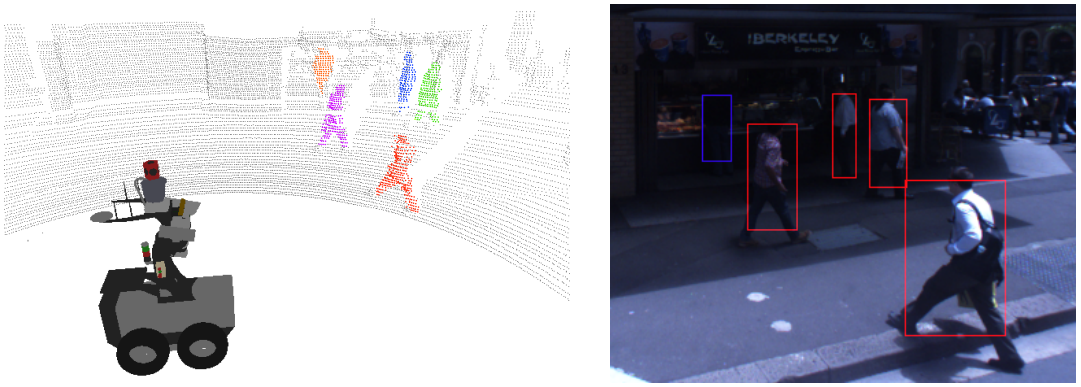
Whilst there are many successful commercial applications of camera-based tracking systems, multiple-target tracking in urban environments remains an active area of research. In the context of this thesis, the most relevant recent works consider the task of tracking interacting pedestrians in cluttered scenes [54, 118, 119].

Much of the work in camera based identity tracking focuses on fixed-infrastructure systems and less so on mobile platforms (although pedestrian tracking from a mobile platform has been demonstrated [31]). In the DARPA Urban Challenge, for example, the top four teams relied entirely on other sensors, such as radar and lidar, for obstacle detection and tracking [4, 25, 56, 67].

There are a number of reasons why cameras are not the preferred sensor for multi-

target tracking from ground vehicles. Firstly, accurate 3D position information is important. A collision avoidance system, for example, must not only be able to detect a pedestrian in the vicinity of the vehicle, but also know exactly where it is such that braking or evasive manoeuvres can be performed. Whilst depth information is possible by using stereo cameras, the depth error increases quadratically with the range to target [19], and changing illumination and amounts of texture make the stereo-matching problem difficult.

Illumination is a problem not just for depth estimation, but for object detection too. Figure 2.11 shows a real-world environment as seen by a 3D lidar (left) and camera (right). The 3D lidar can see five people in this part of the scene, but they are much harder to see in the image on the right due to the camera's limited dynamic range. Lidar can detect people equally well in complete darkness, where a camera would fail, and because it actively illuminates the environment, algorithms do not need to be adapted for different lighting conditions. Both sensors, however, are susceptible to environmental conditions such as rain, dust and snow.



(a) **(b)**

Figure 2.11 – (a) Pedestrians in an urban scene, extracted from 3D lidar data via change detection. (b) Bounding boxes of the same five lidar detections projected into the corresponding camera image. Due to insufficient dynamic range, the pedestrian marked in blue is almost impossible to see.

Cameras are, however, a very useful source of identity information. The visual appearance of objects are often sufficiently distinct that they can be readily identified. We choose, therefore, to use 3D lidar as our primary sensor for tracking, but to make

use of camera data for determining object identity. This topic is explored in detail in Chapters 4 and 5.

2.4 Evaluating tracking performance

So far in this chapter we have presented an overview of tracking approaches, described the concept of target identity, and introduced sensors appropriate to the task of multi-target tracking. In the chapters that follow we will be investigating these areas in greater detail and developing new methods for identity tracking. In order to objectively evaluate and compare tracking approaches we require performance metrics.

In this thesis we are primarily interested in supervised evaluation. Supervised evaluation involves comparison against a "ground truth", object trajectories obtained by manual labelling of data, by per-target instrumentation or other tracking infrastructure. The alternative, unsupervised evaluation, seeks to measure tracking performance in the absence of any ground truth, and statistics such as track length or duration are often considered.

In this section we describe the desirable properties of tracking evaluation metrics. We discuss why the current methods don't adequately measure identity-tracking performance, and present an additional method adapted from the clustering analysis literature. The methods presented in this section will be used throughout the rest of the thesis as we investigate the identity tracking problem.

2.4.1 Evaluation criteria

If a ground truth tracking solution is available, tracking performance can be measured by comparing the output of the system with the ground truth. Whilst the requirements of the comparison may at first seem easy to specify (high score when system output is the same as the ground truth, low score when they differ), many different metrics have been proposed to achieve this same task.

Perhaps the most prominent discussion on performance evaluation came out of the classification of events, activities and relationships (CLEAR) Evaluation Workshop [1]. In work that originated from this workshop, Bernardin and Stiefelhagen define the following two criteria for performance metrics [10].

1. They should allow to judge a tracker's precision in determining exact object locations.

2. They should reflect its ability to consistently track object configurations through time, that is, to correctly trace object trajectories, producing exactly one trajectory per object.

These criteria illustrate the difficulty in defining performance metrics alluded to above. Both specify what a system must do to achieve a perfect tracking score, but do not state how scores should degrade, or what the lowest possible score even means. Particularly with regard to (2), recognising whether a solution is perfect or imperfect is well defined, but quantifying how imperfect is not.

2.4.2 The CLEAR MOT metrics

In an effort to address the criteria listed above and standardise metrics for multi target tracking systems Bernardin and Stiefelhagen proposed the CLEAR multiple object tracking (MOT) metrics [10]. The performance evaluation has two components: multiple object tracking accuracy (MOTA) and multiple object tracking precision (MOTP).

MOTP measures the ability of a tracking system to determine the correct *location* of targets (independent of its estimate of how they are configured). It is defined as in Equation 2.1, where d_t^i is the distance between object i and its corresponding track, and c_t is the number of matches at time t.

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{2.1}$$

MOTA measures the ability of a tracking system to determine the correct *configuration* of targets, that is, which target is where in the scene. Maintaining target configuration over time implies correct determination of object trajectories. MOTA is defined in Equation 2.2 and is $1 - E_{tot}$ where E_{tot} is the error rate of the tracker. The error rate includes m (misses), fp (false positives) and mme (mismatches), and is averaged over the total number of objects present g. For further discussion of the measures see the original paper ([10]).

$$\text{MOTA} = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t} \tag{2.2}$$

Whilst the standardisation of tracking metrics is important for the comparison of tracking systems, some limitations in the CLEAR MOT metrics mean that authors using these metrics still apply custom solutions to evaluate their tracking methods.

A problem arises with the use of MOTA to measure identity switches. When using this metric, identity switches are penalised each time they occur. As a result, if a tracker switches the identity of two targets for a single frame, then corrects them on a subsequent frame, MOTA counts this as two mismatch errors per track; one for the switch into the erroneous state, and a second error for the switch back to the correct state. This corresponds to case (a) in Figure 2.12 and Table 2.1. If the identities were switched for a longer period of time before being corrected, as shown in case (b), then the score will be the same. Note that in case (b), the middle section (between $t = 6$ and $t = 11$) is a completely different track, but the number of identity switches is the same. Additionally, the bias of MOTA against changes in track identity mean that case (c) receives a *higher* score than (a).

2.4.3 GMOTA

The problem shown in parts (a), (b) and (c) of Figure 2.12, whereby a persistent error in identity scores more highly than a temporary one, has lead to the proposal of a MOTA variant, which computes a global identity score. The authors of [9] compute

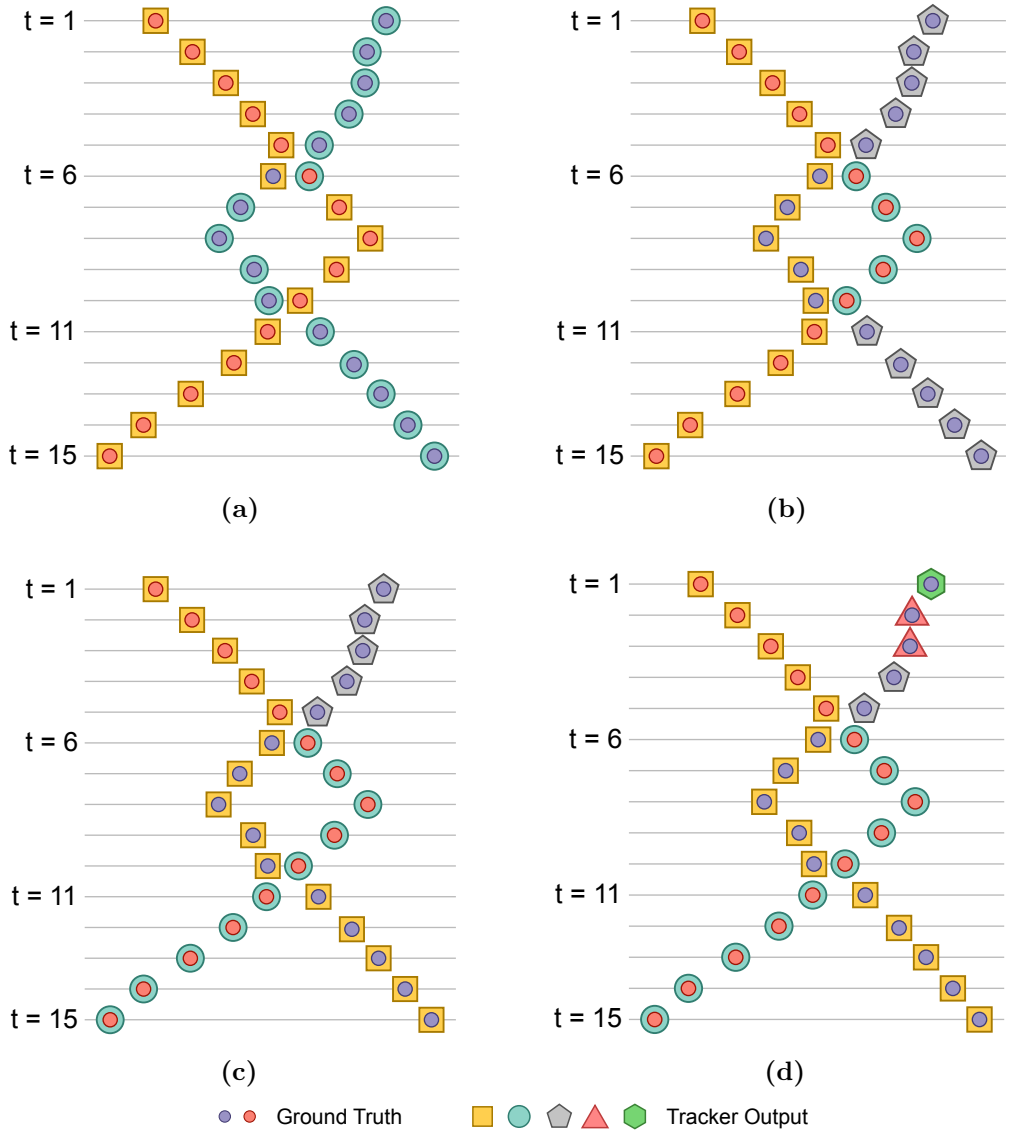


Figure 2.12 – Four different tracking solutions for the same synthetic data, showing varying numbers of identity errors. In (a) the identities are swapped at frame $t = 6$, and then corrected on the next. In (b) identity switches occur at $t = 6$ and are corrected at frame $t = 11$. In (c) errors occurs at $t = 6$ and finally in (d), one labelled object is fragmented across several tracks before stabilising at frame $t = 6$.

	a	**b**	**c**	**d**
MOTA	0.867	0.867	0.933	0.867
GMOTA	0.933	0.667	0.667	0.667
V-measure	0.647	0.440	0.440	0.399

Table 2.1 – Performance metrics for the four tracking solutions shown in Figure 2.12

gmme, or 'global mismatch errors', which accumulates the number of frames in which the identity is wrong after a mismatch error. Zervos et al. [120] also compute *gmme* and use it to replace *mme* in Equation 2.2. They refer to the resulting metric as global multiple object tracking accuracy (GMOTA).

GMOTA is also problematic, however, as the method scores the longest track-to-label match as correct, and treats all other assignments as errors. In case (d) of Figure 2.12, an object trajectory is fragmented across several tracks before stabilising. However, since this portion of the trajectory is not the longest match, the effect of this is ignored by GMOTA and the score for case (c) and (d) is the same.

2.4.4 V-measure

The ability of a tracking system to maintain the correct identity for each tracked object is its ability to group input observations into the correct trajectories. When considered in this way, a measure that captures identity-tracking performance has two requirements:

1. Each track contains observations from the one labelled object.
2. All observations from the one labelled object are assigned to the same track.

These requirements are very similar to those in the classification literature. Consider the following definitions of homogeneity and completeness, quoted from [84].

1. "A clustering result satisfies **homogeneity** if all of its clusters contain only data points which are members of a single class.
2. A clustering result satisfies **completeness** if all the data points that are members of a given class are elements of the same cluster."

If we consider the labelled object trajectories as classes and the tracks as clusters, then we can rewrite the above as:

1. A tracking result satisfies **homogeneity** if all of its tracks contain only observations from a single labelled object.

2. A tracking result satisfies **completeness** if all of the observations from a single labelled object are part of the same track.

These two terms, homogeneity and completeness, are the components of a clustering evaluation measure called V-measure [84], which have their roots in much earlier information theoretic measures such as mutual information [8]. According to the definitions above, V-measure is well suited to the evaluation of identity-tracking performance, so we first define V-measure as used in clustering evaluation, then explain how it is adapted for tracking evaluation.

V-measure is defined as the weighted harmonic mean of homogeneity (h) and completeness (c),

$$V = \frac{(1+\beta) \cdot h \cdot c}{(\beta \cdot h) + c} \tag{2.3}$$

where β is a parameter that controls the weighting of the two components. When $\beta = 1$, V-measure is equivalent to normalised mutual information (NMI) [7, 96]. The presentation of V-measure above, in terms of homogeneity and completeness, however, allows us to compute each component separately, which proves useful for understanding the behaviour of the identity tracking systems in this thesis. Homogeneity and completeness are defined as follows.

$$homogeneity = 1 - \frac{H(C|K)}{H(C)} \tag{2.4}$$

$$completeness = 1 - \frac{H(K|C)}{H(K)} \tag{2.5}$$

where

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|}\frac{n_{c,k}}{n}\log\left(\frac{n_{c,k}}{n_k}\right) \tag{2.6}$$

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\log\left(\frac{n_c}{n}\right) \tag{2.7}$$

and $H(K|C)$ and $H(K)$ are defined in a symmetric manner

$$H(K|C) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\log\left(\frac{n_{c,k}}{n_c}\right) \tag{2.8}$$

$$H(K) = -\sum_{k=1}^{|K|}\frac{n_k}{n}\log\left(\frac{n_k}{n}\right). \tag{2.9}$$

In the definitions above, $n, n_c, n_k, n_{c,k}$ are all computed from the clustering contingency table. n is the total number of samples, n_c and n_k are the number of samples belonging to class c and cluster k, respectively, and $n_{c,k}$ is the number of samples from class c assigned to cluster k.

To adapt V-measure for the analysis of identity-tracking performance we adjust the definition of the contingency table. In this work, $n_{c,k}$ is the number of frames in which labelled object c matches to track k. As for the CLEAR MOT metrics, we use the Hungarian (or Kuhn-Munkres) algorithm [53, 74] to compute the track to labelled-object mapping for each frame, bounded by a threshold distance.

V-measure does not include the concept of "false positives" (clustered points that weren't in the labelled set) or "false negatives" (labelled points not assigned to any cluster). This means that, using V-measure, we evaluate only the ability of a tracking system to maintain target identity. Missed tracks and false positives must be measured using an alternate metric.

2.4.5 Comparison of performance measures

In the preceding sections we discussed a number of tracking performance measures. Limitations in the ability of MOTA and GMOTA to reflect identity tracking performance (a key element of this thesis) led us to propose the use of V-measure for this task.

The examples shown in Figure 2.12 demonstrate some situations in which V-measure provides a more consistent measure of a system's ability to maintain target identities. The values computed by MOTA, GMOTA and V-measure (presented in Table 2.1) provide an ordering for the different tracker outputs, with the highest scores representing the "best" performance according to that measure.

From Figure 2.12, MOTA scores track output (c) highest, whilst GMOTA chooses (a). V-measure, too, scores (a) highest, but additionally provides a more intuitive ordering of (b), (c) and (d). The general properties of the three performance measures can be summarised as follows.

MOTA

MOTA favours temporal consistency, by punishing trackers every time they switch a track's identity. It is better for an identity to continue being wrong after a switch then to correct the mistake.

GMOTA

GMOTA favours overall consistency, by rewarding the longest match between tracks and labelled objects. Parts of tracks that are not in the longest match are punished equally regardless of their consistency.

V-measure

Similar to GMOTA, V-measure will emphasise the agreement between the largest sections of matching tracks and labels, but unlike GMOTA, V-measure will also assess the consistency of the smaller fragment. V-measure evaluates *homogeneity* and *consistency* as described above.

V-measure, as shown in this section, encapsulates the desired properties of an identity-tracking performance measure. It has empirical advantages over MOTA and GMOTA as shown in this section and has a stronger theoretical basis in information theory, which has been applied more diversely for general set assignment problems.

Throughout this thesis we will primarily use V-measure to evaluate identity tracking performance. We will also make use of its component homogeneity and completeness scores to expose greater detail about the behaviour of tracking systems.

However, since V-measure does not consider missed observations and false positives, or measure accuracy in target positions, we must use additional metrics to understand tracking performance in these regards. The MOTP score is a suitable way to record positional accuracy, and since MOTA does include false positives and misses we also report the MOTA score.

2.5 Summary

In this chapter we introduced the background material relevant to the work in this thesis. We considered the problem from a high level and discussed the two main frameworks for multi-target tracking.

We introduced the concept of target identity and the "identity management problem", and how this is relevant to the rest of the work in this thesis. We described the sensor technologies applicable to the task of multi-target tracking in urban scenes and motivated the selection of 3D lidar as our primary tracking modality.

The evaluation of tracking performance can be separated into two components, the ability of a system to determine the *locations* of targets, and its ability to maintain their *identities*. In Section 2.4 we described ways to measure these components. We presented the most common approach from the literature, CLEAR MOT, and demonstrated that MOTA does not adequately capture the ability of a tracking system to maintain object identity. This deficiency has been previously recognised in the literature and an adaptation, GMOTA, proposed. We demonstrated that GMOTA, though

an improvement on MOTA, still has important limitations that decrease its utility.

In this chapter, we proposed the adoption of V-measure to evaluate identity-tracking performance. Whilst it encapsulates the desired properties of an identity-tracking performance measure it does not evaluate the positional accuracy of a system, false positives or false negatives. Therefore, in this thesis we will analyse tracking performance in terms of the MOTP, MOTA and V-measure scores.

Chapter 3

Tracking interacting objects with 3D lidar

This chapter examines methods for multi-target tracking in urban environments. In such environments, scene complexity results in occlusions and ambiguous target interactions and poses significant challenges for a tracking system.

3D lidar sensors are becoming increasingly popular for perception in urban scenes. The FOV, density and accuracy of their data make them fundamentally different to other sensors. Since its introduction in 2006, the Velodyne HDL-64E in particular has become very popular with autonomous driving researchers, being used by five of the six finishing teams in the 2007 DARPA urban challenge [111] as well as for Google's self-driving car project [39].

The recency of the sensor means that tracking methods designed specifically for such data are in their relative infancy (as compared to 2D lidar and radar). The methods used in the 2007 DARPA Urban Challenge generally projected the 3D point cloud into a 2D representation [24, 67] – for example, the Stanford entry re-sampled Velodyne data into 2D "virtual scans" in order to perform segmentation and tracking [80]. Most of the teams relied on the assumption that all moving objects encountered during the challenge would be vehicles, and could therefore be represented by vehicle-specific models. Only more recently have algorithms that can segment the full 3D point cloud

from a Velodyne sensor [28, 43, 69] been proposed and methods designed specifically to track objects in 3D developed [70, 94].

In this chapter we focus on the task of tracking dynamic objects in urban scenes. In particular we are interested in the case where objects are in close proximity to each other, forming groups, and crossing paths. In Section 3.2 we compare a number of techniques for tracking object in segmented point clouds, propose a method for data association that reasons about the possibility of occlusion, and in Section 3.3 we demonstrate the utility of group tracking and higher-level logic to form the most complete understanding of scene dynamics. The methods developed in this chapter are combined with the appearance modelling techniques described in Chapter 4 to provide robust identity tracking in complex urban scenes, in Chapter 5.

3.1 3D lidar and pre-processing

In this section, the fundamentals of lidar technology are described along with the pre-processing steps required to work with lidar data in higher level systems. We introduce the Velodyne HDL-64E, the primary sensor used in this chapter, and describe the coordinate transformations and object detection methods that prepare the data for input to our tracking system.

At the heart of any 3D lidar sensor is one (or more) laser range finder. By emitting a pulse of light and measuring how long it takes for the pulse to be reflected off a target and return to the sensor it is possible to determine how far away an object is. Single beam range-to-target measurements are useful in fields such as construction, surveying, and the military.

For robotics applications, 2D and 3D lidar sensors are more common, because in this form, lidar data can be used to build metric maps of the environment, enabling high level tasks such as obstacle avoidance, tracking and classification. A 2D lidar sensor can be created by rotating a single beam laser range finder around one axis or, more commonly, reflecting the laser off a rotating mirror. Taking multiple range measure-

ments as the beam sweeps across the scene allows a 2D slice of the environment to be reconstructed.

To make the step to 3D, various techniques may be employed. For instance a "nodding" sensor takes a 2D scanning lidar and rotates it around a secondary axis. This axis may be driven by a motor (as in [104]), harmonically oscillated (in scanners such as the Nippon Signal FX8 [44]), or allowed to wobble freely (the "Zebedee" [15]). In this work we utilise the Velodyne HDL-64E, which captures scenes by rotating 64 diverging laser range finders around one axis, rather than moving one range finder around multiple axes.

Whilst some of the pre-processing techniques discussed below are specific to the Velodyne, the general concepts of coordination transforms, ground detection, segmentation and change detection can be applied to other 3D sensors.

3.1.1 Velodyne lidar

The Velodyne HDL-64E comprises 64 lasers rangefinders, arranged so as to sweep out concentric rings from approximately -24° (towards ground) to +2° from horizontal. The entire unit, shown in Figure 3.1, spins around the vertical axis and each of the 64 lasers fires at approximately 21 kHz, resulting in over 1.3 million range samples per second. When spinning at 20 Hz, as used for our datasets, this equates to an resolution in azimuth of less than 0.2°. Sampling in azimuth is controlled by laser firing time and therefore does not necessarily happen at the same angle each revolution.

The data is measured as a continuous stream, and transmitted in groups of 100 'blocks' – where each block contains 32 returns from either the upper or lower set of lasers. For convenience during segmentation and tracking, the point stream is divided into "scans", where each scan represents one 360° revolution of the Velodyne sensor.

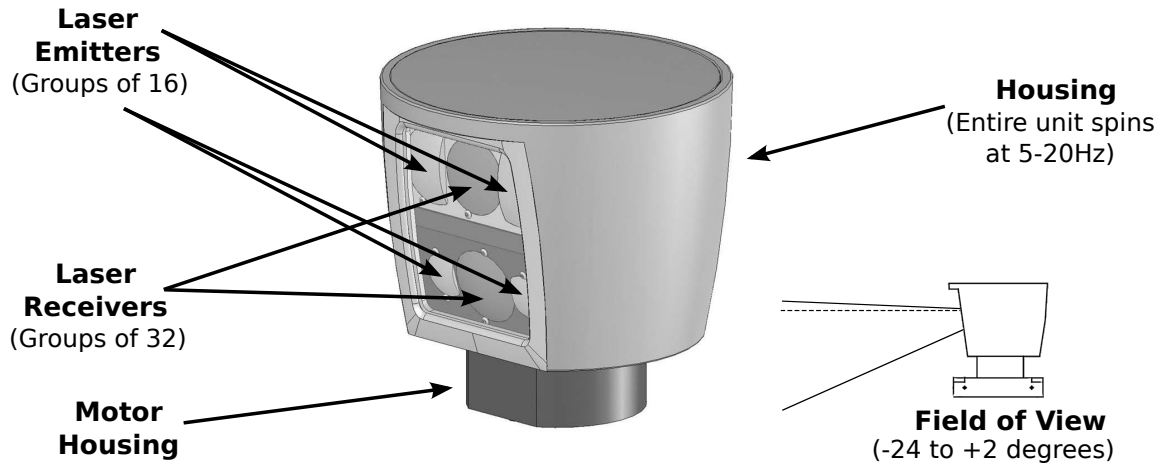


Figure 3.1 – Velodyne lidar (diagram adapted from [112])

3.1.2 Coordinate Transforms

Each data point returned from the Velodyne is a line-of-sight distance from one of the laser emitters to the nearest object in its path. To convert data from these $(range, bearing, elevation)$ measurements into Cartesian coordinates, a number of transforms are required. These transforms are listed below and shown in Figure 3.2.

1. From raw sensor returns into 3D Cartesian points local to the sensor (intrinsic calibration).
2. From the sensor coordinate frame into the vehicle body frame (extrinsic calibration).
3. From the vehicle body frame into global coordinates (via the navigation solution).

Intrinsic Calibration

The Velodyne HDL-64E contains 64 independent lasers and to generate a metrically accurate point cloud, measurements in $(range, bearing, elevation)$ are converted to

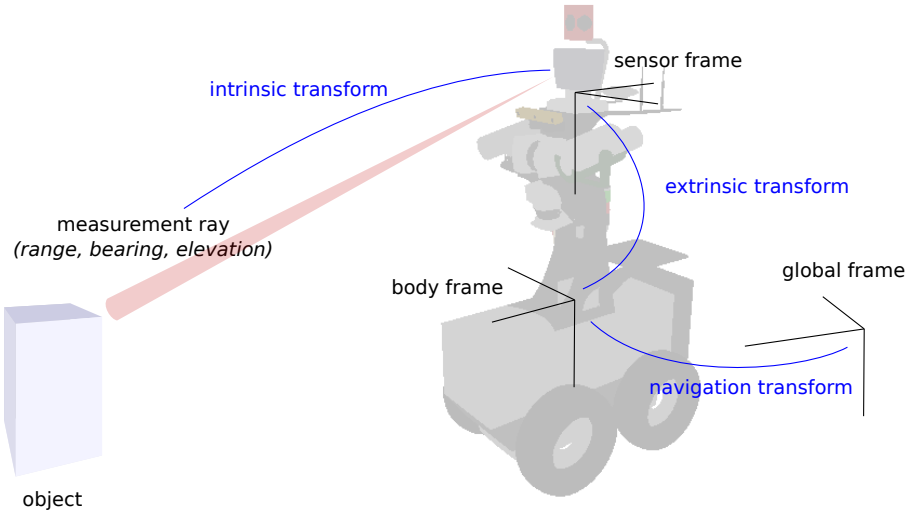


Figure 3.2 – Transforming from raw sensor data to a point-cloud in global coordinates. Measurements from each laser are represented as $(range, bearing, elevation)$. These measurements are converted to the Cartesian sensor frame via an *intrinsic* transform, to the body frame via an *extrinsic* transform, and finally into the global frame via the navigation solution.

(x, y, z) coordinates in the sensor frame. There are a total of five parameters to be optimised for each beam, comprising corrections for each of range, bearing and azimuth, as well as the horizontal and vertical positions of each laser within the housing. Whilst the Velodyne is supplied with a factory calibration, methods that achieve superior accuracy have been developed [59]. The Velodyne used for these experiments was calibrated by an in-house method that optimises sensor pose and the intrinsic parameters jointly by minimising a variant of the iterative closest point (ICP) residual. This method is similar to [59], and the differences are not significant in the context of this thesis. Both methods achieve considerable improvements over the factory calibration.

Extrinsic Calibration

The point cloud is then projected from sensor coordinates into the vehicle's body frame. The transform is obtained using the method described in [108], which can jointly calibrate multiple lidar sources using a navigation solution. An alternative

method for extrinsic calibration, that does not require manual selection of alignment features is presented in [59].

Global projection

To project the points from the body frame into the global frame, the navigation solution (position and orientation of the vehicle body frame with respect to some globally fixed coordinate frame) is used. This navigation solution is usually obtained through the use of a GPS/inertial navigation system (INS), however in city environments, issues such as multi-path [36] may cause jumps in the position estimate. These jumps or "rifts" in the navigation solution can be problematic for tracking, especially if they are large enough such that the tracks obtained up to scan $k-1$ can not be reliably matched to observations at scan k.

If accurate position information is required at all times, then one solution is to use lidar data to build reflectivity maps [58]. Data collected by a survey vehicle is processed offline to generate a globally consistent map, which can then be localised against in real time. Another approach, used by the Stanford and MIT teams in the DARPA Urban Challenge, is to maintain two linked coordinate systems [56, 67]. The "smooth" coordinate frame is formed by integrating velocity estimates from the INS, and the offset between this frame and the global solution (which may contain jumps) is maintained. Whilst the smooth coordinate frame will drift over time, its local consistency makes it better for tracking and other local perception tasks. If data is required in the global frame, then it can be obtained by adding the global offset.

For the purposes of object tracking we require only this smooth coordinate frame. So long as the scan-to-scan navigation estimate is sufficiently accurate to allow reliable data association, then it is not necessary to project the data into a metrically accurate globally registered coordinate system and indeed if this was required then it should be performed at a later stage in the pipeline after data associations have been made.

We obtain this locally smooth solution using an open loop ICP [11, 20] method, where the change in sensor position is found by aligning new scans to previous ones. The

point-to-plane ICP variant [20] is applied, and to limit the drift of the solution, batches of scans are aligned to 'reference' scans chosen at approximately every 10 metres. The resulting transforms are in sensor space, but they can trivially be converted to the body frame, and despite the open-loop nature of the approach we found that the drift rate was low enough to provide a reasonably accurate global solution over the distances considered in our experiments. An example of an ICP-based navigation solution on an urban dataset captured near the Sydney Opera House is shown in Figure 3.3.

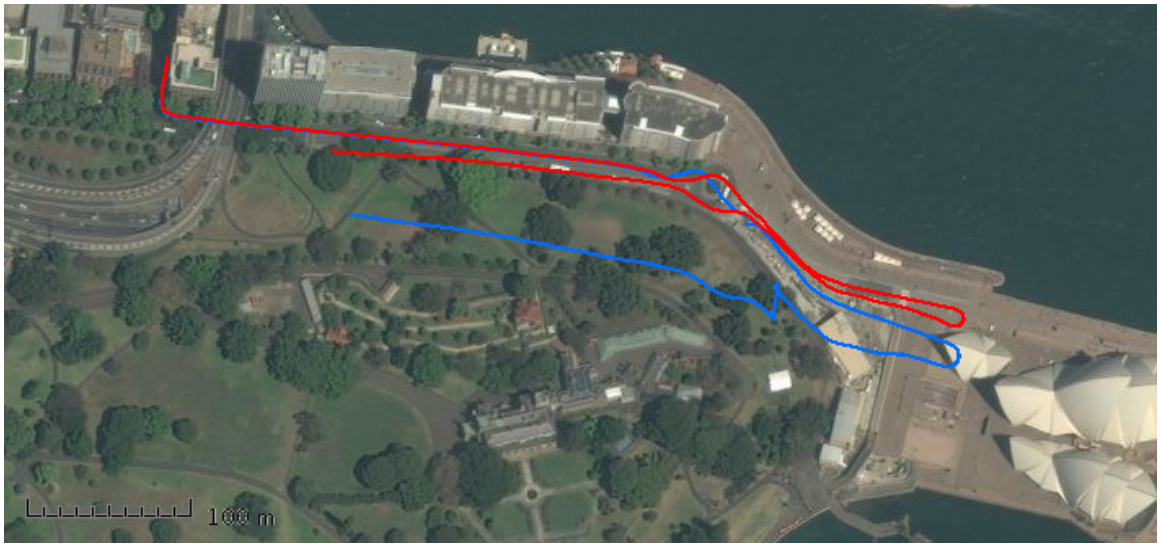


Figure 3.3 – Navigation solution from the GPS/INS (blue) and open loop ICP (red) for data collected near the Sydney Opera House, superimposed on aerial imagery (aerial data © Microsoft Corporation).

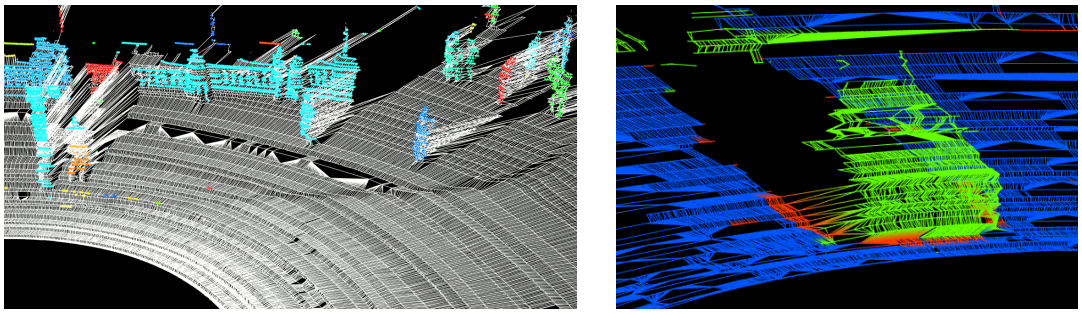
3.1.3 Ground Detection and Segmentation

In Section 2.1.1 we described generically the process of extracting *observations* from a continuous stream of data. Class-specific methods have been developed to detect pedestrians in Velodyne data (for instance [93] which uses an SVM based detector), however it is more common to use a background-subtraction approach to identify separate objects for tracking, without relying on particular object models [43, 76]. Velodyne-based approaches commonly operate on a per-scan basis and partition each

scan into ground and non-ground points, before clustering the non-ground points into separate observations [28, 43, 69, 76]. In this work we apply the "mesh-based" method for ground detection and segment the non-ground points into observations by the "cluster-all" approach (both from [28]).

The "mesh-based" ground detection method does not rely on an assumption of flat ground, nor does it first create an elevation map (which can be confused by overhangs). Rather, it expects the ground to be smoothly varying and uses a mesh data structure designed to suit the scan pattern of the Velodyne lidar. The steps of the algorithm are as follows, but for a more complete explanation, see [28].

1. A mesh is constructed by linking points according to the scan pattern of the rotating Velodyne sensor (Figure 3.4a).
2. A gradient field is computed over all the points, based on their links to neighbours in the mesh.
3. Working from the inner-most scan ring outwards, the ground label is propagated to connected points where the gradient is below a threshold value. Transition zones are also determined by analysing the statistics of the gradient over a sliding window (Figure 3.4b).



(a) **(b)**

Figure 3.4 – Mesh-voxel segmentation. (a) The mesh generated by linking points according to the Velodyne scan pattern. (b) Ground detection result, showing ground points (blue), object points (green), and transition zones (red).

After the ground points have been detected, the remaining points are then clustered according to the 'cluster-all' approach. Non-ground points are placed into a voxel grid of a fixed resolution, and connected voxels are considered to be part of the same cluster. An example of the output of clustering was shown previously in Figure 2.2.

A problem with this clustering approach is that it assumes gaps in the point cloud always indicate boundaries between objects. Consider Figure 3.5, which shows a top down view of two trees in front of a wall, with lidar measurements shown in red. Segmentation has resulted in 4 observations, marked A to D. In 3D space, there is a separation between observations A and C, and the clustering algorithm has correctly separated the two trees. However the presence of the trees means that the section of wall between clusters B and D is not observed. The clustering algorithm makes no distinction between unobserved space and free space, and as there are no lidar measurements connecting B and D it treats them separate objects. The assumption that a gap in lidar data indicates a new object works well for discrete objects such as people, cars and bicycles, but breaks continuous surfaces such as fences or walls at occlusion boundaries.

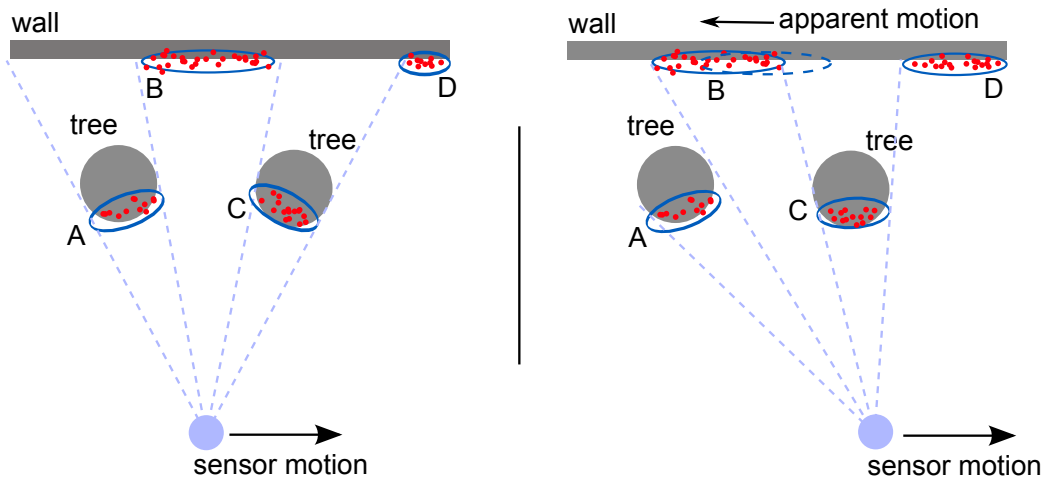


Figure 3.5 – Top down view of a scene containing two trees and a wall, as seen by a lidar sensor on a moving platform. Occlusions cause the wall to be split into two segments (B and D), and as the sensor moves, changing occlusion boundaries make it appear that segment B is also moving.

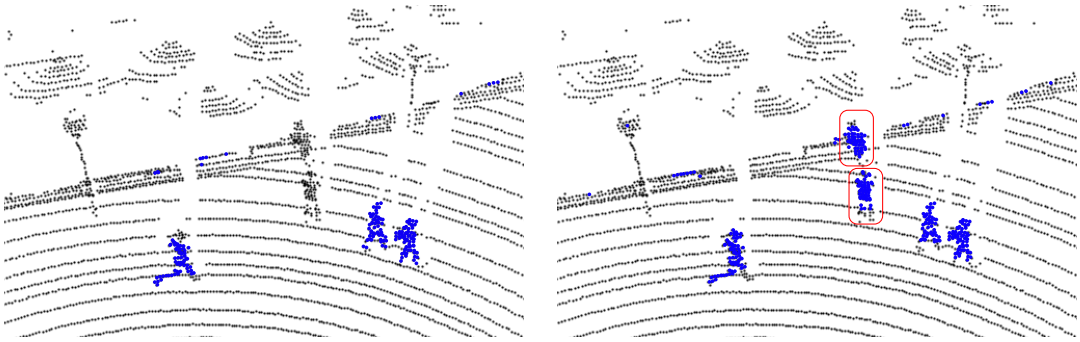
A secondary problem caused by this clustering is "apparent motion", which is also demonstrated in Figure 3.5. The sensor moves from left to right between the two snapshots, measuring the same scene from a slightly different angle. The foreground objects are observed to be in approximately the same location, but the changing occlusion boundaries mean that the section of wall between the two trees appears to have moved.

3.1.4 Change Detection

Change detection is an alternative way to extract observations from point cloud data, and solves the apparent motion problem by explicitly detecting which objects in a scene are moving *before* tracking. Change detection has different strengths and weaknesses compared to the ground detection and segmentation approach described in the previous section, and both are used in this thesis as a component in the overall system.

In this work we use the approach described in [109]. This approach is described as 'explicit' change detection in that it distinguishes between parts of the scene that have been observed to change, and parts that only appear to have changed due to exploration frontiers or occlusions. The method uses ray tracing in spherical coordinates and the computational requirements are linear in the number of points processed, which makes it appropriate for tracking because it can handle data from the Velodyne in real-time.

A key parameter of any change detection algorithm is the time interval between the compared scans. For example, an interval of 0.5 seconds can be used to detect moving objects, whilst an interval of 1 year could detect changes such as the construction of a new building. For the purposes of tracking, we are interested in moving objects so a fixed interval of, say, 0.5 seconds is a sensible first approach. However the moment an object stops moving, it will stop being detected and the track will be lost. Cars stopped momentarily at traffic lights, or pedestrians at intersections will not register as changes.



(a) (b)

Figure 3.6 – Comparison of change detection strategies: (a) A fixed interval of 1.0 seconds. (b) An adaptively sampled set of reference scans with $\Delta t = 1.0s$ and $\Delta x = 3.0m$. Notice the pedestrians who have stopped to take a photo are detected in (b) but not in (a).

There are two conflicting requirements here. Increasing the time interval, Δt, means that objects that are stationary for less than Δt seconds can still be detected, but for a moving sensor vehicle, the increased time interval means that the vehicle will have moved further and the overlap between the two scans is diminished. This will result in fewer changes being detected. The strategy that will maximise the number of changes detected in the environment is to use *all possible values of* Δt, or effectively compare every scan to every other scan that has an overlapping FOV. This, however, is computationally intractable for large datasets.

We developed, therefore, a strategy that balances the number of changes detected with computational requirements. From a dataset, we sample a set of reference scans along the sensor vehicle's trajectory such that each scan is sampled no further than Δx metres from or more than Δt seconds since the previous. During detection, a scan is compared against all overlapping scans in the reference set. This method allows us to choose a sampling time interval that detects the changes we are interested in, whilst ensuring a good spatial coverage even when the sensor vehicle is moving quickly. A comparison between this approach and the fixed-interval sampling mentioned earlier is shown in Figure 3.6. When using the fixed-interval sampling, two pedestrians who have stopped to take a photo are not detected.

3.1.5 The pre-processing pipeline

The pre-processing steps described above form modules in a data pipeline, shown in Figure 3.7, that starts with raw data packets from the Velodyne sensor and delivers a globally registered, segmented, 3D point cloud, which serves as an input to the tracking system.

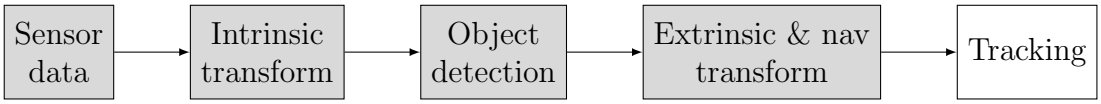


Figure 3.7 – The lidar pre-processing pipeline. Modules may be rearranged depending on the exact implementation used for a component – for example, the extrinsic and navigation transforms might need to be applied before object detection.

This figure represents the main flow of data in a lidar-based perception system, and the modular structure reflects the way the on-board software is separated into components. The "ground detection and segmentation" and "change detection" methods discussed in Section 3.1.3 and Section 3.1.4 are two possible implementations for the "object detection" module, and modules may be re-ordered depending on implementation details (for example object detection might occur after the point cloud is transformed into global coordinates).

The output of this pre-processing pipeline is a stream of Velodyne scans, where each scan has been separated into background points and observations. Each observation comprises a subset of 3D points from the lidar scan, registered in a locally-smooth coordinate frame. In the remainder of this chapter we describe and evaluate tracking systems that build upon this pipeline.

3.2 3D lidar tracking approaches

Tracking approaches for 3D lidar data have historically been based upon techniques developed for 2D lidar which, in turn, were inspired by radar-based techniques from as early as the 1970s. In 1971, Sea described an algorithm that tracks multiple targets

using Kalman filters, and performs nearest-neighbour data association to link observations to tracks [90]. In the intervening years many variants have been designed to better address various aspects of the multi-target tracking problem. For instance the PDAF [5] and joint probabilistic data association filter (JPDAF) [33] were developed to better represent the uncertain relationship between observations and targets in cluttered environments by using data assignment probabilities to weight the various measurements, whilst the MHT [83] creates a tree of data-association solutions that it prunes based on the assignment probability. Sample-based approaches such as the sample-based joint probabilistic data association filter (SJPDAF) have also been proposed [88]. The PHD filter, discussed in Section 2.1.3 has also been applied to lidar data [50].

Amongst the first algorithms for multi-target tracking using a vehicle mounted 3D lidar were those used by teams in the 2007 DUC. These approaches can be understood in terms of the "tracking-by-detection" framework outlined in 2.1.1. Incoming data is separated into observations, which are associated with tracks. Tracks matched with observations are updated, new tracks are created for unmatched observations, and unobserved tracks are propagated until their uncertainty grows too large and they are eventually deleted.

A key change in the transition from radar to 3D lidar data is the increase in the amount of information available about an individual target. Whilst radar systems are usually designed to generate a single return per target, 3D lidar observations may consist of tens or even hundreds of points. Teams in the DUC made use of this information to some extent by modelling cars as rectangles [67, 110]. The Carnegie Mellon team, for instance, detected corners in observations and used these as keypoints to improve estimation of target heading and tracking accuracy. Observations that were not shaped like a car, or were far enough away from the robot that they comprised only a few points were represented by their centroid [110].

"Tracking by detection" methods that are not tied to a specific object model usually require objects to be represented by a keypoint and hence they discard potentially useful information from the 3D point cloud. An alternative approach, designed

specifically for the Velodyne lidar and which makes use of a full 3D object model was proposed in [70]. The technique uses ICP to align tracked objects with Velodyne scans resulting in a 6 degrees of freedom (DOF) trajectory estimate. This method is, however, best suited to tracking rigid objects such as cars, and we have found in prior work [71] that the open-loop nature of the ICP estimate leads to drift in the tracked coordinate frame. For pedestrian tracking, we found that the tracking performance from the algorithm did not justify its increased computational cost [71].

Segmentation errors are a major cause of tracking failure [71, 91]. Particularly in urban scenes with high pedestrian density, targets are often in close proximity to one another, making segmentation based on lidar data unreliable. In this section we focus on tracking performance in these situations and develop methods with improved robustness against segmentation errors.

3.2.1 Methods

In order to evaluate the effects of occlusion and segmentation failure on the tracking of interacting, moving objects, we compare three different tracking strategies. The techniques, which we will refer to as **centroids**, **split** and **split-cost**, are all designed to track arbitrary objects using 3D lidar data. They do not differentiate between class of objects such as pedestrians, bicycles and cars by using specific motion models or target representations. The first, **centroids**, is a standard multi-target tracking approach, whilst **split** was introduced by Shackleton et al. [91] to solve tracking problems caused by under-segmentation. We describe these two approaches, and propose our own method, **split-cost**. These methods are then evaluated in Section 3.2.4.

3.2.1.1 Centroids

The first tracking method, **centroids**, is a standard multi-target tracking approach. The method is the same as that shown Figure 2.1 with the data association step detailed here. Observations are represented by their *centroid*, the mean of their 3D points as measured in the global coordinate frame.

To perform data association a distance matrix is calculated, where the distance between a predicted track state ($\hat{x}$) and an observation (Z) is measured by the Mahalanobis distance (Equation 3.1). S represents the innovation covariance and H is the observation model.

$$D_M(\hat{x}, Z) = \sqrt{(H\hat{x} - Z)^\intercal S^{-1} (H\hat{x} - Z)} \tag{3.1}$$

The innovation covariance is computed from the state covariance (P) by Equation 3.2, where R is the sensor measurement noise.

$$S = HPH^\intercal + R \tag{3.2}$$

From this distance matrix, the "Hungarian" (or "Kuhn-Munkres") algorithm [53] is used to determine the track-to-observation assignments (Ω) that minimise the sum of the assignment distances, or equivalently maximise the observation to track assignment probability.

Whilst observations are represented by their centroids for the purpose of filtering, each observation is made up of a subset of the 3D point cloud. To improve data association accuracy, it is useful, therefore to have a model that takes into account the shape of the observation. For rigid objects, box models [110] (discussed previously) or generic shape models [89] have been proposed, and pedestrian tracking in 2D lidar has been achieved using contour models combined with the Hausdorff distance [63].

For tracking generic objects in 3D lidar data, shape is far less stable, and so rather than explicitly representing the shape of an observation we instead measure the spread of the observed points by a covariance matrix C_Z. Given that a track is made up of observations, we can also estimate the shape of the object being tracked in the same manner ($C_{\hat{x}}$). Whilst the underlying shape of an object is unlikely to be representable as a Gaussian, the compounding of centroid estimation errors due to changing viewpoints, occlusions and sensor noise, make the Gaussian distribution a useful approximation that incurs minimal computational overhead. The adjusted equation for the

innovation covariance is shown in Equation 3.3. Figure 3.8 shows an example of how including the point covariance terms (C_Z and $C_{\hat{x}}$) improves the track-to-observation distance measure.

$$S = HPH^{\intercal} + C_Z + C_{\hat{x}} \tag{3.3}$$

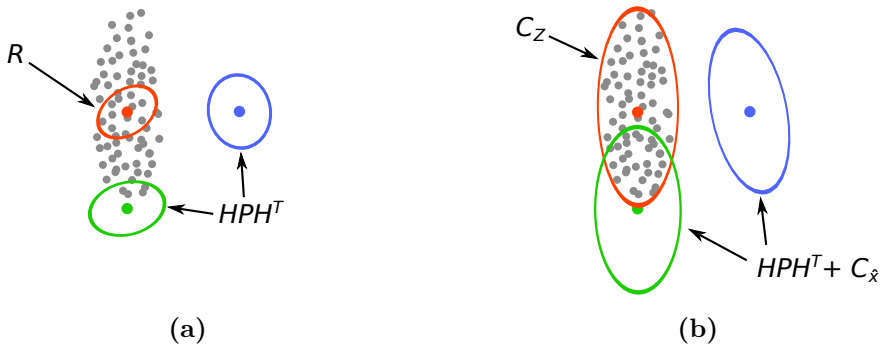
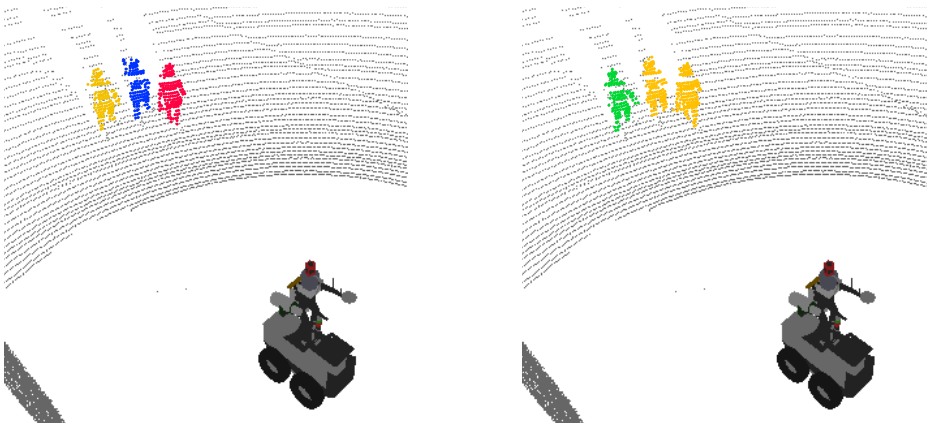


Figure 3.8 – Computing track to observation distance during data association. (a) An observation, shown in grey dots, is represented by its centroid and has uncertainty given by the sensor noise, R (red). Two tracks (green and blue), with state uncertainty P, are equidistant from the observation. (b) If we take into account the extent of the observation (C_Z), as well as the estimated shape of each tracked object ($C_{\hat{x}}$) and apply Equation 3.3, then the observation will be correctly associated with the green track.

3.2.1.2 Split

The second method evaluated, which we will refer to as **split**, was proposed in [91] and attempts to solve problems caused by under-segmentation. The **centroids** method above assumes that there is a one-to-one correspondence between tracks and observations, however due to the nature of the segmentation methods that are popular in the lidar tracking literature, objects that come close together are likely to be grouped together into the same observation. An example of such under-segmentation is shown Figure 3.9.

Systematic under-segmentation caused by objects remaining close together will eventually cause tracking of one or both of the interacting objects to fail. Shackleton



(a) (b)

Figure 3.9 – Example of (a) correct segmentation and (b) under-segmentation of three pedestrians walking side-by-side. Unique colours represent different segments.

et al. solve this problem through the use of a re-clustering step on observations that could be associated with multiple tracks [91]. During the track-to-observation matching process, multiple tracks are allowed to match to the one observation. When this occurs, K-means [65] is is used to split the observation. K-means is initialised using the predicted track states and after convergence tracks are updated using the new clusters.

A scenario in which this method is successful is shown in Figure 3.10a. Under-segmentation causes the lidar points (shown in black) to be clustered into one observation (grey). During data association, both tracks are matched to this observation and re-clustering is performed using K-means, resulting in new observations (shown in red and blue). These observations are used to update the corresponding tracks, and tracking continues successfully.

A disadvantage of **split** is that it will always re-cluster the observation into as many parts as the number of tracks that are associated with it. This makes sense so long as the observation includes points from each object being tracked, however under-segmentation also occurs when objects are occluded. If a tracked object is occluded, the observation contains points from only one object and the **split** method will divide these points between the two tracks. The two tracks will then "lock on" to the

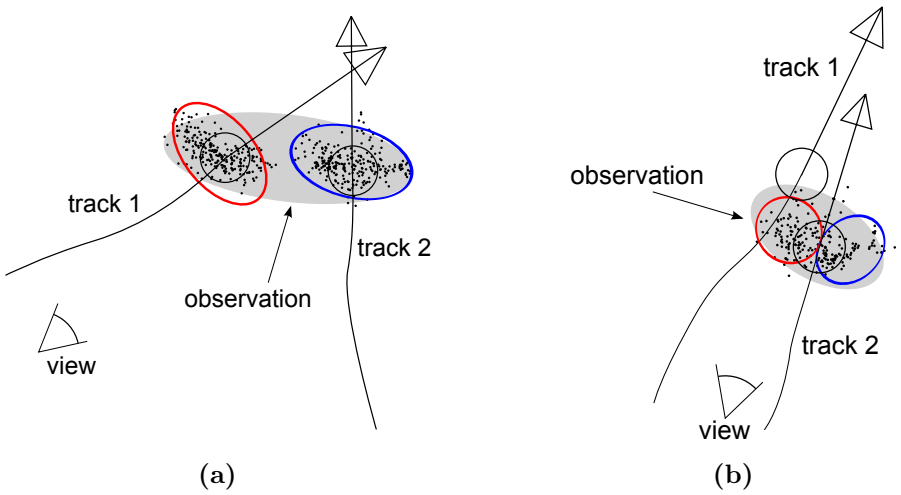


Figure 3.10 – Example of using K-means to re-segment an observation. In (a), the observation (grey) is separated into two parts (red) and (blue) which are used to update track 1 and 2 respectively. In (b) track 1 is occluded and so the observed points are all from track 1. Attempting to split the observation using K-means results in two observations which come from the same object and both tracks end up converging to track the first object.

observable object, resulting in the track of the occluded object being lost. An example of how this can happen is shown in Figure 3.10b.

3.2.1.3 Split-cost

We propose a new method, referred to here as **split-cost**, which avoids this problem whilst maintaining the desirable ability to split the observation in certain cases. It does so by calculating a cost function after K-Means clustering and using this to determine whether to update tracks with the re-clustered points, or to use the original observation as-is.

The cost for matching a track (represented by its predicted state, $\hat{x}$) to the points (p) of a re-clustered observation (Z') is given by Equation 3.4 where the Mahalanobis distance is as shown in Equation 3.1, using the standard innovation covariance from Equation 3.2.

$$cost(\hat{x}, Z') = \sum_{p \in Z'} D_m(\hat{x}, p) \tag{3.4}$$

A data association hypothesis Ω maps tracks to (perhaps re-clustered) observations. The total cost for a given hypothesis is then given by Equation 3.5.

$$cost(\Omega) = \sum_{(\hat{x}, Z') \in \Omega} cost(\hat{x}, Z') \tag{3.5}$$

For both of the scenarios shown in Figure 3.10, three data association hypotheses will be generated:

Hypothesis Ω_1: The observation is associated with track 1, and track 2 is not observed.

Hypothesis Ω_2: The observation is associated with track 2, and track 1 is not observed.

Hypothesis Ω_3: The observation is split into two parts and each track is associated with an observation.

In case (a) of Figure 3.10, $cost(\Omega_3)$ is the smallest and so the observation is split. In case (b), $cost(\Omega_1) < cost(\Omega_3) < cost(\Omega_2)$ and so the observation will be used to update track 1.

Taking the minimum cost works well for the case where objects are both observed, or one object is completely occluded. However, when objects are close to each other the costs can become similar and it becomes ambiguous whether or not the observation should be re-clustered. To detect these ambiguous situations, the ratios of the hypothesis costs are considered. All hypotheses with a cost less than a given multiple α times the minimum cost are considered to be valid. From these hypotheses we then choose the one that updates the fewest number of tracks. Effectively, this means that we bias the tracker towards data association hypotheses that treat some tracks

as occluded and assign the observation to the fully visible tracks. When the tracker detects an ambiguous situation and updates the foreground track, the background track is deemed to be lost and reset.

In this way, the tracker is designed to be conservative and, in terms of the V-measure analysis described in Section 2.4.4, is designed to maximise track homogeneity, possibly at the expense of completeness. The method resets tracks that are ambiguous and thereby aims to prevent an "identity switch", where the same track starts to follow a different physical object. This ability of the tracker to detect ambiguous situations and maximise homogeneity is an important part of the group tracking algorithm that will be described in Section 3.3.1.

Through its hypothesis generation and ranking scheme, **split-cost** copes with some degree of under-segmentation without having problems due to naïve re-clustering that is possible in the **split** method. The effectiveness of this approach is evaluated in the experiments that follow.

3.2.2 Dataset

To evaluate the performance of these different tracking methods, a range of scenarios of varying complexity were designed. The focus of these scenarios is on tracking objects as they interact with each other and so the experiments were performed on the unmanned aerial vehicle (UAV) runway at the ACFR facility "Marulan", which is an open space free of clutter from trees, buildings or other moving objects (Figure 3.11).

These experiments were executed using the ACFR mobile sensor platform "Shrimp": a skid-steer vehicle based on the Segway RMP-400 platform, which carries a tactical-grade real time kinematic (RTK) GPS/INS system, on-board computing and a comprehensive sensor payload (Figure 3.12). For the experiments in this chapter, only the Velodyne lidar sensor was used to perform tracking.

A number of interaction scenarios of varying complexity involving pedestrians and cyclists were collected and are shown in Figures 3.13 and 3.14. The scenarios were

Figure 3.11 – Photograph of the area used to perform the "Marulan" experiments, with the sensor vehicle 'Shrimp' in the centre. (Photo taken during the MERGE-BB experiment from Figure 3.14.)

deliberately designed to test tracker performance in ambiguous situations, by including trajectories that cross and merge. The complexity ranges from two pedestrians following planned trajectories to four pedestrians or two pedestrians and two cyclists interacting in an unscripted manner.

3.2.3 Ground Truth

The ground truth trajectories of each object were manually labelled using the open-source 3D creation software, "Blender" [102]. To accelerate the labelling process for this thesis, we developed a custom extension module that displays Velodyne and camera data simultaneously and allows 3D bounding boxes to be drawn around each object. Object bounding boxes were labelled at key frames spaced 0.5 to 1.0 second apart, and interpolation was then used to give the full object trajectory. Through this process, 54 separate object tracks were labelled across the 16 datasets in a total of total of 22072 scans. The centroids of these labelled bounding boxes form the trajectories shown in Figures 3.13 and 3.14. A screenshot of the application is shown in Figure 3.15. The fact that bounding boxes are labelled rather than just centroids means that the 3D lidar data corresponding to each object can be extracted. An example of the observations extracted from a labelled trajectory is shown in Figure 3.16.

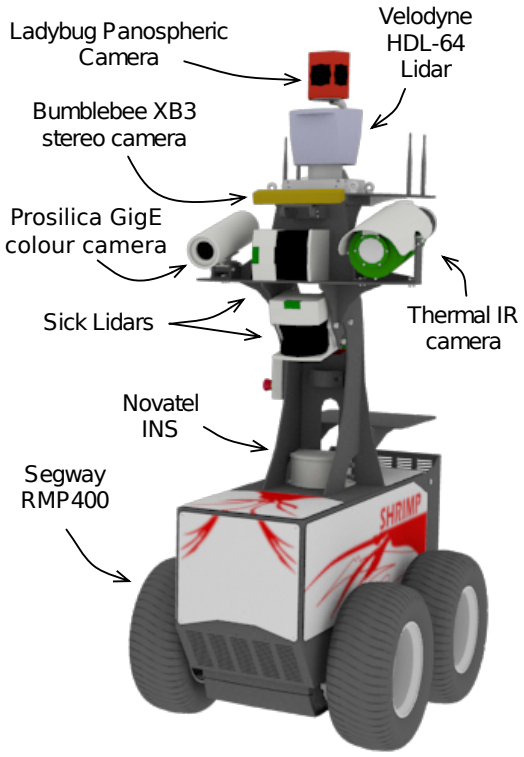


Figure 3.12 – The ACFR perception robot "Shrimp" that was used to acquire data for the experiments in this thesis.

3.2.4 Results and discussion

The tracking performance of the methods listed in Section 3.2.1 were evaluated according to the metrics described in Section 2.4. Figure 3.17 shows the MOTA and MOTP for each tracker on the dataset described in Section 3.2.2. The mean values across the different experiments are shown in Table 3.1.

In terms of MOTP, where better performance is reflected by lower scores, **split** and **split-cost** outperformed **centroids** across all datasets. The ability of those trackers to re-cluster under segmented observations means the lidar returns used to build each track are more likely to come from a single target. As a result, the object locations estimated by these methods were closer to the ground truth.

In Figure 3.17b, **split-cost** achieves the highest scores on average and therefore exhibits best performance on the MOTA measure. Whilst there are a few instances where **split-cost** performs slightly worse than **split**, it outperforms **centroids** in all

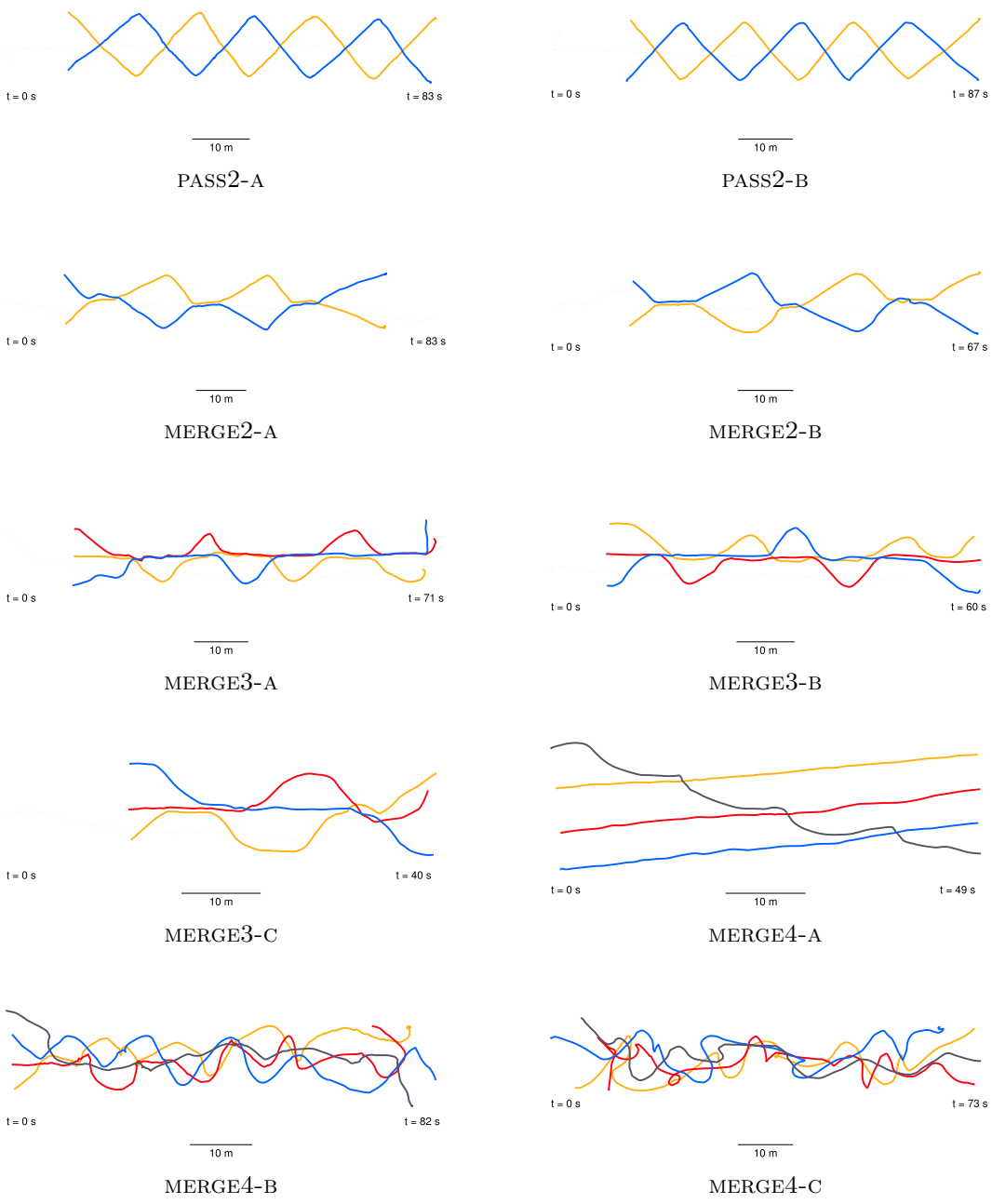


Figure 3.13 – Pedestrian interaction scenarios used to evaluate trackers. Each plot shows manually labelled object centroids, coloured per track, as seen from above. Targets move from left to right in the figures, and the start and end time is indicated. The sensor vehicle is manually driven approximately 10-15m behind the targets. In the 2 and 3 person interaction scenarios there is an extra track represented as a dashed grey line. This person is controlling the sensor vehicle and walks behind it, not interacting with the other people.

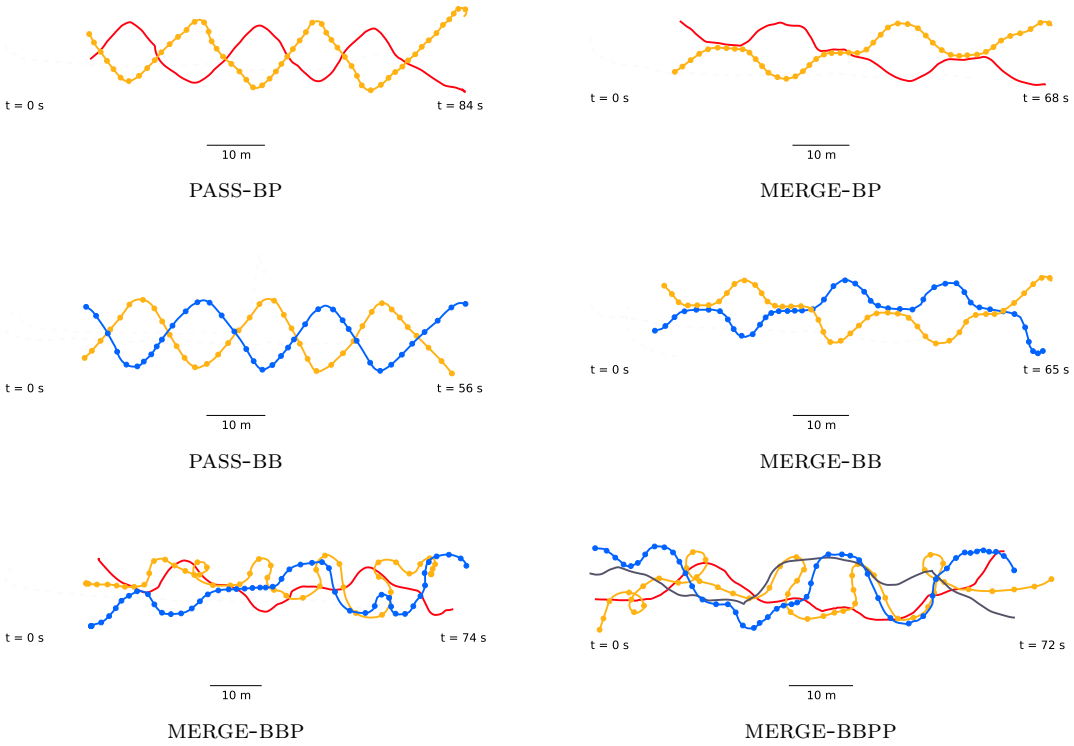


Figure 3.14 – Pedestrian and cyclist tracking scenarios. As in the previous figure, each plot shows manually labelled object centroids, coloured per track, as seen from above. Targets start at the left and end on the right, and observer trajectories are shown as dashed lines. Cyclist trajectories are indicated with additional circular markers.

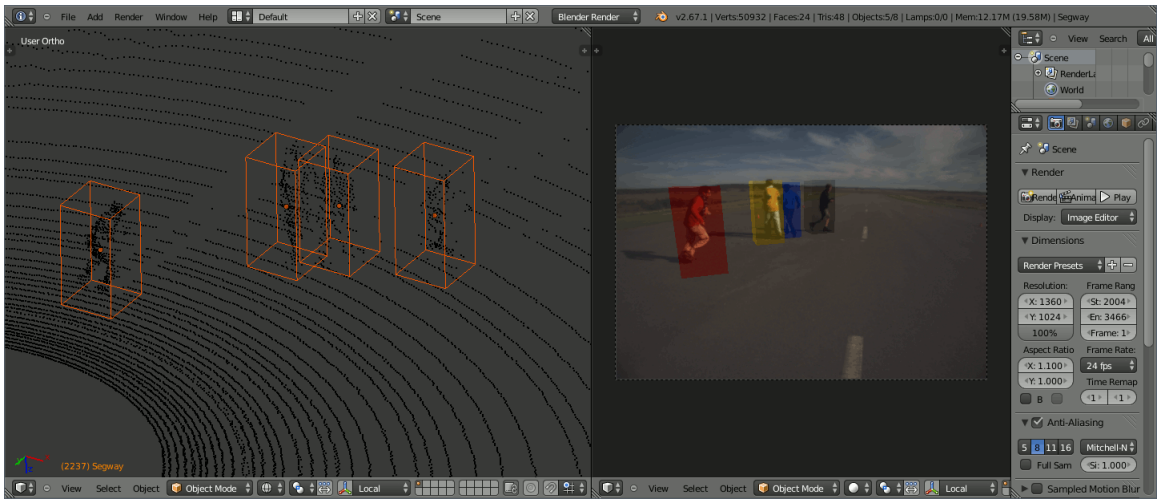


Figure 3.15 – Screenshot of the open-source 3D creation software **Blender** [102] as used to label object trajectories. The left-hand pane shows 3D bounding boxes around the pedestrian targets, whilst the right hand pane shows these bounding boxes projected into a image from the forward-facing camera.

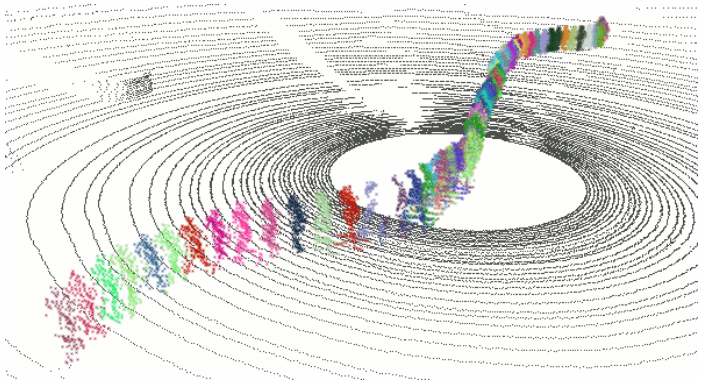
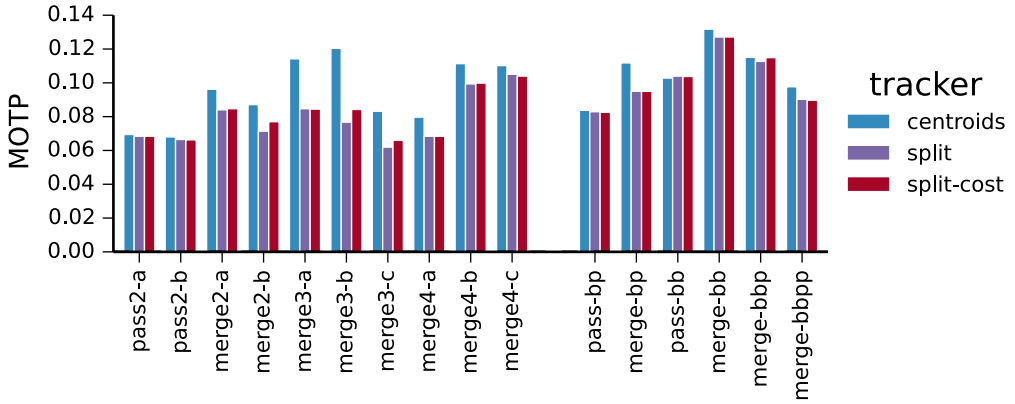


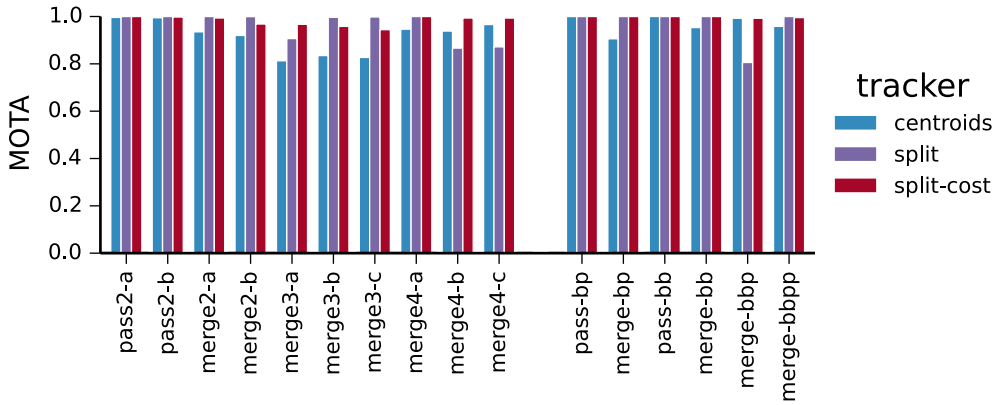
Figure 3.16 – Example trajectory labelled with Blender. The labelled object is shown in a different colour for each scan. For clarity, only every 10th scan is displayed, and to give some context the background points from one scan are shown in black.

cases. The **split** method also outperformed **centroids** except in three instances. In each of these cases, the tracking failure was caused by the problem shown in Figure 3.10b, where an incorrect split results in trackers converging on the same target and producing a duplicate trajectory. To further illustrate this issue, the result of **split** on the dataset MERGE4-B is shown in Figure 3.18.

V-measure, when used as a tracking metric as described in Section 2.4, is designed to evaluate the ability of a tracker to maintain the correct identity for each target



(a)



(b)

Figure 3.17 – MOTP and MOTA for the Marulan datasets. Lower numbers indicate better MOTP, whilst higher numbers represent better MOTA.

	MOTP	MOTA	homogeneity	completeness	vmeasure
centroids	0.097	0.926	0.816	0.640	0.712
split	0.084	0.951	0.921	0.889	0.904
split-cost	0.089	0.979	0.976	0.854	0.908

Table 3.1 – Mean tracking performance for the Marulan datasets.

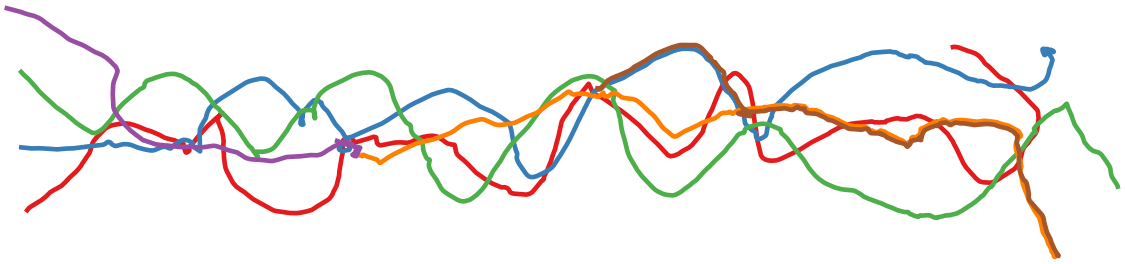


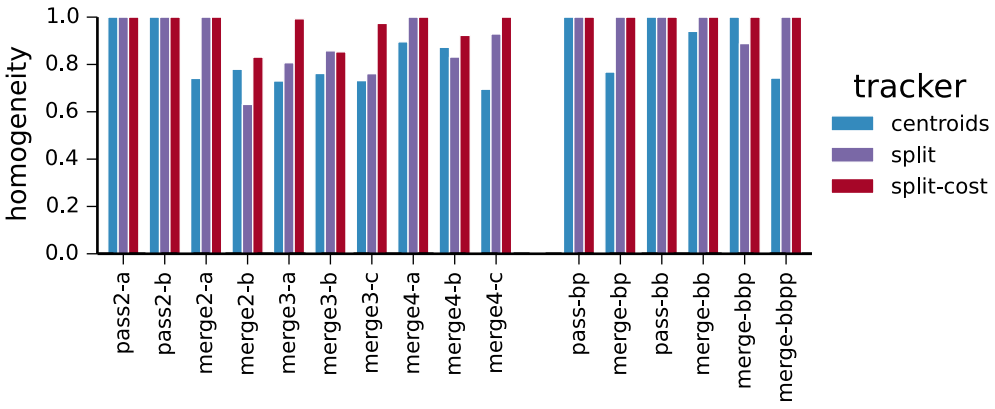
Figure 3.18 – Tracking result of **split** on the MERGE4-B scenario. There are only four different objects in the scene, but 5 tracks terminate at the right hand side of the figure. A tracking error has caused the track that ends at the lower right of the figure to be duplicated.

and the homogeneity, completeness and V-measure scores of the tracking methods are shown in Figure 3.19 and summarised in Table 3.1.

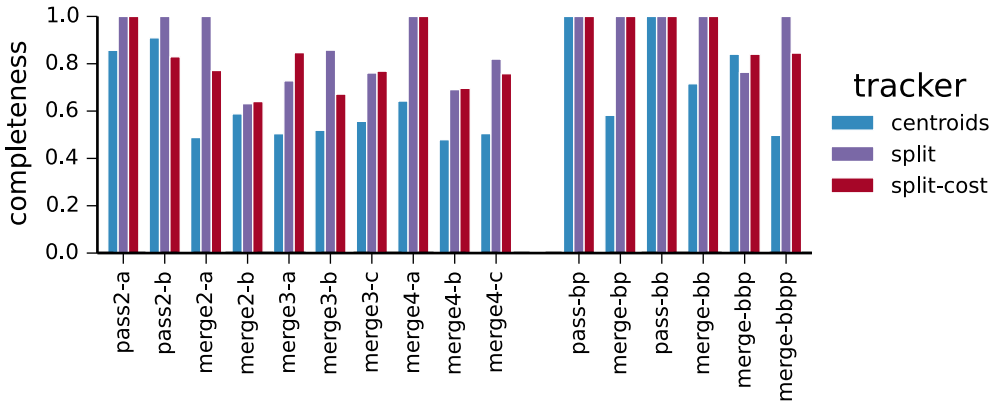
Split and **split-cost** generally achieve higher V-measure scores than **centroids**, but the difference between **split** and **split-cost** is less clear. **Split-cost** is designed to be conservative with its data association, so that when there is an ambiguous assignment it will favour the foreground track and reset occluded tracks. This helps the method avoid the re-clustering errors of **split** discussed above and results in very high homogeneity scores (Figure 3.19a). The downside to this high homogeneity is that the method will sometimes be overly conservative, and this reduces its completeness score (Figure 3.19b). Since **split** always re-clusters and doesn't reset occluded tracks, it achieves higher completeness but lower homogeneity. V-measure, which combines homogeneity and completeness, shows a slight advantage to **split-cost**.

An interesting example that highlights the difference between MOTA and V-measure as a metric is MERGE3-C. The trajectories estimated by each method are shown in Figure 3.20, and the various metrics are shown in Table 3.2.

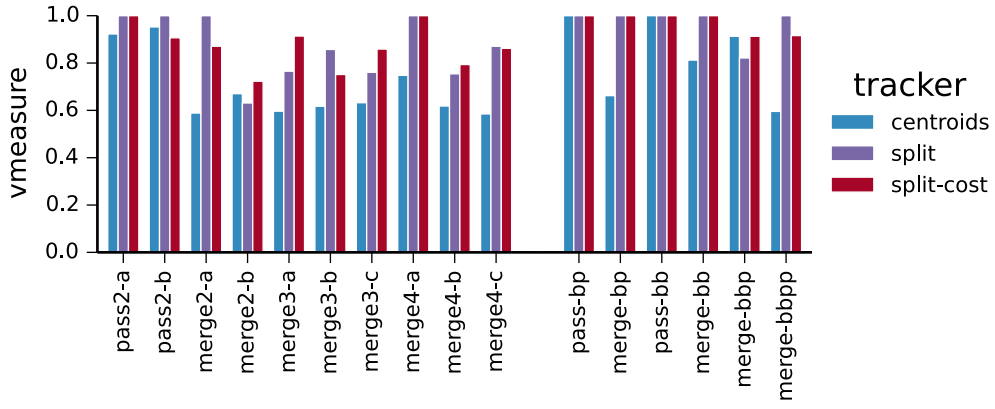
On this example, **split** performs very well according to MOTA with a score of 0.998 (due to 2 misses, 2 false positives and 2 mismatches), whilst **split-cost** (with 174 misses and 5 mismatches) achieves a MOTA of 0.944. In terms of V-measure, however, the ordering is reversed, and **split-cost** outperforms **split** significantly. The reason for this becomes apparent upon comparison of the estimated trajectories to the ground



(a)

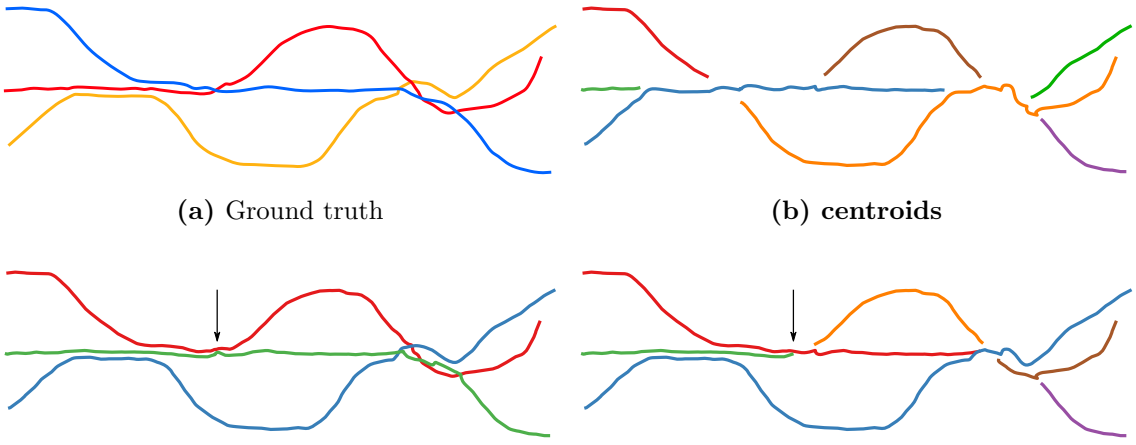


(b)



(c)

Figure 3.19 – Homogeneity, completeness and V-measure for the Marulan datasets. The highest homogeneity is achieved by **split-cost** in all cases, whilst **split** occasionally has better completeness. In terms of the combined metric, V-measure, **split-cost** has the best performance.



(a) Ground truth **(b) centroids**

(c) split **(d) split-cost**

Figure 3.20 – Comparison of trajectories estimated by the different tracking methods for the MERGE3-C dataset. Although the **split** method results in three continuous trajectories, the identities switch at the point marked by the arrow. Both the **centroids** and **split-cost** method create more tracks for the same number of targets, but **split-cost** does a better job. The track corresponding to the robot operator is removed for clarity.

	MOTP	misses	false_positives	mismatches	MOTA	vmeasure
centroids	0.083	549	0	8	0.827	0.632
split	0.062	2	2	2	0.998	0.761
split-cost	0.066	174	0	5	0.944	0.859

Table 3.2 – Tracking performance on MERGE3-C dataset

truth. At the point in the experiment marked by an arrow in Figure 3.20, **split** switches the identity of two tracks. According to MOTA this is two mismatch errors, which reduces the score by only a small amount, however this makes the rest of the trajectory also incorrect resulting in a low V-measure score. In terms of providing a track estimate that is as consistent as possible, starting a new track in the case of ambiguous associations is desirable. As V-measure measures identity consistency (defined in terms of homogeneity and completeness) it is a more appropriate metric to capture this property.

These experiments demonstrate that when objects get close together, conventional tracking methods fail and methods that naïvely perform re-clustering can become confused when the data association becomes ambiguous. We propose in **split-cost** an extension to the re-clustering approach which is shown to improve upon **centroids** without failing as badly as **split** in difficult cases. As we have seen, though, **split-cost** pays for its improved homogeneity with reduced completeness. The desire for a tracking approach that can maximise both homogeneity *and* completeness leads us to 'group tracking', which is the focus of the next section.

3.3 Group Tracking

When tracking closely spaced targets such as interacting pedestrians the difficulty of matching observations to tracks has inspired a number of 'group tracking' approaches, whereby the data association algorithms lift the assumption that each track corresponds to one physical object. Instead, tracks can represent groups of objects with their formation either explicitly modelled, or implicitly obtained from the data. Grouping approaches have been developed for 2D lidar data [55, 73] as well as vision [97, 106]. The benefits of group tracking have also been applied with the PHD filter [21].

Grouping tracks together improves performance in terms of computation time (tracking fewer targets is faster), consistency (by avoiding 'hard' data association decisions)

and can also give semantic information about the relationship between objects. However, in order to reason about the identities of individual targets after they emerge from a group, a history of interactions needs to be maintained. Such a data structure is called an 'event graph'.

In this section we describe an event graph framework for lidar based tracking. This is only part of an identity tracking system. To be able to recover the identities of individual targets after they emerge from groups, we need appearance models and graph reasoning algorithms, which will be presented in Chapters 4 and 5, respectively.

3.3.1 Event Graph

An 'event graph' or 'track graph' [77, 106] is a data structure used to represent the higher-level interactions between tracked targets in a scene. In this thesis, we adapt the event graph formulation of [106] to 3D lidar data. The complete event graph tracking framework shown in Figure 3.21 (which we presented in [72]) includes the use of camera data to solve the identities of each node in the graph. This section however, focuses on the construction of the graph from lidar data (the modules shaded in grey in the Figure 3.21); appearance models are discussed in Chapter 4, and identity reasoning in Chapter 5.

3.3.1.1 Track extraction

In conventional tracking (such as the methods in Section 3.2.1.1), data association ensures each track is updated by one observation at each time step. As part of this there is an assumption that each observation contains only one object.

Whilst objects remain well separated from each other this assumption is not violated and the track extraction module can use the same centroid-tracking methods discussed previously to estimate the trajectory of each target. If the module detects that the expected one-to-one track-observation matching has been violated, then track extraction stops and a new event will be recorded in the event graph.

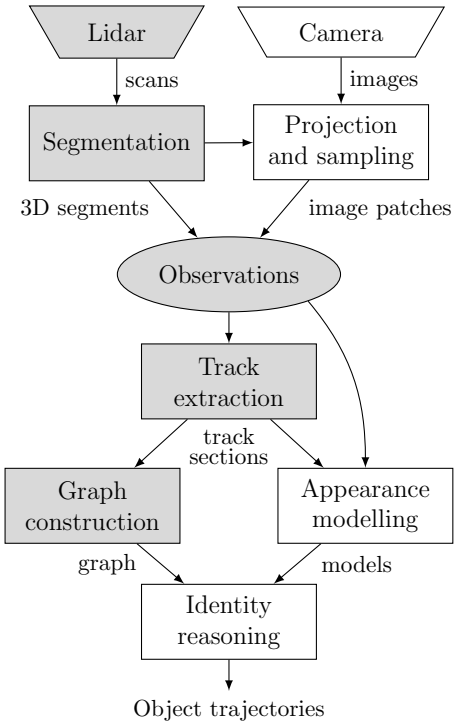


Figure 3.21 – The event graph tracking framework. Parts of the framework discussed in this chapter are shaded in grey.

Groups of people may be tracked in the same way – for example if two people are consistently segmented into one observation, then a "track section" will be created for these two objects together.

The role of the track extraction module, therefore, is to associate consistent sets of observations (represented by their centroids) together into 'track sections'. The module is designed to be conservative in the same way as the **split-cost** method from earlier in this chapter. Track sections should be homogeneous, and the reduction in completeness that this entails is compensated for elsewhere in the system.

3.3.1.2 Graph Construction

The event graph represents the relationships between the track sections extracted in the previous module. Each track section is represented as a node in the graph and edges represent the relationship between sections (e.g. two objects coming together to form a group, or a group splitting into multiple objects).

An example of an event graph is shown in Figure 3.22. Two pedestrians walk from top to bottom of the image starting as two tracks (shown as nodes 1 and 2 in the graph (Figure 3.22a)).

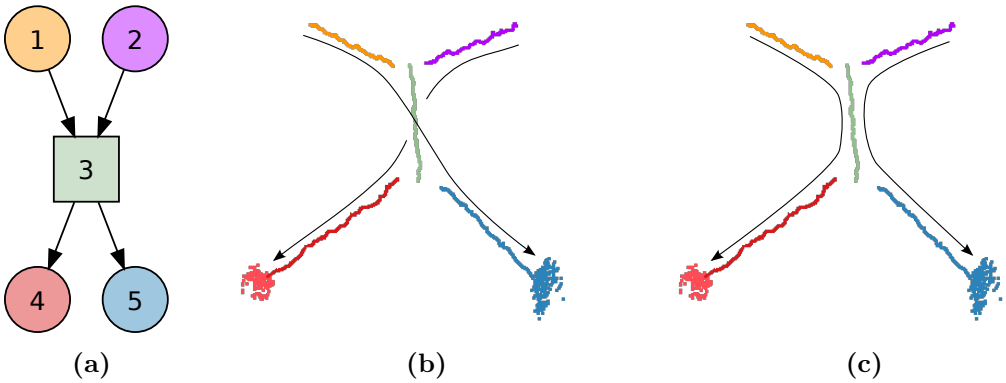


Figure 3.22 – Event graph (a) for the merge-split trajectories shown in (b) and (c), where two people walk from the top of the figure, merge and split apart again. The event graph captures the ambiguity in the objects' final positions, and scenarios (b) objects cross and (c) objects don't cross are equally likely.

As the pedestrians move together, the data-association between observations and tracks becomes ambiguous. The track extraction module stops tracking the individual objects, and starts a new track for the group. The relationship between this group track (node 3) and its parent tracks is captured in the graph.

Whilst the objects remain together, under-segmentation causes them to be measured as a single observation so the data association is clear and the group track (node 3) continues to be updated. As the objects move apart, the system receives multiple observations that overlap with the group track. This causes the track extraction module to stop updating the group track and spawn two new single-object tracks that are linked into the event graph (nodes 4 and 5). The event graph captures the ambiguity inherent in centroid-only tracking, encoding the fact that the target represented by node 5, for example, could be the same object as either node 1 or node 2.

3.3.1.3 "Ambiguity"

The preceding sections described that track extraction stops and an event is created in the graph any time the data association becomes ambiguous. The only requirement for this is that a tracker declares the situation to be 'ambiguous' *before* it makes any data association errors. Therefore exactly what is defined as ambiguous depends on the capabilities of the tracking algorithm.

In the original version of this work, which we published in [72], data association was driven entirely by the observations generated by the segmentation process. The track-to-observation distance matrix (introduced in Section 3.2.1.1) is used to match observations to tracks and ambiguous situations occur when one track matches multiple observations, multiple tracks match one observation, or there is a many-to-many matching.

An implementation detail worth noting is that a single-object track section might be observed to split. This could occur when, for example, two pedestrians enter the field of view of the sensor whilst close together, then move apart. In this case, a split event is recorded, and the parent node is promoted from a single-object track to a group.

This method is referred to here as **centroids-graph** as it is based upon **centroids** from Section 3.2.1.1. Its behaviours in the various ambiguity cases are described below, where n_x and n_Z refer to the number of tracks and number of observations involved in a match

$(n_x > 1, n_Z = 1)$
: One observation matches multiple tracks – objects have merged together. Create a new group track which is a child of the parent tracks.

$(n_x = 1, n_Z > 1)$
: A track has split into multiple parts – create two new tracks which are children of the current track.

$(n_x > 1, n_Z > 1)$
: Multiple tracks and multiple observations, but since they are all within associ-

ation range of each other the correct decision is ambiguous. The observations are merged together into one, and then a group node is created in the same manner as the $(n_x > 1, n_Z = 1)$ case.

As shown in Section 3.2.4 tracking performance can be improved by allowing for a re-clustering step during data-association. **Split-cost** improves upon the basic re-clustering of **split** by using a cost function to determine the best data association decision as well as to detect ambiguous situations.

We propose here a new event graph tracking method, called **split-cost-graph**, which builds upon the logic of **split-cost** by modifying the handling of ambiguous situations. Whereas **split-cost** uses the cost term to make a conservative tracking update, **split-cost-graph** uses ambiguity as a trigger to create a merge event in the graph.

Figure 3.23 shows this process. Firstly, all data association hypotheses are evaluated. The set of hypotheses below the ambiguity threshold are found, and any common data association decisions are applied. In the example shown, hypothesis Ω_4, Ω_5 and Ω_6 are considered ambiguous, but all have a common assignment for the blue track so it can be updated independently. The remaining tracks are then merged into a group.

3.3.2 Results and discussion

The description of group tracking approaches in this thesis so far covers the formation of an event graph, but does not explain how to reason about the objects represented by each node. Nodes in the graph correspond to track sections, which represent only part of an object's trajectory and are expected to be homogeneous but not complete. These track sections must be linked together in order to form full trajectories. This graph solving logic first requires the introduction of appearance models, which we present in Chapter 4. The complete tracking framework is then described in Chapter 5.

In the absence of a full graph solution, we can not analyse the identity-tracking performance of **centroids-graph** and **split-cost-graph** in terms of MOTA and V-

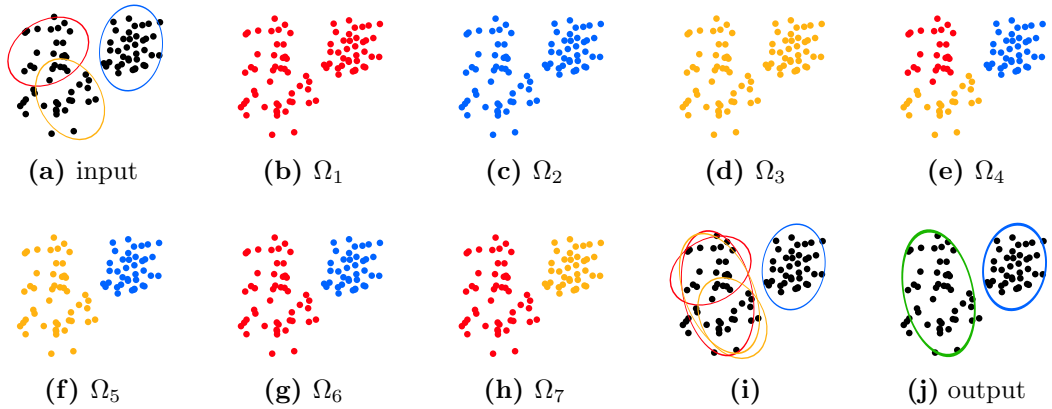


Figure 3.23 – Example of **split-cost-graph** data association process. (a) During minimum cost matching, 3 tracks (red, yellow and blue) map to the same observation (lidar points represented by black dots). Various hypotheses are generated and costed, with the colour of the points indicating the track to which they are assigned. The hypotheses are: (b-d) one object was observed, (e) all three objects were observed, (f-h) two objects were observed. After generating these hypotheses, Ω_4, Ω_5 and Ω_6 are below a pre-defined cost threshold. These three hypotheses each assign the same set of points to the blue track (i), so it is updated directly. However the hypotheses disagree about the assignment of points to the red and yellow tracks (i.e. the association is deemed ambiguous), so these tracks are merged together into a group (j).

measure. Instead, we can evaluate how successful graph formation was by comparing the number of individual tracks created to the actual number of objects in the scene. For example, Figure 3.24a shows the graph created by the **split-cost-graph** method for the MERGE3-C dataset. The fact that the graph has four start nodes (nodes with no parents) and four end nodes (nodes with no children) means that four people have been tracked through this dataset.

Figure 3.25 shows the difference between the number of tracks created by the tracker and the number of labelled objects in each scenario. Because the **split-cost** method is designed to be conservative, and starts tracking from new when an object becomes occluded, it creates more new tracks than does **split**. On the pedestrian-only dataset, **split-cost-graph** estimates the number of objects correctly in all cases, whilst **centroids-graph** does well in most experiments but creates two extra tracks on MERGE4-B. On the bike and pedestrian datasets, **centroids-graph** has a bit more difficulty, creating extra tracks on PASS-BP, PASS-BB and MERGE-BBP, and

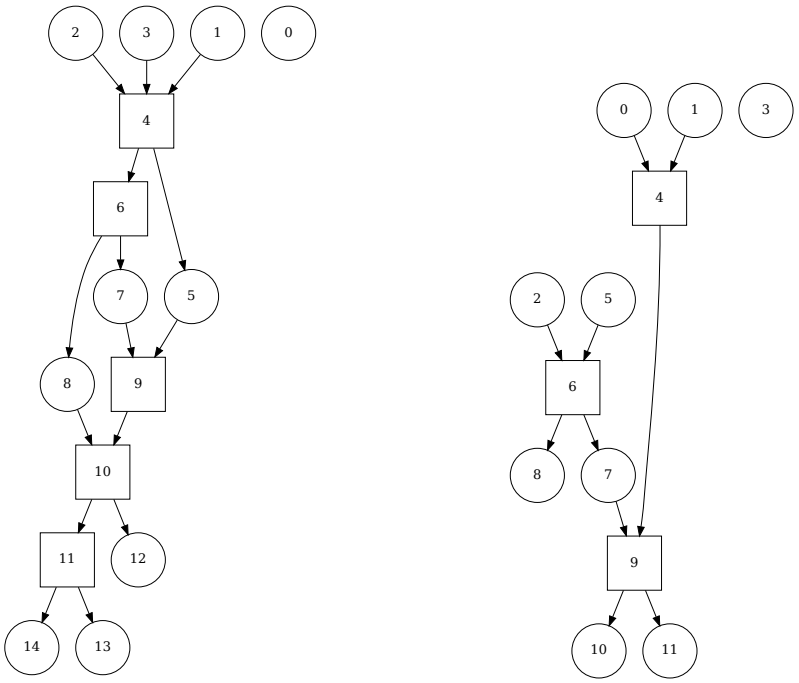


Figure 3.24 – (a) Event graphs created by the **split-cost-graph** approach. (a) Successful graph generation for the MERGE3-C experiment. (b) Inconsistent graph created for the MERGE-BBP experiment. There are five starting nodes but only four end nodes. A correct graph solution would have a link between nodes 4 and 5.

split-cost-graph has graph errors on MERGE-BBP and MERGE-BBPP. An example of a graph error is shown in Figure 3.24b.

Of the evaluation metrics used in Section 3.2.4, MOTP and homogeneity can be used to analyse the performance of these graph methods, even though we have not yet discussed how to determine object identities. MOTP measures the positional accuracy of a tracker, irrespective of any identity decisions it has (or in this case, has not) made and the results are shown in Figure 3.26.

The event graph approaches incur a MOTP performance penalty whenever they create a group node, since the estimated centroid of the group as a whole will be further away from the ground truth than estimates of each object's individual centroid. This MOTP trade off is a necessary part of the event graph's ability to represent identity ambiguity but approaches that minimise the effect are desirable. The ability of **split-**

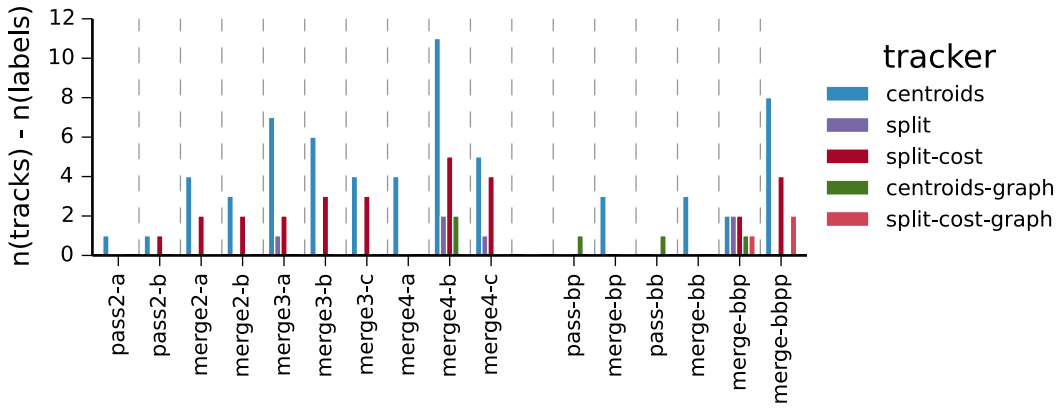


Figure 3.25 – Number of extra tracks created by the trackers on each dataset. **Split-cost-graph** gets the number of targets correct on all but the MERGE-BBP and MERGE-BBPP experiments.

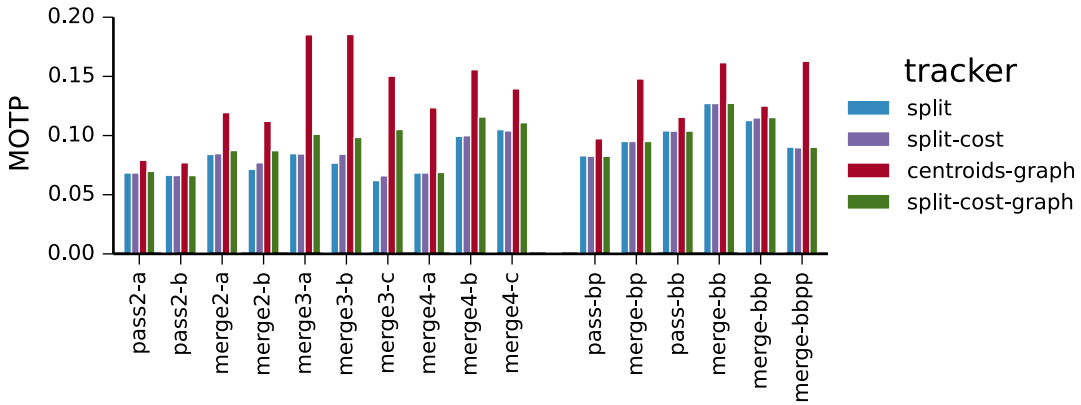


Figure 3.26 – MOTP results (recall that lower MOTP indicates better positional accuracy). **Centroids-graph** creates many more groups then **split-cost-graph**, and this results in increased MOTP.

cost-graph to re-cluster observations means that it creates many fewer groups than **centroids-graph** and as a result the impact of its grouping behaviour on MOTP is less noticeable. This is evident in Table 3.3, which shows the average MOTP across the data shown in Figure 3.26. The average MOTP score of **split-cost-graph** is much closer to **split** and **split-cost** than is **centroids-graph**.

Figure 3.27 shows the homogeneity of the various tracking solutions. Since we haven't yet linked graph nodes together into object trajectories, this graph only considers the homogeneity of individual track sections. **Split-cost-graph** obtains high homogene-

	MOTP
split	0.087
split-cost	0.089
centroids-graph	0.134
split-cost-graph	0.095

Table 3.3 – Average MOTP across the data shown in Figure 3.26.

ity scores across almost all datasets, indicating that if we can increase completeness by correctly linking track sections together into full trajectories we will be able to achieve high V-measure scores.

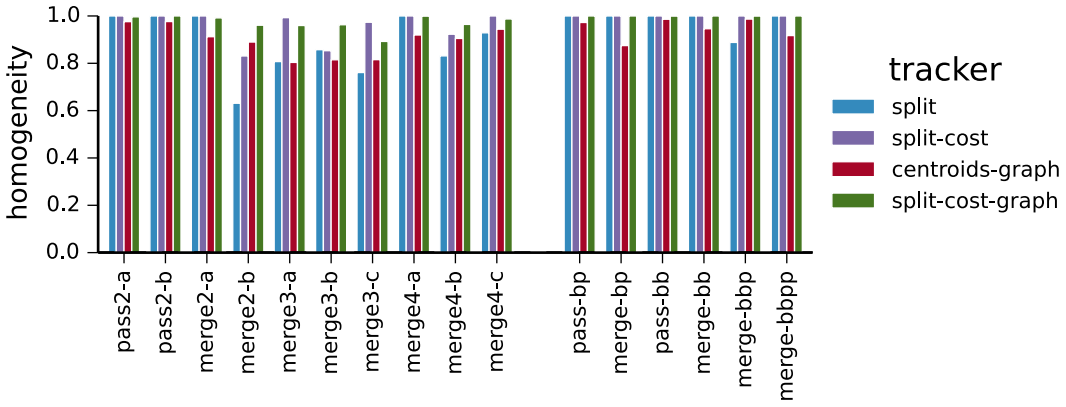


Figure 3.27 – Homogeneity score of the tracking methods.

An additional benefit of the **split-cost-graph** method's ability to re-cluster observations is that it creates simpler event graphs. For the same input data, **split-cost-graph** encounters ambiguous situations less often than **centroids-graph** and records fewer merge/split events. Figure 3.28 shows the number of nodes created by the two graph-based methods. This reduction in complexity will become relevant in Chapter 5 when we introduce methods that solve for node identities. A graph with fewer nodes for the same data means that longer track sections have been extracted. As was demonstrated in Figure 3.27, **split-cost-graph** acheived these longer track sections whilst still obtaining the highest homogeneity scores.

The experiments in this section have established the utility of our proposed graph-based tracking approaches. **Split-cost-graph** in particular was able to correctly

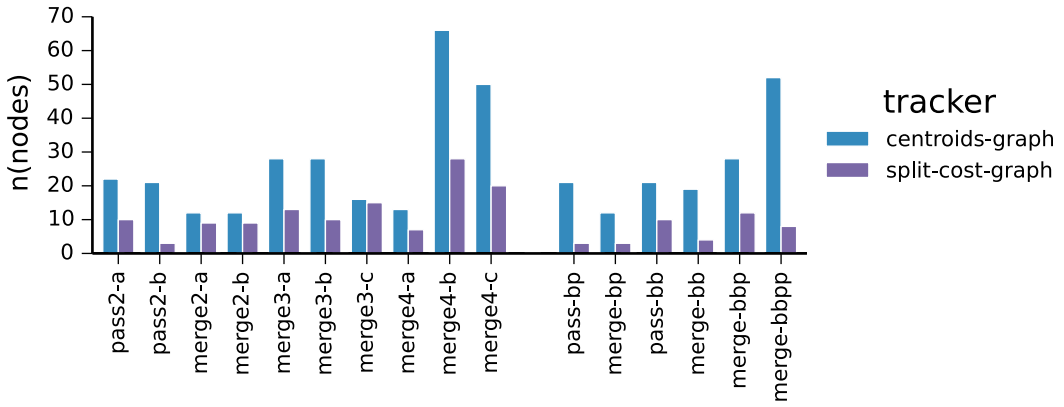


Figure 3.28 – Number of nodes in the event graphs created by **split-cost-graph** and **centroids-graph**. **Split-cost-graph** results in simpler graphs, which becomes relevant when we solve for node identities in Chapter 5.

determine the number of objects present in a scene in almost all cases, with only a slight decrease in metric tracking accuracy. The method achieved high homogeneity indicating that track sections, as desired, correspond to stable configurations of objects. As objects interact, form groups and move apart, **split-cost-graph** is able to record a graph structure that correctly reflects these events.

The results shown in this section are not the complete picture. False positives and missed observations have not been analysed nor have identity assignments. As presented, there is no way to associate the identity of objects to nodes in the event graph, and the full trajectory of individual objects can not be recovered. These algorithms will be covered in Chapter 5 and it is in that chapter that we will evaluate the full performance of these tracking methods in terms of V-measure and MOTA.

3.4 Summary

In this chapter we considered methods for tracking interacting objects using lidar data. We described the pre-processing steps required to convert lidar data into observations (the input to the tracking system) and considered issues such as calibration, coordinate transforms and segmentation via both ground detection and clustering, and change detection.

We compared a number of different approaches to tracking, including a novel method that offers improved robustness against under segmentation, whilst achieving a significant increase in track homogeneity. From the detailed experimental evaluation of these methods, we concluded that there was a need for tracking methods which could maintain this level of homogeneity whilst increasing the completeness of the trajectories obtained.

This motivated the exploration of group tracking approaches, with a specific focus on methods that maintain a record of object interactions such that sets of ambiguous identities can be explicitly determined. To this end, we adapted an "event graph" strategy from the computer vision literature, and demonstrated that graph complexity can be significantly reduced by applying our our proposed lidar-based tracking method.

In the next chapter, we introduce the concept of "appearance models", target representations that can be used to resolve node identities after periods of ambiguity. Algorithms that combine event graphs and appearance models to determine full object trajectories are presented in Chapter 5.

Chapter 4

Appearance modelling for tracking

In the previous chapter, we presented the case for a tracking system that allows for objects to be tracked as groups rather than individual targets during times of ambiguity. Occlusion, or other problems with segmentation caused by the proximity of tracked targets to each other, means that more stable tracking can be achieved by merging tracks together.

Whilst the event graph approach described in Section 3.3.1 does maintain some aspect of object identity by recording the links between track sections, the position-based tracking system cannot resolve the mapping between the parent and child nodes of a group. Referring to the diagram of a basic merge and split, which is repeated for convenience in Figure 4.1, the graph represents that the positional tracker deems situations (b) and (c) to be equally likely.

Some situations do, however, require that the identities of tracked targets be preserved and to achieve this, additional information is needed. This chapter discusses the concept of appearance modelling, that is, representing tracked objects in ways that enable their identities to be recovered after periods of ambiguity such as interactions or occlusions. Appearance information can be extracted from the 3D lidar itself via additional processing, or from a complementary sensor such as a camera.

In this chapter we introduce the concept of appearance models and how they can

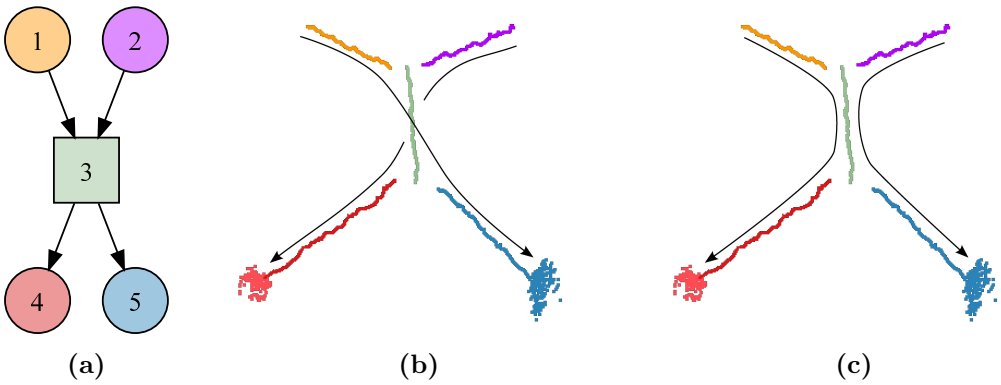


Figure 4.1 – Event graph (a) for the merge-split trajectories shown in (b) and (c), where two people walk from the top of the figure, merge and split apart again. The event graph captures the ambiguity in the objects' final positions, and scenarios (b) "objects cross" and (c) "objects don't cross" are equally likely.

be used to provide identity information for tracking. We discuss extracting appearance information from 3D lidar scans and camera images, and how to use 3D lidar observations to drive the selection of image data. We describe and experimentally examine the various parameters of an appearance modelling system, and show how these models perform over time and as the complexity of interactions increases.

4.1 Identity and Appearance

We define in this thesis an appearance model as a representation that captures the remotely observable properties of an object in a manner that is invariant to both the state (position, orientation, velocity) of the object, and the environment surrounding it (lighting, background, etc.).

Appearance modelling is strongly related to two concepts, recognition and categorisation. Recognition is the task of retrieving a label for an object, given that the object has been seen before. The object could have been previously seen in a different context, from a different angle or in a different environment. Categorisation is the task of labelling object type. Having seen instances of a type of object (for example, cars), a categorisation system seeks to generalise the label such that it can determined

whether or not a previously unseen object is also a car. A human *categorising* an object might say something like "that is a cup", whilst when *recognising* and object could say "that is *my* cup".

Whilst the recognition and categorisation problems both notionally make use of the concept of object appearance, the properties of the models they require are quite different. When performing categorisation, the goal is to 'abstract away' or 'average out' appearance details that are irrelevant to the task at hand. For example, if we are trying to categories vehicles as sports cars or sport utility vehicles (SUVs) the shape of the vehicle is a more relevant detail than its colour.[1] If, however, you are trying to recognise your car in a parking lot, colour becomes more important.

Determining the identities of tracked objects after interaction events is an object recognition problem. Furthermore, it is a constrained version of the problem in that the set of objects under consideration is clearly defined. Usually, when performing object recognition, the test object may or may not have been seen previously, and the set of possible objects (both seen and unseen) could be arbitrarily large. However when using appearance information in a tracking scenario, the event graph limits the objects we need to 'recognise'. For example, to determine the identity of a track that split from a group of three objects, we only need to recognise which of the three objects it is.

In this sense, recognition for tracking can be posed as a self-supervised multi-class classification problem, where the different classes represent individual objects in the scene. When resolving the identities of objects after a merge-split event, the input nodes provide labelled training data. Output nodes are then the test sets, which can be classified against the trained models to determine identities.

An advantageous aspect of self-supervision in the context of tracking is that track sections provide a whole sequence of training data. Tracking an object before an interaction results in not one but a set of observations, potentially from a changing viewpoint whilst the object being tracked changes poses and moves in front of varying

[1] excepting maybe a prior on the colour red.

backgrounds. Likewise after a split event, the tracker may observe the object for some period of time. This gives the system the opportunity to learn models which take into account the appearance variations caused by such changes.

4.2 Feature Extraction

In this section we describe how appearance information can be extracted from sensor data. There are many options for the choice of appearance features, which is an active area of research in its own right, and advancing the state-of-the-art in this area is outside the scope of this thesis. Instead, this chapter uses some common features from the literature to examine the requirements of appearance models in the context of identity tracking. This section presents the appearance features that will be used in this thesis, and describes the additional processing requirements when combining lidar and camera data.

An overview of the extraction process is shown in Figure 4.2. Lidar scans are segmented to form observations, the basic input to the tracking system (as described in Chapter 3). Appearance features can be computed from the lidar data directly (Section 4.2.1) or extracted from the corresponding camera data (Section 4.2.2). To obtain the camera data corresponding to lidar observations, the 3D laser measurements are projected into the camera frame (Section 4.2.3.1), whereupon a sampling strategy is applied in order to select the relevant image data (Section 4.2.3.2).

4.2.1 Appearance from lidar data

To date, most of the work relating to appearance information from 3D lidar has been with respect to the *categorisation* problem described earlier. Himmelsbach et al. look at the binary classification problem of finding cars in lidar data [42], Wang et al. classify foreground versus background objects [115], and Teichman and Thrun use tracking information to increase classification performance on cars, pedestrians and

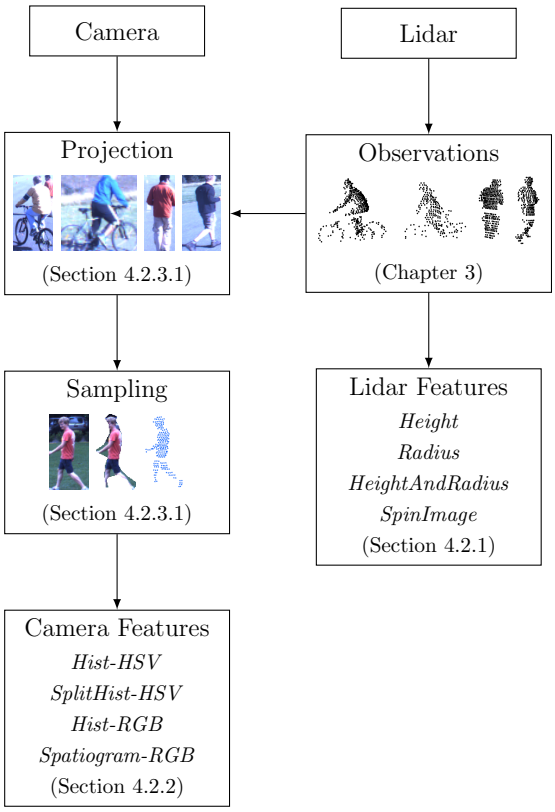


Figure 4.2 – Feature extraction from lidar and camera data. The sections in which the various elements are discussed are shown in parentheses.

bicycles when starting from a small labelled set [101]. Quadros et al. develop a feature specifically for object categorisation using Velodyne data to separate the classes "building", "car", "pedestrian", "sign", "tree" and "trunk" [82]. These methods are all designed to represent the appearance differences between classes, and minimise the effects of intra-class variations.

To discriminate targets, however, we're interested in capturing the appearance variation between all objects regardless of class. For a human looking at Velodyne data this is a difficult problem. Figure 4.3 shows some examples of point cloud observations of pedestrians and cyclists. Notice that it's much easier to tell the cyclists from the pedestrians than it is to tell one pedestrian or cyclist from another.

Despite this difficulty, some methods have been proposed to make use of 3D lidar data for determining target identity. Schöler et al. use target height as part of the

Figure 4.3 – Cyclists and pedestrians from Velodyne data. Note it is much easier to see the variation between classes (pedestrians versus cyclists) than it is to see the variation within classes.

particle weighting function in a tracking system [86], and Shackleton et al. use spin images [47] when computing the track association cost function [91]. Whilst height was found to provide some benefit in [86], the authors of [91] state that the "impact of [spin images] is minimal, and only affects the most pathological cases." Both of these methods combine appearance information directly into tracking and the utility of the appearance models have not been tested in isolation. In this work we explicitly measure the ability of these appearance models to determine the identity of tracked objects from Velodyne data.

In the experiments in Section 4.4 we employ height and radius, both independently and in combination, as a source of appearance information. Observation height is measured by the difference between the maximum and minimum lidar returns with respect to the global Z axis, whilst radius is computed from the minimum size circle that encloses all lidar returns projected on the X/Y plane. We also present results for the use of spin images. We use an image size of 1 metre, and as per [91] compute 20 spin images per observation.

4.2.2 Appearance from camera data

Camera data, particularly colour video, is a more intuitive choice for capturing the appearance of objects for the purposes of recognition. As humans, we make use of both object shape and colour to recognise objects, with an increased reliance on

Figure 4.4 – Camera observations for the same objects as in Figure 4.3. It is much easier (for a human at least) to separate the pedestrians and cyclists from each other in camera data than it is with 3D lidar.

colour for recognition amongst categories with similar shape [99, 103]. In remote-sensing systems we can also measure both shape and colour data. Whilst 3D lidar captures object shape, and can therefore be used to categorise objects with differing geometric appearance, colour data is a more powerful cue for recognition.

At least for a human, telling the difference between objects is much easier with camera data than it is with lidar data. Figure 4.4 shows the same four objects as Figure 4.3, this time as seen by a colour camera. A human can immediately see that these are four distinct objects, not one cyclist and one pedestrian, both imaged twice.

Camera-based multi-target tracking is an active area of research and many state of the art approaches have commonalities with the event graph approach described in Section 4.2.1, which is itself adapted from the camera-based tracking literature [106]. In [119] track sections are extracted based on local associations, and then trajectories are formed by linking these sections together based on appearance models, whilst in [9] ambiguities stemming from object interactions are resolved using appearance models.

Recent work in camera-based object categorisation has moved away from the manual specification of image features towards "feature learning" strategies that determine the best object representation directly from data (e.g. [52]). However, appearance modelling for tracking often relies on representing colour statistics with simple histograms [9, 119]. More advanced models seek to improve performance by training discriminative models that can better separate objects with similar appearance [117].

Name	Colour Space	Bins	Comment
Hist-HSV	HSV	6,3,3	As per [106]
SplitHist-HSV	HSV	6,3,3	Independent *Hist-HSV* for top and bottom half of image
Hist-RGB	RGB	8,8,8	For comparison with *Spatiogram-RGB*
Spatiogram-RGB	RGB	8,8,8	As per [13]

Table 4.1 – Colour models used in the appearance evaluation experiments. The "Bins" column gives the number of quantisation levels for each channel of the joint histogram. For example, *Hist-HSV* has 6 bins in the hue axis, and 3 in both saturation and value, for a total of $6 \times 3 \times 3 = 54$ bins. The chosen colour space affects the computation of histogram similarity measures, so both the RGB and HSV colour space are tested.

For reasons discussed later in this chapter, models that can be trained independently for each track section are preferred over discriminative models, so we focus our attention on histogram-based approaches. Whilst these methods are not generally considered to be the state of the art they are sufficiently powerful to highlight general properties of the appearance matching problem, which will be discussed in Section 4.4. Table 4.1 is a summary of the histogram models employed.

Colour histograms measure only the frequency of colour ranges within an image region and do not account for the spatial distribution of colour in an image. Various strategies that do incorporate this information have been proposed in the literature, including the *spatiogram* [13], which represents the spatial distribution of each colour bin as a 2D Gaussian. Some pedestrian-specific methods separate observed objects into parts (e.g. head, torso, legs) and learn colour models for each part [12]. We remove the pedestrian-specific assumption of [12], whilst retaining some concept of spatial distribution, by learning models on the top and bottom halves of each image. The four histogram features compared in this work are therefore joint histograms with HSV and RGB colour spaces (*Hist-HSV*, *Hist-RGB*), spatiograms (*Spatiogram-RGB*) and independent histograms from the top and bottom halves of an image region (*SplitHist-HSV*).

An additional modality also tested in this work is thermal infrared (IR). The IR camera installed on "Shrimp" has a spectral response of 7 to 14 μm and provides only

relative thermal information, not calibrated temperature readings. Pixel intensities encode thermal information and the IR camera has a narrower field of view than the Ladybug camera, but in all other respects the data can be processed in the same pipeline as colour camera data. A sample of the IR data is shown in Figure 4.5.

Figure 4.5 – Sample of thermal infrared camera data

Since the data from the IR camera is monochrome, the colour histograms described are not used. Instead we evaluate a single channel histogram *Hist-gray*, and *SplitHist-gray* which splits the image into upper and lower regions and computes independent histograms in the same manner as *SplitHist-HSV*.

4.2.3 Combining camera and lidar data

So far we have discussed ways to extract appearance information from an observation obtained from a 3D lidar point cloud or region of an image. In the system described in this thesis, tracking is driven by lidar and we wish to augment the lidar observations with camera data. To do so, we must be able to match point cloud segments with image regions that correspond to the same physical object or objects in the environment.

The first step to enabling this is to ensure that the sensors have an overlapping FOV. To maximise this overlap, the Velodyne is paired with an omnidirectional camera,

the Point Grey Ladybug-3 [81], shown mounted above the Velodyne on the ACFR experimental platform "Shrimp" in Figure 3.12. This positioning ensures that the Ladybug-3 has an uninterrupted 360° view of the scene that fully encompasses the FOV of the Velodyne lidar.

The two sensors (being different modalities) have very different data formats, sampling patterns, and acquisition times, yet so long as their field of views overlap, their data can be combined. Laser points can be used to add a sparse depth channel to the colour images or colour information can be added to the 3D point cloud.

Due to the differences in sensor positioning, sampling and timing, there are a number of issues that must be considered when combining camera and lidar data. We discuss two of these issues, calibration and sampling in the sections that follow. For a more complete examination of the topic, the reader is referred to [85].

4.2.3.1 Camera to lidar calibration

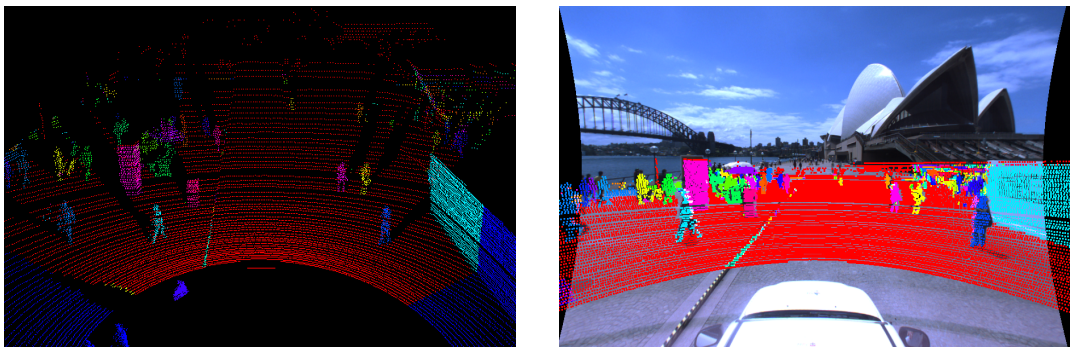
When projecting 3D lidar points into the camera frame, there are three important considerations: the intrinsic parameters specific to each sensor, the "extrinsic" transform between the sensors, and any timing differences between them. We discussed the intrinsic parameters of the Velodyne in Section 3.1.2 and for the Ladybug we make use of the factory-calibrated parameters supplied with the camera.

The transform between the sensors is established by calibration. The pairing of the Ladybug-3 and Velodyne sensors has proven popular in the literature and a number of methods of performing the requisite calibration have been proposed [60, 78]. In this work we used the approach of Levinson and Thrun [60], which is an automatic method that requires neither hand-labelling nor a specific calibration scene.

There are two factors relating to the timing of data acquisition that must be taken into consideration. The first is that any clock offsets between the Velodyne and Ladybug sensors must be accounted for. Whilst this offset could potentially be optimised as part of the same method used for computing the sensor transform, in this work it was adjusted manually by synchronising events visible in both the lidar and camera data.

The second consideration is the differences in the sampling times of the lidar and camera systems. On our robot, the Velodyne is configured to rotate at 1200 RPM (20 Hz), whilst the Ladybug acquires images at approximately 5.25 Hz. Therefore, to project points into the camera frame, they must be transformed from the navigation solution (the pose of the vehicle in a global frame) at the time each point was measured to the navigation solution at the time the image was taken[2]. Additionally, we only associate Velodyne scans with the closest camera image in terms of timestamps, which means that only about 1 in 4 lidar scans have associated image data.

Figure 4.6 shows an example Velodyne data projected into the camera frame. Accurate calibration and time synchronisation ensures that laser points sampled from objects in 3D coordinates are projected onto those same objects in image coordinates.



(a) **(b)**

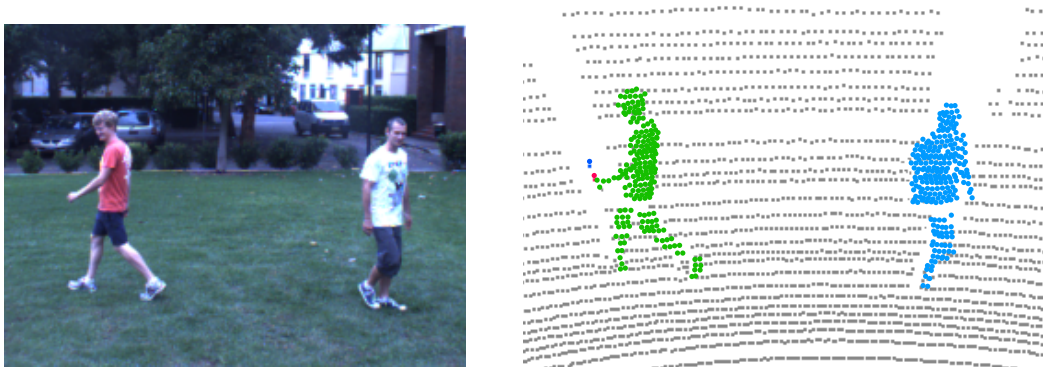
Figure 4.6 – (a) Segmented data from the Velodyne lidar. (b) The same segmented data projected into the Ladybug camera frame.

4.2.3.2 Image sampling

Once the intrinsic and extrinsic parameters and time synchronisation offsets have been accounted for, we have the ability to project lidar data into the camera image. However, to extract the camera data that corresponds to a lidar observation, more design decisions must be made.

[2]For efficiency during data projection we make the approximation that all lidar points are captured at the same (i.e. we do not compensate for motion). Since the dynamics of the sensor vehicle are slow with respect to the frame rate of the sensor, this is a reasonable approximation

Figure 4.7 shows an example of synchronised camera and lidar data. Processing of the lidar data has produced observations of two targets, shown in green and blue. To build appearance models of each target, we want to extract *all* the pixels in the image that correspond to each person, but using the image data corresponding to the projected Velodyne points provides a sparser sampling.



(a) **(b)**

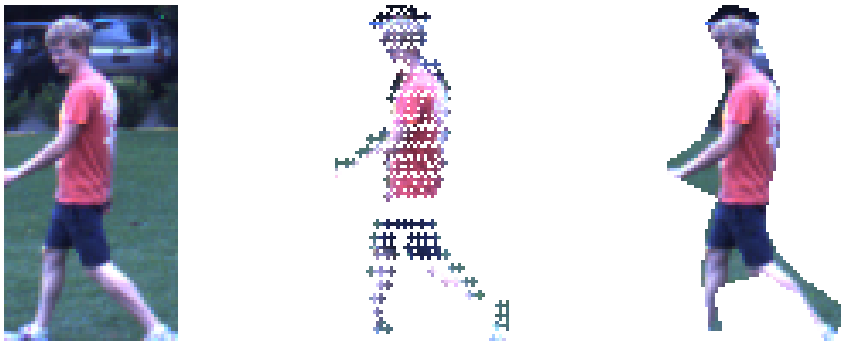
Figure 4.7 – Synchronised camera and lidar data

In this section we describe several alternatives for how to extract camera data for a given set of 3D lidar points projected into an image, which we describe as 'sampling'. An example of each option is shown in Figure 4.8, and the performance of these different methods will be evaluated in Section 4.4.

The first such method, *rect*, refers to the bounding rectangle of all the lidar points after projection into the image plane. Sampling all pixels within this rectangle results in an image patch that fully encompasses all the lidar returns, but also potentially includes a lot of the background (Figure 4.8a).

The sampling method *points* samples the colour values of the pixels within a given radius of each project laser return. For example *points-0* refers to the pixels which correspond exactly to projected Velodyne points, whilst *points-5* includes all image data within a 5 pixel radius. An example of the *points* sampling strategy is shown in Figure 4.8b.

Increasing the selection radius in the *points* method includes more of the image data from between projected lidar returns, but for data at the very edge of an object will



(a) (b) (c)

Figure 4.8 – Image sampling from Velodyne data (a) *rect*, (b) *points-1*, (c) *hull*

also sample additional background pixels. The sampling strategy, *hull*, is designed to include all image data from within the projected lidar segment, but maintain closely-cropped edges. The *hull* is computed as the alpha shape (or α-shape) [30] of the projected lidar points and is a generalisation of the convex hull which allows for some amount of concavity, depending on the parameter, α. Figure 4.8c shows the α-shape for the same observation as before.

The performance of the various sampling strategies on the example image of Figure 4.8 are shown in Table 4.2. Performance was measured with respect to a manually labelled image. Perfect precision would mean that the sampled pixels all correspond to the target, whilst perfect recall would mean that all pixels corresponding to the target were sampled. The F_1 score or "F-measure" is the harmonic mean of precision and recall and reflects overall performance.

	# pixels	precision	recall	F_1
rect	5049	0.31	0.92	0.47
points-0	209	0.80	0.10	0.17
points-1	948	0.79	0.43	0.56
points-3	2043	0.68	0.80	0.74
points-5	2739	0.57	0.91	0.70
hull	1762	0.76	0.77	0.77
manual	1728			

Table 4.2 – Performance of the image sampling strategies on the example observation shown in Figure 4.8.

This table demonstrates the potential gains from using a more sophisticated sampling strategy such as *hull*. Using the projected lidar points alone means that only 10% of the image pixels corresponding to the target were selected. We can increase this to over 90% by using *rect* or *points-5*, however these methods increase the number of background pixels selected and therefore achieve lower precision. The best balance of precision and recall (reflected by the F_1 score) is achieved by *hull*, closely followed by *points-3*. Note that this table is computed for one image only and is in intended for illustration purposes only. A more complete examination of how sampling affects appearance model quality is presented in Section 4.4.

4.3 Appearance models

The preceding section discussed how to extract features from 3D lidar and/or camera observations. In this section we use these features to create appearance models, and describe how these models are used to determine target identity.

Figure 4.9 shows the steps to compute an appearance model for a track section. Track sections are constructed by using data association to group segmented lidar observations together into tracks as discussed in Chapter 3. Using the methods described in Section 4.2.3, these observations can be augmented with camera data. From each observation, appearance features are computed and therefore every track section is associated with a set of features.

There is no limit to the duration of a track section. So long as an object does not interact with others or become occluded, the methods described in Chapter 3 can continue to track it indefinitely. This means that the number of observations in the track section, and hence the number of features extracted, grows at a constant rate. An appearance model, therefore, may choose to maintain only a representative set of features, or compress the feature set in other ways.

As well as a set of features, an appearance model also specifies how those features are used to compare the appearance of different track sections. During identity reasoning,

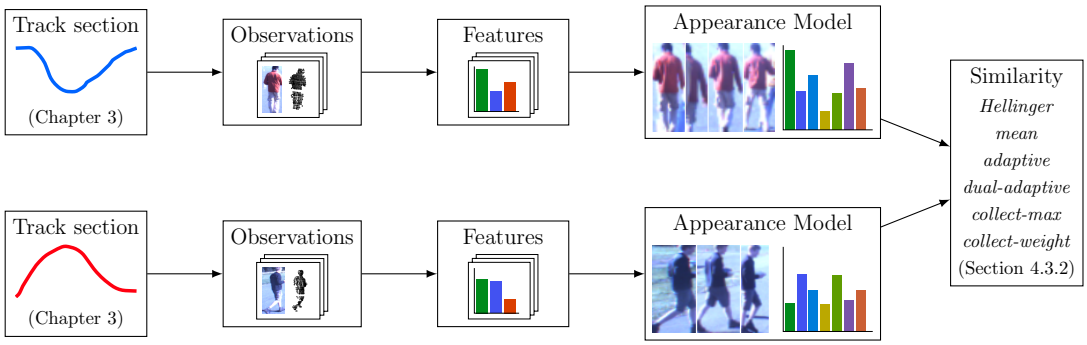


Figure 4.9 – Building appearance models from track sections. Each track section is made up of a set of observations. Features are computed for each observation independently, and these features combine to form an appearance model. Appearance models also specify a similarity measure (described in Section 4.3.2), which is used to compare pairs of track sections and determine if they share the same identity.

similarity measures are used to measure the likelihood that two appearance models represent the same target. In Section 4.3.1 we explain why appearance models that allow the computation of similarity measures are the best fit for event graph identity reasoning. We discuss the computation of these similarity measures in Section 4.3.2.

4.3.1 Using appearance models to determine identity

Appearance models are needed to resolve targets whose identities have become ambiguous during tracking. Chapter 3 discussed how this ambiguity occurs and proposed tracking methods that could detect such situations. Ambiguous situations are represented by group nodes in the event graph; when the group splits apart again and individual tracks are observed, appearance models can be applied.

Figure 4.10 shows an example merge/split event. Four track sections (T_1, T_2, T_3, T_4), merge to form a group, which then splits and we observe four new track sections (T_6, T_7, T_8, T_9). The task of determining the identity of a single output node (e.g. T_6) demonstrates why we described this as a self-supervised classification problem: the features corresponding to each input node provide training data to a model which can then be used to classify the output. Indeed, in this example it is possible to solve the problem using an off-the-shelf classification algorithm such as a multi-class SVM.

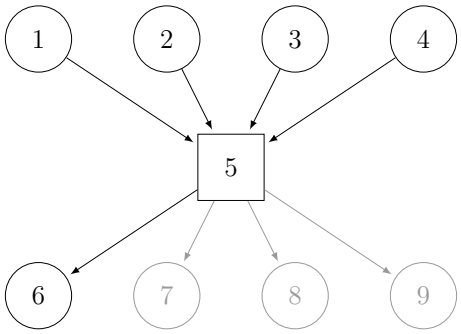


Figure 4.10 – The *independent* appearance evaluation problem. Four track sections (nodes 1 to 4) merge to form a group. When the objects move apart, four new track sections (6 to 9) split from the group node. The identity of an output node (e.g. node 6) is determined by comparing its appearance model to the appearance model of each input node. The other output nodes are not considered in this case.

However, in this thesis, we use appearance models that allow for the computation of similarity measures, rather than building discriminative classifiers. Some of the reasons for this are as follows.

1. In the discriminative case, a new classifier (SVM or otherwise) must be learnt for every interaction that occurs. That is, a classifier used to distinguish objects 1 and 3 must be completely re-trained to distinguish objects 2 and 3, and again to distinguish all three objects together. When using a similarity measure, models can be learnt individually for each track section and used to solve arbitrary interactions without retraining.

2. The event graph provides additional constraints that become difficult to incorporate into discriminative models such as an SVM. For example, when determining the identity of T_6 in the example from Figure 4.10, we also have appearance information from output nodes $T_7, ..., T_9$. We know there must be a one-to-one mapping from inputs to outputs and this constraint should be taken into account when determining node identities.

3. An output node from a merge/split even can also act as an input node to another event. Consider the graph shown in Figure 4.11. The identity of node 6 could be the same as node 1, 2 or 5, and if node 6 matches to nodes 1 or 2 then it

must also be the same object as node 4. Further to this, knowing something about the identity of node 3 also helps constrain the identity of nodes 6 and 7. In fact in this example, the only nodes that don't have any dependency are 3 and 5.

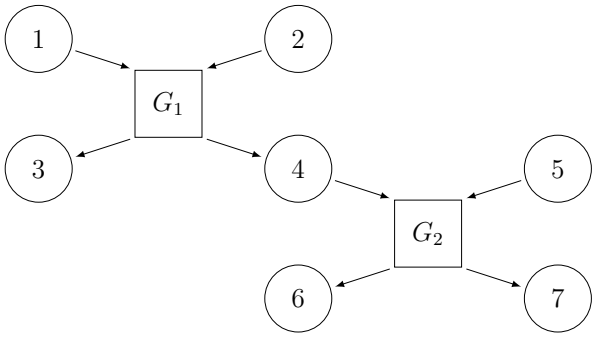


Figure 4.11 – A more complex interaction scenario. Tracks 1 and 2 merge to form a group, then split again. One of these tracks then merges with another track before splitting once more. To fully exploit the appearance information of each node requires modelling of the dependencies of the tracks.

For these reasons we use appearance models that allow for the computation of a pairwise similarity measure. Similarity measures give the likelihood that the corresponding track sections have the same identity. Given two track sections T_i and T_j, with associated appearance models, A_i and A_j, the likelihood that T_i is the same object as T_j, can be computed from the appearance models as:

$$P(T_i = T_j) = P(A_i, A_j) \approx s(A_i, A_j) \tag{4.1}$$

Where $s(A_i, A_j)$ is a similarity function in the range $[0, 1]$, the computation of which we will describe in the following section. The identity of a node is computed as the maximum likelihood estimate, therefore, for the example shown in Figure 4.10, the identity of T_6 is found as

$$T_6 = \underset{T_i}{\operatorname{argmax}} \left(P(T_6 = T_i) \right), \text{ where } T_i \in \{T_1, T_2, T_3, T_4\} \tag{4.2}$$

The equations above are for the *independent* case, where we consider the identity of T_6 with respect to the input nodes only. We can, however, choose to consider the identities of all output *jointly*, and impose the additional constraint that two output nodes cannot share the same identity. We assume that the appearance models themselves are independent, and therefore the joint likelihood of output node identities is the product of the appearance model similarities. The joint probability is this likelihood normalised over all possible identity configurations.

In Chapter 5 we discuss two frameworks which use these pairwise similarity models to solve target identities, whilst taking into account the constraints imposed by the event graph.

4.3.2 Similarity measures

In the previous section we assumed the availability of a pairwise similarity function between appearance models ($s(A_i, A_j)$). The form of this function depends on the features used in the appearance model.

Histogram features

The visual appearance of an object changes as it moves about in the scene due to changing view point, occlusions and varying illumination. An ideal model would capture this variability by combining the information from multiple features. Figure 4.12 shows example observations from a tracked object. The object interacted with another object midway through tracking, and therefore the trajectory is broken into track sections (A) and (B), separated by a group. A good appearance model will recognise that track sections (A) and (B) correspond to the same object, by reporting a high similarity between the models. The event graph will contain other nodes for the other targets in the scene, but here we focus on just this one object.

Normalised image histograms are probability density functions and there are numerous ways to compute similarity measures between them [18]. Appearance models,

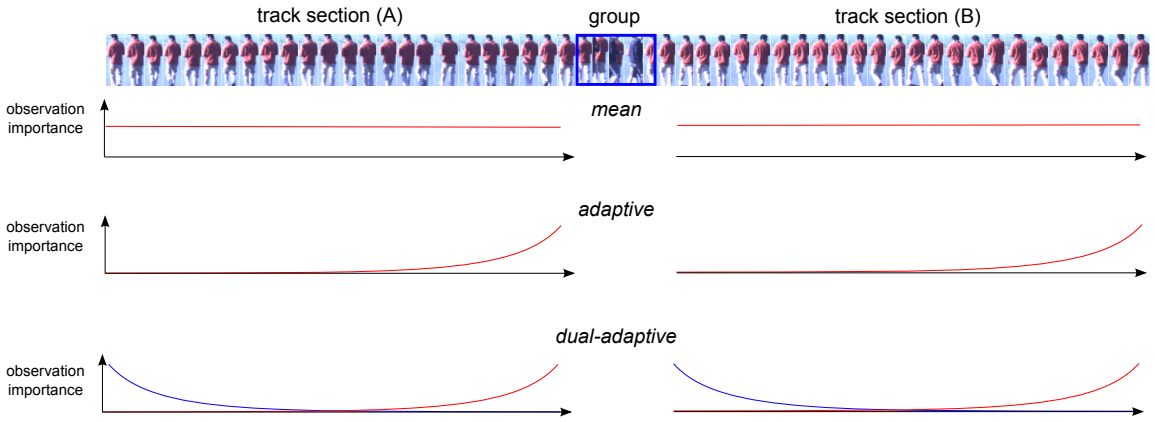


Figure 4.12 – Observation importance in different histogram strategies.

however, consist of many features computed from different observations of the same object so we need a way to compute the similarity between sets of features. Five methods for achieving this are evaluated in this work: *mean*, *adaptive*, *dual-adaptive*, *collect-max* and *collect-weight*.

One simple strategy is to simply take the *mean* histogram. That is, sum up all the histograms from the different observations within a track section and re-normalise (Equation 4.3, where H_M is the mean histogram, and H^k is the histogram computed from the observation at time k).

$$H_M = \frac{1}{N} \sum_{k=1}^{N} H^k \tag{4.3}$$

This method gives each observation equal weight, as represented in Figure 4.12. The similarity between two appearance models is then computed as the similarity of their mean histograms, which in turn is computed by a standard histogram similarity measure. In this work we use the histogram intersection (Equation 12 from [18]).

Another technique from the literature is to sequentially blend the histograms from each observation into the model [106, 116]. This "*adaptive*" histogram (H_A) is computed from observations up to time k as

$$H_A^k = (1 - \alpha) H_A^{k-1} + \alpha H^k \tag{4.4}$$

The parameter α affects the adaptation rate of the histogram. As for the mean histogram, the similarity between adaptive histograms is calculated using histogram intersection. This approach has the advantage of adapting to the changing appearance of a tracked object, whilst having constant storage requirements regardless of track length. As a result of blending, the importance of an individual observation to the appearance model decays over time with the most recent observations being the most important.

It is likely, however, that the observations immediately after a split event are most similar to the observations of the object before the merge, and should perhaps be given higher weight. As a track section increases in length it is likely that the appearance will drift from what it was at the time of the interaction and the *adaptive* adaptation strategy may reduce the performance of the appearance model.

We introduce therefore, a new method, referred to as *dual-adaptive*. It creates, in effect, two adaptive histograms, one of which gives *initial* observations the highest weight. This results in the effect shown in red and blue lines in Figure 4.12. When comparing the similarity of two appearance models, they are ordered temporally. The forwards adaptive histogram (red) of the first track section is compared to the reversed adaptive histogram (blue) of the second. Observations closest (in terms of time) to the group event will therefore have the highest weight in the models, which may result in better performance.

Finally, we propose two methods which collect representative histograms from the set of appearance features, rather than blending them all together with some weight. Each track section maintains a set of histograms $\mathbf{H}$, and associated counts, c. Initially $\mathbf{H}$ is empty. For an observation at time k, the histogram H^k is computed and compared to the current set of histograms (by the histogram intersection measure as before). If the maximum of these similarities is above a threshold, then that observation is deemed to be adequately represented by the current set, and the counter, c, corresponding to the most similar histogram is incremented. Otherwise, the histogram H^k is added to the set $\mathbf{H}$ and the corresponding counter is set to one.

When computing the similarity between appearance models A_1 and A_2, we can take

the maximum similarity between their two sets of representative histograms ($\mathbf{H_1}$ and $\mathbf{H_2}$). This method is referred to in the experimental section as *collect-max*.

$$s(A_1, A_2) = \max_{H_1 \in \mathbf{H_1}} \left(\max_{H_2 \in \mathbf{H_2}} s(H_1, H_2) \right) \tag{4.5}$$

Alternatively, we can take the sum of the product of histograms, each weighted by their normalised counts. This method is referred to as *collect-weight* and is calculated as shown below, where $H_{1,i}$ refers to the i^{th} histogram in the set $\mathbf{H_1}$, and $c_{1,i}$ is its associated 'count'.

$$s(A_1, A_2) = \sum_{i=1}^{i=|\mathbf{H_1}|} \sum_{j=1}^{j=|\mathbf{H_2}|} c_{1,i} \cdot c_{2,j} \cdot s(H_{1,i}, H_{2,j}) \tag{4.6}$$

Lidar features

For the height and radius features, we assume that the measurements are Gaussian distributed around the true values.[3] From the set of observations in a track section, we compute the mean and variance, and compute the similarity via the Hellinger Distance [41].

For spin images the computation is more involved. The spin image correlation [47] provides a similarity measure between pairs of spin images, but as was the case for the appearance models, we need to measure the similarity between *sets* of features. We also need to reduce the number of features in the model; computing 20 spin images per observation over a 10 second track section, with the Velodyne spinning at 20Hz, would result in a total of 4000 spin images. To more efficiently compare tracks, we use K-means clustering to reduce the feature set to 50 representative images.

To compute the similarity between two sets of spin images (denoted $\mathbf{S_1}$ and $\mathbf{S_2}$), by matching each spin image in $\mathbf{S_1}$ to its closest neighbour in $\mathbf{S_2}$ and computing the average correlation. In the equation shown below $S_{1,i}$ represents the i^{th} spin image in

[3] Sampling effects due to the angular resolution of the laser, and partially occluded objects can invalidate this assumption.

the set $\mathbf{S_1}$ and R is the spin image correlation coefficient defined by [47]. Since spin image correlation ranges from 1 (completely correlated) to -1 (anti-correlated), the result is divided by 2 and 0.5 added to give a similarity value in the range $[0, 1]$.

$$s(A_1, A_2) = \left(\frac{1}{2|\mathbf{S_1}|} \sum_{i=1}^{|\mathbf{S_1}|} \max_{j \in \{1..|\mathbf{S_2}|\}} R(S_{1,i}, S_{2,j}) \right) + 0.5 \tag{4.7}$$

4.4 Evaluation of appearance models

In this section we evaluate appearance models by testing their ability to recognise previously-tracked targets. Manually labelled data is used to create "interaction scenarios" and appearance models are used to infer the identities of the targets involved. This section describes the definition of interaction scenarios and introduces the datasets used for evaluation, followed by experimental results and analysis.

4.4.1 Methodology

When evaluating appearance models, we wish to answer questions of the form "if person A interacted with person B, would the model be able to recognise who is who after they separate?". Interactions cause an object's trajectory to be divided into "track sections", and the ability of an appearance model to determine identity can be tested on labelled interactions.

Whilst in a given dataset, objects may only interact once, or not at all, we can simulate an arbitrary number of interactions by manually dividing trajectories into a number of track sections. Each track section is simply a set of observations, and an interaction experiment selects some sections as inputs, others as outputs, and tests if the appearance model can correctly link inputs to outputs. To isolate the performance of appearance modelling from segmentation and tracking we use as input to this process manually labelled trajectories obtained through the method described in Section 3.2.3.

For example, manually labelling the dataset MERGE4-B gives us four trajectories. If each trajectory is split in three, we obtain a total of 12 sections, pictured in Figure 4.13. We will refer to these track sections as 1A, 1B, ..., 4C. We can then experiment with arbitrary hypothetical interactions. For example, in one experiment, we test what would happen if target 1 and 3 interact. Sections of each trajectory (say 1A and 3B) are used to train appearance models. Then, we want to examine the behaviour of the appearance models if target 1 is observed again, and so a different section from target 1's trajectory (e.g. 1C) is selected. This hypothetical interaction is depicted in Figure 4.14a. If the model correctly groups the section from target 1 together then the test is successful. A second example, depicting another valid hypothetical test situation is depicted in Figure 4.14b.

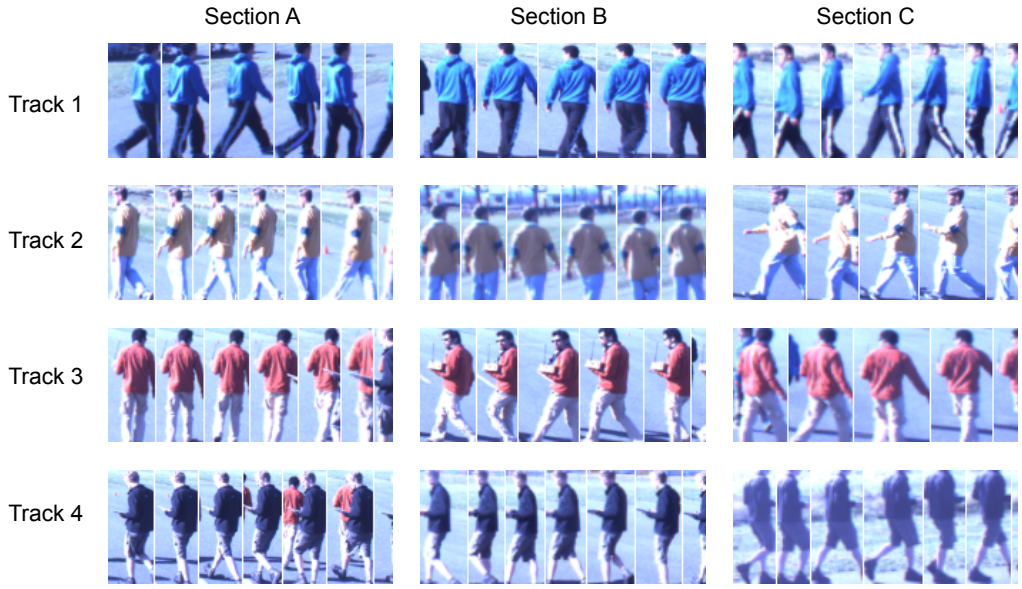


Figure 4.13 – Example track sections extracted from labelled trajectories.

This method of testing treats the observed output node independently of any others and so corresponds to Figure 4.10, above. We average the accuracy of the appearance models across all possible interactions. For a given dataset with N labelled trajectories, each divided into M sections, where we simulate interactions between T of those targets ($T <= N$), the total number of simulations, S, is given by Equation 4.8.

Figure 4.14 – Example of simulated interactions, using the data from Figure 4.13. Both (a) and (b) simulate interaction of the same objects, but use different track sections and test a different outcome. This is for the "independent" appearance reasoning case, where we only consider the appearance model of one of the output nodes.

$$S = \binom{N}{T} M^T (M-1)T \tag{4.8}$$

In Section 4.3.1 we described how it is possible to jointly solve for the identity of multiple nodes which split from a group. The number of simulations to perform in this case to exhaustively try all combinations is given by Equation 4.9.

$$S = \binom{N}{T} [M(M-1)]^T \tag{4.9}$$

These numbers grow very quickly in the size of N, M and T. For example in a dataset with 4 labelled objects split into 10 section, there are 48600 interactions to simulate between 2 objects. This jumps to 2916000 for 3 objects, making the simulation of all interactions intractable, so for large combinatorial sets we may randomly sample from the possible interactions. The results below are across *all* possible interactions unless explicitly noted.

In the experimental results that follow, trajectories were split into 5 second sections. For trajectories with a much longer duration than this, the time separation between the track sections in an interaction scenario will vary. Some interaction scenarios will include track sections which are immediately subsequent to one another, whilst others

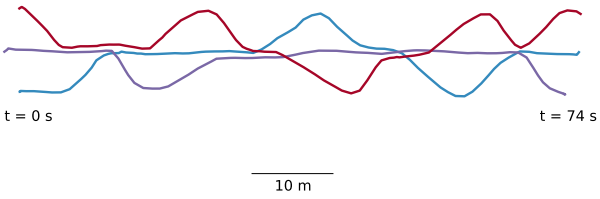
will be separated by some multiple of 5 seconds. This variable separation simulates a group node or occlusion of this same duration.

4.4.2 Dataset

For the evaluation of appearance models, we make use of some of the experimental data presented in Chapter 3, which was gathered with the ACFR research platform "Shrimp" (Figure 3.12). Whilst the experiments in Chapter 3 required only navigation and lidar data, here we also make use of the Ladybug-3 omnidirectional camera and thermal IR camera.

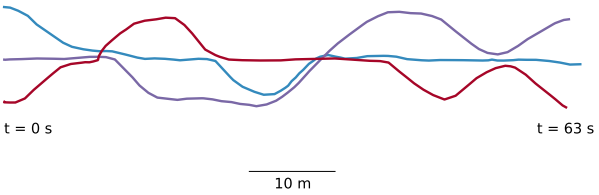
In addition to the scenarios selected from those already presented in Figure 3.13 and Figure 3.14, we include SAME-SHIRT-A and SAME-SHIRT-B, 3-person interaction experiments in which two of the pedestrians are wearing similar-coloured shirts. The goal of these scenarios was to see if the appearance models were able to discriminate targets wearing similar colours, however, it as can be seen from the image in Figure 4.15b that the appearance of the two targets is still fairly distinct.

We also introduce a city dataset. Whilst the Marulan datasets comprise controlled interaction scenarios of varying complexity, the city datasets were captured from driving throughout the city of Sydney and therefore represent realistic appearance information for an urban scene. Tracks were extracted from two scenes, a pedestrian crossing at an intersection in the CBD and a stretch of road near the Sydney Opera House. INTERSECTION contains 14 labelled tracks and a combined 3118 observations, manually labelled using the same procedure as described in Section 3.2.3. For the OPERA-HOUSE dataset, the **split-cost** tracking method described in Chapter 3 was used to initialise labelling. Tracks less than 5 seconds in duration were removed, and the remaining tracks were manually verified and corrected as required using Blender. A total of 111 tracks were labelled in this manner, comprising 32088 observations. Some example images taken during the Sydney city dataset are shown in Figure 4.16.



(a) SAME-SHIRT-A

(b)



(c) SAME-SHIRT-B

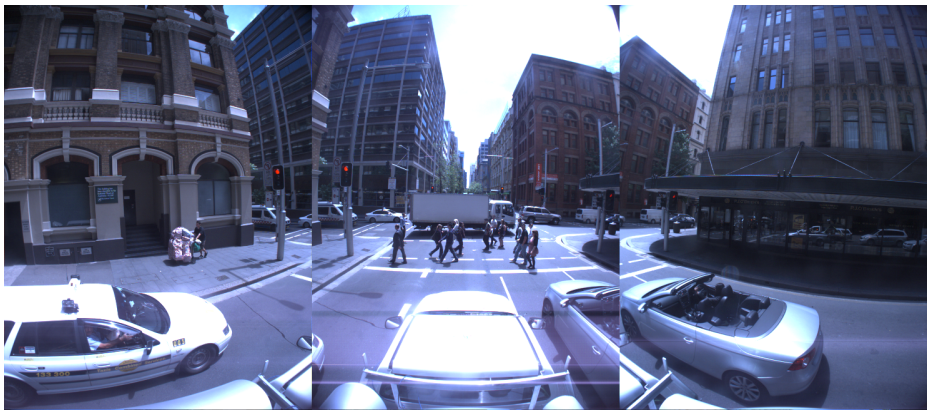
Figure 4.15 – Labelled trajectories from the SAME-SHIRT-A and SAME-SHIRT-B experiments, along with an image taken by the forward-facing camera during the trials.

4.4.3 Results and Discussion

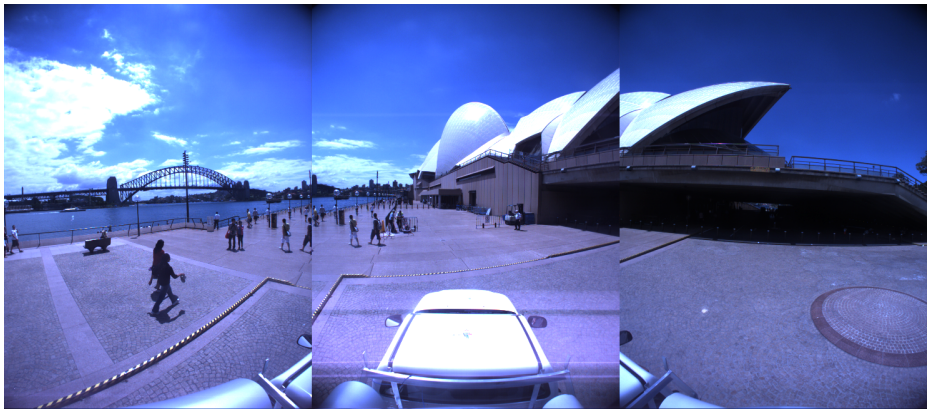
The results and discussion presented in this section are divided according to sensor. We first evaluate appearance features for lidar data, followed by colour camera data and finally thermal IR.

4.4.3.1 Lidar

Figure 4.17 shows the performance of lidar based appearance models from Section 4.2.1 on the interaction simulations. The average performance across all datasets is summarised in Table 4.3. The experiments are simulations of two interacting objects, so a random assignment would have an accuracy of 0.5. Given this, an appearance model which makes use of observation radius does not do that much better than chance, however height does appear to be discriminative, successfully resolving almost 80% of interactions. Using height and radius jointly results in performance about as good as height, indicating that the radius measurement is not greatly influencing the



(a)



(b)

Figure 4.16 – Images from the intersection and opera house dataset. The robot "Shrimp" was mounted on the back of a utility vehicle, which was driven around the Sydney CBD.

model.

The one experiment where radius does have an impact on performance is MERGE-BBPP, which was deliberately chosen because it includes objects which are geometrically quite different. The scenario contains four objects: two cyclists and two pedestrians.

Figure 4.18 shows the radius and height statistics of the four objects (in different colours) for each of the track sections in MERGE-BBPP. Blue and yellow (tracks 1 and 2) represent cyclists and red and black (tracks 3 and 4) are pedestrians. It is evident that the observations of bikes have an (unsurprisingly) larger radius and so the data in Figure 4.18a can be separated into two classes. However, radius does not

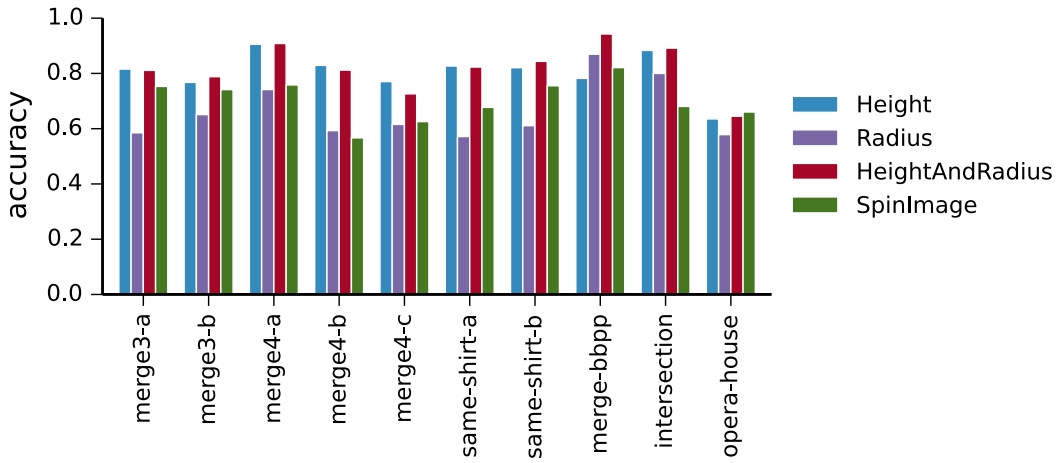


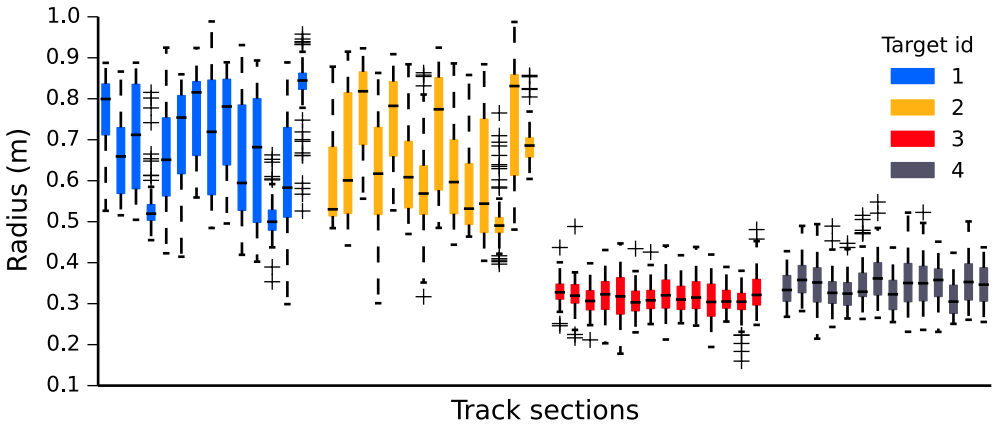
Figure 4.17 – Classification performance using lidar features. Height is a more useful feature than radius when tracking pedestrians. The MERGE-BBPP experiment (shown in Figure 3.14) involves two bicycle riders (b) and two pedestrians (p).

Appearance Model	Accuracy
Height	0.793
Radius	0.655
Height & Radius	0.810
Spin Image	0.721

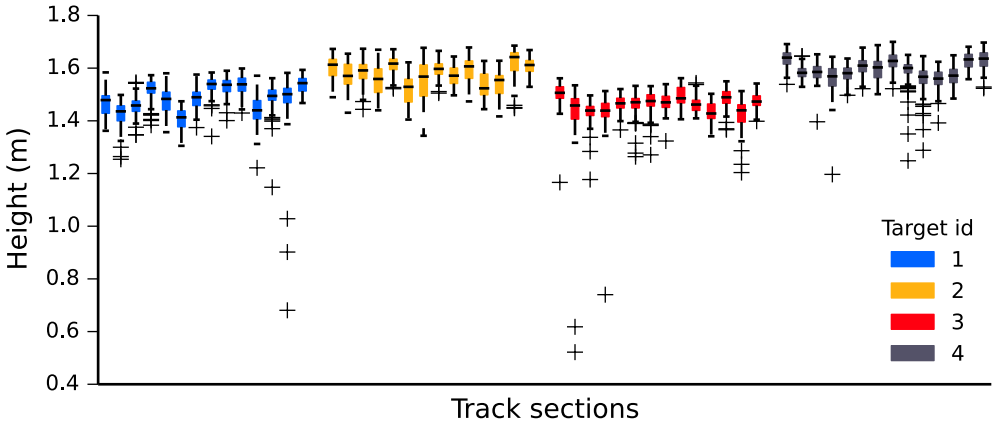
Table 4.3 – Average classification performance using Velodyne-based appearance models across all datasets.

help to distinguish the cyclists from one another, nor does it help separate the two pedestrians. Figure 4.18b shows the height statistics for the four objects. Targets 2 and 4 are marginally taller than 1 and 3. Serendipitously, though there are pairs of people with approximately the same height, in each case one was riding and the other walking. Therefore height and radius both split the problem along different dimensions, and the resulting classifier shows an increase in performance over both the individuals. This is a very convenient dataset, in that if targets 2 and 4 were both riding bicycles the height would likely give us no useful information, but does illustrate that different geometric features can be combined without degrading the performance of the more informative feature used individually.

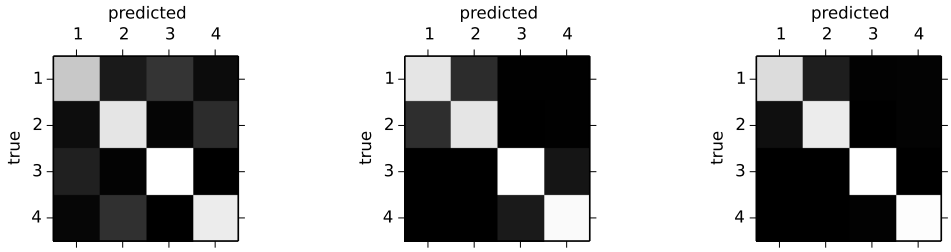
The results for the spin image appearance models are also shown in Figure 4.17. On



(a)



(b)



(c) Height (d) Radius (e) Combined

Figure 4.18 – Radius (a) and height (b) of target track sections in the MERGE-BBPP dataset. The bikes (targets 1 and 2, shown in blue and yellow) are separable from the pedestrians (targets 3 and 4, shown in red and black) by radius, whilst height can separate the targets 2 and 4 from 1 and 3. This results in the combined *HeightAndRadius* feature having significantly improved performance on this dataset in Figure 4.17. (c-e) show the confusion matrices for the *Height*, *Radius* and *HeightAndRadius* features.

average, spin images did not perform as well as the height feature, but did perform significantly better than chance. Spin images capture object shape, and therefore would be expected to show the best performance when separating different types of objects. This is reflected by the fact that the spin image appearance model receive its highest score on the MERGE-BBPP dataset.

Whilst spin images capture the height of an observation to some extent, an appearance model which incorporates all of the features described above may show increased performance, and in general the topic of combining different types of appearance features would be an interesting avenue for future research.

4.4.3.2 Camera

In this chapter we have discussed three main design decisions for camera-based appearance models: the image sampling strategy, choice of feature, and definition of the similarity measure. Table 4.4 lists the various implementation options presented. The implementation options for each decision are largely independent, in that we can choose to combine any sampling strategy, feature and similarity measure together. The exception to this are *Spatiogram-RGB* features, which, because of the way they represent positional information, cannot readily be added or averaged and therefore are only compatible with the *collect-max* and *collect-weight* similarity measures. Visualising the possible combinations of all these parameters simultaneously is difficult, so in this section we examine projections of this parameter space.

Sampling (Section 4.2.3.2)	**Feature** (Section 4.2.2)	**Similarity** (Section 4.3.2)
rect	*Hist-HSV*	*mean*
points-0	*SplitHist-HSV*	*adaptive*
...	*Hist-RGB*	*dual-adaptive*
points-n	*Spatiogram-RGB*	*collect-max*
hull		*collect-weight*

Table 4.4 – The three main parameters for the camera-based appearance models presented in this chapter, shown with the implementations options discussed.

Figure 4.19 shows the accuracy of the appearance models at the task determining target identity after a two-object interaction. This is the *average* performance across all features, sampling strategies and similarity measures and the results will be broken down further in subsequent figures. The average performance across all datasets was 0.95 indicating that the models could on average resolve 95% of two-object interactions.

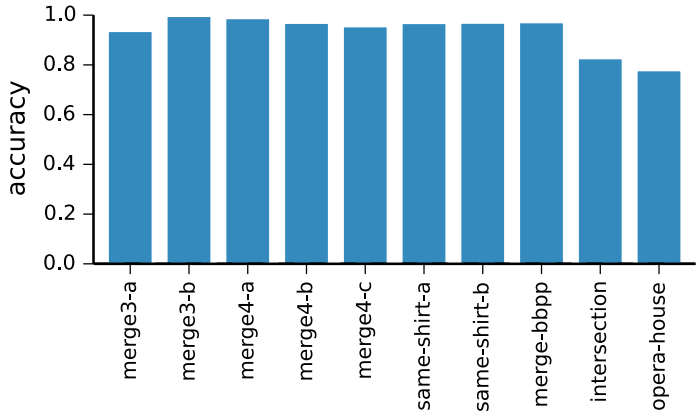


Figure 4.19 – Identity solving accuracy using colour image data from the Ladybug camera, averaged across all parameters presented in Table 4.4.

We now consider the different measures for computing the similarity between sets of features, discussed in Section 4.3.2. Figure 4.20 shows the accuracy of appearance models using the various strategies averaged over the other parameters. The accuracies all appear fairly similar except for a peak for *dual-adaptive* on INTERSECTION. The reason for this peak is to do with the way the track sections are extracted. As described earlier, trajectories are broken up into 5 second pieces which are used for model evaluation. In INTERSECTION most trajectories are quite short and many of the trajectories are divided into only two parts. This means that the testing data is sampled immediately after the training data from within a trajectory. The *dual-adaptive* algorithm, which uses forward and reverse histograms, will weight the observations closest to the training/testing split the highest and therefore achieve better performance. This result is artificially high, because in a real use-case some kind of interaction occurs between the track sections, and the appearance may change more significantly in this time.

Looking at the average score of the methods across all datasets (Table 4.5) shows an advantage to *mean*. However, the differences are only minor, and for the remainder of the experiments in this work we use *collect-weight*. We choose this method because it has comparable performance to the others and the way it collects exemplar histograms is helpful for visualisation and diagnostics.

	accuracy
adaptive	0.900
collect-max	0.939
collect-weight	0.937
dual-adaptive	0.950
mean	0.945

Table 4.5 – Accuracy for the different similarity measures averaged across all datasets.

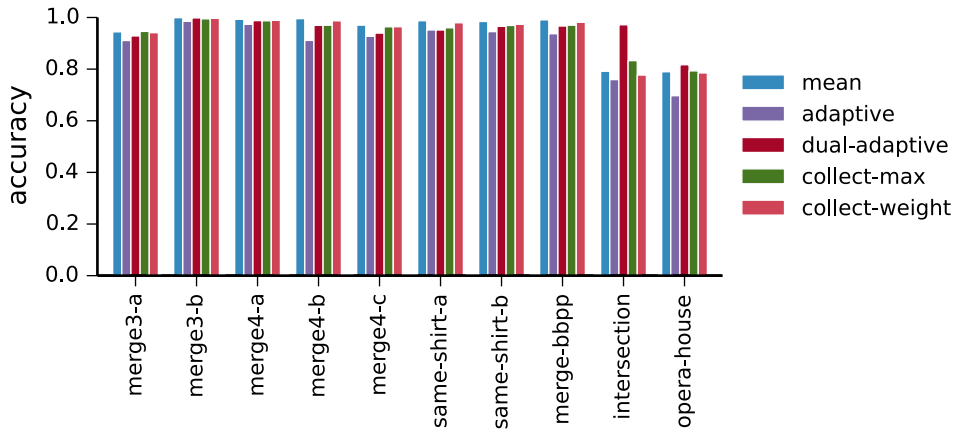


Figure 4.20 – Appearance model performance when using different similarity measures.

The next parameter we analyse is the feature computed from each observation, as described in Section 4.2.2. The results are shown in Figure 4.21, and the mean performance across all datasets is shown in Table 4.6. As mentioned, this and the following results are computed using the *collect-weight* similarity measure.

In this set of experiments, *SplitHist-HSV*, which uses two histograms to capture the colour distribution of the top and bottom halves of the image independently had the best performance. However, *Spatiogram-RGB* which captures spatial colour

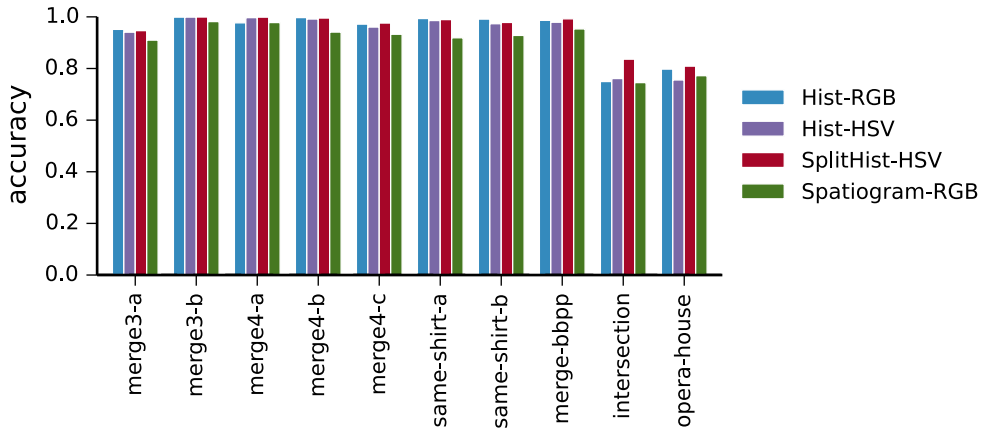


Figure 4.21 – Effect of histogram choice on appearance model accuracy (Ladybug data).

	accuracy
Hist-HSV	0.934
Hist-RGB	0.942
Spatiogram-RGB	0.906
SplitHist-HSV	0.952

Table 4.6 – Accuracy for the different colour image features averaged across all datasets.

distribution in more detail actually has the worst performance of all the methods. This suggests that *Spatiogram-RGB* may be over fitting the data, and that more colour variation occurs during tracking than what is predicted by the model. The advantage of *SplitHist-HSV* was more pronounced on the city scenes, where there were many more targets, and having the additional spatial information added to the ability of the method to discriminate targets. We continue to use the best feature, *SplitHist-HSV*, for the remainder of the experiments in this section.

Figure 4.22 shows the effects of image sampling on appearance model accuracy, given the choices of *SplitHist-HSV* as feature and *collect-weight* as similarity measure. Recalling Figure 4.8, *rect* samples image data from the rectangular bounding box of the projected lidar points. This bounding box often includes background pixels, so as expected it has the worst accuracy in this set of experiments. Sampling from the

α-shape hull of the points improves performance as the majority of the background can be excluded whilst preserving foreground pixels.

Interestingly, the *points-0* strategy, which samples image data at the locations of the projected lidar points does about as well as *hull*, showing that even though less image data is sampled (as was shown in Table 4.2) it contains sufficient information for identity discrimination.

Sampling all data within a 1-pixel radius of the projected Velodyne points (*points-1*) improves performance slightly, and then there is a gradual fall off as this radius increases. This effect reflects the results from Table 4.2. Increasing the sampling radius improves the proportion of foreground pixels selected, but also increases the number of background pixels included into the model. There is a trade-off between the two, and at some point increasing the sampling radius will decrease overall performance. From the results shown in Table 4.7, this occurs at a radius of somewhere between 1 and 3 pixels.

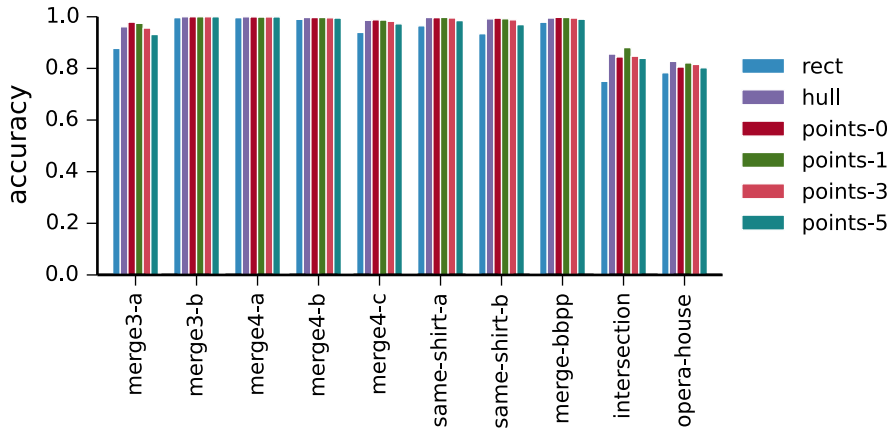


Figure 4.22 – Effect of image sampling on appearance model accuracy (Ladybug data).

The experiments above explore part of the parameter space that an engineer must consider when developing a system that combines lidar and camera data to extract appearance information. Other properties of the appearance modelling problem are determined by the data, so whilst one can not necessarily control these effects, understanding them will aid in the design of reliable appearance models.

	accuracy
hull	0.961
rect	0.921
points-0	0.960
points-1	0.965
points-3	0.958
points-5	0.948

Table 4.7 – Accuracy for the image sampling methods averaged across all datasets.

One parameter over which the designer has little control is the amount of time between observing a target, and it interacting with another. A shorter time limits the number of observations available to train the appearance model, which may limit its ability to recognise a target after an interaction. Figure 4.23 shows the results of an experiment which limited the number of camera observations used to train the appearance model. As expected, the performance is worst when training with only one observation and improves as the number increases. This improvement tails off as the number of observations increases, suggesting that for the datasets tested, around 15 observations are sufficient to learn a good appearance model of the target. Note that for the Sydney city datasets there were an insufficient number of tracks with more than 15 camera samples to calculate this trend. Therefore, only the results from the Marulan experiments are presented. The mean performance across these experiments is shown in Table 4.8.

	accuracy
1	0.945
2	0.956
5	0.978
10	0.983
15	0.988
20	0.993
25	0.996
no limit	0.996

Table 4.8 – Average accuracy of the camera based appearance models when trained with a limited number of image samples

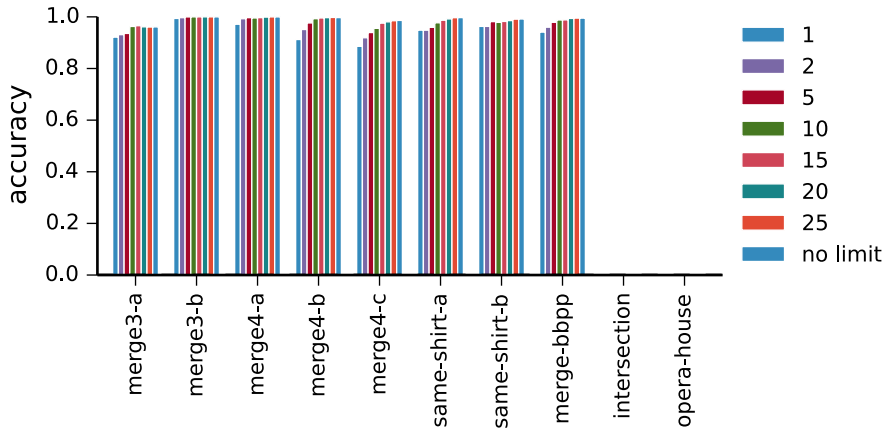


Figure 4.23 – Comparison of appearance model performance as the number of images used to train the model changes.

A second important factor from the data is how many targets interact at a given time. All the experiments so far show the appearance model recognition accuracy when two random targets interact. In this case, represented previously in Figure 4.10, the job of the appearance model is to determine which input track the output is most similar to. As more targets interact, the appearance model has to discriminate between more and more targets, and we would expect the accuracy of the system to decrease. This is confirmed by the experimental results shown in Figure 4.24. As the number of inputs (shown on the x-axis) increases, the ability of the appearance model to resolve them decreases. This figure is shown only for the INTERSECTION experiment, but a similar trend is observed across the other datasets.

Figure 4.24 also shows the difference between reasoning jointly about object identities versus determining the identity of each output node independently. If we observe multiple output nodes splitting from a group node, then we can make use of the knowledge that there must be a one-to-one mapping between inputs and outputs. Computing the joint likelihoods greatly improves classification performance.

This increase in performance motivates the use of a framework that can reason about appearance identity jointly, even in situations where there is not a simple mapping of inputs to outputs (like the example shown in Figure 4.11). Such a framework will be discussed in the following chapter.

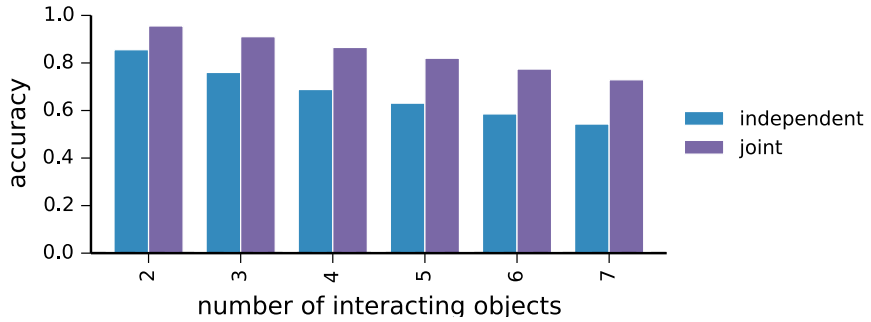


Figure 4.24 – Comparison of "independent" and "joint" reasoning using colour appearance models from the Ladybug camera on the INTERSECTION dataset. As the number of targets involved in an interaction increases, the appearance model accuracy decreases. Joint reasoning shows a distinct performance advantage.

4.4.3.3 Thermal IR

We also evaluated the accuracy of appearance models based upon a thermal IR sensor. From the example image in Figure 4.5 it is evident that a human may have trouble determining the identities of pedestrians based solely on IR data. The results in Figure 4.25 show that the appearance models described in this chapter also struggle to identify the individuals, and identity recognition accuracy was not much greater than random chance. This is with the exception of the SAME-SHIRT-A experiment.

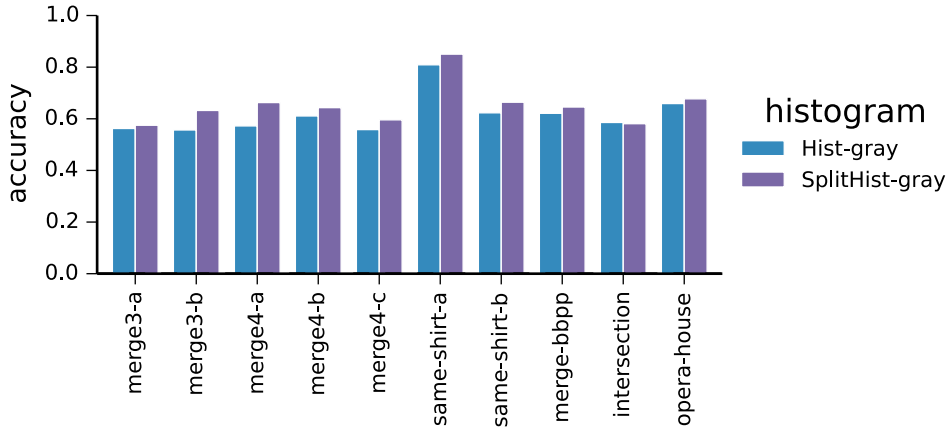
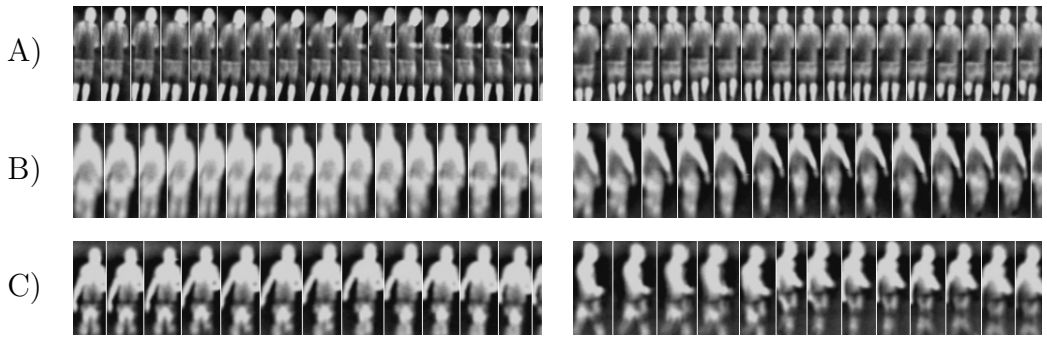


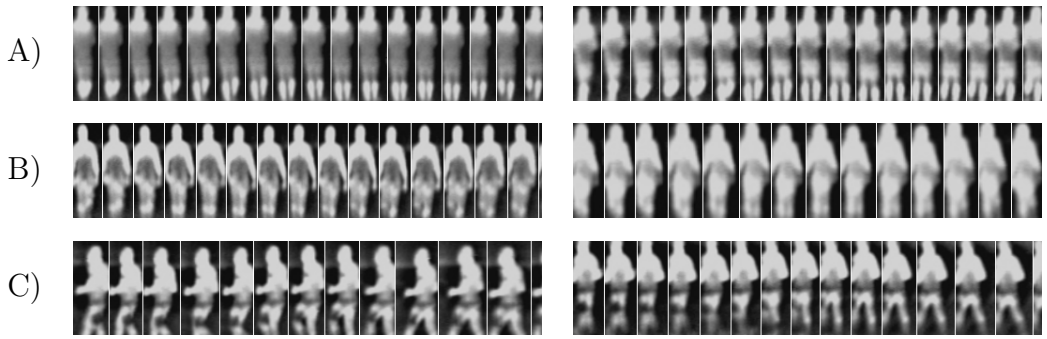
Figure 4.25 – Identity solving accuracy using IR data

The reason for the increased performance on this one experiment is difficult to determine. Figure 4.26 shows some track sections extracted from both the SAME-SHIRT-A

and SAME-SHIRT-B experiments, which involve the same pedestrian targets. The data for SAME-SHIRT-B was recorded immediately after SAME-SHIRT-A. Each row of the figure corresponds to one of the three pedestrians (A, B and C), and shows two of the track sections extracted for appearance model evaluation. The figure shows that the difference between target A and targets B and C is slightly more pronounced in the SAME-SHIRT-A experiment, which could explain the increased performance on this dataset in Figure 4.25.



(a) SAME-SHIRT-A

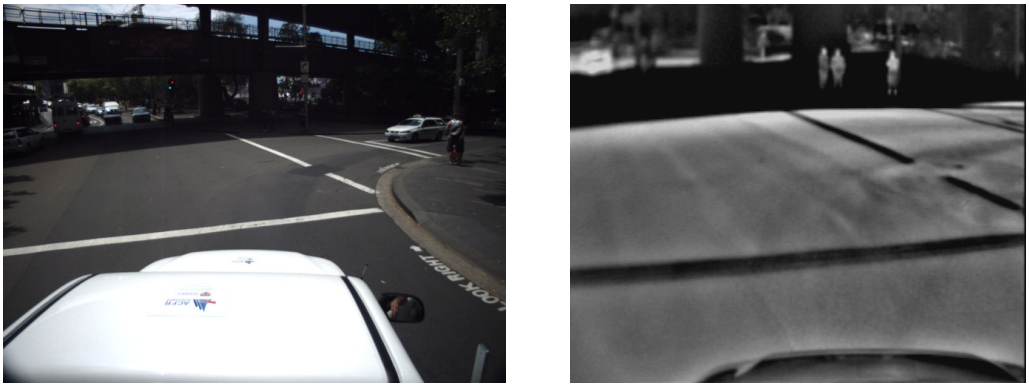


(b) SAME-SHIRT-B

Figure 4.26 – Thermal IR data from the SAME-SHIRT experiments. Each row corresponds to one target, and shows two different sections extracted from the trajectory. Target A is slightly more distinctive in the SAME-SHIRT-A experiment, which might explain the higher accuracy in Figure 4.25.

Whilst a more sensitive IR camera may be able to read the temperatures of targets accurately enough to be able to identify the differences between them, the sensor used in this thesis did not have the ability to do so. Thermal IR data offers advantages, such the ability to detect pedestrians in situations where colour vision might fail (an example of this is shown in Figure 4.27), but for the type of identity tracking problems

discussed in this thesis, colour images were found to provide more useful information.



(a) Colour camera image **(b)** Thermal IR image

Figure 4.27 – Synchronised thermal IR and colour camera images captured in an urban environment. The pedestrians, shadowed by an overpass, are difficult to see in the camera data (a), but stand out in thermal IR data (b)

4.5 Summary

In this chapter we examined the concept of appearance models and how they can be used to represent identity information extracted from tracked targets. We covered possible sources of identity information from 3D lidar and discussed using histograms to represent colour information.

Section 4.2.3 studied methods to combine appearance information from 3D lidar and camera data, and included a discussion of the issues of calibration and time synchronisation. We considered the problem of sampling high resolution image data based on the projection of much sparser Velodyne data points and proposed the use of the α-shape for computing the region within which to sample. The utility of this method was demonstrated in the experiments of Section 4.4.

Experiments showed that lidar data often provides sufficient information to determine target identities, but that thermal IR does not. We demonstrated that, for colour images, representing the top and bottom half of observations as independent histograms captures additional useful information, but that the use of spatiograms

decreases recognition performance. Additionally, methods for computing a similarity measure between appearance models were evaluated.

Appearance modelling parameters relevant to a system designer were explored in extensive experimentation using real-world data gathered both under controlled conditions and in a city environment, using a testing scheme that simulated arbitrary object interactions from labelled data. We demonstrated empirically that increasing numbers of interacting objects make appearance methods more likely to fail, but that significant performance gains can be achieved by jointly computing the identity of sets of outputs.

Appearance models designed with respect to the lessons of this chapter will be used in Chapter 5 to solve the 3D lidar event graphs and provide robust identity tracking in complex scenes.

Chapter 5

Robust identity tracking

In the preceding chapters we have described the components of a perception system for robust identity tracking of dynamic targets. Chapter 3 studied multi-target tracking using 3D lidar data, resulting in the proposal of a grouping strategy and the event graph framework shown in Figure 5.1. We covered the aspects of the framework which relate to lidar data, namely segmentation, track extraction and graph construction.

Chapter 4 considered appearance models, and we analysed their ability to resolve object interactions independent of any graph structure. We discussed the parts of the event graph framework relating to camera data: projection and sampling and appearance model generation.

In this chapter we describe the remaining module, "identity reasoning". This is the process of taking the appearance models, learnt according to the methods discussed in Chapter 4, and using them to reason about the identities of nodes in the graph.

Section 5.1 describes two different methods for solving identities in the event graph; a greedy approach that links the most similar graph nodes together (**HGraph**) and a global approach that considers fully the dependencies of each node in the graph (**BNet**). We analyse the performance of these methods and demonstrate that they enable robust identity tracking even in situations involving complex interactions between objects.

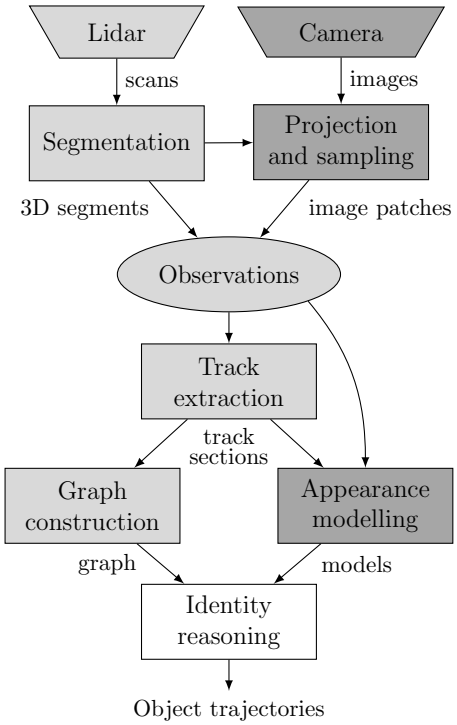


Figure 5.1 – Event-graph tracking framework. Parts of the framework discussed in Chapter 3 are shaded in light grey and parts discussed in Chapter 4 are shaded in a darker grey. The remaining module, "identity reasoning" is the subject of this chapter.

Section 5.2 describes experiments on data collected in a city environment. A dataset of this size is infeasible to manually label, so we present a detailed analysis of two smaller sections of the data. Additionally, we present tracking statistics from experiments as a whole, and some conclusions drawn from the data.

5.1 Solving the event graph

Solving the event graph, referred to in this section as 'appearance reasoning' or 'graph inference', is the process of taking an event graph and determining which objects are represented by each node. When nodes merge together to form a group, then the group contains all of the objects represented by its parent nodes, however when a group splits, appearance information must be used to perform data association. The 'path' of an object through an event graph is the sequence of nodes in which that

object appeared. An example of a solved event graph is shown in Figure 5.2. The 'path' of node 12 was determined to be $\{2, 4, 5, 9, 10, 12\}$.

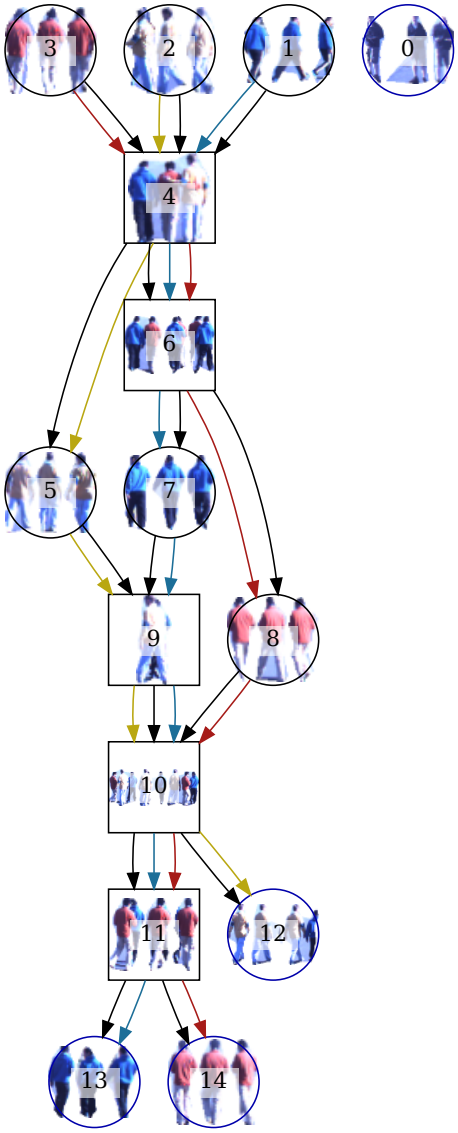


Figure 5.2 – Example event graph solution. Following the coloured lines shows the *path* of each object through the graph.

In this section we consider two different ways to solve the event graph. The first (**HGraph**) is a greedy method that has the advantage of being fast to compute, whilst the second (**BNet**) provides a close approximation to the global optimal solution.

The common input to the methods described below is the combined results of the work described in Chapter 3 and Chapter 4, that is, an event graph where each node

(i) is associated with an appearance model (A_i). From the appearance models we are able to calculate the similarity between nodes $s(A_i, A_j)$. So long as this similarity can be calculated then the exact nature of the appearance model is not important. The abilities of different models were explored in Chapter 4, and the same set of models will be used in the experimental results that follow in Section 5.1.3.

5.1.1 Hypothesis Graph (HGraph)

The first approach we describe was proposed by Torabi and Bilodeau in 2009 [106]. Whilst their work uses camera and thermal IR data [106, 107], the event graph is a more general construct and their appearance reasoning method applies equally well to graphs and appearance models constructed from 3D lidar data. The method makes use of a secondary graph structure called the "hypothesis graph" which is constructed in parallel with the event graph.

For each single-object node in the event graph, a node is created in the hypothesis graph. Edges connect each hypothesis node to all of its descendants in the corresponding event graph. This means that each hypothesis node is linked to all nodes which could share the same identity, whilst respecting the temporal constraint that nodes must occur in the order they were observed. Referring to Figure 5.3, which shows an example event graph and corresponding hypothesis graph, hypothesis node 0 is linked to nodes 5, 6, 8, and 9. This reflects the fact that according to the event graph the person observed in node 0 could have been observed in track section 5, 6, 8 or 9 after the merge event which created node 4.

Edges are weighted according to the similarity of the nodes they connect (computed from their appearance models) and the solid lines Figure 5.3b show the highest weight edge emanating from each node. Note that the method in [106] computes the *distance* between appearance models and therefore the 'preferred' edges are those with the lowest weight, where in this work we seek to maximize *similarity* scores. We choose to work with similarity rather than distance here because similarity is required for the **BNet** method discussed below and it is convenient to keep a common implementation.

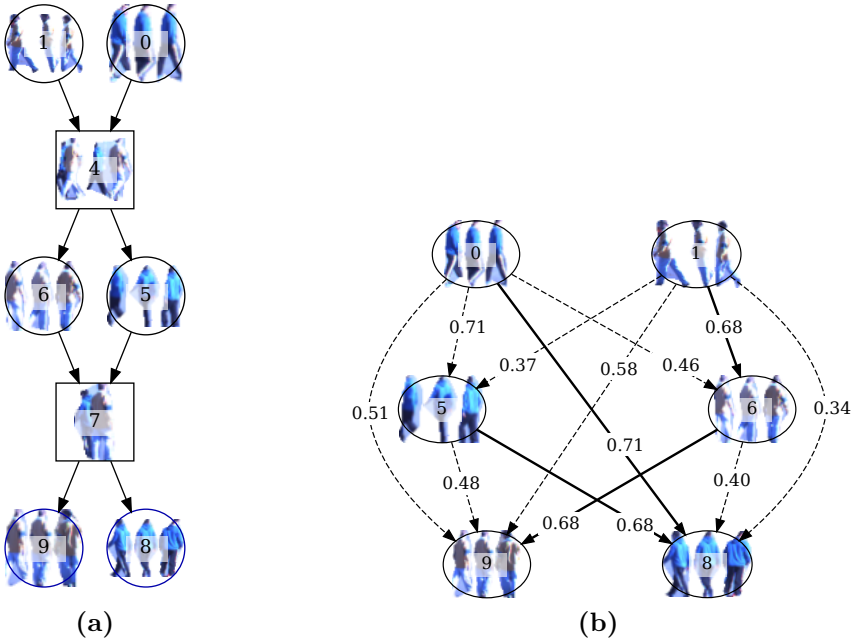


Figure 5.3 – An event graph (a) and corresponding hypothesis graph (b). Each node in the hypothesis graph represents a non-group node in the event graph, and is connected to nodes which may have the same identity. Edge weights show the similarity between pairs of nodes, with a solid line connecting a node to it's most similar child.

Since we are only interested in the ordering of edge weights, maximising similarity is equivalent to minimising distance.

Preferred edges are used to define "best hypothesis" or BH sets that link nodes to their ancestors. In Figure 5.3b, $BH(n_8) = \{n_5, n_0\}$ because node 8 is the highest weighted link from both node 0 and node 5. Similarly, $BH(n_9) = \{n_6\}$, $BH(n_6) = \{n_1\}$ and $BH(n_5) = \varnothing$. Identities are determined by tracing back from a node to the ancestors in its BH set; node 8 will be linked to nodes 5 then 0 (since they appear in its BH set). Following the same algorithm (described in more detail in [106]) node 9 will be linked to node 6, then $BH(n_6)$ will be expanded and the link will be made to node 0. The final paths are therefore found to be $\{0, 4, 5, 7, 8\}$ and $\{1, 4, 6, 7, 9\}$.

This particular example could be solved using a simpler algorithm; isolate each merge/split event and solve it independently, by matching input nodes to their most similar output nodes. Solving the merge/split event around group 4 would link nodes

0 and 5, whilst solving group 7 would link 5 to 8 thereby recovering the full trajectory.

However the **HGraph** algorithm has two advantages over such an approach. The first is that it can handle the case where some appearance information is ambiguous. If for example, node 5 was partially occluded and the appearance model wasn't very good, but the appearance models of nodes 8 and 0 indicated that they were the same object with high probability, then **HGraph** can cope with the poor appearance information in node 5 and determine that nodes 0 and 8 are on the same path.

The secondary advantage of the method is that it can reason about node identities in cases where the merge/split events are not well separated like they are in the previous example. Figure 5.2 showed an event graph where we cannot solve the merge/split mapping for each group node independently. Group 6 splits into two nodes (7 and 8) but its input is a group node. Therefore we need to consider further back into the graph history to make an identity decision. This is handled automatically by the **HGraph** algorithm.

Whilst the **HGraph** algorithm can handle missing or poor appearance models and determine an object's path through complex sequences of merges and splits, it can be considered a 'greedy' algorithm. The method always connects nodes with the most similar appearance models together, and so if two different paths 'compete' for the same node then the highest similarity will win and the other path will be cut short. In this case the correct identity decision might be to accept a locally sub-optimal node assignment in order to create the globally optimal solution. A method that achieves this is described in the following section.

5.1.2 Bayesian Network (BNet)

The method presented in this section (referred to as **BNet**) is adapted from [77] and includes a contributed extension that expands the variety of graphs that the method is able to solve.

An outline of the method is shown in Figure 5.4. As for **HGraph**, the method takes as input an event graph and associated appearance models and determines the

identity of each graph node. We first present the method as developed by Nillius et al. [77], then describe our extension in Section 5.1.2.4. Experimental results verifying the performance of the method are shown in Section 5.1.4.

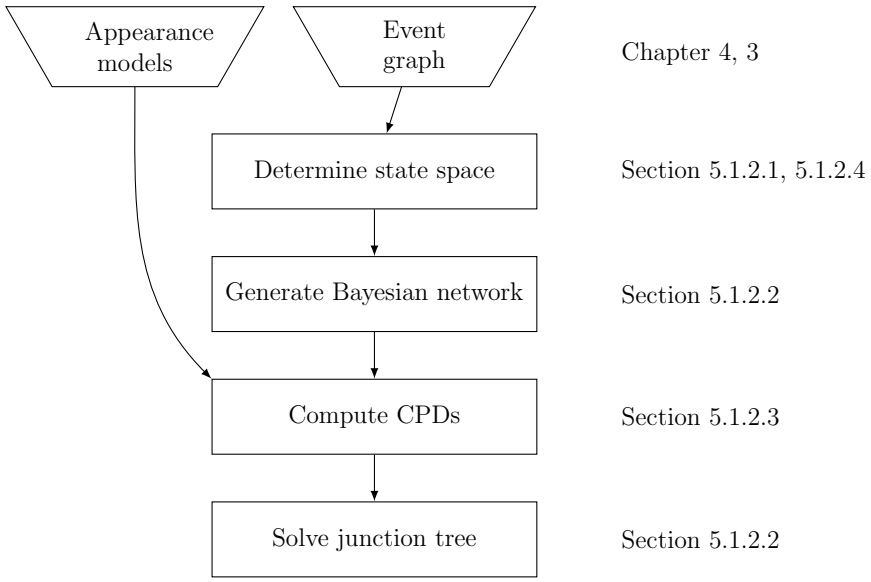


Figure 5.4 – Overview of the steps in solving the event graph using the **BNet** approach. The column on the right indicates in which sections the corresponding step is discussed.

5.1.2.1 'Switch' states

An innovation in the work of Nillius et al. [77] was to determine the minimal set of variables that represent all possible paths of targets through the graph. They recognised that the path of each target through the graph can be represented by the mapping of inputs to outputs for each group node. Each group node can be seen as a 'switch' that maps inputs to outputs, and the solution state space is composed of one state variable for each switch.

To adopt the notation from [77], nodes in the graph (track sections) are represented as T_i. If T_i is a group node, then it has a discrete state variable S_i (the 'switch' state) which represents the way a node's input edges connect to its outputs. All possible graph solutions can then be represented by $\mathcal{S}$, the set of state variables for all the group nodes (Equation 5.1).

$$\mathcal{S} = \{S_i; T_i \text{ is a split node}\} \tag{5.1}$$

The number of values that each S_i can take on must be chosen such that the state space uniquely represents all possible paths taken by objects through the graph. For a group node containing N targets that splits into m tracks (each output track comprising n_i targets) the number of choices per track is combinatorial and the total number of states for the switch node is given by Equation 5.2 [77].

$$\prod_{i=1}^{m} \binom{N - \sum_{j=1}^{i-1} n_j}{n_i} \tag{5.2}$$

For example, when a group containing two targets splits into two tracks, there are only two possible ways in which the inputs can be mapped to the outputs (Figure 5.5a). Track sections may contain multiple targets and so the situation can arise where a node with three inputs has only two output tracks. If the number of targets in each output are known to be 2 and 1 respectively, then there are three possible mappings Figure 5.5b. When multiple input objects map to the same output track, their relative ordering is preserved.

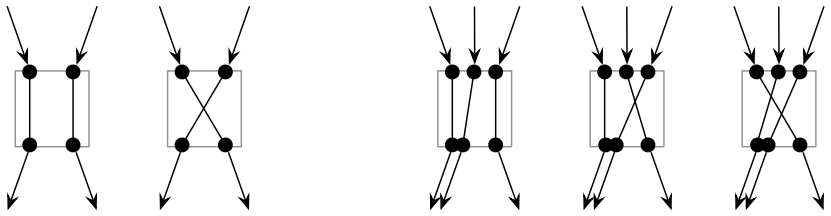
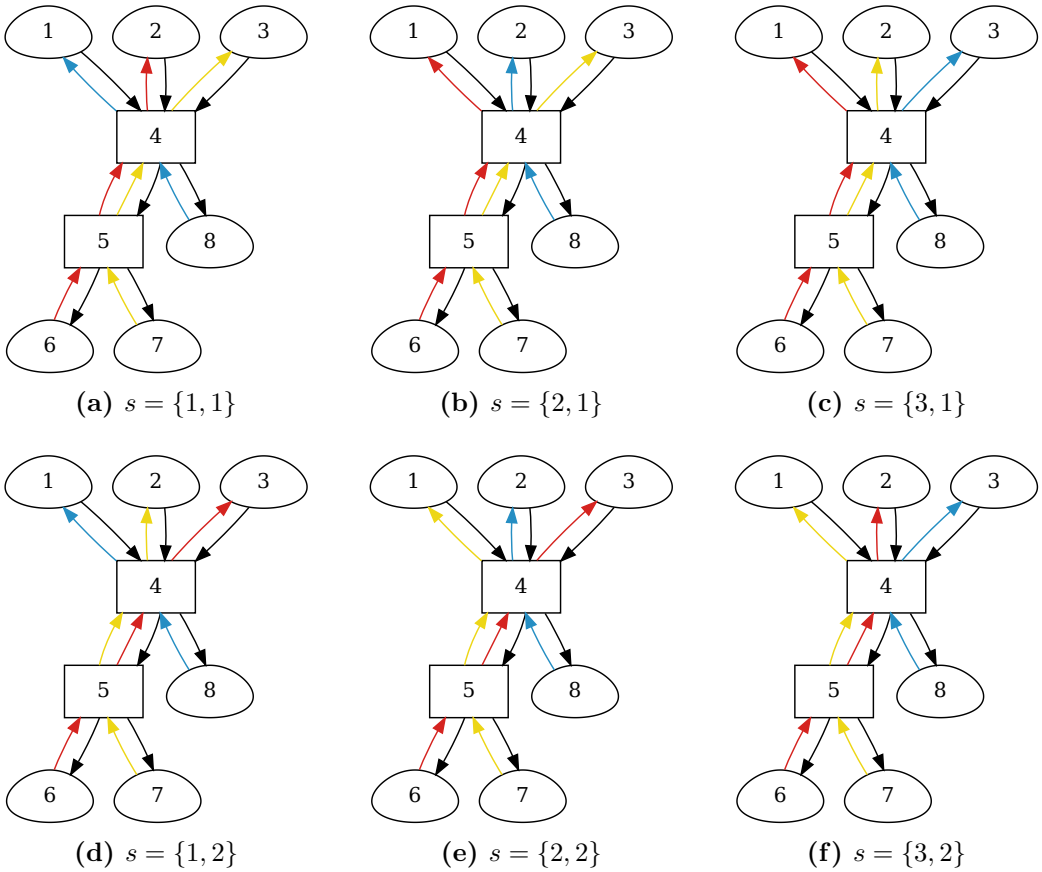


Figure 5.5 – Switch states for (a) a node with two input and two output tracks. (b) a node with three input and two output tracks, where the number of objects on each output are known to be 2 and 1 respectively.

An event graph comprising both of these group nodes is shown in Figure 5.6. Of the three objects that entered node 4, two of them remain together and become group node 5 before splitting apart. Node 5 (which is the same as Figure 5.5a) can switch the order of its outputs, so the fact that node 4 preserves their relative order ensures

that there is no redundancy in the state space. Notice that whether the red and yellow arrows switch sides of each other depends only on S_5. The total state space comprises 6 possible solutions, since we have $S_4 \in \{1, 2, 3\}$ and $S_5 \in \{1, 2\}$. This makes sense in that if we collapse the two groups together the problem is effectively to map 3 inputs to 3 outputs. The number of possible permutations in such a case would be $\frac{3!}{(3-3)!} = 6$.



(a) $s = \{1, 1\}$ **(b)** $s = \{2, 1\}$ **(c)** $s = \{3, 1\}$

(d) $s = \{1, 2\}$ **(e)** $s = \{2, 2\}$ **(f)** $s = \{3, 2\}$

Figure 5.6 – All possible graph solutions, s, for the state space $\mathcal{S} = \{S_4, S_5\}$ where $S_4 \in \{1, 2, 3\}$ and $S_5 \in \{1, 2\}$. Each colour represents the path of one object through the graph.

5.1.2.2 Bayesian network representation

Once the state space of the graph has been determined, solving the graph involves finding the most probable state given the observed appearance information

$$\hat{\mathcal{S}} = \underset{\mathcal{S}}{\operatorname{argmax}}\, P(\mathcal{S}|\mathcal{A}) \tag{5.3}$$

where $\mathcal{A}$ is the set of appearance models from the single-target track sections.

$$\mathcal{A} = \{A_i; T_i \text{ is a single-target track section}\} \tag{5.4}$$

Using Bayes' rule, we can convert this into a product of the prior and a likelihood function.

$$P(\mathcal{S}|\mathcal{A}) \propto P(\mathcal{A}|\mathcal{S})P(\mathcal{S}) \tag{5.5}$$

This enables us to represent the problem in a Bayesian network where the hidden nodes are the 'switch' states described in the previous section and the observed nodes represent the appearance information gathered during tracking, according to the methods described in Chapter 4.

As described in the previous section, each state configuration defines a unique set of paths through the graph, where each path represents one distinct object. The set of appearance models for each single (i.e. non-group) node on such a path is described as the *appearance path* and is given by Equation 5.6, where A_i represents the appearance model of track section T_i.

$$\text{path}(A_i, s) = \{A_j; T_j \text{ is on the same path as } T_i \text{ given } \mathcal{S} = s\} \tag{5.6}$$

Taking the event graph from the previous section as an example, the appearance path of a node can be found by tracing the coloured lines back from a given node. For example, under the state hypothesis $s = \{1, 1\}$, the appearance path of node 6 is found by tracing the red line in Figure 5.6a. The object in node 6 started in node 2, and was present in group nodes 4 and 5. Hence

$$\text{path}(A_6, s = \{1, 1\}) = \{A_6, A_2\}.$$

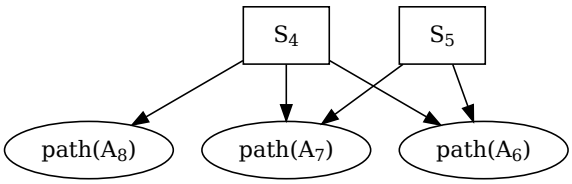


Figure 5.7 – Bayesian network for the event graph shown in Figure 5.6. The rectangles represent 'switch' variables and the ellipses are observed nodes which define the appearance CPDs described in Section 5.1.2.3. The path through the graph that leads to node 8 is not dependent on switch state S_5, and so they are not linked.

A path always ends in a node with no children (a tail node), and so the paths of the tail nodes fully define a graph solution. Therefore, the likelihood function from Equation 5.5 can be factorised as

$$P(\mathcal{A}|\mathcal{S}) = \prod_{A_i \in \text{tail nodes}} P(\text{path}(A_i, s)|\mathcal{S} = s). \tag{5.7}$$

The appearance paths of objects may be dependent on multiple 'switch' state variables (for example in Figure 5.6, path(A_6) is dependent on S_4 and S_5). These conditional dependencies are captured as links between 'switch' nodes and 'appearance path' nodes in the Bayesian network. The Bayesian network for the example graph presented in the previous section is shown in Figure 5.7.

As in [77] inference is performed by message propagation using the junction tree algorithm [46] and we reduce the complexity of the problem by limiting the depth in the graph within which we allow nodes to be considered dependent. This approximation limits the size of the cliques in the junction tree, hence speeding up the message passing. We use max-marginalisation in the message propagation (as implemented in the Bayes Net Toolbox for MATLAB [75]) to solve the inference problem.

5.1.2.3 Node likelihoods

The previous section described the structure of the Bayesian network used to solve the inference problem. We now define the node conditional probability distributions

(CPDs). The node CPDs are tables of probabilities for each appearance path, conditioned on the state variables to which they are linked. For example, the table of CPDs for path(A_6) in Figure 5.7 will contain 6 entries (1 for each permutation of $\{S_4, S_5\}$), each computed according to Equation 5.8 [77].

$$P(path(A_i, s)|S = s) \approx \prod_{A_j \in path(A_i,s) \setminus A_i} P(A_i, A_j) \tag{5.8}$$

$P(A_i, A_j)$ is the pair-wise similarity function for the appearance models A_i and A_j, as defined in Equation 4.1. Computing these similarity functions for specific appearance models was discussed in Section 4.3.2.

5.1.2.4 Number of Targets

The previous sections made the assumption that we know the number of targets represented by each node in the event graph. For example, to determine that the size of state space for the node in Figure 5.5b is three, we need to know that two targets leave split to the left, and one to the right. Nillius et al. describe a technique for computing the number of targets in each node of an event graph, however they note in [77] that "in practice there will be inconsistencies and some links will be left undefined".

An example of such a situation where a node represents an unknown number of tracks is shown in Figure 5.8a. Node 5 is a group node containing 3 targets, and it splits into two tracks (6 and 7). They merge again to form node 8. From looking at the graph we can conclude that node 8 must represent three targets, and that track 6 and 7 *together* represent three targets, but how these targets are distributed between the tracks is unknown. In this example there is one free parameter, but more complicated event graphs may yield more. In this section we contribute a novel algorithm that finds the minimal number of free parameters (referred to as 'count' variables) required to represent the number of targets in all graph nodes, and determines their range of allowable values. We then show how these variables can be incorporated into the

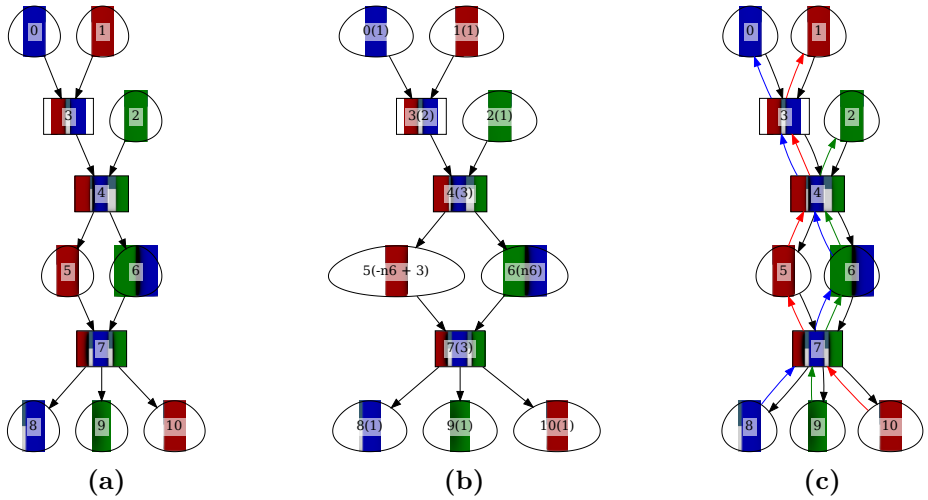


Figure 5.8 – The **BNet** algorithm applied to simulated data. (a) shows the event graph built by applying the track extraction algorithm to segmented point cloud data. Each track section is associated with a set of image patches, an example of which is shown in each of the nodes. (b) shows the number of targets for each node (in parentheses), as determined by Algorithm 1. In this example there is one free variable with the allowed values $n_6 \in \{1, 2\}$. The solution to the graph determined by solving the state variables is shown in (c). The inference process has determined that $n_6 = 2$.

Bayesian network formulation described above such that they can be solved using appearance information.

The event graph naturally leads to a set of constraints; for example if two groups merge, then the number of objects in the merged group is the sum of both groups that formed it. Likewise, when a group splits, the number of objects in the child nodes must sum to that of the parent group. Starting nodes (i.e. nodes with no parents) are assumed to contain only one object each. This logic can often be used to correctly propagate the number of objects to each node in the graph, however if breaks down when a group node known to contain n objects splits into m tracks, where $m < n$. Each resulting track contains one or more objects, and the way the n objects are distributed amongst m tracks may be unclear.

Algorithm 1 establishes a set of constraints that determines the number of targets in each node. Lines 2 through 7 start by setting the initial constraints that single start nodes contain one target, and that the minimum number of targets represented by a

Algorithm 1 Determine constraints on the number of objects in each node.

Require: *graph* made up of *nodes* and *edges*
node.min = minimum number of objects in node
$node.num, edge.num$ = number of objects in node, edge
node.in = set of input edges to a node
node.out = set of output edges from a node

1: $constraints \leftarrow \varnothing$
2: **for** *node* in *graph* **do**
3: **if** $|node.in| = 0$ and not group node **then**
4: $constraints.add\,(node.num = 1)$
5: **end if**
6: $node.min \leftarrow \max(|node.in|, |node.out|)$
7: **end for**
8: **for** *node* in *graph* **do**
9: **if** $|node.in| > 0$ **then**
10: $constraints.add\,(node.num = \sum\{edge.num \text{ for } edge \in node.in\})$
11: **end if**
12: **if** $|node.out| > 0$ **then**
13: $constraints.add\,(node.num = \sum\{edge.num \text{ for } edge \in node.out\})$
14: **end if**
15: $constraints.add\,(node.num \geq node.min)$
16: **end for**
17: **return** *constraints*

node is the number of input or output edges (whichever is higher). This logic is based on the fact that each edge represents at least one target, or otherwise that link would not be present in the graph. Line 10 states that the number of targets represented by a node is equal to the sum of the number of targets represented by each input edge, whilst line 13 states that this number is also equal to the sum of the number of targets on each output edge. Finally, line 15 stipulates that the number of targets represented by a node must be equal to or greater than the minimum number, determined in line 6. Note that the constraints in lines 10 and 13 rely on the assumption that targets don't spontaneously appear or disappear within groups. Of course, pedestrians don't randomly materialise or vanish, and this should be a reasonable assumption to make, however in practice these situations can arise, such as when the lidar-based tracker loses an object and spawns a new track later. In such cases the constraints will be invalid and this method is unable to compute a graph solution.

Whilst this might seem like a limitation of the **BNet** approach, the problem really lies in the event graph methods computing inconsistent graphs. The feedback of the constraints determined using Algorithm 1 into the online tracking algorithms could perhaps aid in graph consistency, and is an area for future research.

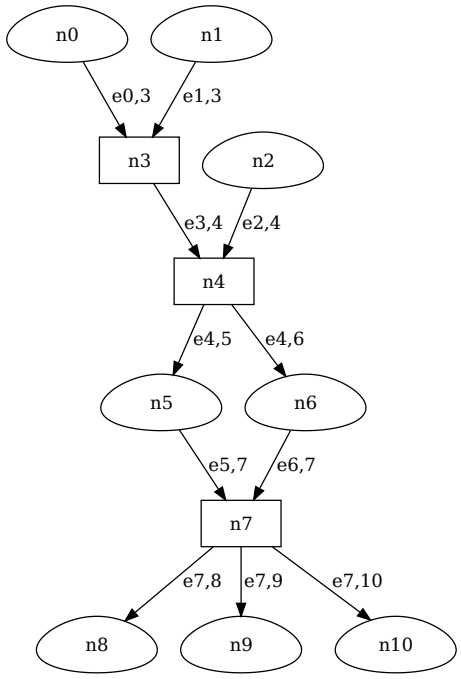
The constraints determined by Algorithm 1 are a set of linear equations, which can include inequalities. Due to the way the constraints are created, many equations are trivial and variable elimination can be used to reduce the constraint set. After variable elimination, variables may be fully determined, or their range may be constrained.

Figure 5.9 shows the constraints generated by Algorithm 1 for the event graph shown in Figure 5.8. After variable elimination, we are left with one free 'count' variable, n_6. Knowing that n_5 and n_6 are integers of value 1 or greater, we can determine that n_6 must be in the set $\{1, 2\}$.

In practise, the constraints are solved in two parts. First, the equality constraints (lines 10 and 13 of Algorithm 1 define a system of linear equations which are solved using fraction-free Gaussian elimination (as implemented in SymPy [98]). Line 10 adds inequality constraints and serves to limit the ranges of the remaining variables. These constraints may be multivariate and cannot presently be solved by SymPy, so instead we iterate through all permutations of the remaining variables, eliminating permutations which do not meet the constraints.

Any free variables left remaining after the constraint reduction ('count' variables) are part of the Bayesian network's solution space. The appearance paths of nodes, which were previously conditional upon 'switch' states may additionally become dependent on 'count' states. Figure 5.10 shows the Bayesian network that represents the event graph from Figure 5.8. Note the variable "n6" which represents $n_6 \in \{1, 2\}$. These additional state variables increase the complexity of the Bayesian network, but do not change its overall form, meaning that the maximum likelihood for the state variables ('switch' *and* 'count' states) can be found, as before, through a max-marginalisation message-passing algorithm.

An additional complexity is that 'count' states may be dependent on one another. For



(a)

reduced constraints
$n_0 = 1$
$n_1 = 1$
$n_2 = 1$
$n_3 = 2$
$n_4 = 3$
$n_5 \geq 1$
$n_6 \geq 1$
$n_5 = 3 - n_6$
$n_7 = 3$
$n_8 = 1$
$n_9 = 1$
$n_{10} = 1$

(b)

line	constraint
(4)	$n_0 = 1$
(13)	$n_0 = e_{0,3}$
(15)	$n_0 \geq 1$
(4)	$n_1 = 1$
(13)	$n_1 = e_{1,3}$
(15)	$n_1 \geq 1$
(4)	$n_2 = 1$
(13)	$n_2 = e_{2,4}$
(15)	$n_2 \geq 1$
(10)	$n_3 = e_{0,3} + e_{1,3}$
(13)	$n_3 = e_{3,4}$
(15)	$n_3 \geq 2$
(10)	$n_4 = e_{2,4} + e_{3,4}$
(13)	$n_4 = e_{4,5} + e_{4,6}$
(15)	$n_4 \geq 2$
(10)	$n_5 = e_{4,5}$
(13)	$n_5 = e_{5,7}$
(15)	$n_5 \geq 1$
(10)	$n_6 = e_{4,6}$
(13)	$n_6 = e_{6,7}$
(15)	$n_6 \geq 1$
(10)	$n_7 = e_{5,7} + e_{6,7}$
(13)	$n_7 = e_{7,8} + e_{7,9} + e_{7,10}$
(15)	$n_7 \geq 3$
(10)	$n_8 = e_{7,8}$
(15)	$n_8 \geq 1$
(10)	$n_9 = e_{7,9}$
(15)	$n_9 \geq 1$
(10)	$n_{10} = e_{7,10}$
(15)	$n_{10} \geq 1$

(c)

Figure 5.9 – Using Algorithm 1 to solve for the number of targets in each node. (a) The same event graph data as in Figure 5.8, with node and edge labels. (b) The reduced set of constraints, after applying variable elimination. (c) The full set of constraints where n_i represents the number of targets in node i, as produced by Algorithm 1 and annotated with the line of the algorithm that generated the constraint. From the reduced constraints, we are left with one free variable (n_6) and can determine that it must be either 1 or 2.

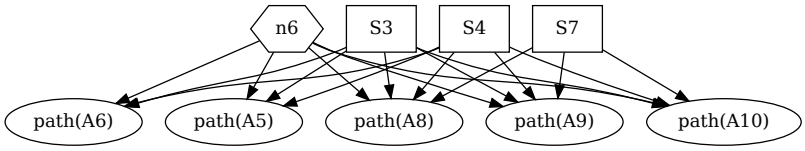


Figure 5.10 – Bayesian network for the event graph shown in Figure 5.8. The rectangles represent 'switch' variables (which determine the way inputs link to outputs in group nodes). The hexagon represents a 'count' variable (n_6 in Figure 5.8b), and the ellipses are observed nodes which define the appearance CPDs described in Section 5.1.2.3. n_6 affects the appearance path of all the tail nodes and so is connected to each.

example Figure 5.11 shows an event graph where a node containing 5 targets splits into two nodes. One of these child nodes then splits again before they all recombine, and the number of targets represented in nodes 9, 10, 11, 12 and 13 are dependent upon two variables which are constrained with respect to each other. In this case, the variables are considered jointly and are represented by a meta 'count' state. In this case, the meta 'count' state is $(n_{12}, n_9) \in \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$. These meta states are fit into the Bayesian network in the same way as regular 'count' states (Figure 5.12).

5.1.2.5 Computational Complexity

In the **BNet** algorithm, the biggest computational cost is the evaluation of the node CPDs. The size of each CPD is exponential in size of the state space of the linked nodes, so for densely connected graphs this cost grows quickly. To improve the runtime performance of the algorithm, we make use of the approximation described in [77] and mentioned earlier, that limits the size of cliques in the network and speeds up message passing. This approximation, by ignoring dependencies between nodes with sufficient separation in terms of the original event graph, also limits the size of the CPDs.

Whilst our implementation was written in Python and not optimised for speed, our experience suggests that **BNet** would not be able to achieve identity reasoning at the sensor frame rate of 20Hz. This is not necessarily a problem, however, since identity

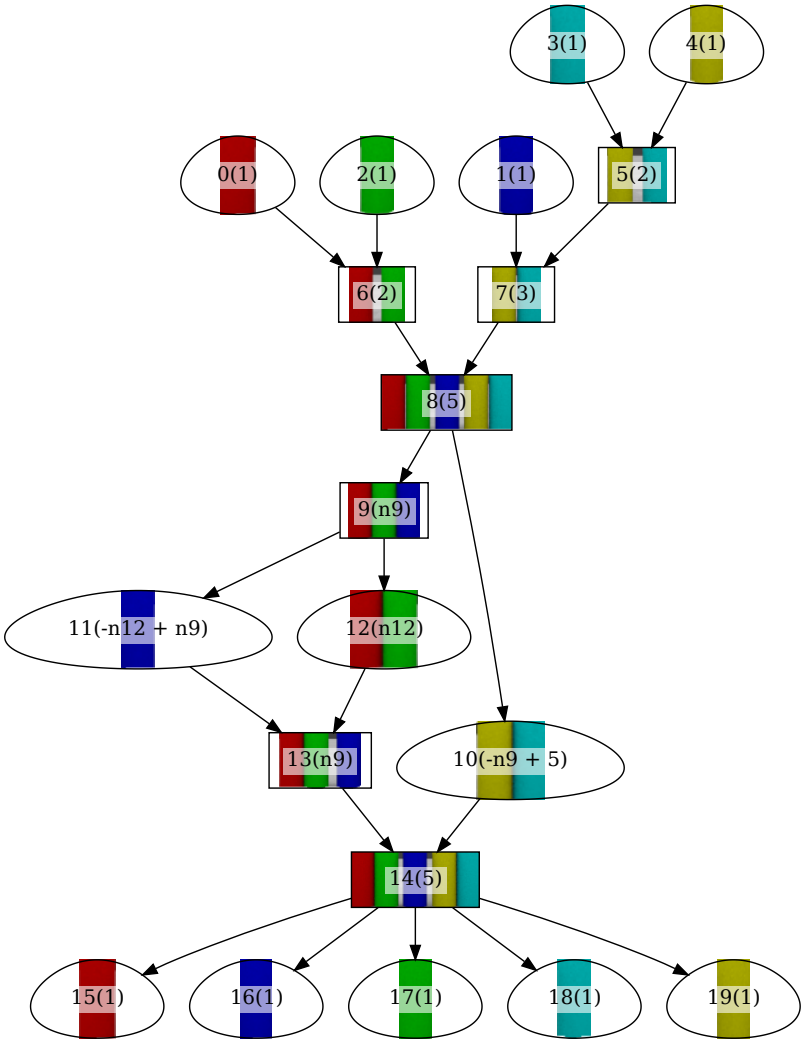


Figure 5.11 – An event graph with dependent 'count' states. The number of targets in nodes 9, 10, 11, 12 and 13 are unknown but can be represented by two free parameters. After applying the constraints from Algorithm 1 and solving for the minimal set of variables, the variables n_9 and n_{12} are found to be dependent. Therefore a meta 'count' state is created, with possible values $(n_{12}, n_9) \in \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$.

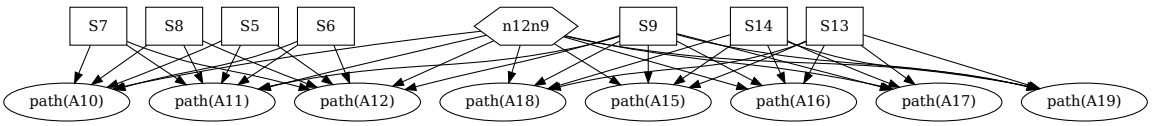


Figure 5.12 – Bayesian network with meta 'count' state, computed for the event graph above. n_9 and n_{12} are dependent on each other and so are merged into the one state variable.

reasoning is carried out independently from graph construction. Tracking and event graph construction can proceed at full frame rate, with an identity solution computed perhaps every few seconds, or as required by the particular application.

The **HGraph** method, which greedily computes identities, requires fewer model comparisons, and is therefore suitable for identity reasoning at every frame.

5.1.2.6 Marginalisation and information gain

A benefit of using a Bayesian network to solve for node identities that wasn't explored in [77] is the ability to determine the marginal likelihoods of specific variables.

Before adding appearance information, the prior probabilities of a 'switch' state variable with two choices is $\{0.5, 0.5\}$, which corresponds to an entropy of 1 bit. If, after incorporating appearance information, the marginal probabilities for the state variable are $\{0.1, 0.9\}$ the corresponding entropy is 0.47 bits. This enables the system to examine how 'certain' a particular data association decision is, which could be used to request intervention from a human operator, or data from an additional sensor.

The latter option, requesting data from additional sensor, is limited in the sense that appearance information must have been captured by that sensor *before* the interaction occurred. Observing two objects after they split is not sufficient to determine their identities unless you have the appearance models from before they merged. One case in which this could be useful is if a particular sensor is expensive to process but can be logged continuously. Then, if an interaction occurs and the entropy in that state variable remains high, additional information can be extracted from the sensor log. This is quite applicable to the Ladybug as it is a high bandwidth sensor, for which operations such as rectification, Bayer decoding and subsequent processing are non-trivial. Only processing the data as needed would reduce the overall computational resources required by the system.

Even without such a system, it is still useful for the identity reasoning algorithm to be able to understand the uncertainty in its own predictions. A further use case for such information is in the context of multiple sensor systems. As part of the appearance

model evaluation in Chapter 4, we introduced the concept of simulated interactions. By simulating interactions between targets, then measuring the entropy in the graph solution, a tracking system is able to predict target interactions that will result in ambiguity. This could enable the system to bring another sensor on line, or request an additional perspective from another sensor platform before the targets even interact.

The marginal probabilities computed by the **BNet** algorithm will be considered in more detail in the experimental section that follows.

5.1.3 Experiments

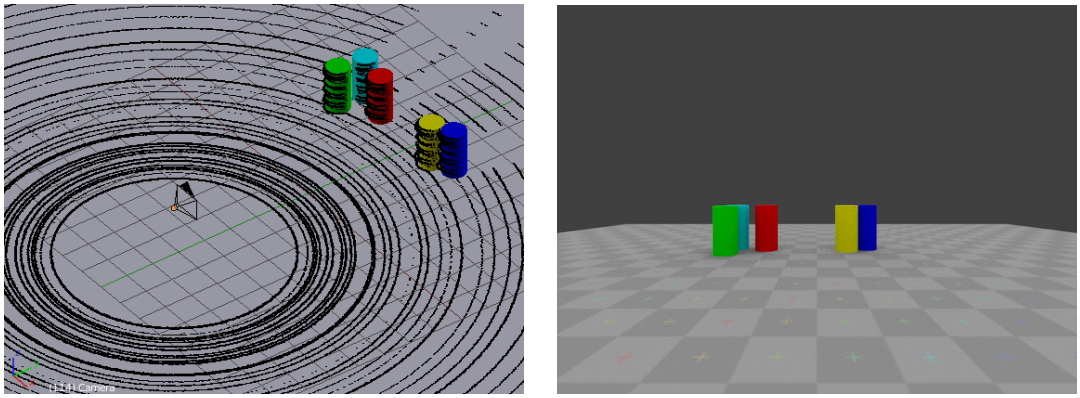
To evaluate the ability of the **HGraph** and **BNet** methods to solve the identities of nodes in an event graph, we make use of the same datasets presented previously in Chapters 3 and 4. As in Chapter 3, identity-tracking performance is measured with respect to a hand-labelled ground truth and reported in terms of the V-measure score.

The simulated data used for the scenarios shown in Figures 5.8 and 5.11 was generated using "BlenSor" [37], an extension to "Blender" [102] (the open source 3D content creation application used for labelling in Section 3.2.3). "BlenSor" simulates the 3D data captured by a variety of range sensors, and combined with the image rendering capabilities of "Blender" was used to generate simulated Velodyne and camera data. Figure 5.13 shows a screenshot from the application and a simulated camera frame.

Due to the modular nature of the tracking systems described in this thesis, the process of obtaining object trajectories can be factored into three separate stages:

1. 3D lidar tracking is performed as per the methods in Section 3.3, resulting in an event graph. The camera data associated with each observation is also recorded.

2. Appearance models (as described in Chapter 4) are constructed for each node in the event graph, using the recorded camera data.



(a) (b)

Figure 5.13 – Sensor simulation. (a) shows a screen shot of the "BlenSor" [37] simulation environment, with a generated Velodyne scan. The properties of the simulated Velodyne scan closely match real data. (b) Simulated camera data.

3. The graph is 'solved' using the methods described in this chapter, and track sections (nodes) are linked together to form complete trajectories.

This separation allows us to compare specific implementations of parts of the system independent of other design choices. For instance we can compare **HGraph** and **BNet** on the same event graph, using exactly the same appearance models. Alternatively, we could the measure impact of changing the appearance model, without changing the event graph structure or appearance reasoning algorithm.

5.1.4 Results and Discussion

In this section we evaluate the event graph solving methods described in Section 5.1.1 and Section 5.1.2. The 3D lidar tracking approaches are as described in Chapter 3, and so the analysis of MOTP and the difference between the predicted and true numbers of independent objects (shown in Figures 3.25 and 3.26 respectively) still apply. In Chapter 3, however, we had no way to determine which nodes in the event graph corresponded to which objects, so we did not analyse the identity tracking performance.

We begin by quantitatively evaluating tracking performance on the simulated scenarios used previously in this chapter. We then turn our attention to real-world data and measure the identity tracking ability of the **HGraph** and **BNet** methods, comparing their performance on appearance models built from camera, lidar and finally thermal IR data. The results from **split-cost**, reported in Chapter 3, are included for comparison.

5.1.4.1 Simulated data

Our extension to the **BNet** method, presented in Section 5.1.2.4, can use appearance information to solve for the number of targets represented by nodes in the event graph in situations where the graph itself is ambiguous. How often such situations occur depends both on the observed data and on the nature of the graph building algorithm itself, so to focus on the determination of the 'count' state variables we make use of simulated data.

Figure 5.14 shows the tracking performance of the **HGraph** and **BNet** methods on the simulated data, with **split-cost** included for comparison. The event graph for SIM-THREE was already used as an example in Figure 5.8, where the **BNet** method correctly determined that there were two objects present in node 6. The **HGraph** algorithm is not designed to handle nodes containing an unknown number of targets, and this is reflected by the lower V-measure score.

The second dataset SIM-FIVE corresponds to the event graph shown in Figure 5.11. In this case there are two dependent variables that must be solved as one 'meta' state in the Bayesian network. **BNet** automatically calculates the state space of these 'count' variables, and solves for them using the appearance models, result in the paths shown in Figure 5.15, and a high score for this method in Figure 5.14.

5.1.4.2 Camera-based appearance models

From the evaluation of appearance models in Section 4.4 we found that Ladybug camera data can be used to reliably solve multi-object interaction scenarios. We

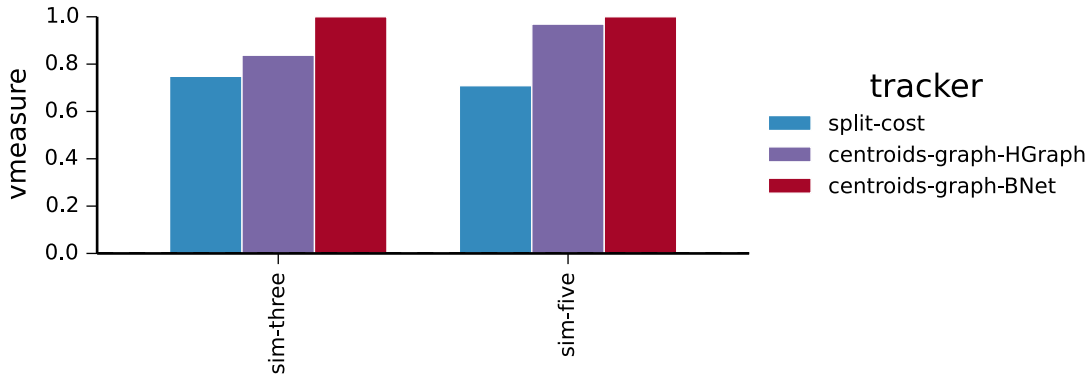


Figure 5.14 – Tracking performance on simulated data. The **BNet** solutions for SIM-THREE and SIM-FIVE are shown superimposed on the event graphs, in Figure 5.8c and Figure 5.15 respectively.

would expect, therefore, that combining these models with the event graph reasoning methods described in this chapter should result in a tracking system that can correctly maintain object identity.

Figure 5.16 shows the V-measure score of both the **HGraph** and **BNet** reasoning methods applied to the event graph created by **split-cost-graph**. As a reference, the V-measure score from **split-cost**, the best of the non graph trackers, is also shown. As expected, the event graph methods consistently matched or outperformed **split-cost** across all of the datasets with near-perfect scores. Note, however, that the event graph created for MERGE-BBP was inconsistent, so appearance information could not be used to solve for node identities using the **BNet** method. In this case, the V-measure score is that of the graph prior to integrating appearance information.

BNet and **HGraph** performed similarly well across the datasets, with **BNet** showing slightly better performance on the SAME-SHIRT-A and MERGE-BBPP experiments. With SAME-SHIRT-A, the event graph created by **split-cost-graph** contained one split event where the number of targets in each branch was ambiguous. This section of the event graph is shown in Figure 5.17. **BNet** correctly determines that there is one object in node 4 and two in node 5. **HGraph**, on the other hand, is not able to reason about the number of objects in such nodes, resulting in a lower V-measure score.

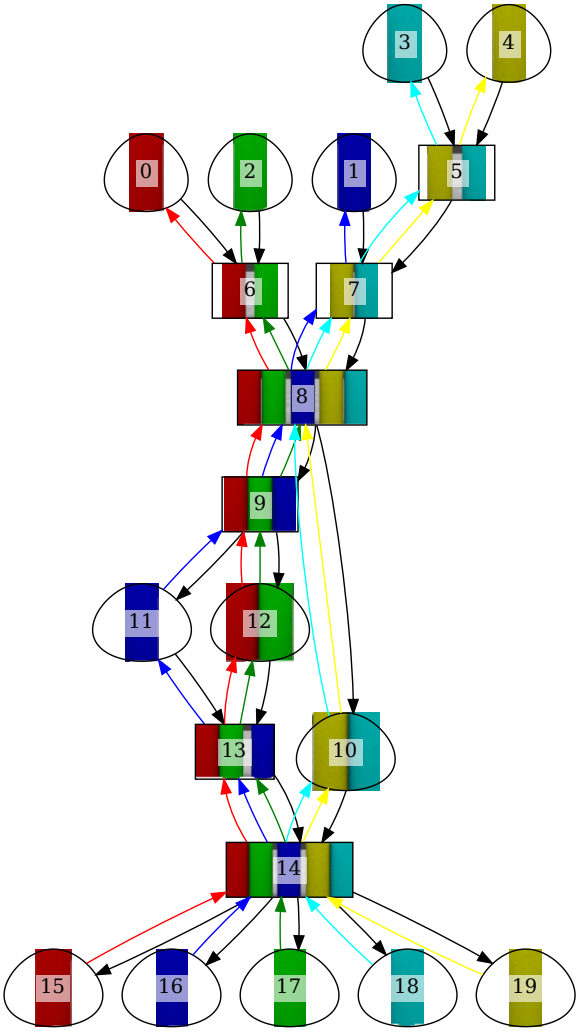


Figure 5.15 – Solved graph for SIM-FIVE. The **BNet** algorithm has determined that there were three objects in node 9 and two in node 12, and has correctly solved the paths of all the objects in the graph.

The **centroids-graph** tracking approach also produces event graphs and so we evaluate the performance of the **HGraph** and **BNet** methods applied to these graphs. Figure 5.18 shows the V-measure scores obtained. As described in Chapter 3, **centroids-graph** creates more groups than **split-cost-graph**, which often results in a more complicated event graph, possibly involving nodes that contain an unknown number of targets. **BNet**, with its ability to resolve these 'count' variables, outperforms **HGraph** on such datasets.

The **centroids-graph** approach is also more likely to create inconsistent event graphs.

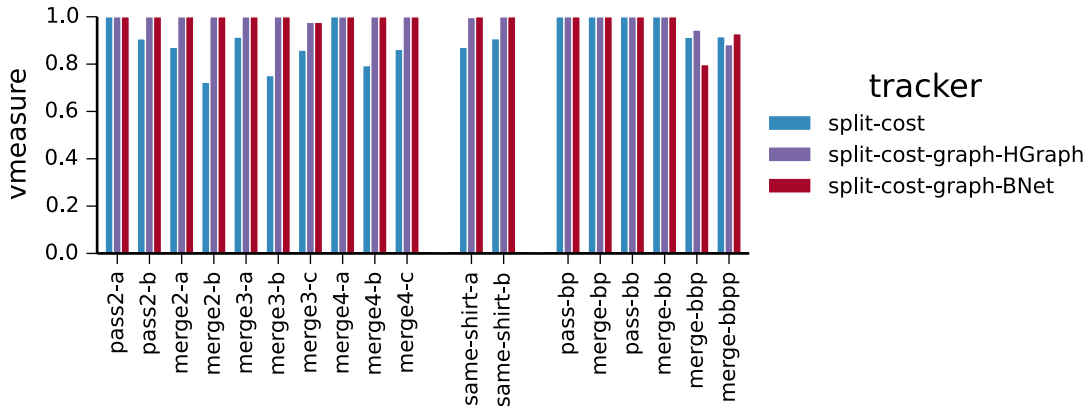


Figure 5.16 – Identity tracking performance with Ladybug camera data.

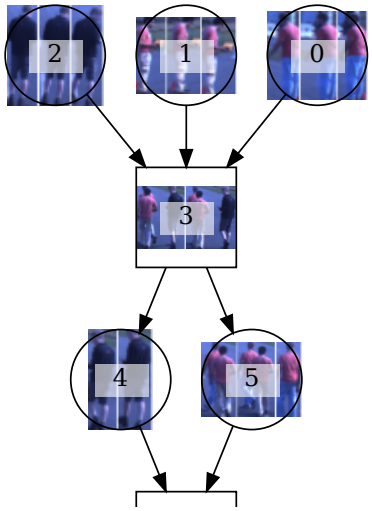


Figure 5.17 – Section of the event graph for SAME-SHIRT-A. The number of targets in nodes 4 and 5 is successfully determined by **BNet** to be one and two respectively

Referring back to Figure 3.25, there were four cases in which **centroids-graph** incorrectly determined the number of objects in the scene. For these experiments, **BNet** is unable to generate a solution, and the V-measure score is that of the graph prior to identity reasoning.

The mean performance across these datasets is summarised in Table 5.1, which includes all the tracking methods from Chapter 3. Overall **split-cost-graph** is the best-performing method, and achieves best results when paired with the **BNet** graph solving algorithm. As mentioned in Chapter 3, the MOTP performance of the graph

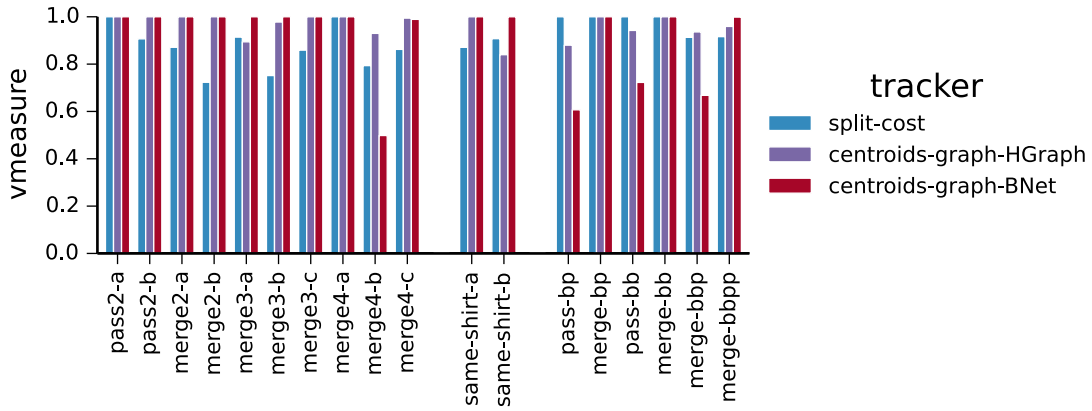


Figure 5.18 – Identity tracking performance using Ladybug data and **centroids-graph**. Four of the event graphs recorded were inconsistent, so **BNet** fails to generate a solution.

methods is worse than that of **split** and **split-cost**, due to the presence of group nodes. **Centroids-graph** creates groups more often then **split-cost-graph** so has the worst MOTP.

	MOTP	MOTA	homogeneity	completeness	vmeasure
centroids	0.099	0.929	0.810	0.625	0.700
split	0.086	0.961	0.917	0.884	0.900
split-cost	0.086	0.989	0.976	0.848	0.905
centroids-graph-HGraph	0.137	0.991	0.995	0.938	0.964
centroids-graph-BNet	0.138	0.991	0.991	0.880	0.916
split-cost-graph-HGraph	0.094	0.997	0.999	0.980	0.989
split-cost-graph-BNet	0.094	0.997	0.999	0.972	0.983

Table 5.1 – Mean tracking performance with Ladybug camera data used in the **HGraph** and **BNet** appearance models.

The Bayesian network applied in the **BNet** approach allows us to compute the uncertainty of a graph solution from the entropy associated with each state variable. This is an important advantage of the **BNet** method – it means the system can monitor its own performance and could, for example, trigger warnings or request human intervention when uncertainty passes a threshold. Figure 5.19 shows the total entropy of the graph before and after appearance reasoning was applied. The reduction in entropy effectively shows the amount of information added to the system by the appearance

models. The reason why a few of the experiments show zero entropy is because the number of state variables in each of these track graphs was zero. Since there were no appearance decisions to be made, the prior entropy is zero, and adding appearance information can not reduce it further.

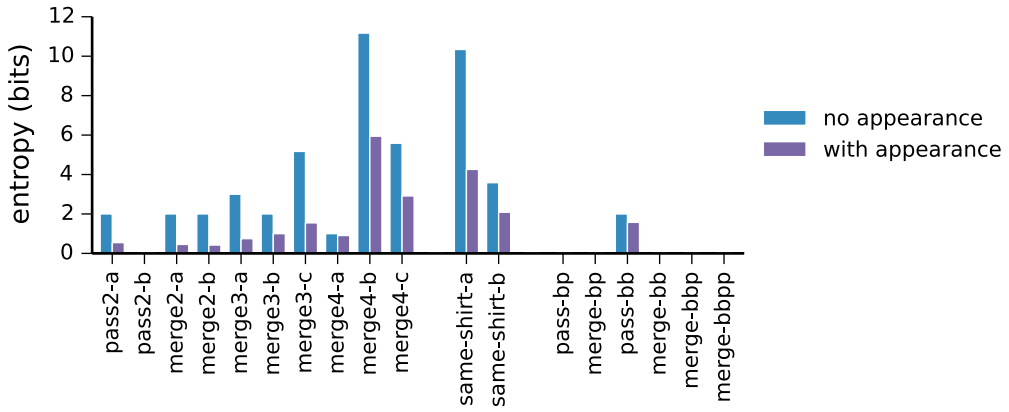


Figure 5.19 – Event graph entropy before and after applying appearance reasoning.

5.1.4.3 Lidar-based appearance models

Whilst Chapter 4 showed that appearance models built from camera data were the most useful for resolving multi-object interactions, we demonstrated that in some cases shape statistics from the 3D lidar data aid in object recognition. Figure 5.20 shows the appearance reasoning results when using observation height and radius (as measured from 3D lidar data) as appearance features. The entropy in the graph variables before and after integrating appearance information is shown in Figure 5.21.

In most cases, the graph-based methods outperform **split-cost**, indicating that the lidar features could be used to discriminate between targets. However, this was not always the case and **HGraph** in particular performed poorly in a few instances.

The MERGE3-C experiment was the only one in which **HGraph** outperformed **BNet** and so warrants explicit attention. The experiment involves four people (one of whom is an operator and doesn't interact with the others). The event graph recorded by **split-cost-graph** is shown in Figure 5.22a. **BNet** allows us to examine the

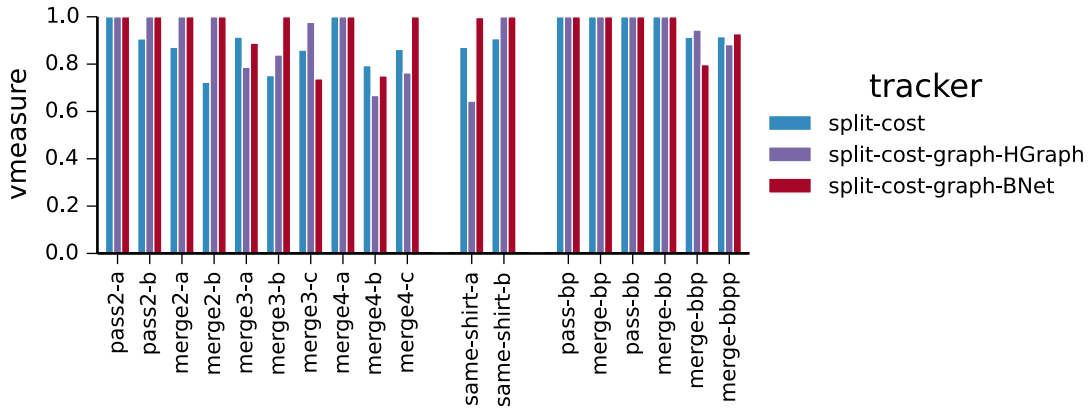


Figure 5.20 – Appearance reasoning performance, using observation height and radius as the appearance model.

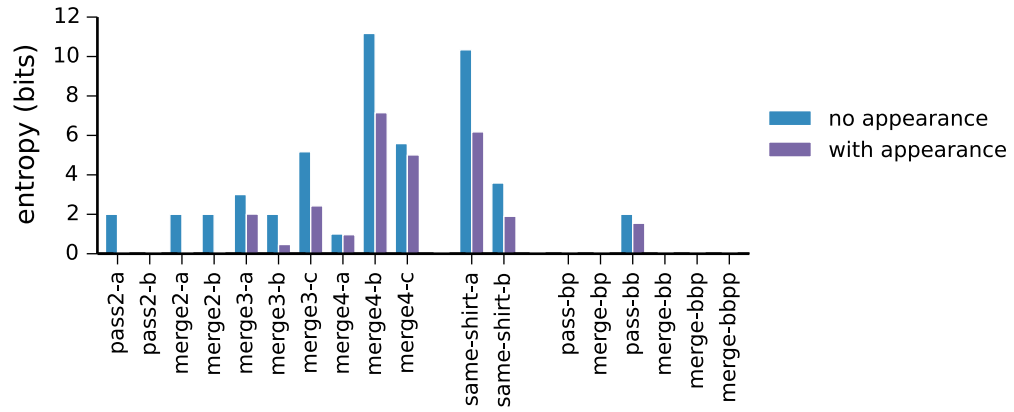


Figure 5.21 – Entropy of geometry-based appearance reasoning solution.

uncertainty in a graph solution as shown in Figure 5.21, however these entropies are marginalised over all state variables. To get a closer understanding of the decisions made by **BNet**, we can examine the marginal probability distribution over individual state variables. Looking at node 6, where the red and blue targets split, the marginal probability distribution was $\{0.501, 0.499\}$, indicating that the system did not know which target went in which direction.

A more intuitive way of understanding the uncertainty in the graph solution is to map these state uncertainties onto node identity likelihoods. Figure 5.22b shows a representation of these likelihoods: rows correspond to nodes from the event graph, columns to individual object identities and the value in a cell represents the likelihood

that an object is present in the corresponding node. Cells are also coloured by object id, with the intensity proportional to the likelihood.

Initially, the system is confident it is tracking four distinct objects, since they haven't yet interacted. Then, three objects merge into a group, and node 4 contains objects B, C and D. The 'switch' state of node 6 was very uncertain, and this is reflected by the fact that the system gives equal likelihood to the presence of objects B and D being in node 7. From this point onwards, objects B and D remain confused. The system knows when they are in the same group (e.g. nodes 10 and 11), but when they are separated it can't tell which is which.

The reason for the confusion between objects B and D is simple. The heights of pedestrians A, B, C and D are 1.83m, 1.72m, 1.80m and 1.69m respectively, and the 3D lidar data is not sufficiently accurate to measure this 3cm height difference. Given that there is a similarly small height difference between pedestrians A and C, we would expect that if they interact then their identities will become confused.

This is indeed the case in MERGE4-B. The final section of the event graph and the corresponding identity likelihood table is shown in Figures Figure 5.22c and Figure 5.22d respectively. All four targets interact, and there is confusion primarily between the pairs (A, C) and (B, D). Even though there is a height difference of approximately 10cm between the two pairs, there is still a small chance that any of the pedestrians could be in any of the final four nodes. This is due to the fact that the lidar doesn't measure the height of every target perfectly each time. Occlusions and other noise means that there is some uncertainty in height and radius measurements and this uncertainty is reflected in the identity likelihoods. Also, according to **BNet**, object B is the most likely object for both nodes 24 and 25. Whilst the **BNet** algorithm resolves this conflict when generating the maximum likelihood solution, in this case the solution it finds does not match the ground truth and this is reflected in the lower V-measure score for MERGE4-B in Figure 5.20.

The V-measure analysis does not take into account the uncertainty of node identities, and evaluates only the maximum likelihood solution provided by **BNet**. This means that cases where there is ambiguity, **BNet** will sometimes get it right, and at other

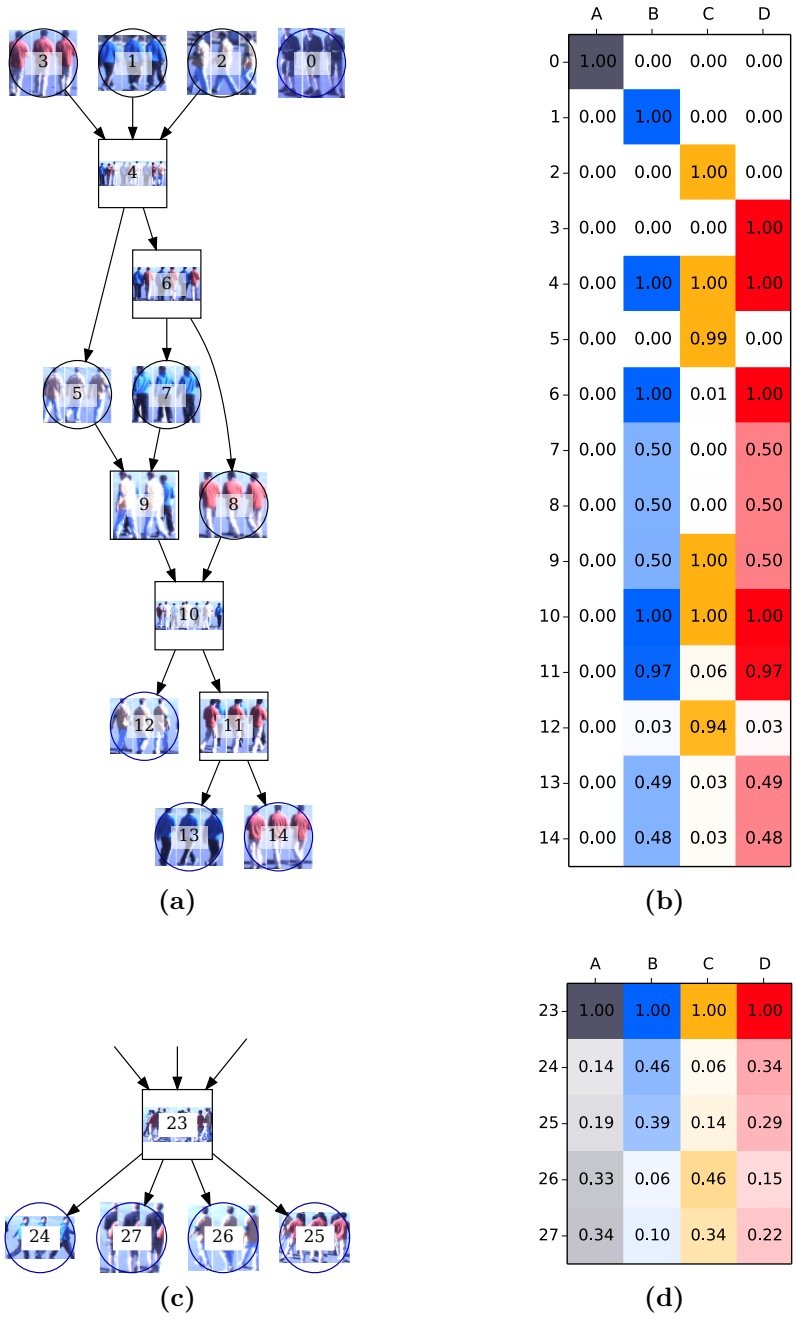


Figure 5.22 – Node identity likelihoods, computed from lidar-based appearance models. (a) and (b) correspond to the MERGE3-C experiment, whilst (c) and (d) are from MERGE4-B. The tables on the right give the likelihood that a given object (columns) is present in a node (row). For example in (b), node 6 contains targets B and D. Node 12 is very confidently target 12, whilst node 13 could represent either target B or D. Each column is coloured by the approximate colour of the target with saturation corresponding to the value of the cell.

times it will fail. Likewise **HGraph**, which maintains no concept of uncertainty, will sometimes settle upon the correct solution when the appearance information is ambiguous. This is the reason why **HGraph** performs better than **BNet** on MERGE3-C; the appearance information is ambiguous, and **HGraph** happened to make the right decision, whilst **BNet** did not. The ability to obtain uncertainty information from **BNet** makes it a more attractive algorithm for real-world applications.

5.1.4.4 Thermal IR

Finally we consider thermal IR data. In Chapter 4 we showed that appearance models built with thermal IR data would likely be ineffective at resolving object interactions. This is confirmed by the experiments in this section.

Figure 5.23 shows the V-measure of the tracking results. Although appearance reasoning based on thermal IR data did, in a number of instances, improve tracking performance over the comparison method, **split-cost**, on the majority of datasets there was no change in the graph entropy. This means that the system did not reduce its overall uncertainty by incorporating appearance data, or put another way, no identity information was gained from the IR data.

As described in the previous section, the V-measure score is computed on the maximum likelihood solution; in the case of ambiguous appearance information when there are multiple solutions with very similar likelihoods, the system will sometimes make the correct decision and sometimes fail. The fact that **BNet** receives a high V-measure score for MERGE3-A, but a low score for MERGE3-B, is an example of this. In both cases, Figure 5.23 shows that the graph entropy is not reduced by adding appearance information.

Since the entropy reported in Figure 5.24 is measured across the entire graph, a small decrease in entropy does not necessarily imply an increase in performance and therefore V-measure. A reduction in entropy might indicate that an object is sufficiently distinct from the others such that it's identity can be determined. For example, in the example presented in Figure 5.22a and 5.22b, the decrease in overall entropy cor-

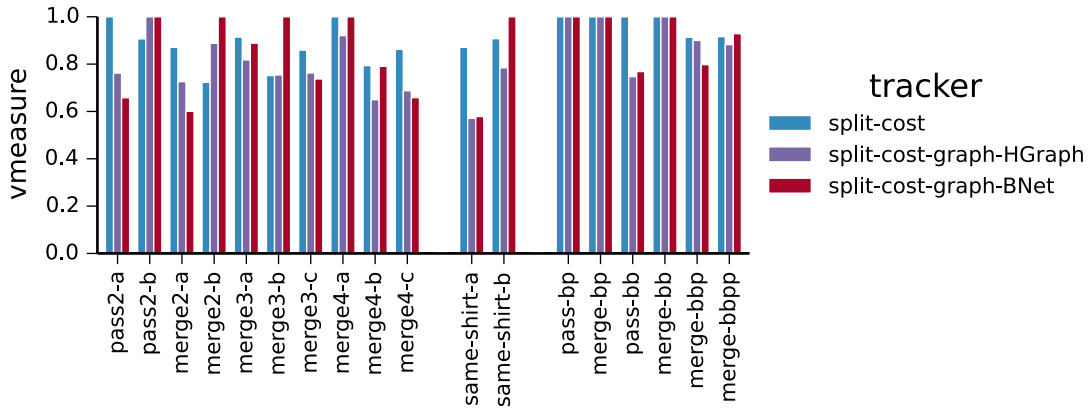


Figure 5.23 – Tracking performance using thermal IR.

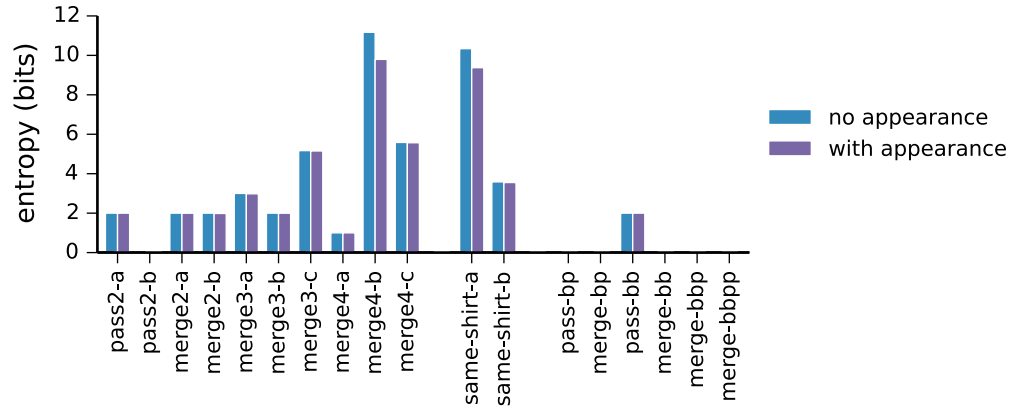
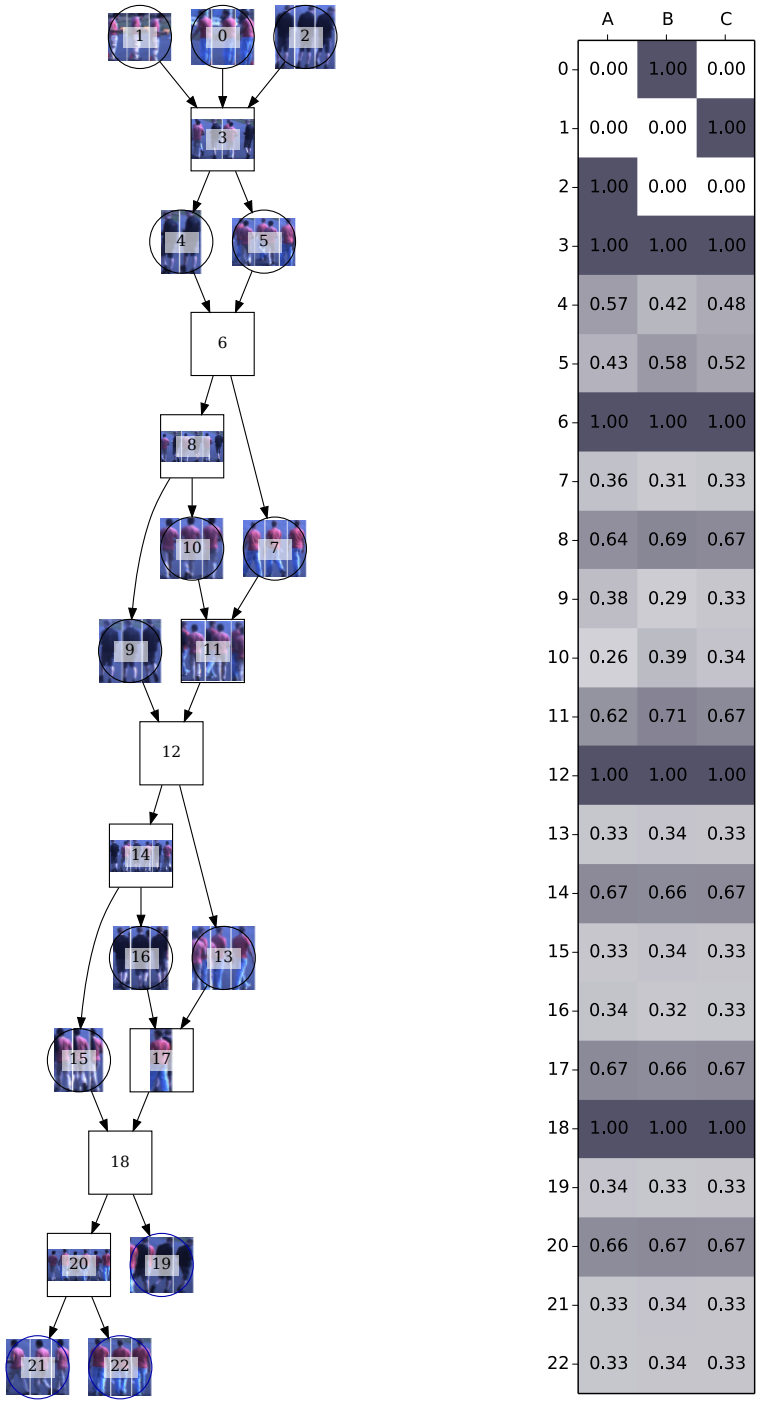


Figure 5.24 – Graph entropy with thermal IR data.

responded to the fact that the path of object C was completely resolved, and the remaining entropy indicated that objects B and D remained confused. However, a reduction in entropy might also be distributed across many state variables, indicating a slight decrease in uncertainty, but not resolving the path of any one object. This is the case for SAME-SHIRT-A when using IR appearance models. Even though the appearance models reduced the overall graph entropy, the identity of any one given node (shown as a likelihood table in Figure 5.25) is still unclear.



(a) **(b)**

Figure 5.25 – Node identity likelihoods, computed from IR-based appearance models for the SAME-SHIRT-A experiment. Despite the reduction in entropy observed in Figure 5.24, the identity of any given node is uncertain. The shade of each cell reflects the likelihood, with darker shades corresponding to higher likelihood.

5.2 Large scale experiments

The experiments so far in this thesis have for the most part shown tracking results on experimental data gathered in controlled conditions. We have demonstrated the ability of our proposed methods to track the position and identities of interacting targets, in situations with up to four targets in close proximity. In this section we apply the proposed methods to datasets collected in an unconstrained urban environment.

We qualitatively show the nature of the performance of **split-cost-graph** in combination with the **HGraph** and **BNet** appearance reasoning algorithms, highlighting their strengths and weaknesses in particular unconstrained examples.

The analysis in this section builds upon the results of Section 5.1. There, we demonstrated, through detailed experimentation on a large number of manually labelled scenarios, that our graph-based tracking approach extracts homogeneous track sections that can be linked together via appearance reasoning to form complete object trajectories. In this section we consider datasets that are too large to manually label. We present, therefore, a detailed analysis of two scenarios extracted from the data (which are small enough to label) as well as overall tracking statistics, which, when viewed in the context of Section 5.1 demonstrate the utility of our approach.

5.2.1 Experiments

For these large scale experiments, we applied our tracking algorithms to two urban datasets, recorded in the city of Sydney. In order to gather the datasets, "Shrimp" was mounted on a utility vehicle and driven through city streets. Batch change detection (described in Section 3.1.4) was used to find moving objects, which were then tracked using the methods described in this thesis.

Opera House

The OPERA-HOUSE dataset is the same as that used for appearance model evaluation in Chapter 4. However, unlike in Chapter 4 where we made use of labelled tracks

to test the appearance models, here we apply the full tracking pipeline, from change detection and segmentation to event graph construction and appearance reasoning. The dataset was gathered on the Sydney Opera House promenade, which is a predominantly pedestrian area and no other vehicles are observed in the dataset. Figure 5.26 shows the path of the sensor vehicle and the trajectories of all tracked objects (obtained using the **split-cost-graph** method) superimposed on aerial imagery. The duration of the dataset was 2 minutes and it comprises 2400 Velodyne scans. Due to the nature of the area in which the data was gathered the sensor vehicle was driven at a fairly slow pace, with a median speed of 8.3 km/h. Figure 5.27 (duplicated from Figure 4.6b) shows Velodyne points superimposed on a camera image taken during the experiment.

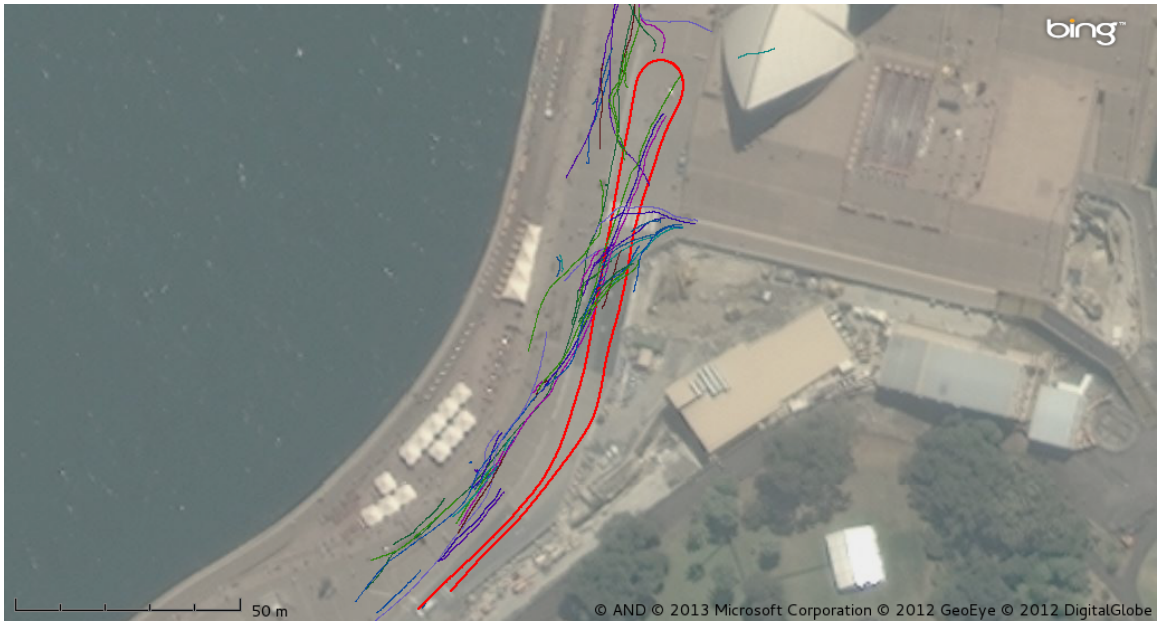


Figure 5.26 – Tracking output of **split-cost-graph** on the OPERA-HOUSE dataset. The trajectory of the sensor vehicle is shown in red. The short trajectory in the upper right of the image, which may appear to be an error, is actually a correctly-tracked pedestrian walking in the open area under the Opera House. Aerial imagery © Microsoft Corporation.

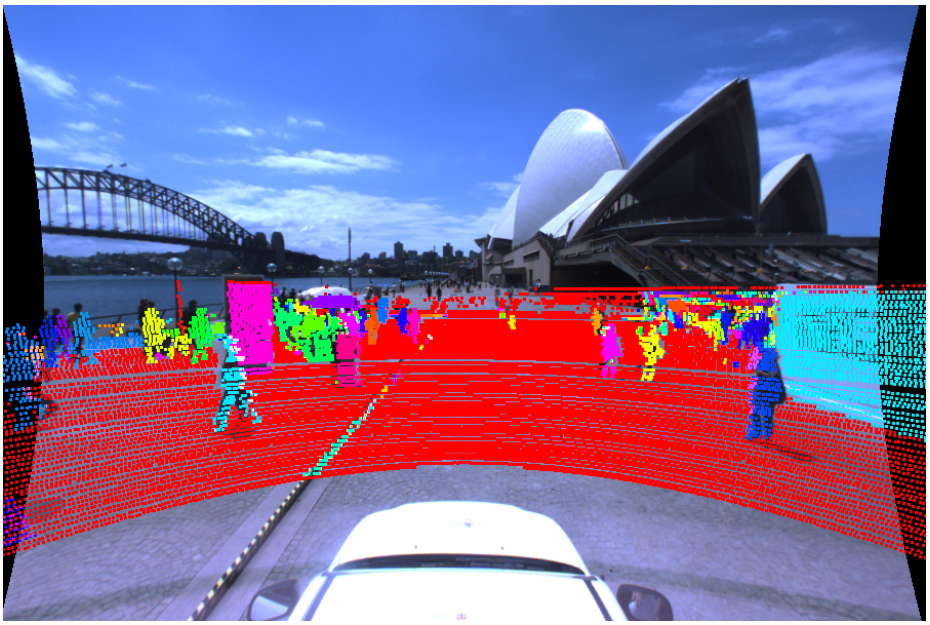


Figure 5.27 – Segmented Velodyne lidar data superimposed upon an image from the Ladybug camera.

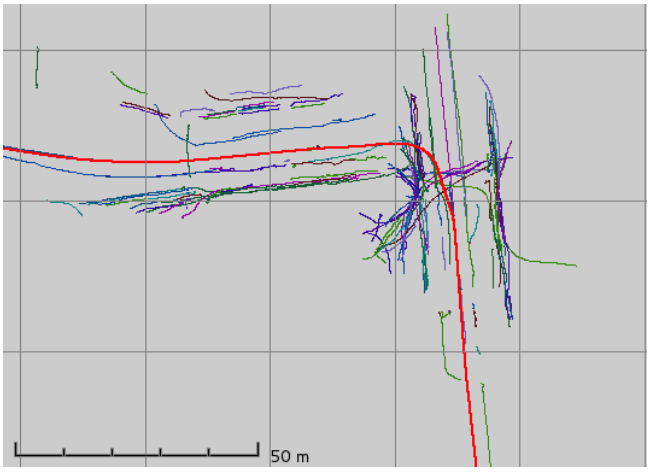
CBD

The CBD dataset was gathered in the Sydney central business district on a Friday during lunchtime hours. Heavy traffic and frequent stopping for traffic lights meant that the speed of the sensor vehicle varied from completely stopped up to a maximum of 30 km/h. The dataset lasts for 20 minutes and 50 seconds, and comprises 24975 Velodyne scans. Figure 5.28a shows the route taken by the sensor vehicle during the experiment. A zoomed in region from the route, showing tracks obtained by the **split-cost-graph** method, is shown in Figure 5.28b. Figure 5.29 shows examples of objects detected by the change detection algorithm.

5.2.2 Evaluation of specific scenarios

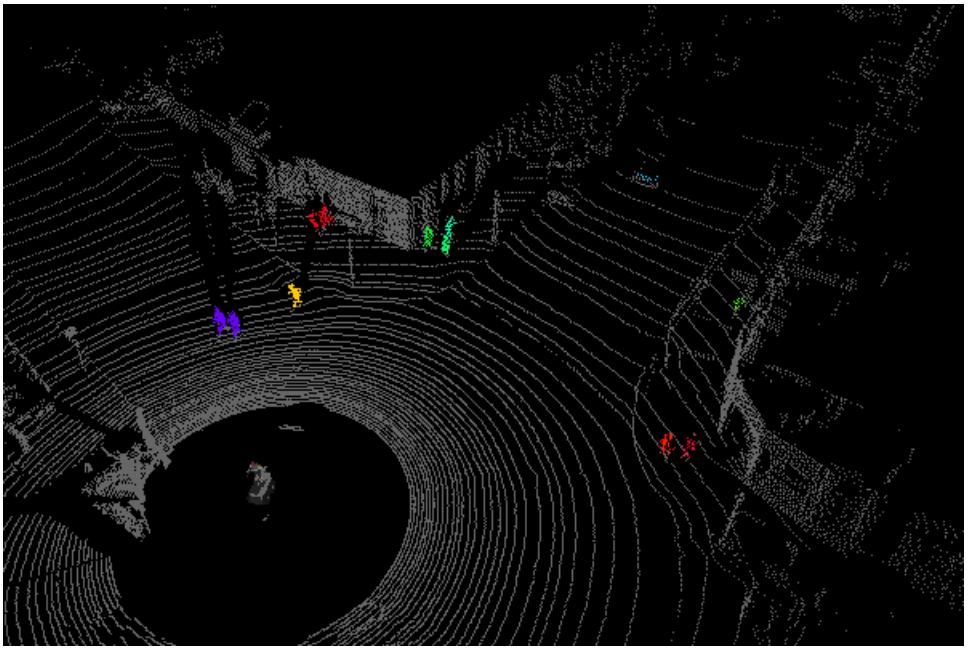
Evaluating tracking performance in the same manner as we did for the previous section would require the position of all objects to be labelled in all scans. Labelling is a time consuming process, and it would not be practical to label all 27375 frames of the datasets. To understand the behaviour of the object tracking methods on

(a)

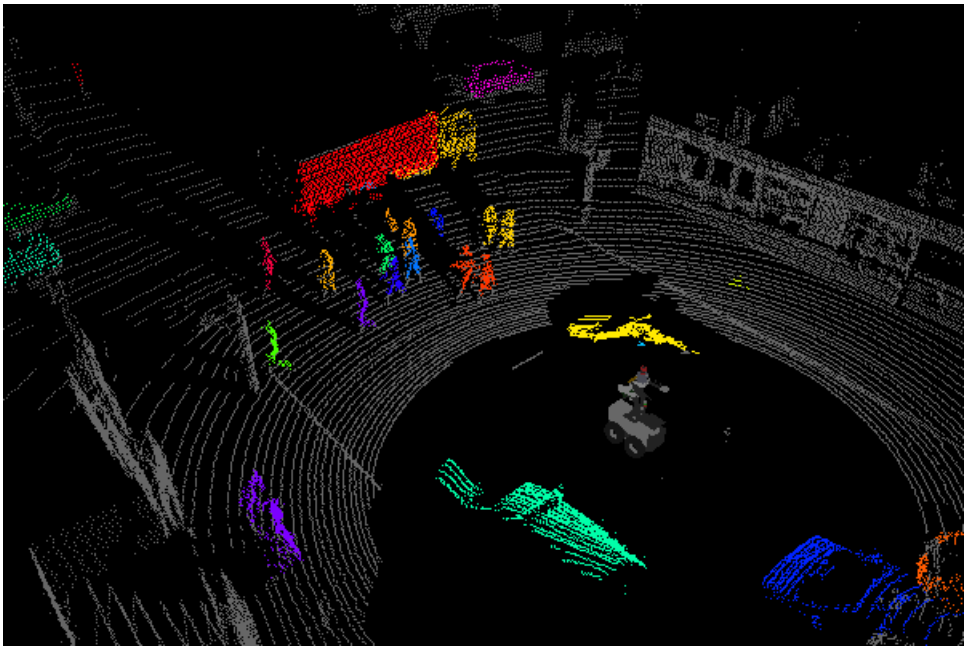


(b)

Figure 5.28 – (a) Route of the CBD dataset through the city of Sydney, shown overlaid on aerial imagery. (b) Zoomed in region (marked with a blue rectangle on (a)) showing a section of the tracking result that contains a large number of tracks crossing at an intersection. The trajectory of the sensor vehicle is shown in red. Aerial imagery © Microsoft Corporation.



(a)



(b)

Figure 5.29 – Objects extracted by change detection in the CBD dataset. Background points are shown in grey, and segments are drawn in different colours.

real-world data, therefore, we explore some specific cases selected manually from the longer datasets.

5.2.2.1 Opera House - Three Pedestrians

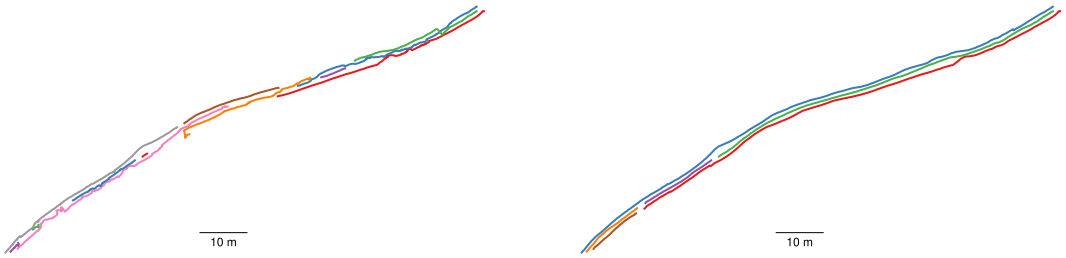
This section analyses some trajectories extracted from the OPERA-HOUSE experiment. Three pedestrians cross the road and walk parallel to it, moving in the same direction as the sensor vehicle. Figure 5.30 shows some images captured by the forward facing camera during the time the pedestrians were being tracked.

(a) **(b)** **(c)**

Figure 5.30 – Three pedestrians walking side-by-side during the OPERA-HOUSE experiment, as seen by the forward facing camera on the sensor vehicle. Images are taken at approximately 90, 105 and 127 seconds into the experiment.

The three pedestrians walk side-by-side and maintain the same positioning relative to each other, so a correct tracking solution would record three parallel trajectories. Figure 5.31 shows the results of the **centroids** and **split-cost** method applied to this data. Since the pedestrians are walking close together, they are occasionally under-segmented, and the tracking system receives only two or even one observation. **Centroids** is not designed to handle this case, and the extracted trajectories shown in Figure 5.31a demonstrate its poor performance. When under-segmentation occurs, tracks are lost or their identities become confused.

Split-cost, on the other hand, is able to re-cluster these under-segmented observations and does a much better job of determining object trajectories. In order to maximise homogeneity, the tracker resets tracks in the case of data association ambi-



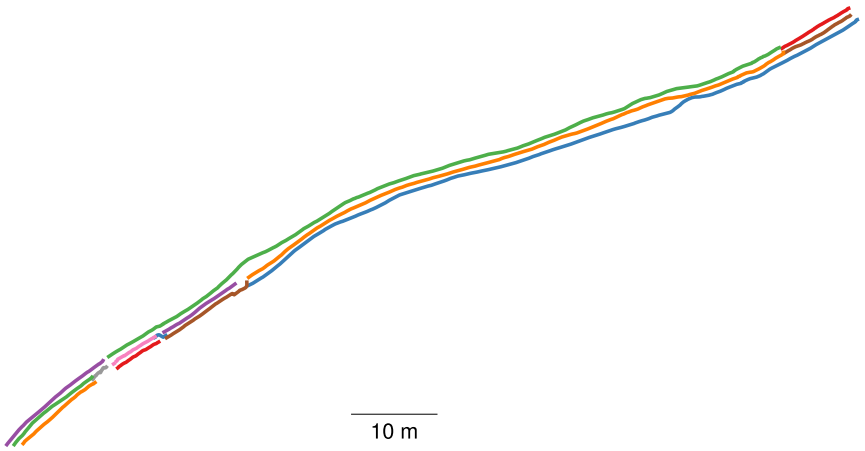
(a) centroids **(b) split-cost**

Figure 5.31 – Tracking results from (a) **centroids** and (b) **split-cost** extracted from a section of the OPERA-HOUSE experiment that contains three people walking side-by-side. **Centroids** creates a total of 12 tracks for the 3 objects as a result of its inability to cope with under-segmented data. **Split-cost** fragments one object into 3 tracks, one into 2 tracks, and the other is tracked correctly. The tracks generated by **split-cost** are *homogeneous* in that they each contain only observations from one object.

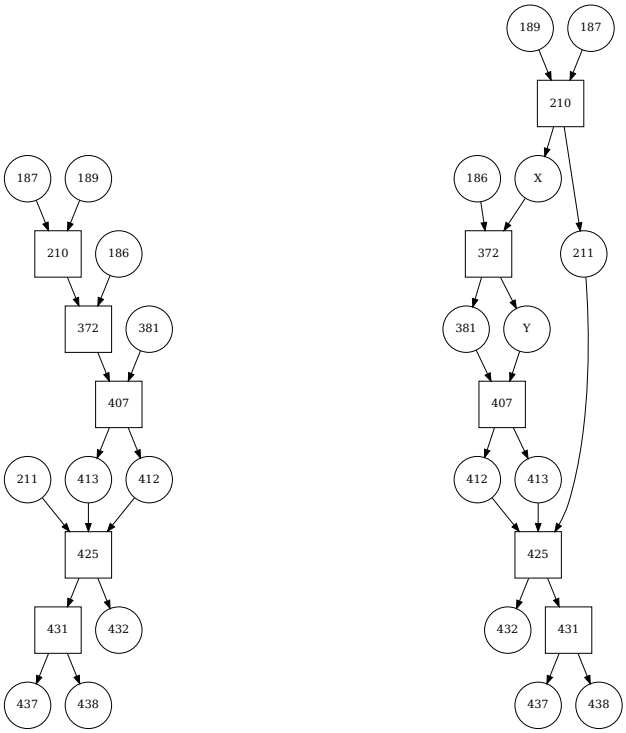
guity. This causes some of the trajectories to be fragmented into several pieces, and therefore results in reduced completeness.

Split-cost-graph behaves similarly to **split-cost** in that it seeks to maximise the homogeneity of individual track sections. Unlike **split-cost**, however, it also maintains the relationships between track sections in the event graph, and can therefore link these sections into longer trajectories, thereby increasing the completeness of the tracking solution. The track sections extracted by **split-cost-graph** (each of which corresponds to a node in the event graph) are shown in Figure 5.32a. The corresponding event graph is shown in Figure 5.32b.

The event graph recorded by **split-cost-graph** during tracking contains errors. The most apparent of these is the fact that there are five start nodes (nodes with no parents) in the graph, which means the tracker found five distinct objects when there were really only three. This problem is caused by occlusion during group formation, and is explored in detail in Figure 5.33. If one object is occluded for a period of time, the group track is updated with data from the other object only and its state estimate shifts to match that object's location. When the other object reappears, it may do so outside the association distance of the group and so instead of being interpreted



(a)



(b) (c)

Figure 5.32 – Behaviour of **split-cost-graph** whilst tracking the three pedestrians. (a) shows the track sections, each colour corresponds to a separate node in the event graph, shown in (b). The event graph in (b) contains errors; for instance there are five start nodes (nodes with no parents), which implies five distinct objects in the scene. In reality, though, there are only three. The graph can be fixed by adding split events to nodes 210 and 372. This requires two additional nodes (X and Y) and corresponding links. The manually corrected event graph is shown in (c).

as a split event, a new track is created.

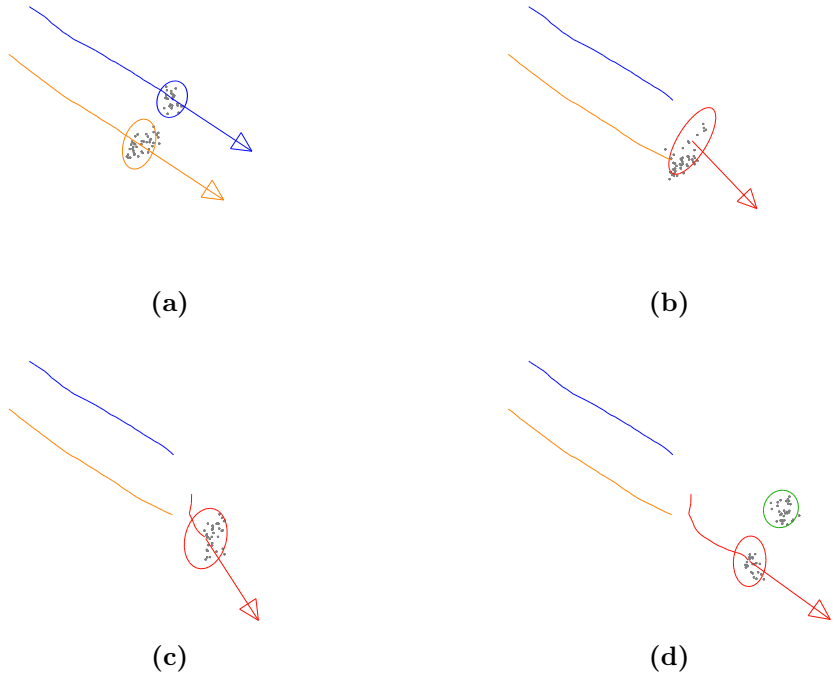


Figure 5.33 – Example of event graph construction error. (a) Successfully tracking two objects. (b) Under-segmentation results in a group observation and causes tracks to be merged into a group track. (c) One object is occluded, and the group track "locks on" to the other object. (d) Second object reappears, and the group should split. However, the observation is outside the association distance of the group track, so a new track is created.

To evaluate the performance of the appearance reasoning algorithms in isolation from any errors in the graph itself, we manually adjusted the graph to reflect what actually happened in the data. The resulting graph is shown in Figure 5.32c. Two groups (nodes 210 and 372) exhibited the problem described in Figure 5.33, whereby a new track should have been linked to the group in a split event. To correct this, we manually created split events, linking the appropriate tracks (211 and 381) to the corresponding group nodes. Additionally, any observations that were assigned to the group node after the split event occurred are separated into a new track section and form the other output of the split event. These extra track sections are shown as nodes X and Y in Figure 5.32c.

We now apply the appearance reasoning methods described in Section 5.1 to this

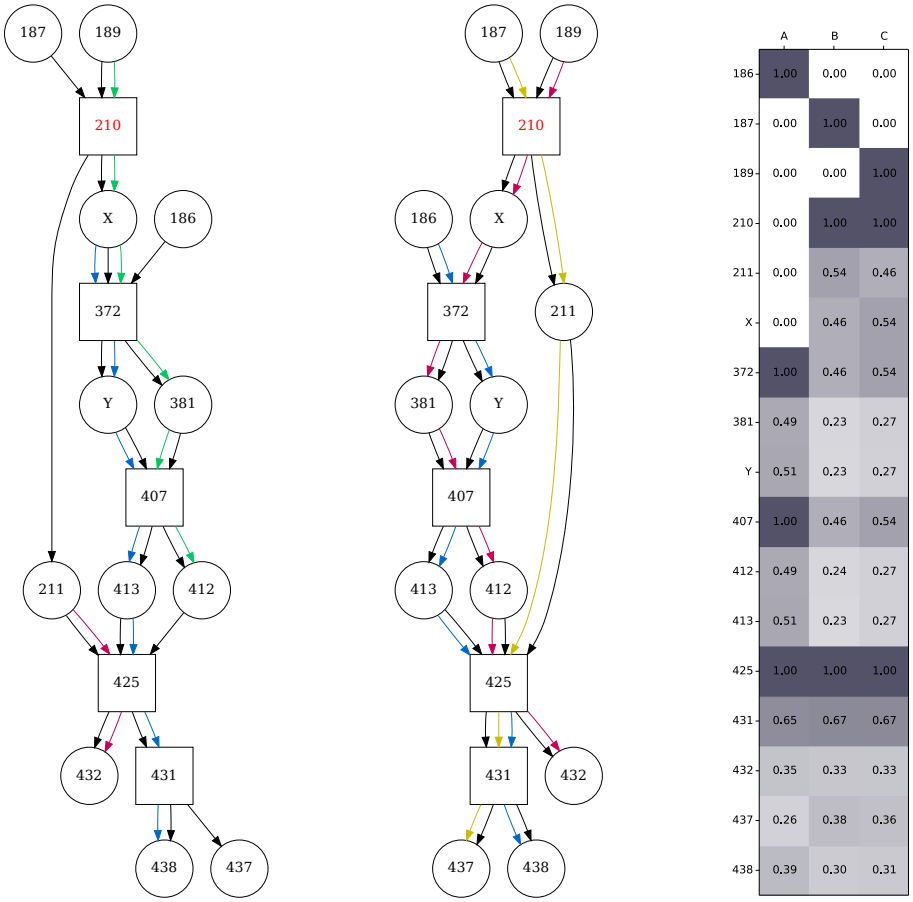
corrected event graph. Figure 5.34 shows the graph solutions obtained by applying the **HGraph** and **BNet** methods along with trajectories coloured by identity.

The correct solution is three pedestrians walking side-by-side. They do not change their relative positions, so whilst **BNet** outperforms **HGraph**, it makes an error when it changes the track order (as seen in the lower left corner of Figure 5.34e). From the likelihood table presented in Figure 5.34c, we can see that the identity assignments of the last few nodes have high uncertainty.

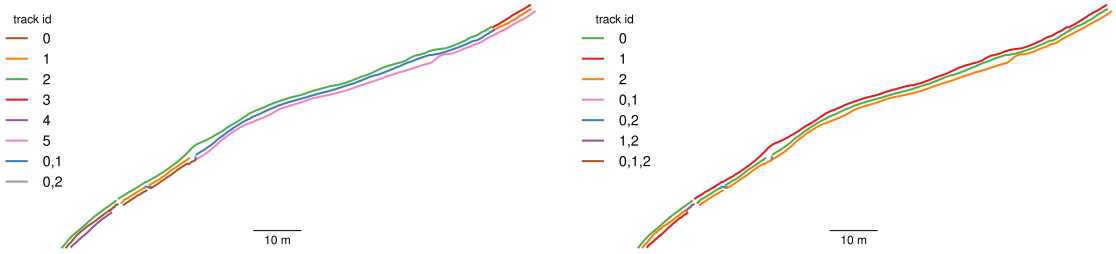
One reason the appearance models performed relatively poorly is because of the way the pedestrians were positioned relative to the sensor vehicle. At the start of the experiments, when nodes 186, 187 and 189 were created, the pedestrians were aligned parallel to the camera's imaging plane, as shown in Figure 5.35a. This meant that the region of camera data corresponding to each target can be extracted cleanly, as indicated by the bounding boxes marked.

Later in the experiment, when the appearance models for nodes 211, 412 and 413 were being created, the targets were arranged perpendicular to the sensor. As Figure 5.35b shows, the bounding boxes of the targets overlap and even though the *hull* sampling strategy (discussed in Chapter 4) improves the likelihood that appearance information will be sampled from the desired target, the occlusion means that the quality of the models is reduced. Although appearance models dynamically update and should reflect the changing target perspective, systematic occlusion means that little information from the furthest of the targets can be extracted. This increases the uncertainty in target identities.

Given this scenario, where we have a sensor vehicle moving through an environment seeking to track all dynamic objects, there is not much that can be done to resolve this particular issue. If the targets were of particular interest, then the sensor vehicle could be actively positioned to reduce the uncertainty in the object identities, or a secondary sensing vehicle could be used to get a better perspective of the targets. These solutions are beyond the scope of this thesis, however it is worth noting that a probabilistic identity tracking solution such as this is an ideal input for active perception. The control loop can be closed around this measure of identity uncertainty.

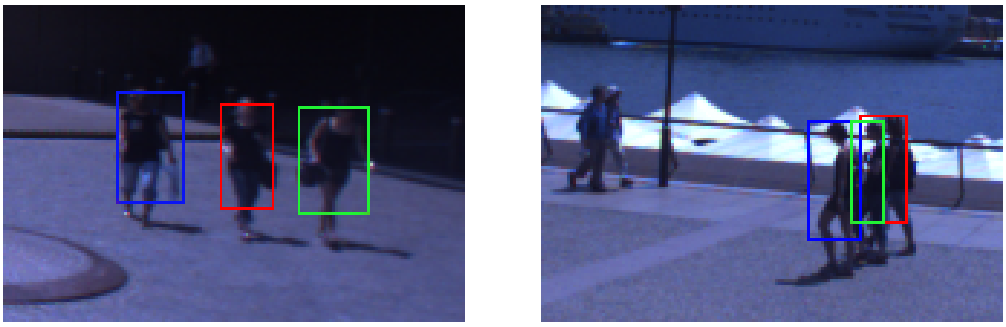


(a) HGraph **(b) BNet** **(c)**



(d) Trajectories from **HGraph** **(e)** Trajectories from **BNet**

Figure 5.34 – Event graph solutions via the **HGraph** and **BNet** methods. **BNet** gets closer to the correct solution, but switches the ordering of the three pedestrians after node 425. The likelihood table (c), in which each cell gives the likelihood that that node (row) contains a given identity (column), shows that the **BNet** identity assignment has low confidence.



(a) (b)

Figure 5.35 – Example of the changing target perspective during tracking. Early in the dataset, the targets are arrange parallel to the camera's imaging plane, and so the appearance information from the three targets is well separated. Later during tracking, the targets are perpendicular to the image plane and the targets occlude each other.

This worked example qualitatively examined the performance of the tracking algorithms in an unconstrained urban setting, demonstrating both strengths and weaknesses of the event graph and appearance reasoning approaches. We showed in Figure 5.33 an important failure mode of the event graph construction algorithm (a potential solution to this problem is discussed in Chapter 6 as a topic for future work) and demonstrated the advantages of the **BNet** appearance reasoning algorithm, particularly with regard to the estimation of identity uncertainty.

5.2.2.2 CBD - Intersection

The previous section analysed the behaviour of the tracking systems on three closely spaced targets, observed walking together for a significant length of time (75 seconds). In this section we consider a section of data collected at a busy city intersection, during which 14 pedestrians cross paths. This is the same section of data used in Chapter 4 for the evaluation of appearance models and the change detection results from one scan were shown in Figure 5.29b.

A region of interest (ROI) around the intersection area was manually selected, and the trajectories of all objects within that region labelled. An overlay of the tracked objects projected onto an image from the forward facing camera is shown in Figure 5.36, along

with an overhead view of the trajectories.

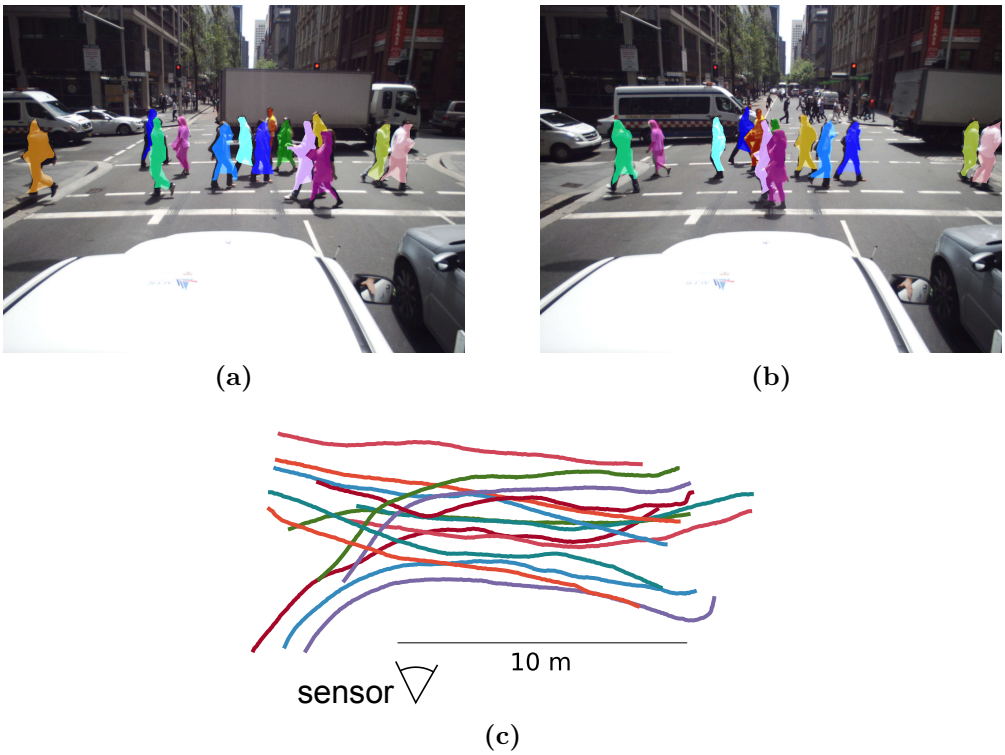


Figure 5.36 – Labelled objects overlaid on camera data, for the CBD intersection example. Frame (b) is approximately 1.25 seconds after frame (a). Image levels have been manually adjusted to improve contrast. (c) Shows the object trajectories, viewed from above.

Given these ground truth trajectories, we can compute the same supervised evaluation metrics that have been use throughout this thesis. The results of all the different tracking methods discussed in this thesis are shown in Table 5.2. This is a complex dataset and determining which tracking method had the best performance requires the comparison of all available different metrics.

According to the MOTP metric (where lower values are preferred) the **split-cost** had the best performance, followed closely by **split**. With their ability to correct under segmentation issues, these methods achieve higher positional accuracy. **Split-cost-graph** can also re-cluster observations, but sometimes creates group nodes. Group nodes result in a less accurate positional estimate, and therefore slightly higher MOTP.

	id-sensor	# tracks	MOTP	misses	FP	MOTA	homogeneity	completeness	vmeasure
centroids	N/A	16	0.118	637	**4**	0.791	0.937	0.916	0.926
split	N/A	15	0.091	270	23	0.903	0.987	0.981	**0.984**
split-cost	N/A	17	**0.088**	186	**4**	0.937	0.980	0.954	0.966
centroids-graph-HGraph	Ladybug	12	0.181	487	**4**	0.837	0.882	0.842	0.862
	Lidar	12	0.181	487	**4**	0.836	0.891	0.834	0.862
centroids-graph-BNet	Ladybug	12	0.181	487	**4**	0.841	0.918	0.958	0.937
	Lidar	12	0.181	487	**4**	0.842	0.956	**0.997**	0.976
split-cost-graph-HGraph	Ladybug	15	0.100	**94**	**4**	0.965	0.975	0.932	0.953
	Lidar	15	0.100	**94**	**4**	0.966	0.988	0.948	0.968
split-cost-graph-BNet	Ladybug	15	0.100	**94**	**4**	**0.967**	**0.989**	0.959	0.974
	Lidar	15	0.100	**94**	**4**	**0.967**	**0.989**	0.959	0.974

Table 5.2 – Tracking performance on the INTERSECTION example (FP refers to the number of false positives). There are 14 labelled objects in the dataset, but none of the methods manage to determine the correct number of tracks. Although **split** has the highest V-measure, it generates a large number of misses and false positives, and **split-cost-graph** has a higher MOTA score. **Split-cost-graph** combined with the **BNet** appearance reasoning results in the highest overall performance. **Split-cost** and **split** do a slightly better job of determining the locations of targets, although only by around 1cm on average.

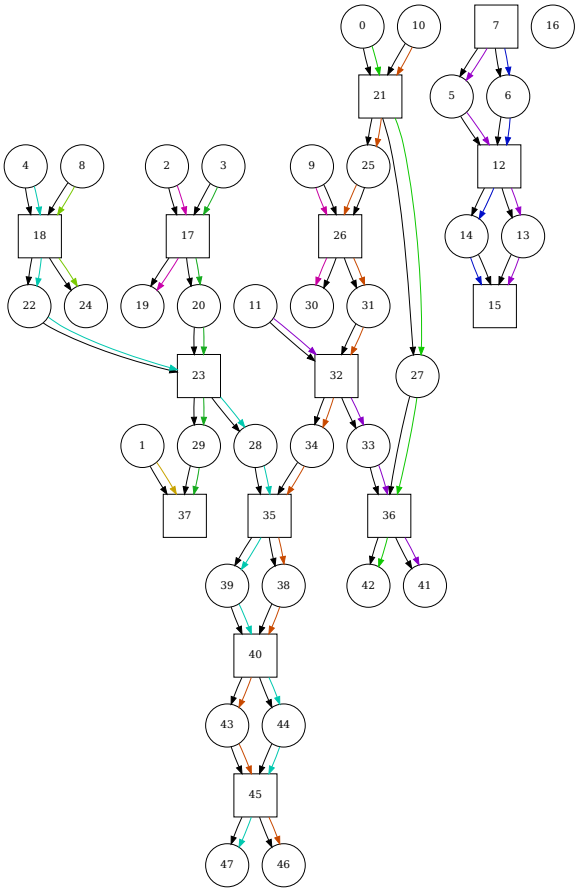
Split-cost-graph in combination with the **BNet** appearance reasoning algorithm achieved the highest performance in terms of MOTA and did so using either camera or lidar based appearance models. **Split-cost-graph** shows considerably fewer missed observations than the other methods and also achieves highest track homogeneity.

In terms of V-measure, however, **split** achieved the highest results. This illustrates a disadvantage of the V-measure score. Since it only evaluates identity decisions it must be considered with the additional context of false positives and misses. In this case, although **split** achieves highest V-measure it has 3 times the false positive rate, and over 5 times the false negative (miss) rate of **split-cost-graph**. Overall, therefore, **split-cost-graph-BNet** is the best performing tracking method. Despite V-measure's disadvantages in cases where there are a high number false positives and misses, its ability to measure the quality of identity assignments is important. For this reason, V-measure and it's component scores are included in Table 5.2.

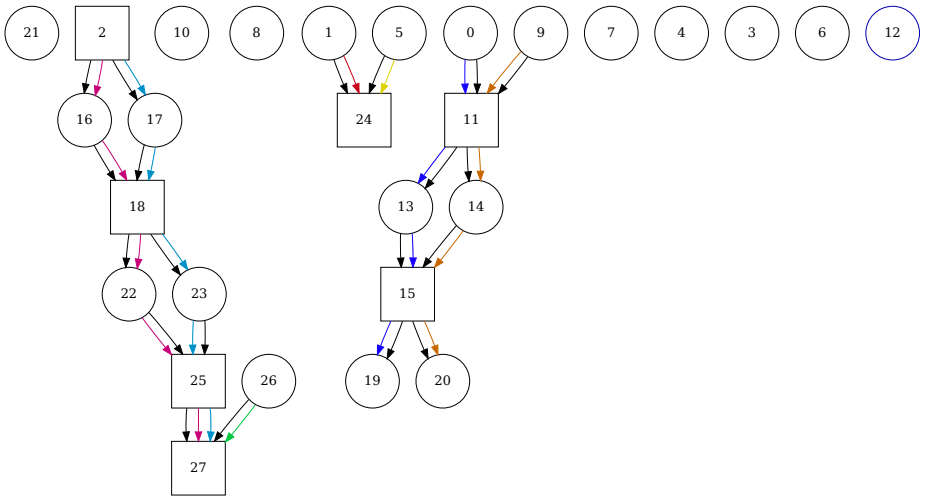
Figure 5.37 shows the event graphs generated by **centroids-graph** and **split-cost-graph** for the intersection scene. Recall that **centroids-graph** relies on the segmentation result provided to it and creates a group any time multiple tracks match to one observation. As a result, it is much more likely to create group nodes than **split-cost-graph**, which first attempts to re-cluster such observations. The propensity of **centroids-graph** towards creating group nodes is reflected in Figure 5.37a. The event graph created by **split-cost-graph** Figure 5.37b is not only simpler, but also more accurate, as demonstrated by the higher performance scores in Table 5.2.

The trajectories computed from a subsection of this data is shown in Figure 5.38. In this example, three pedestrians pass close to each another, two at a time, and **centroids-graph** groups them together. **split-cost-graph**, however, correctly tracks the objects without creating any merge/split events. This demonstrates the utility of the **split-cost-graph** approach. Periods of under segmentation that do not lead to identity ambiguity can be tracked accurately without resorting to appearance reasoning, and group nodes are only created when they represent genuine confusion between object identities.

The controlled experiments in Section 5.1 tested specific scenarios to show the im-

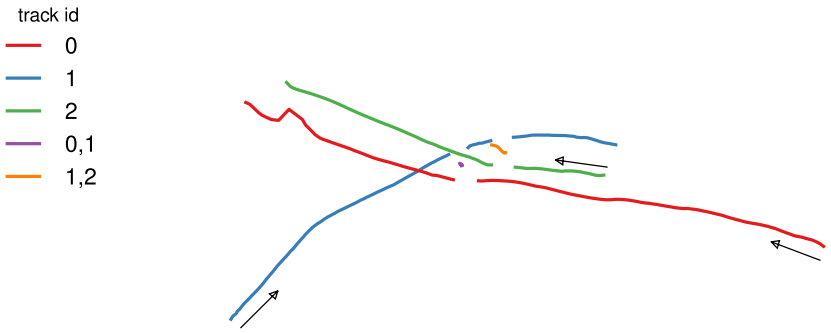


(a) centroids-graph

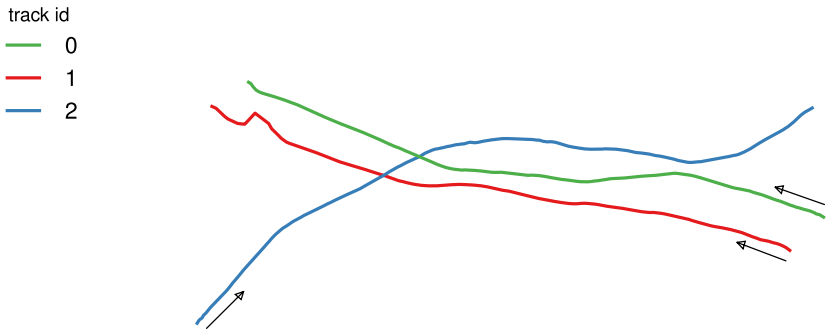


(b) split-cost-graph

Figure 5.37 – Event graphs created for the intersection scene. **Split-cost-graph** is less likely to create group nodes than **centroids-graph** as it is able to re-cluster observations. This is reflected by the much less complicated graph structure in (b).



(a) centroids-graph



(b) split-cost-graph

Figure 5.38 – Tracks extracted from the intersection example, as three pedestrians cross paths (moving in the directions indicated by arrows). As they cross paths **centroids-graph** creates group nodes and records two merge/split events (a), whilst **split-cost-graph** is able to keep the tracks separated (b).

proved performance for graph-based methods for particular types of object interactions. The unconstrained data evaluated in this section, however, contains all kinds of interactions – including some that are trivial and some that are roughly equivalent to those tested in the controlled experiments.

That the **split-cost-graph** methods only perform slightly better than **split** and **split-cost** in this example is likely due to the nature of the interaction patterns observed. Across a large dataset, the extent to which the graph-based approaches show an improvement over methods that don't consider target identity is tied to the frequency with which complex target interactions occur.

5.2.3 Overall results

In the previous section we considered specific tracking scenarios extracted from long datasets collected in real-world conditions on the roads of Sydney. These were only brief snapshots of much longer datasets, so in this section we present some statistics from each dataset as a whole. We compared the performance of **split-cost-graph** against **centroids** (the baseline approach) and **split-cost** (our conservative variant of **split**).

5.2.3.1 Opera House

During tracking, **split-cost-graph** records interactions between objects in an event graph. There is no ambiguity between the identities of objects that do not interact with each other, so to solve the event graph we can separate it into a number of subgraphs. From a graph theory point of view, we separate the graph into sets of nodes that are weakly connected [27]. Consider the event graph for the INTERSECTION scenario, shown in Figure 5.37b. This event graph contains 11 different subgraphs, 8 of which are trivial because they contain only one node. There is no ambiguity between identities of nodes in different subgraphs, since the objects never interacted.

As a result, we can solve the subgraphs independently of one another, using either of the algorithms described in Section 5.1. As we showed in previous experiments

BNet generally leads to better performance, but is only compatible with consistent graphs. **HGraph**, however, can be applied in all cases. Therefore, to maximise the performance of our system on this data we split the overall event graph into independent subgraphs, then apply the **BNet** method. If the graph is found to be inconsistent, we automatically switch to the **HGraph** appearance reasoning method instead.

Table 5.3 shows some statistics about the tracking result recorded by **split-cost-graph** on the OPERA-HOUSE dataset. During tracking 479 track sections we extracted, and of these 105 did not interact with any other targets. The remaining track sections were contained within of one of 45 subgraphs, which on average contained 8.3 nodes each.

The majority of graphs were consistent, meaning they could be solved using the **BNet** approach. After identity reasoning, the 479 track sections had been combined into a total of 261 tracks. The duration results in this table are for individual nodes, prior to appearance reasoning.

	opera-house graph stats
number of track sections	479
mean track section duration (s)	4.847
number of non-interacting tracks	105
mean duration of non-interacting track (s)	8.093
number of subgraphs (more than one node)	45
mean number of nodes per subgraph	8.311
number of subgraphs solved by **BNet**	30
number of subgraphs solved by **HGraph**	15
number of tracks extracted	261

Table 5.3 – Graph statistics for the OPERA-HOUSE dataset.

Table 5.4 shows some statistics for comparison against the other tracking methods evaluated on the same data. After appearance reasoning, **split-cost-graph** creates approximately the same number of independent tracks as **centroids** and **split-cost**. The average length of the tracks it extracts however are considerably longer; **split-cost-graph** shows on average a 16% increase in track distance and 19% increase in

track duration.

	centroids	split-cost	split-cost-graph
number of tracks	262	284	261
mean track distance (m)	10.957	10.395	12.679
mean track duration (s)	8.475	8.423	10.082

Table 5.4 – Overall statistics for the OPERA-HOUSE dataset.

A histogram of track durations for the three different methods is shown in Figure 5.39. **Split-cost**, as described previously, will conservatively kills tracks in order to preserve homogeneity, meaning that it results in a higher proportion of tracks in the 0 to 5 second range. **Split-cost-graph**, by combining track sections together, obtains longer track durations, results in a distribution that is shifted slightly to the right.

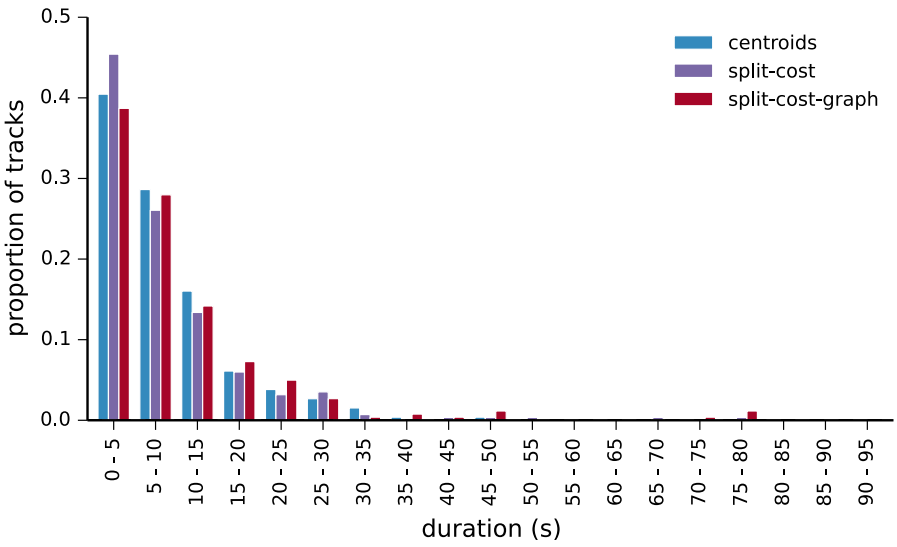


Figure 5.39 – Histogram of track durations from the OPERA-HOUSE dataset.

In a histogram like this, longer tracks are less visible. For example an object tracked correctly for 48 seconds results in a count of 1 in the 45 to 50 second bin. However, if the same track was broken into 24 pieces, each 2 seconds long, then the histogram would show 24 counts in the 0 to 5 second range. This results in the histograms being skewed heavily to the left.

A less biased way to present the same information is to compute the histogram on a

per-observation basis, weighting each track by the number of observations in it. At a rate of, 20 observations per second (as provided by the Velodyne) the same example would result in a histogram 960 counts in the 45 to 50 second range in the first case, and 960 counts in the 0 to 5 second range in the second. This observation-weighted histogram is shown in Figure 5.40.

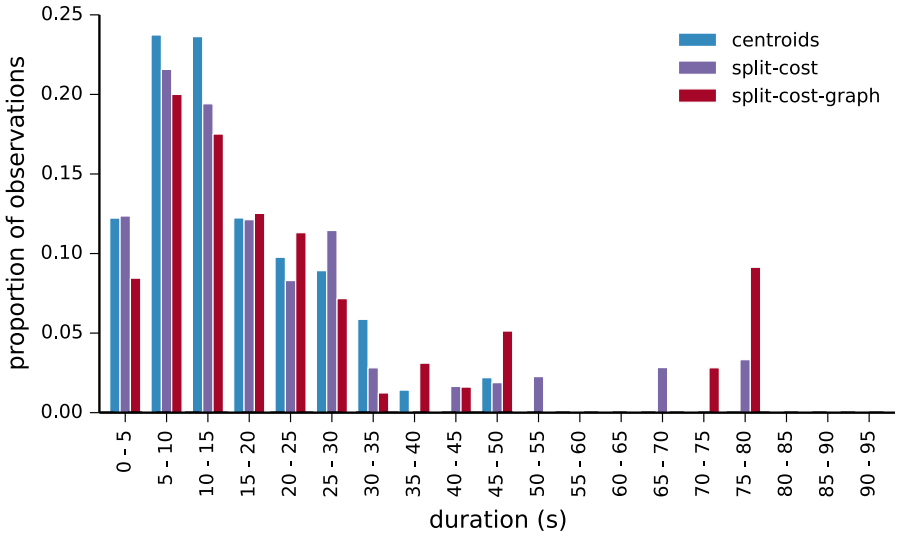


Figure 5.40 – Histogram of track durations from the OPERA-HOUSE dataset, weighted by the number of observations in each track.

This representation emphasises the ability of **split-cost-graph** to extract longer tracks and reveals that **split-cost** extracted some longer tracks than **centroids** too. Whilst **split-cost** does, in times of ambiguity, cut tracking short to preserve homogeneity, resulting in a shorter mean track duration (shown in Table 5.4) it also has the ability to re-cluster observations, meaning that sometimes it can maintain a track through instances of under segmentation that cause **centroids** to fail.

5.2.3.2 CBD

A similar analysis was performed for the CBD dataset. The graph statistics shown in Table 5.5 reveal that a total of 5565 track sections were created. Of these 2466 were independent, i.e. objects that did not interact with any other objects. The

remaining track sections were divided between 454 subgraphs, each with an average of 6.8 nodes. As was the case for the OPERA-HOUSE dataset, the majority of graphs could be solved using the **BNet** appearance reasoning method, leaving 96 to be solved by the fall-back method, **HGraph**.

	cbd graph stats
number of track sections	5565
mean track section duration (s)	3.616
number of non-interacting tracks	2466
mean duration of non-interacting track (s)	4.335
number of subgraphs (more than one node)	454
mean number of nodes per subgraph	6.826
number of subgraphs solved by **BNet**	358
number of subgraphs solved by **HGraph**	96
number of tracks extracted	3691

Table 5.5 – Graph statistics for the CBD dataset.

In terms of the comparison with **centroids** and **split-cost**, **split-cost-graph** again computed longer tracks. The amount of improvement, however, was reduced with **split-cost-graph** showing a 3.7% increase in track distance and a 4.7% increase in track duration.

	centroids	split-cost	split-cost-graph
number of tracks	4027	3899	3691
mean track distance (m)	7.755	8.128	8.429
mean track duration (s)	4.941	5.398	5.651

Table 5.6 – Overall statistics for the CBD dataset.

This small improvement means that the differences in histograms are more subtle. The track duration histogram is shown in Figure 5.41 and the observation-weighted version is shown in Figure 5.42.

The tracking statistics for the OPERA-HOUSE and CBD experiments presented here demonstrate that the **split-cost-graph** approach achieves an increase in average track length. Coupled with prior experiments showing that **split-cost-graph** extracts the most homogeneous tracks across a wide variety of interaction scenarios,

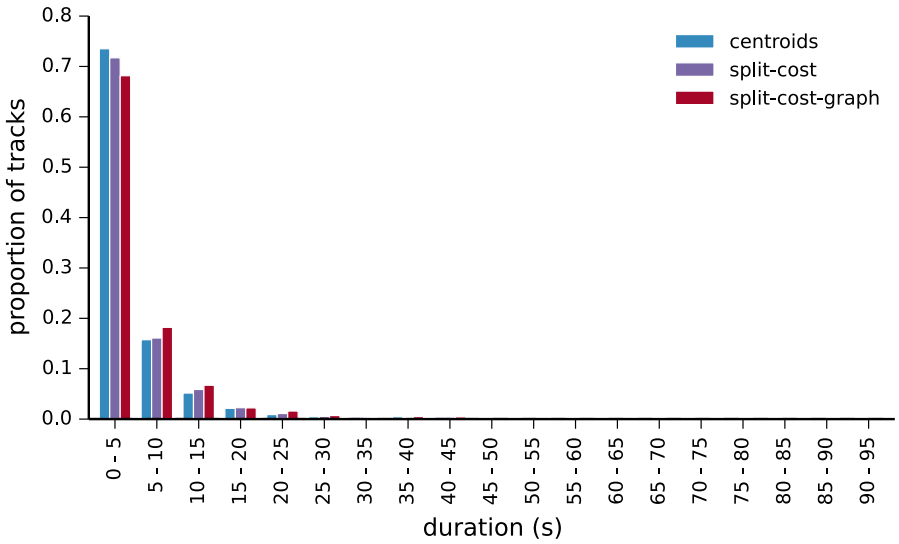


Figure 5.41 – Histogram of track durations from the CBD dataset.

the results in this section demonstrate the utility of graph-based tracking coupled with identity reasoning.

5.2.4 Discussion

The track distance and duration results presented above are unsupervised metrics and thus do not allow the same depth of analysis as the supervised metrics used in previous sections. However, given the size of the datasets, manual labelling (a prerequisite for supervised evaluation) is infeasible.

The results do, however, demonstrate that the **split-cost-graph** method is able to link track sections together into complete object trajectories that show an improvement in length of other methods. Results presented in Sections Section 5.1.4 and Section 5.2.2 give us confidence that our methods will also achieve higher identity-tracking performance on these datasets.

Visual inspection of the tracks and graph solutions obtained on the OPERA-HOUSE and CBD datasets leads us to the following conclusions.

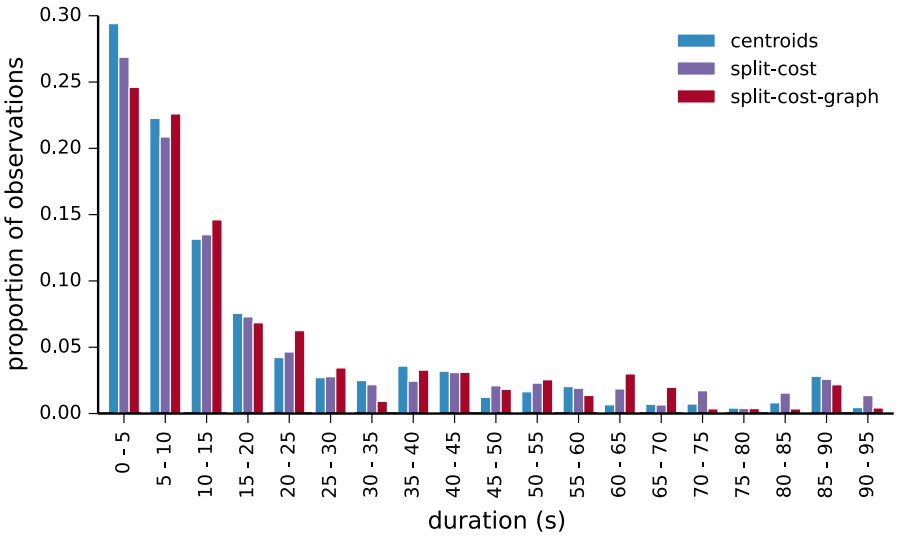


Figure 5.42 – Histogram of track durations from the CBD dataset, weighted by the number of observations in each track.

Tracking works best at close ranges

Inspection of individual sub-graphs leads us to conclude that there is a strong relationship between range to targets and graph consistency. Inconsistent graphs are often caused by interactions near the limits of sensor range.

This is unsurprising given the nature of that data. Cameras and rotating lidar sensors such as the Velodyne have a fixed angular resolution. This means that as range increases, the Cartesian resolution of the data decreases and fewer measurements are sampled from each target. Reduced resolution in the point cloud data makes data association more ambiguous, which can result in more grouping events, or incorrect graphs. Additionally, the reduced resolution in the image data extracted for targets means that appearance models become less discriminative.

Object-specific methods would likely improve performance

In this work we focused on tracking and appearance modelling without consideration of the particular type of objects involved. Pedestrians, cyclists, trucks, cars and buses were all segmented in the same pipeline and tracked under the same framework. In the

scope of this thesis, we could not consider the state of the art detection, segmentation and classification methods for each and every object type. The methods presented in this thesis, could, however, be used in conjunction with object-specific methods to potentially boost performance beyond what either framework can achieve on its own.

One area in particular in which existing approaches could improve the performance of our methods is vehicle tracking. The teams in the DUC, for example, used strong shape priors and more specific motion models to navigate amongst traffic and avoid collisions. Vehicle-specific models would also improve segmentation.

The primary assumption of the segmentation method (that objects can be separated based on the distance between neighbouring points) is violated for larger objects, particularly in the presence of occlusion. This results in over-segmentation, an example of which is shown in Figure 5.43.

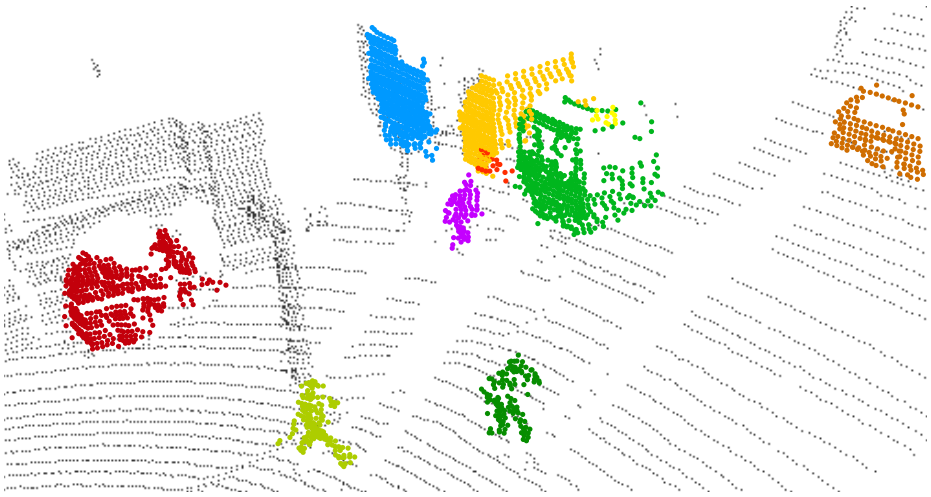


Figure 5.43 – Over-segmentation of a truck in the CBD dataset.

Whilst our tracking system can handle occlusions and segmentation errors, changing segmentation often meant that the tracker would switch between tracking parts of a vehicle separately, and representing the whole vehicle as a group. In many cases, the appearance reasoning methods allow full object trajectories to be extracted but the meaning of the result can be unclear.

For an application specific deployment of our tracking system, we would recommend the approaches developed in this thesis in conjunction with the best available object-

specific and application-specific methods for segmentation, classification and motion prediction.

5.3 Summary

This chapter evaluated a full framework for robust identity tracking using appearance information to solve the event graphs computed from 3D lidar data.

In Section 5.1 we discussed two frameworks for event graph based identity reasoning. The first, **HGraph** is a greedy, local approach from [106]. The second, **BNet** [77], uses a Bayesian network to achieve an approximation to the globally optimal graph solution. We adapted both of these methods from the computer vision literature and successfully applied them to 3D data. Additionally, we proposed an extension to **BNet**, which increases the range of graphs to which the method can be applied.

Through detailed experimentation we showed that these graph reasoning methods, when combined with our proposed **split-cost-graph** graph extraction method from Chapter 3 consistently achieve the best results in terms of identity tracking performance. We analysed the ability of **BNet** to provide an estimate of identity uncertainty, which could be used to "close the loop" around perception and control, and demonstrated that in many cases lidar data provides sufficient appearance information to resolve target identity.

Section 5.2 describes the application of our proposed methods to data gathered in unconstrained urban environments. We examined some specific tracking cases extracted from the larger datasets and presented overall tracking statistics. Building upon the results in Section 5.1, which showed that our tracking methods reliably compute homogeneous trajectories, we demonstrated that on these large datasets our methods achieve higher average track duration. This indicates that the **HGraph** and **BNet** approaches were able to use appearance information to combine the track sections extracted by **split-cost-graph** into full object trajectories.

Chapter 6

Conclusion

This thesis has proposed methods that allow for the tracking of multiple, interacting targets in complex urban scenes. It focused, in particular, on the task of maintaining the identity of targets in situations where phenomena such as occlusions and interactions may cause them to become ambiguous.

To address this task, this thesis proposed new methods for tracking closely-spaced targets using 3D lidar data (including the adaptation of a graph-building strategy that explicitly represents identity ambiguity), developed appearance models that allow objects to be recognised amongst a set of candidate tracks, and adapted and extended existing frameworks that combine track graphs and appearance models to solve the identity management problem. The methods were validated on large, manually labelled datasets consisting of both planned interaction scenarios and unconstrained data collected in an urban environment, showing improved performance in complex real-world scenarios where these phenomena occur.

This chapter summarises the contributions of this thesis and presents avenues for future research.

6.1 Summary of contributions

The identity tracking framework developed in this thesis is shown in Figure 6.1. The system makes use of both lidar and camera data and we discussed the processing steps required to extract the observations that form the input to the tracking system. Lidar-based segmentation and change detection methods were presented, and the requirements relating to the fusion of lidar and camera data described. The methods developed in this thesis extract "track sections", homogeneous sets of observations that represent a consistent object or set of objects. An appearance model is computed for each track section. The relationships between these track sections are maintained in an event graph, and identity reasoning methods are used to determine complete object trajectories.

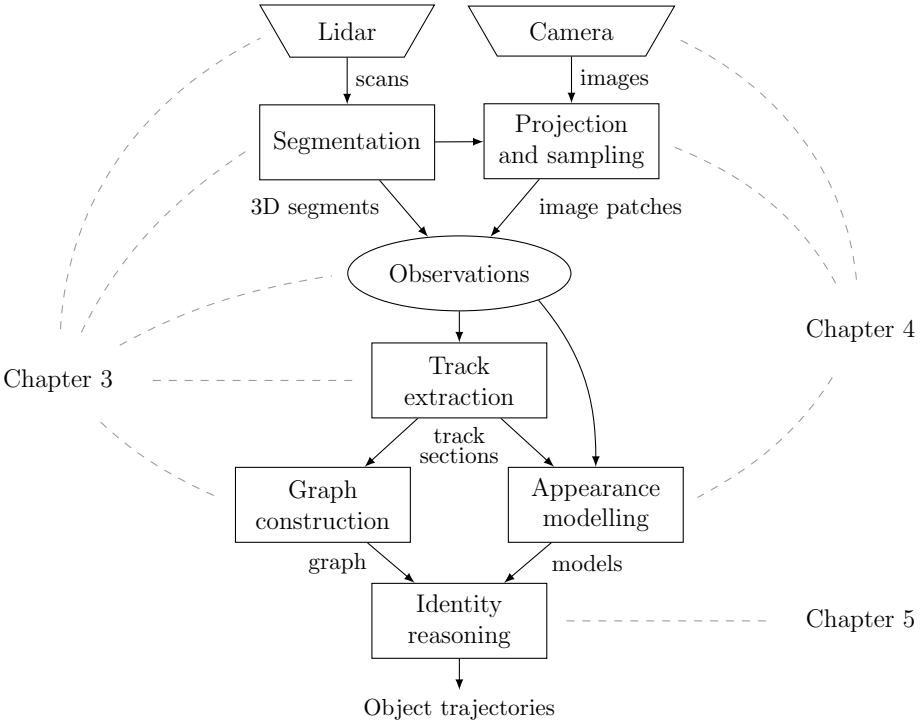


Figure 6.1 – The tracking framework developed in this thesis. Modules are linked to the chapters in which they were discussed.

The development of this framework was described across three chapters that explored lidar based tracking, appearance models and identity reasoning. To evaluated iden-

tity tracking performance, we proposed the adoption of the V-measure metric from clustering analysis. The contributions of this work are summarised below.

6.1.1 Tracking performance evaluation

In order to understand the performance of the tracking methods developed in this thesis, performance measures are required. We reviewed a standard method from the literature, the CLEAR MOT metrics, and demonstrated that MOTA does not adequately capture the ability of a system to maintain object identity.

In Section 2.4 we proposed that V-measure, from the field of clustering analysis, be adopted as a metric for identity tracking performance. Homogeneity and completeness, the components of V-measure, reflect important qualities of a tracking system and were used throughout this thesis to understand tracking behaviour.

6.1.2 Tracking interacting objects with 3D lidar

In Chapter 3 we developed lidar based tracking methods that allow accurate tracking of interacting targets in complex scenes. We proposed **split-cost**, a method that offers improved robustness against under segmentation whilst achieving a significant increase in track homogeneity and validated this method using a large dataset of manually labelled interaction scenarios.

The analysis of **split-cost** motivated the development of tracking methods which could maintain this level of homogeneity on extracted track sections whilst maintaining a record of object interactions. We adapted an event graph strategy from the computer vision literature and demonstrated the effectiveness of our method.

6.1.3 Appearance modelling

The lidar based tracking methods developed in Chapter 3 explicitly represent identity ambiguity but do not provide a way to resolve target interactions. A target's identity

is not directly observable, but the methods presented in Chapter 4 provide a way to learn information sufficient to recognise targets within the constraints imposed by the event graph.

In Chapter 4 we explored the appearance modelling parameters relevant to a system designer through extensive experimentation using real-world data, using a testing scheme that simulated arbitrary object interactions from labelled data. This experimentation demonstrated important properties of the identity recognition problem and motivated the use of event graph reasoning approaches that compute joint identity likelihoods.

6.1.4 Robust identity tracking

Chapter 5 identified two frameworks that use appearance models to solve the identities of nodes in an event graph, **HGraph** and **BNet**. We adapted these methods to work with the track graphs obtained from lidar data, and proposed an extension to the **BNet** method that increases the range of graphs that it is able to solve.

An analysis of tracking performance, both in terms of metric accuracy and identity maintenance was presented and the ability of the **BNet** framework to measure its own reduction in identity uncertainty was explored. These experiments demonstrated that the combination of event graphs and appearance reasoning achieves increases in identity tracking performance. Finally, the results of a large scale experiment, performed in an unconstrained urban environment, were discussed.

6.2 Future Directions

Perception in dynamic urban environments remains a difficult problem. Whilst the methods discussed in this thesis go some way to addressing identity ambiguity due to phenomena such as occlusions and interactions, the complex nature of urban scenes means that robust identity tracking remains a challenge.

The methods developed in this thesis lead to some immediate areas of future research and development:

- The event graphs described in Chapter 3 are recorded online during tracking. Whilst group nodes provide for multiple hypotheses over object identities, they do not allow the tracking system to go back and adjust the merge/split events on the basis of future evidence. For offline applications it would be advantageous to consider the complete sequence of observations when constructing the graph. For example, a newly appearing track could be predicted backwards in time to see if it should in fact be linked to a group splitting event. This would likely solve the problem shown in Figure 5.33 and improve event graph consistency.

- For deployment in a real-world tracking system, the tracking framework described in this thesis should be combined with the current state of the art vehicle tracking systems. Knowledge of road layout including traffic direction and lanes provides strong priors for vehicle tracking and determination of object type can also improve segmentation performance. The parts of a scene not tracked by vehicle-specific models could then be processed using our more generic identity-tracking framework.

- Chapter 4 considered appearance models built from either lidar or camera data. Reliability could likely be increased by combining the two modalities in a joint feature space. Multi-modal classification and recognition is a large area of research and these techniques could likely be applied directly.

- An alternative way of using appearance information from multiple sensors is via a cascade classifier. Given we are already tracking objects with lidar data, simple geometric features are cheap to compute, and Chapter 4 showed that these features are often powerful enough to determine object identity. As the system has the ability to monitor its own uncertainty (the **BNet** approach), then identity decisions that remain ambiguous after the application of lidar-based appearance models can be deferred to camera-based models, which in this work were shown to be more accurate. This approach may be preferable to

the multi-modal models mentioned in the previous bullet point, as the camera features may incur a significant computational cost.

- In this thesis, positional tracking and event graph construction were performed completely independently of appearance reasoning. This had the advantage of allowing us to develop and validate the systems separately, and also evaluate the contribution of each part of the system to overall tracking performance. However feedback from identity reasoning back into tracking could increase the capability of the system. In particular, the ability to determine the number of objects in each node of the graph (as provided by **BNet**) could allow better re-clustering strategies. Also, identity recognition would aid in determining if an newly-observed object is in fact a new target, or a previously observed target emerging from an occlusion.

- The availability of identity uncertainty from the **BNet** method enables the development of "active perception" approaches, in which feedback from perception becomes part of the control loop. The integration of tracking and control would allow sensor platforms to plan trajectories which maximise information gain and reduce identity uncertainty.

Bibliography

[1] CLassification of Events Activities and Relationships (CLEAR) Evaluation and Workshop. `http://www.clear-evaluation.org/`. 2007.

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[3] K. Arras, B. Lau, S. Grzonka, M. Luber, O. Mozos, D. Meyer-Delius, and W. Burgard. Range-based people detection and tracking for socially enabled service robots. In E. Prassler, M. Zllner, R. Bischoff, W. Burgard, R. Haschke, M. Hgele, G. Lawitzky, B. Nebel, P. Plger, and U. Reiser, editors, *Towards Service Robots for Everyday Environments*, volume 76 of *Springer Tracts in Advanced Robotics*, pages 235–280. Springer Berlin Heidelberg, 2012.

[4] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. V. Covern, and M. Webster. Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.

[5] Y. Bar-Shalom. Tracking methods in a multitarget environment. *Automatic Control, IEEE Transactions on*, 23(4):618 – 626, August 1978.

[6] Y. Bar-Shalom, X. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. Wiley-Interscience, 2001.

[7] H. Becker. *Identification and characterization of events in social media*. PhD thesis, Columbia University, 2011.

[8] C. B. Bell. Mutual information and maximal correlation as measures of dependence. *The Annals of Mathematical Statistics*, 33(2):587–595, 06 1962.

[9] H. Ben Shitrit, J. Berclaz, F. Fleuret, and P. Fua. Tracking multiple people under global appearance constraints. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 137–144. IEEE, 2011.

[10] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008, 2008.

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

[12] A. Bialkowski, S. Denman, S. Sridharan, C. Fookes, and P. Lucey. A database for person re-identification in multi-camera surveillance networks. In *Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on*, pages 1–8, 2012.

[13] S. Birchfield and S. Rangarajan. Spatiograms versus histograms for region-based tracking. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1158 – 1163 vol. 2, june 2005.

[14] Bosch Automotive Technology. *LRR3: 3rd generation long-range radar sensor*, 2009.

[15] M. Bosse, R. Zlot, and P. Flick. Zebedee: Design of a spring-mounted 3-D range sensor with application to mobile mapping. *Robotics, IEEE Transactions on*, 28(5):1104 –1119, oct. 2012.

[16] A. Bruce and G. Gordon. Better motion prediction for people-tracking. In *ICRA 2004*, 2004.

[17] R. Cavallaro, M. Hybinette, M. White, and T. Balch. Augmenting live broadcast sports with 3d tracking information. *MultiMedia, IEEE*, 18(4): 38–47, 2011.

[18] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[19] C. Chang and S. Chatterjee. Quantization error analysis in stereo vision. In *Signals, Systems and Computers, 1992. 1992 Conference Record of The Twenty-Sixth Asilomar Conference on*, pages 1037–1041 vol.2, 1992.

[20] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724 –2729 vol.3, April 1991.

[21] D. Clark and S. Godsill. Group target tracking with the gaussian mixture probability hypothesis density filter. In *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*, pages 149 –154, December 2007.

[22] J. Coales, H. Calpine, and D. Watson. Naval fire-control radar. *Electrical Engineers - Part IIIA: Radiolocation, Journal of the Institution of*, 93(2): 349–379, 1946.

[23] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In C. Schmid, S. Soatto, and C. Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[24] M. Darms, P. Rybski, and C. Urmson. Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1197 –1202, 4-6 2008.

[25] M. Darms, P. Rybski, C. Baker, and C. Urmson. Obstacle detection and tracking for the urban challenge. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3):475 –485, sept. 2009.

[26] B. Davies. A review of robotics in surgery. *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, 214(1): 129–140, 2000.

[27] R. Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, second edition, 2000.

[28] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel. On the Segmentation of 3D LIDAR Point Clouds. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2011.

[29] B. Douillard, J. Underwood, N. Melkumyan, S. Singh, S. Vasudevan, and C. Brunner. Environment modeling methods for autonomous operations in semi-urban environments. *ACFR Internal*, 2009.

[30] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4): 551–559, 1983.

[31] A. Ess, B. Leibe, K. Schindler, and L. Van Gool. A mobile vision system for robust multi-person tracking. In *NIPS Robotics Workshop*, 2008.

[32] Federal Aviation Administration. Automatic dependent surveillance-broadcast (ads-b). 2011.

[33] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *Oceanic Engineering, IEEE Journal of*, 8(3):173 – 184, jul 1983.

[34] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[35] K. Goto, K. Kidono, Y. Kimura, and T. Naito. Pedestrian detection and direction estimation by cascade detector with multi-classifiers utilizing feature interaction descriptor. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 224–229, 2011.

[36] M. S. Grewal, L. R. Weill, and A. P. Andrews. *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2001.

[37] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree. Blensor: Blender sensor simulation toolbox. In *Advances in Visual Computing*, volume 6939 of *Lecture Notes in Computer Science*, pages 199–208. Springer Berlin Heidelberg, 2011.

[38] L. Guibas. The identity management problem; a short survey. In *Information Fusion, 2008 11th International Conference on*, pages 1 –7, July 2008.

[39] E. Guizzo. How google's self-driving car works. 2011. Retrieved 07-Nov-2013.

[40] D. Helbing, I. Farkas, P. Molnàr, and T. Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. In M. Schreckenberg and S. D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 21–58, Berlin, 2002. Springer.

[41] E. Hellinger. Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *J. Reine Angew. Math.*, 136:210–271, 1909.

[42] M. Himmelsbach, T. Luettel, and H.-J. Wuensche. Real-time object classification in 3d point clouds using point feature histograms. In *IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, pages 994–1000, Piscataway, NJ, USA, 2009. IEEE Press.

[43] M. Himmelsbach, A. Müller, T. Lüttel, and H.-J. Wünsche. Lidar-based 3d object perception. In *Proceedings of 1st International Workshop on Cognition for Technical Systems*, München, October 2008.

[44] Hitachi. http://www.hitachi-hta.com/products/advanced-materials/electronic-devices/nippon-signal-mems-3d-laser-sensor.

[45] S. Hoogendoorn and P. Bovy. Gas-kinetic modeling and simulation of pedestrian flows. *Transportation Research Record: Journal of the Transportation Research Board*, 1710(-1):28–36, January 2000.

[46] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15:225–263, 1996.

[47] A. E. Johnson. *Spin-Images: A representation for 3-D surface matching*. PhD thesis, Carnegie Mellon University, 1997.

[48] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart. Generative object detection and tracking in 3d range data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3075–3081, 2012.

[49] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[50] B. Kalyan, K. Lee, S. Wijesoma, D. Moratuwage, and N. Patrikalakis. A random finite set based detection and tracking using 3d lidar in dynamic environments. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 2288 –2292, 2010.

[51] B. Kluge, C. Kohler, and E. Prassler. Fast and robust tracking of multiple moving objects with a laser range finder. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1683 – 1688 vol.2, 2001.

[52] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

[53] H. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[54] C.-H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1217–1224, 2011.

[55] B. Lau, K. Arras, and W. Burgard. Multi-model hypothesis group tracking and group size estimation. *International Journal of Social Robotics*, 2(1): 19–30, March 2010.

[56] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727 – 774, October 2008.

[57] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4372 –4378, may 2010.

[58] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[59] J. Levinson and S. Thrun. Unsupervised calibration for multi-beam lasers. In *Proc. of the International Symposium on Experimental Robotics (ISER)*, 2010.

[60] J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Proc. of Robotics: Science and Systems*, 2013.

[61] L. Lin, Y. Bar-Shalom, and T. Kirubarajan. Track labeling and phd filter for multitarget tracking. *Aerospace and Electronic Systems, IEEE Transactions on*, 42(3):778 –795, july 2006.

[62] M. Lindstrom and J.-O. Eklundh. Detecting and tracking moving objects from a mobile platform using a laser range scanner. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 3, pages 1364 –1369 vol.3, 2001.

[63] M. Luber and K. O. Arras. Multi-hypothesis social grouping and tracking for mobile robots. In *Proc. of Robotics: Science and Systems*, 2013.

[64] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras. People tracking with human motion predictions from social forces. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 464 –469, 3-7 2010.

[65] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[66] R. Mahler. Multitarget bayes filtering via first-order multitarget moments. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1152 – 1178, 2003.

[67] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The Stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[68] F. Moosmann and C. Stiller. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393 –398, june 2011. **.

[69] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. pages 215 –220, 2009.

[70] F. Moosmann and T. Fraichard. Motion estimation from range images in dynamic outdoor scenes. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 142 –147, 3-7 2010.

[71] P. Morton, B. Douillard, and J. Underwood. An evaluation of dynamic object tracking with 3D LIDAR. In *Proc. of the Australasian Conference on Robotics & Automation (ACRA)*, 2011.

[72] P. Morton, B. Douillard, and J. Underwood. Multi-sensor identity tracking with event graphs. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4742–4748, 2013.

[73] M. Mucientes and W. Burgard. Multiple hypothesis tracking of clusters of people. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 692 – 697, 2006.

[74] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):pp. 32–38, 1957.

[75] K. Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

[76] L. E. Navarro-Serment, C. Mertz, N. Vandapel, and M. Hebert. Ladar-based pedestrian detection and tracking. In *Proc. 1st. Workshop on Human Detection from Mobile Robot Platforms, IEEE ICRA 2008*. IEEE, May 2008.

[77] P. Nillius, J. Sullivan, and S. Carlsson. Multi-target tracking - linking identities using bayesian network inference. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2187 – 2194, 2006.

[78] G. Pandey, J. McBride, S. Savarese, and R. Eustice. Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. In *IAV2010*, 2010.

[79] K. Panta, D. Clark, and B.-N. Vo. Data association and track management for the gaussian mixture probability hypothesis density filter. *Aerospace and Electronic Systems, IEEE Transactions on*, 45(3):1003 –1016, july 2009.

[80] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26:123–139, 2009.

[81] Point Grey Research, Inc. http://www.ptgrey.com/products/ladybug3/.

[82] A. Quadros, J. Underwood, and B. Douillard. An occlusion-aware feature for range images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4428–4435, 2012.

[83] D. Reid. An algorithm for tracking multiple targets. *Automatic Control, IEEE Transactions on*, 24(6):843 – 854, dec 1979.

[84] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[85] S. Schneider, M. Himmelsbach, T. Luettel, and H.-J. Wuensche. Fusing vision and lidar - synchronization, correction and occlusion reasoning. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 388 –393, 2010.

[86] F. Schöler, J. Behley, V. Steinhage, D. Schulz, and A. B. Cremers. Person tracking in three-dimensional laser range data with explicit occlusion adaption. 2011.

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

[88] D. Schulz, W. Burgard, D. Fox, and A. Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99, 2003.

[89] M. Schutz, N. Appenrodt, J. Dickmann, and K. Dietmayer. Simultaneous tracking and shape estimation with laser scanners. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 885–891, 2013.

[90] R. Sea. An efficient suboptimal decision procedure for associating sensor data with stored tracks in real-time surveillance systems. In *Decision and Control, 1971 IEEE Conference on*, volume 10, pages 33–37, 1971.

[91] J. Shackleton, B. VanVoorst, and J. Hesch. Tracking people with a 360-degree lidar. In *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, pages 420 – 426, 2010.

[92] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593 –600, jun 1994.

[93] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart. A layered approach to people detection in 3d range data. In *Proc. of The AAAI Conference on Artificial Intelligence: Physically Grounded AI Track (AAAI)*, 2010.

[94] L. Spinello, M. Luber, and K. O. Arras. Tracking people in 3d using a bottom-up top-down people detector. In *Proc. of The International Conference in Robotics and Automation (ICRA)*, 2011.

[95] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[96] C. Studholme, D. Hill, and D. Hawkes. An overlap invariant entropy measure of 3d medical image alignment. *Pattern Recognition*, 32(1):71 – 86, 1999.

[97] J. Sullivan and S. Carlsson. Tracking and labelling of interacting multiple targets. *Computer Vision–ECCV 2006*, pages 619–632, 2006.

[98] SymPy Development Team. *Sympy: Python library for symbolic mathematics*, 2013.

[99] J. Tanaka, D. Weiskopf, and P. Williams. The role of color in high-level vision. *Trends in cognitive sciences*, 5(5):211–215, 2001.

[100] A. Technologies. Ptz auto-tracking module.

[101] A. Teichman and S. Thrun. Tracking-based semi-supervised learning. *The International Journal of Robotics Research*, 31(7):804–818, 2012.

[102] The Blender Foundation. Blender. `http://www.blender.org/`.

[103] D. J. Therriault, R. H. Yaxley, and R. A. Zwaan. The role of color diagnosticity in object recognition and representation. *Cognitive Processing*, 10(4):335–342, 2009.

[104] S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, D. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. Whittaker. Autonomous exploration and mapping of abandoned mines. *Robotics Automation Magazine, IEEE*, 11(4):79–91, 2004.

[105] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[106] A. Torabi and G.-A. Bilodeau. A multiple hypothesis tracking method with fragmentation handling. In *Canadian Conference on Computer and Robot Vision*, pages 8–15, May 2009.

[107] A. Torabi, G. Massé, and G.-A. Bilodeau. An iterative integrated framework for thermal visible image registration, sensor fusion, and people tracking for video surveillance applications. *Computer Vision and Image Understanding*, 116(2):210 – 221, 2012.

[108] J. Underwood, A. Hill, T. Peynot, and S. Scheding. Error modeling and calibration of exteroceptive sensors for accurate mapping applications. *Journal of Field Robotics*, 27(1):2–20, 2010.

[109] J. Underwood, D. Gillsjo, T. Bailey, and V. Vlaskine. Explicit 3d change detection using ray-tracing in spherical coordinates. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4735–4741, 2013.

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

[111] Velodyne Lidar. http://www.velodynelidar.com/lidar/hdlabout/origins.aspx.

[112] Velodyne Lidar. *HDL-64E S2 Users Manual*, 2008.

[113] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[114] C. F. Wakim, S. Capperon, and J. Oksman. A markovian model of pedestrian behavior. In *2004 IEEE lnternational Conference on Systems, Man and Cybernetics*, 2004.

[115] D. Wang, I. Posner, and P. Newman. What could move? finding cars, pedestrians and bicyclists in 3d laser data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4038–4044, 2012.

[116] P. Withagen, K. Schutte, and F. Groen. Likelihood-based object detection and object tracking using color histograms and EM. In *Image Processing, 2002 IEEE International Conference on*, volume 1, pages 589–592, 2002.

[117] B. Yang and R. Nevatia. An online learned crf model for multi-target tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2034–2041. IEEE, 2012.

[118] B. Yang and R. Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *Computer Vision and*

Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 1918–1925. IEEE, 2012.

[119] A. R. Zamir, A. Dehghan, and M. Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[120] M. Zervos, H. B. Shitrit, F. Fleuret, and P. Fua. Facial descriptors for identity-preserving multiple people tracking. Technical report, Swiss Federal Institute of Technology, Lausanne (EPFL), 2013.