Using probabilistic graphical models to detect dynamic objects for mobile robots

by

Daniek Brink

Dissertation presented for the degree of Doctor of Engineering in the Faculty of Engineering at Stellenbosch University

Promoter: Dr C.E. van Daalen

Co-promoter: Prof. B.M. Herbst

December 2016

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2016 .....

Abstract

An autonomous mobile robot must be able to identify moving objects in its environment, continually as it is operating, for accurate environment mapping and collision-free navigation. This is not an easy task, since most of what might be observed will appear to be moving due to the robot's own motion. The task is further complicated by the inherent uncertainty in the pose estimates and environment measurements captured by the robot.

In this work we focus on features in the environment whose 3D locations are measured over time, such as triangulated stereo image features. Our aim is to separate dynamic features from static ones and also to group the dynamic ones into separate objects. Existing approaches generally assume that the exact pose of the robot is known at every time step or, in order to estimate the robot pose, they assume that the environment is predominantly stationary. We avoid these assumptions through thoughtful consideration for the uncertainties involved.

In order to model the uncertainties, as well as the statistical dependencies between observations and latent variables, we present a novel application of probabilisitic graphical models (PGMs) for dynamic object detection. Our PGM can be divided into two interacting components. The first relates to motion segmentation, in which all observed features are classified as static or dynamic, and the second relates to object segmentation, in which dynamic features on the same objects are clustered together. We also take care to accommodate for semi-static objects, which are objects that can be both stationary and dynamic during the observation period. Our design choices lead to a PGM containing both discrete and continuous variables. Tractable inference in such a hybrid model can be challenging, and we pay particular attention to this issue. It turns out that messages sent from continuous to discrete variables can be pre-computed, before loopy belief propagation is performed over the discrete variables.

Experiments on the KITTI benchmark datasets indicate that our PGM approach performs well, and it often outperforms a state-of-the-art feature-based algorithm. We find that motion segmentation accuracy tends to improve as more observations of the same features become available, and that our method has the ability to handle semi-static objects successfully. The ability of our PGM to segment different objects is also seen to perform superior.

Opsomming

'n Outonome vry-bewegende robot moet oor die vermoë beskik om voortdurend bewegende voorwerpe wat in sy omgewing voorkom te identifiseer, ten einde akkurate kartering en botsing-vrye navigasie te bewerkstellig. Die taak word verder gekompliseer deur die inherente onsekerheid in die postuurafskattings en omgewingsmetings wat die robot neem.

In hierdie proefskrif fokus ons op kenmerke in die omgewing waarvan die 3D posisies oor tyd gemeet word, soos verdriehoekte stereo-beeldkenmerke. Ons poog om dinamiese en statiese kenmerke van mekaar te skei asook om die dinamiese kenmerke in verskillende voorwerpe te groepeer. Bestaande benaderings aanvaar tipies dat die eksakte posisie en oriëntasie van die robot op elke tydstap bekend is of, in 'n poging om dit af te skat, word aanvaar dat die omgewing grotendeels staties is. Ons vermy hierdie aannames deur welbedagte oorwegings vir die betrokke onsekerhede.

Om hierdie onsekerhede asook die statistiese afhanklikhede tussen waarnemings en latente veranderlikes te modelleer, stel ons 'n nuwe toepassing van grafiese waarskynlikheidsmodelle vir dinamiese voorwerpherkenning voor. Ons grafiese model kan in twee wisselwerkende komponente verdeel word. Die eerste komponent is verwant aan bewegingsegmentering, waarin alle waargenome kenmerke as staties of dinamies geklassifiseer word, terwyl die tweede komponent verwant is aan voorwerpsegmentering, waarin dinamiese kenmerke op dieselfde voorwerp saam gegroepeer word. Ons tref voorsorg om semi-statiese voorwerpe, wat kan beweeg en stilstaan in die waargenome tydperk, te akkommodeer. Ons ontwerpkeuses lei tot 'n grafiese model wat diskrete en kontinue veranderlikes bevat. Uitvoerbare inferensie in so 'n hibriede model kan uitdagend wees en ons skenk veral aandag aan hierdie kwessie. Dit blyk dat boodskappe wat van kontinue na diskrete veranderlikes gestuur word, vooraf bereken kan word, voordat vertroue-sirkulasie ("loopy belief propagation") oor die diskrete veranderlikes uitgevoer kan word.

Eksperimente op die KITTI maatstaf-datastelle dui aan dat ons grafiese model benadering goed presteer en dat dit gereeld 'n vooraanstaande kenmerk-gebaseerde algoritme uitstof. Ons bevind dat die akkuraatheid van bewegingsegmentering neig om toe te neem namate meer waarnemings van dieselfde kenmerke beskikbaar word en dat ons metode die vermoë het om semi-statiese voorwerpe suksesvol te hanteer. Die vermoë van ons grafiese model om verskillende voorwerpe te segmenteer blyk ook beter te vaar.

Acknowledgments

I would like to express my sincere gratitude to my promoters, Dr Corné van Daalen and Prof. Ben Herbst, for their guidance and support, as well as to Prof. Johan du Preez for the use of his belief propagation implementation. I would also like to thank the DAAD-NRF and ARMSCOR for funding this work.

Contents

Chapter 1

Introduction

For successful operation an autonomous mobile robot must be able to navigate its environment safely. In particular, the robot must be able to identify obstacles in its environment and plan a path that avoids collisions. This by itself is not an easy task, because the onboard sensors used to detect obstacles may return measurements tainted by noise. Also, since the robot is moving through the environment, the locations of obstacles relative to the robot change over time. The robot's pose estimators may also be noisy, resulting in the robot becoming uncertain of both its own position and the geometry of the environment.

To complicate matters even further, many obstacles in the environment may be moving independently from the robot. In order to deal with this added problem the robot must be able to do three things. First it must detect dynamic objects and separate them from static objects. Next the robot has to track those dynamic objects over time and, finally, the robot must attempt to predict the trajectories of the dynamic objects before deciding on its own plan of action. In this work we focus on the first step towards navigating dynamic environments, namely dynamic object detection.

1.1 Challenges in dynamic object detection

Separating dynamic objects from static ones in a robot's environment presents a number of challenges. Firstly, in order to perceive obstacles in its environment, the robot is usually equipped with sensors that return measurements of the environment relative to the robot. These sensors can include stereo cameras, laser range scanners, radar or sonar. Measurements returned by environment sensors may be noisy due to sensor imperfections, random interference or the assumptions required to process raw data into a useful format. Consequently, environment measurements carry uncertainty, which leads to uncertainty regarding obstacle locations relative to the robot.

If the robot is moving while measurements are captured, it may appear from the sensor's point of view as if everything in the environment is moving. In order to distinguish between apparent movement of stationary objects and truly independent motion of dynamic objects, knowledge of the robot's own motion (ego-motion) is vital. For this reason the

robot is typically equipped with an inertial measurement unit (IMU), a global positioning system (GPS) or the more accurate differential GPS (DGPS). These sensors provide an estimate of the robot's pose (position and orientation) at different times.

Since these sensors can also return noisy measurements, it makes sense to combine pose estimates obtained from them with pose estimates determined from the locations of identified and tracked landmarks in the environment. For the robot to localize itself relative to the landmarks, the locations of the landmarks are required. However, in order to place the landmark locations in a global map, knowledge of the robot pose when the measurements are captured is required. This leads to a chicken-and-egg type problem, aptly named the simultaneous localization and mapping (SLAM) problem [21, 22]. Much research has been done on the SLAM problem and it is now considered solved on a theoretical level. However, existing solutions typically assume that the environment is stationary, which means that dynamic objects have to be identified and discarded before SLAM can be performed. A solution to this problem has been proposed in the form of SLAM with detection and tracking of moving objects (DATMO). There the estimation problem is factored into two separate estimates, thus reducing computational complexity [81].

Regardless of the manner in which localization estimates are obtained, they will always carry uncertainty. Therefore, ideally, localization estimates should be modelled as probability distributions over the pose variables. A small error in the estimate of robot orientation, for example, can have a drastic effect on the obstacle location estimates in the global map, especially for objects far away from the sensor. Since more accurate pose sensors are generally more expensive, it is desirable for a dynamic object detection system to incorporate and reason under significant pose uncertainty, rather than to neglect it.

The dynamic objects themselves also present some difficulties. We identify three types of objects that the robot may encounter in its environment. The first is a stationary object which remains stationary for the entire period that it is observed. The second type is a dynamic object that moves for the entire period. The third type, however, includes semi-static objects that are moving for only part of the period that it is observed. Examples include vehicles stopping at a stop sign, or parked cars that start to move while being observed. In our view a dynamic object detection system must be able to identify all three types of objects, track both dynamic and semi-static objects, and be able to handle measurement uncertainty as well as pose uncertainty.

1.2 Related work

We proceed with a review of existing work on the subject of dynamic object detection in mobile robotics. We split the review into two parts: motion segmentation and clustering.

1.2.1 Motion segmentation

The first step towards dynamic object detection is to separate measurements of the environment into measurements of static objects (including the background) and mea-

surements of dynamic objects. This process is called motion segmentation or motion detection. The literature on the subject is vast, and we limit this discussion to work representative of the main techniques.

1.2.1.1 Background subtraction

One of the earliest approaches to motion detection is frame differencing [31]. The technique subtracts consecutive images from a video sequence pixel-wise. The camera is assumed to be stationary, and the only differences should occur in regions of independent motion. This approach is sensitive to noise and illumination changes, and having a stationary camera is not suitable for our application.

Attempts have been made to use frame differencing for moving cameras [6, 61]. An accurate ego-motion at each time step is assumed to be known from an IMU or another means of motion estimation, but the methods do not account for uncertainty in this process. By compensating for the ego-motion a predicted image is computed and background subtraction can be performed.

In an attempt to deal with noise, it has been proposed to build a probabilistic model for the background over time [58, 64]. A popular approach is the one of kernel density estimation over the pixel intensities [76, 77]. The density for each pixel of an incoming image is computed and if it deviates from the background model according to some similarity measure, that pixel is classified as dynamic. However, these methods are again only applicable to stationary cameras and are more popular in surveillance applications.

If the majority of the environment is static, and the camera is moving, the induced image motion satisfies certain geometric constraints [26] and a basis for the trajectories of the static features can be found using RANSAC [1]. All features with trajectories that deviate from such a computed trajectory model are then classified as dynamic [12, 29]. Otherwise, if the ego-motion can be accurately estimated, the image plane can be transformed accordingly and independent motion can be detected. To date, ego-motion compensation appears to be the most common strategy for independent motion detection [16]. These techniques are limited by the assumption that the majority of the scene is static and may fail in highly dynamic environments.

1.2.1.2 Optical flow

Optical flow presents another vision-based solution to detect moving objects in an image sequence [28]. It can be seen as the apparent movement or flow of intensity values over time, and is computed for every pixel. Features with similar optical flow are then grouped together by a similarity criterion. In the past, optical flow methods have been criticized for their high computational cost, but advances in computational technology lead to a rise in popularity. It has now been used successfully for driver assistance purposes [35], indoor navigation [49] and obstacle avoidance for unmanned aerial vehicles [42].

Optical flow methods assume brightness constancy, which states that the image brightness

at a specific pixel $I(x, y, t)$ at a given frame t can be expressed as

$$I(x, y, t) = I(x - u(x, y), y - v(x, y), t + 1), \tag{1.1}$$

where $u(x, y)$ and $v(x, y)$ represent the horizontal and vertical displacements of the pixel after one frame. By expanding (1.1) into a Taylor series and assuming a small displacement, a linear system of equations can be obtained. This is usually solved by means of the Lucas-Kanade method [46], where the image gradients are computed for a window around the pixel and substituted into the system of equations. The Lucas-Kanade optical flow method, however, fails when the instantaneous velocity of a pixel is too large with respect to the window where motion is being considered [17]. Also, when a point has a strong rotational component, the assumption of similarity in displacement vectors can fail.

In order to compute the optical flow, certain constraints are required which lead to assumptions regarding the structure of the viewed scene [28]. In practice, where motion discontinuities exist, these assumptions are often violated and result in inaccurate estimates [2]. Most optical flow methods also compute the displacement vectors using pairs of images [75], which is generally not enough information for motion segmentation and can lead to inaccurate results. Other drawbacks include sensitivity to noise, the inability to deal with occlusions, and semi-static objects that remain stationary for a short period of time. Furthermore, optical flow methods were originally designed for stationary cameras and therefore require a known robot pose at each time step.

1.2.1.3 Wavelets

Wavelet transforms are commonly used in signal processing applications [47]. They express a function in terms of a set of mutually orthogonal basis functions localized in both time and frequency. This enables the wavelet transform to isolate components of a certain frequency for a specific time duration.

Wavelet motion segmentation has been a popular research topic [38] as it facilitates analysis of the frequency components of images. From these components motion parameter estimates are made and object edges are detected, allowing for clustering into individual objects. Wiskott [83] integrated Gabor- and Mallat-wavelet transforms in an attempt to overcome the aperture problem [70] as well as the correspondence problem [59]. The wavelet transform as a solution to motion segmentation is limited to simple cases, such as translational object motion, and research interest seem to have waned [86].

1.2.1.4 Statistical approaches

Contrary to the methods mentioned thus far, many approaches to motion segmentation formulate the problem probabilistically or employ traditional statistical methods.

Probabilistic graphical models (PGMs), in general, specify a factorization over the joint distribution of a set of variables in the form of a graph, where the variables are depicted as nodes and the dependencies between the variables are specified by the presence (or

absence) of edges between the nodes. This allows for the exploitation of conditional indepencencies between the variables, either for modelling purposes or for inference. We note that graphical models have been employed in the context of motion segmentation for stationary observers, and graphical models form part of our proposed solution.

Markov random fields (MRFs), which are a type of PGM, have been employed to detect motion in image sequences [58]. First, a model of the background is built and ego-motion compensation is applied. Next, regions that do not conform to the estimated model are detected by means of a statistical regularization approach and are classified as moving objects. A maximum a posteriori MRF (MAP-MRF) framework has also been employed to detect objects by using a foreground and background model [67]. These models are built through non-parametric density estimation over image intensity as well as pixel proximity, similar to the background kernel density estimation technique mentioned in Section 1.2.1.1. Both MRF methods operate on the image plane, which has the drawback that information is lost during the projection from 3D to 2D.

The conditional random field (CRF), which is a variant of the MRF, has been used to segment dynamic objects from static objects [20]. CRFs are generally employed to train discriminative classifiers and are able to represent spatial and temporal similarities between measurements. Douillard et al. [20] present a semi-supervised learning algorithm to train the classifiers but require the number of possible classes to be known. Even though the method utilizes semi-supervised learning, manual data annotation must still be performed and a representative set of training examples is required.

A classic Bayesian approach (not related to graphical models) to detect dynamic objects in range measurements has been proposed by Kaestner et al. [33]. They estimate the probability that a point is dynamic, given the range measurement at a particular time. They assume that every observation is caused by one of K objects, which can be either static or dynamic. Furthermore, they model the true range distance for each cell in the range image using a Gaussian distribution. Their goal is to estimate the posterior over all states, the parameters describing all the Gaussian distributions, as well as the number of objects K.

Some have viewed the motion segmentation as a soft clustering problem, in which a probabilistic mixture model is employed to explain change in appearance for a specific pixel [8]. These changes are assumed to be explained by object or camera motion, illumination changes, specular reflections and object-specific changes, or a combination of these factors. By following this approach, pixels that change as a result of object-specific changes can be extracted as moving obstacles.

Another popular statistical motion segmentation trend is to formulate the motion detection problem as one of determining whether a change occurred at a specific pixel after ego-motion compensation. Hypothesis testing, where the null hypothesis is formulated as “no change occurred”, can then be employed to detect moving objects [63].

1.2.1.5 Layering

The problem of motion detection in an image sequence can be seen as one of fitting a set of motion models to the data and extracting all the pixels that correspond to a specific motion model [12, 18].

The EM algorithm has been employed to build a mixture model for layered segmentation iteratively [82], although these methods are sensitive to initialization and are computationally complex [12]. A more recent approach detects features, tracks them over time, clusters them according to their motion and separates them into different objects once enough evidence is obtained [62]. Sun et al. [75] attempt to estimate the number of layers in a scene and simultaneously solve the depth ordering of the layers for occlusion handling using graph-cut optimization methods [11]. Learning layers of an image sequence for motion segmentation has also been proposed [40]. First, moving objects are identified by computing rigid transformations between two consecutive frames. Second, a clustering algorithm is employed to learn appearance models for the individual objects.

Min and Medioni [52] note that an image sequence can be seen as a 3D volume with variables (x, y, t) describing the position of a pixel in the image at time t. They convert this representation to 5D by adding a velocity at the pixel with components (v_x, v_y). By assuming spatio-temporal smoothness, which implies that neighbouring pixels belonging to one object moves similarly and also that a pixel moves smoothly over time, they can extract objects with similar velocities and thus identify individual layers in the image by employing a tensor voting framework [50]. They make another assumption that a layer or segment can only contain pixels from a single object which, if violated, leads to algorithm failure.

Layering is arguably the most natural solution to the occlusion problem [86]. However, the main drawbacks of layering approaches include high computational cost and a large number of parameters that must be tuned manually.

1.2.1.6 Learning

Some algorithms detect specific objects that are known to be dynamic. Luber et al. [45] train a classifier to detect people in RGBD data, while Wojek et al. [84] use support vector machines (SVM) to detect pedestrians, cars and trucks. Specific objects in images can be detected through use of template matching [15]. However, methods that learn foreground activity generally do not account well for measurement uncertainty [33]. These methods also typically require vast amounts of labelled representative data, which can be hard to come by and time-consuming to process.

Another approach is to learn the background on-line and then classify each pixel as static or dynamic [3]. In such methods an ensemble of weak classifiers is combined into a strong classifier using AdaBoost. However, learning the background on-line can be risky for a moving camera since the background is always changing.

1.2.1.7 Feature-based approaches

Most of the methods mentioned thus far are dense, in the sense that they consider all the pixels in an image. Such methods suffer from high dimensionality and high computational cost. An alternative is to consider only interest points or features detected in the images. These feature-based approaches can enable fast tracking and, if the image features are also to be used for SLAM or visual odometry, the identified dynamic features can be excluded from the feature set employed by those modules.

Despite their advantages, feature-based methods have attracted surprisingly little attention. Müller et al. [56] track features using a Kalman filter [34] and use edge scoring in a graph structure to segment features into different objects. A similar approach calculates scene flow for each feature and clusters the flow vectors into objects [72]. Sparse optical flow methods have the advantage that they are less sensitive to noise than their dense counterparts [74], but also require a known pose. Lenz et al. [43] combine ideas from these two into a method called triTrack, that also removes edges based on uncertainty in the measured 3D position of each feature. An additional advantage of feature-based approaches, especially for long sequences, is the possibility of compactly representing motion information as a feature interval graph [51] which is similar to the Kalman filter. A sparse motion segmentation that learns the robot's ego-motion has also been proposed and applied to humanoid robots [39].

It seems as if most existing motion detection methods rely heavily on accurate ego-motion estimation and do not accommodate for the uncertainties inherent in the robot localization process. Statistical methods in general can be well suited to handle such uncertainties. Existing statistical approaches, however, operate on the image plane and how robot pose uncertainty can be dealt with in that paradigm is unclear.

This concludes our review on segmenting measurements into static and dynamic classes. Next we consider the problem of grouping dynamic points into separate objects.

1.2.2 Clustering

Once measurements of the environment have been received and separated into static and dynamic measurements, it may be necessary to group the dynamic measurements into objects. Some of the motion segmentation algorithms mentioned in Section 1.2.1 have an inherent clustering module as part of the algorithm, but many do not. Grouping measurements or points into individual objects is important because, in the next component of a typical path planning and navigation system, these objects have to be tracked.

Clustering algorithms can generally be divided into two classes: hard and soft. A hard clustering method makes a hard decision by assigning one specific label to each data point. On the other hand, soft clustering methods provide a probability distribution over all labels for each data point, from which the most likely label can then be inferred.

One of the most popular hard clustering methods is the k-means algorithm [25]. It requires

knowledge of the number of clusters, and the algorithm starts by randomly selecting k data points, one for each cluster, that form the initial set of means. All data points are clustered by choosing the closest mean according to some metric. New means are computed for each of the clusters and the process is repeated until convergence.

Another approach, especially popular in computer vision applications, is to perform kernel density estimation over all the data points. A mean-shift algorithm [14] is employed to obtain a hard clustering and the number of clusters simultaneously. The mean-shift algorithm computes, for each data point, the direction of the largest gradient, after which each data point is translated in that direction. These steps are repeated until convergence is obtained.

In some approaches it is assumed that the underlying density representing the data is a Gaussian mixture model (GMM) and the EM algorithm is employed to estimate the parameters of this GMM. This is also an iterative procedure, which starts with an initial estimate for the parameters, performs clustering and adjusts the parameters to maximize the likelihood of the clustering until it converges [30]. The result is also a hard clustering.

In fuzzy clustering each cluster consists of a fuzzy set of all the data points [30]. The algorithm starts with an initial fuzzy clustering and a matrix containing the grade of membership of the data points to a cluster. A fuzzy criterion function of this matrix is then iteratively minimized and the matrix is recomputed until convergence. These algorithms lead to soft clustering. PGMs have also been used in the context of soft clustering image regions into objects based on colour, texture and shape [65].

Soft clustering methods are preferred, since they can provide more information to subsequent modules (such as an object tracker or a path planner). We also note that while most methods assume knowledge of the number of clusters beforehand, this information is rarely available in practice.

1.3 Problem description

From a review of the literature it is our conclusion that almost none of the currently available methods incorporate pose uncertainty into their motion segmentation approaches. Since pose uncertainty is often significant, and subsequent components in a navigation system are dependent on the accuracy of the motion segmentation, we regard the incorporation of pose uncertainty as a necessity.

Furthermore, most approaches operate on the image plane, where information is lost during the projection from 3D to 2D. Illumination change is also a problem for appearance-only motion segmentation. We instead focus on methods that can be applied to any of the sensors commonly used for environment perception by performing the motion segmentation on 3D features rather than on image level. In doing so we can also incorporate measurement uncertainty.

A common assumption made in motion segmentation algorithms is that the predominant motion in the measurement is a result of the motion of the robot. We want to avoid this assumption since situations can arise where the environment is dominated by independently moving objects.

If we can also avoid the requirement of prior information regarding the shape or appearance of the objects in the scene, our approach may become suitable for AUVs, UAVs as well as robots moving in unstructured environments. Therefore, even though we may know for example that traffic scenarios typically contain pedestrians, cyclists and vehicles, we refrain from training a representative object classifier which, in any case, requires large amounts of labelled data.

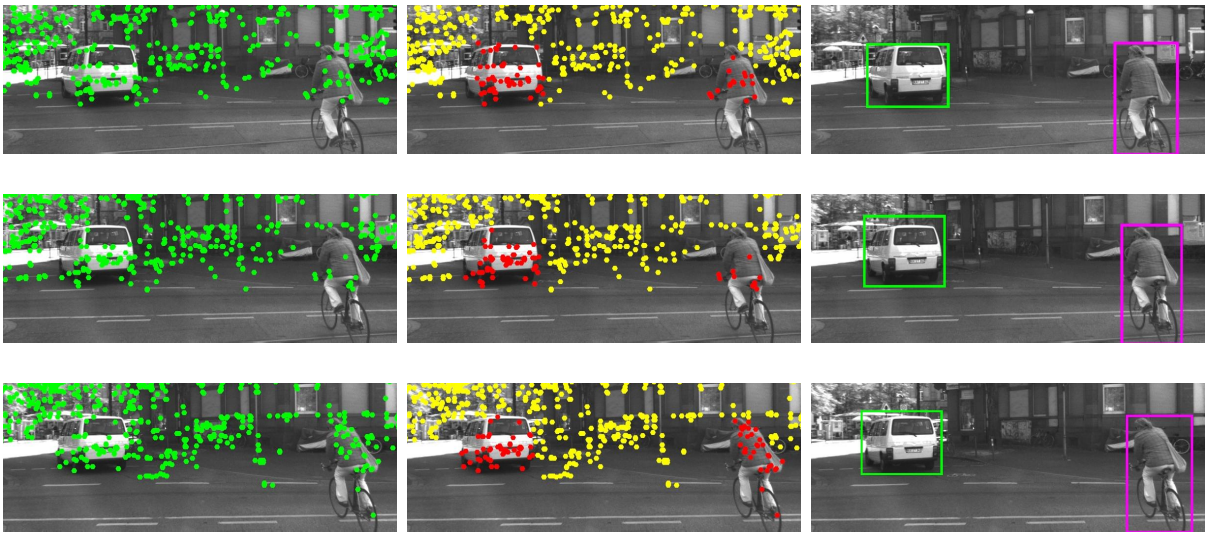
At every time step we assume access to a belief distribution over the robot's pose, obtained from some type of localization estimator. We assume that features can be identified and measured as 3D locations relative to the robot (by, for example, LIDAR or the triangulation of stereo features). We also assume a set of known feature correspondences between the current time step and the previous one, in order to compute feature velocities, as well as a measurement model over the individual features in order to characterize measurement uncertainties.

Our first goal is to classify all observed features as static or dynamic, while also making provision for features that can switch states over time (semi-static features). The second goal is to cluster dynamic features together that belong to the same object. Note that, unlike most existing methods that assume the pose of the robot to be known exactly, we want to incorporate the inherent uncertainty in the pose estimation process into our model. In order to suit a variety of robots and environments, we also want to refrain from training an environment-specific object classifier. To further the generality of the algorithm to sensor type, we choose to operate in 3D space, rather than image space, so that the system will not be restricted to vision sensors.

As a concrete example, consider Figure 1.1. The sensor is a stereo pair of cameras, and the Libviso [24] image features detected in the video feed from one of the cameras over three consecutive time steps are shown in (a). The goal of this work is firstly to label every feature as static or dynamic, as indicated in (b), and secondly to group features on the same dynamic objects together, as indicated in (c).

1.4 Our approach

Probabilistic graphical models are specifically designed for situations where we have imperfect knowledge about a situation, partial and uncertain observations, and have to draw conclusions based on such observations. PGMs are well suited for classification and clustering problems [36, 65], and should therefore be able to handle both our motion detection problem and object detection problem. Surprisingly, however, PGMs have not been a popular approach in this regard. This may be partly due to the fact that the entire problem must be carefully modelled and that inference in large networks can be compu-



(a) detected features (b) motion segmentation (c) object segmentation

Figure 1.1: An example of (a) features detected in three consecutive frames, (b) desired motion segmentation output where dynamic features are indicated in red, and (c) desired object segmentation output where dynamic objects are demarcated with bounding boxes.

tationally complex. This creates a trade-off between having a realistic representation of the problem and enabling tractable inference. In our work we address this issue, in an effort to unlock the true potential of employing PGMs for dynamic object detection.

Our approach requires feature correspondences over time from which 3D feature positions can be retrieved during triangulation [26]. For reduced complexity in feature matching, lost features reappearing in later time steps are considered to be new features. We assign a time-dependent binary state variable to every feature that indicates whether or not it is dynamic, and another indicating whether or not it is capable of movement (to accommodate semi-static features that switch states over time). Pairs of features are connected via binary variables to indicate whether or not they belong to the same object. Inference is enabled by incorporating feature velocities and relative distances between features. Modelling the problem in this way results in a hybrid PGM that consists of both continuous and discrete variables, for which tractable inference is challenging. However, a few simplifying assumptions allow closed-form expressions for messages sent from the continuous variables to the discrete ones, and inference can be performed on a set of binary discrete variables.

1.5 Original contributions

We present a novel application of probabilistic graphical models especially designed to solve the dynamic object detection problem for mobile robots. Our PGM can be divided into two interacting components. The first is the motion detection component, which is developed from first principles and offers a new way of describing the problem. The second component builds on an idea from Shental et al. [68], where pairs of features are

connected via a binary random variable that indicates whether or not they belong to the same object. The idea is adapted to interact with the first component as well as to fit into the PGM framework. We further adapt the model to accommodate semi-static objects. We also propose an incremental inference scheme that avoids increased complexity over time and facilitates tractable inference.

Existing approaches generally assume that the exact pose of the robot is known at every time step or, in order to estimate the robot pose, they assume that the scene is predominantly stationary. In this work we avoid both assumptions. As a consequence, our proposed method enables a robot to detect dynamic objects in highly dynamic scenes. Also, having an expensive localization sensor is no longer a requirement since we accommodate for uncertainty in the robot pose estimates.

The rest of this dissertation is organized as follows. We proceed with some background on probabilistic graphical models in Chapter 2. In Chapter 3 we present the design of our PGM, and address the hybrid PGM inference issue in Chapter 4. Our model requires a set of measurements that may need pre-processing, and this is discussed in Chapter 5. Results are presented and discussed in Chapter 6, followed by conclusions and future work in Chapter 7.

Chapter 2

Probabilistic graphical models

Probabilistic graphical models (PGMs) model the relationship between what we want to know (in our case whether objects in the environment are static or dynamic), and the observations (robot pose and feature measurements) which have their own uncertainties. Given observations, the PGM allows us to infer what the most likely situation is. Put differently, PGMs provide a platform to model what we do know about a certain situation in the form of interactions between components (such as the observations and what we want to know) and, given uncertainties, draw conclusions about the true state of the situation.

If a mobile robot is equipped with stereo cameras, for example, comparing images from consecutive time steps can give the impression that all observed features are moving. However, some of the observed motion is explained by the robot's ego-motion and some by independently moving objects, and we do not know which is which. We do know that stationary features will have a velocity of zero in the world coordinate system, and dynamic features will not. However, we cannot observe these velocities directly and the feature velocities we do observe, carry uncertainty. In a dynamic object detection system we have imperfect knowledge of the environment and uncertain observations, and we want to infer which objects are dynamic and which are static. PGMs therefore seem to provide a natural framework in which to solve the independent motion segmentation problem.

In this chapter we highlight certain aspects of PGMs required for our proposed solution and implementation. The reader is referred to the books by Koller and Friedman [36] and Barber [4] for further reading.

2.1 Modelling

When a probabilistic model is defined to describe or model a problem, it has to specify the joint probability distribution over all the random variables involved. A particular model can specify a factorization of the joint distribution by making some independence assumptions. PGMs encode the model-specific factorization as well as the independence assumptions. After observations are made, inference can be performed by exploiting the graph-structure of the PGM in order to draw conclusions about the model. We first

discuss how a PGM is constructed.

The first step is to decide on which random variables to include in the model and exactly what they represent. There can be latent random variables, which are variables of interest that cannot be observed, as well as related random variables that can be observed. We are usually interested in the latent variables, and want to infer distributions over the values they can assume from what can be observed.

As a simple example, suppose a doctor wants to infer the disease a patient has from a number of observed symptoms. We can define a set of binary random variables $D = \{d_1, \ldots, d_n\}$, such that a value of $d_i = 1$ indicates that the patient has disease i. These are latent random variables since they are not observed directly. We can also include in our PGM a set of binary random variables $s = \{s_1, \ldots, s_m\}$, where $s_j = 1$ indicates that the patient exhibits symptom s_j. This set denotes the observed random variables.

The next step is to decide how the variables are related to one another. It is of course important that some dependencies exist between observations and latent variables, for knowledge to be transferred from the observations to the latent variables.

2.1.1 Bayesian networks

Relationships between diseases and symptoms are causal (diseases cause symptoms). Bayesian networks are a specific type of graphical model well suited to model such relationships. An example of a Bayesian network for the symptom-disease scenario is shown in Figure 2.1(a). The joint distribution over all the variables present in Figure 2.1(a) can be factorized (in general) as

$$p\left(d_1, d_2, s_1, s_2, s_3\right) = p\left(d_1\right) p\left(s_1 \middle| d_1\right) p\left(d_2 \middle| s_1, d_1\right) p\left(s_2 \middle| s_1, d_1, d_2\right) p\left(s_3 \middle| s_1, s_2, d_1, d_2\right). \tag{2.1}$$

However, for simplicity in our model we make the following conditional independence assumptions:

$$d_2 \perp s_1, d_1, \tag{2.2}$$
$$s_2 \perp s_1 \mid d_1, \tag{2.3}$$
$$s_3 \perp s_1, s_2, d_1 \mid d_2, \tag{2.4}$$

(a) (b)

Figure 2.1: (a) A simple example of a directed PGM or Bayesian network. (b) An example of a Markov network representation of (a).

where $\perp$ indicates statistical independence. The joint distribution then factorizes as

$$p(d_1, d_2, s_1, s_2, s_3) = p(d_1)\, p(s_1 | d_1)\, p(d_2)\, p(s_2 | d_1, d_2)\, p(s_3 | d_2)\,. \tag{2.5}$$

Bayesian networks are defined in such a way that independencies can be deduced from the graph structure. If, for each variable, a conditional distribution is defined over that variable given its parents in the graph, the product of all such distributions will form the model-specific factorization of the joint probability distribution.

In order to put this more formally, let us introduce the concept of d-separation. In general, sets of variables X and Y are d-separated given a set of observations Z if one of the following criteria is met:

- for any chain $X \longleftarrow M \longleftarrow Y$, at least one of the variables in the set M is in Z;
- there exists a fork $X \longleftarrow M \longrightarrow Y$ and at least one of the variables in the set M is in Z;
- there exists a v-structure $X \longrightarrow M \longleftarrow Y$ and none of the variables in M nor any of their descendants are in Z.

It can then be said that X and Y are independent given an observation set Z if X and Y are d-separated by all paths between them.

Once the conditional independence assumptions for the model are specified, all factors in the resulting model-specific factorization must be known and supplied to the inference algorithm. For our symptom-disease example, we would have to specify the probability distributions that various symptoms are observed given the disease ($p(s_1 | d_1)$, $p(s_2 | d_1, d_2)$ and $p(s_3 | d_2)$) as well as prior probability distributions that diseases occur ($p(d_1)$ and $p(d_2)$). Note that there may exist more than one graph representation for the same factorization.

In some cases the required distributions can be obtained through logical reasoning or expert knowledge. In other cases it may be necessary to learn such distributions from data samples. Kernel density estimation [71] is an option if no restriction is placed on the class of target distribution, while maximum likelihood estimation [57] or expectation maximization [54] can be employed for distributions assumed to be of a particular class.

2.1.2 Markov random fields

Another type of PGM is a Markov random field (MRF) in which the connections are not directed. A Bayesian network can be converted to an MRF by a process called moralization, where parent nodes with a common child are connected, in addition to already connected nodes. An MRF version of the Bayesian network in Figure 2.1(a) is shown in Figure 2.1(b). Markov random fields are used when no meaningful causal relationship

exists between the variables, i.e. in situations where it is more natural to specify a joint probability over a subset of the random variables than a conditional probability.

Similar to Bayesian networks, the independence assumptions can be deduced from the graph. For Bayesian networks we introduced the notion of d-separation. For MRFs, a set X is independent of another set of variables Y given an observation set Z if all paths (in any direction) between X and Y pass through Z. From Figure 2.1(b) we see that all paths between s_1 and s_2 must pass through d_1, and all paths between s_3 and the set $\{s_1, s_2, d_1\}$ must pass through d_2, so that (2.3) and (2.4) follow. The assumption (2.2) does not follow from the graph and is an example of some of the expressiveness lost by converting a Bayesian network to a Markov network.

The power of PGMs emerges in networks where independence assumptions are made, since such assumptions lead to a more sparsely connected PGM and simplified inference. Specifying the independence assumptions therefore introduces a possible trade-off between computational complexity and accurate modelling of the problem. Also, since specifying the factors and deciding on the independence assumptions go hand-in-hand, it may require a number of design iterations to reach an appropriate model. In some cases relationships between random variables can also be learned [36].

2.1.3 Factor graphs

A further representation of graphical models exists in the form of factor graphs, which are also undirected. Both Bayesian and Markov networks can easily be converted into factor graphs. When drawing factor graphs, random variables are represented by circles, while every factor in the joint factorization is represented by a rectangle. A variable can only be connected to a factor, and is connected only when it occurs in that factor. For inference on PGMs these factors do not have to be normalized, and are therefore not necessarily probability distributions. For this reason the notation $\phi(\cdot)$ is introduced to represent both probability distributions and unnormalized distributions or potentials.

An example of a factor graph for the symptom-disease model used previously is shown in Figure 2.2, where

$$\begin{aligned}
\phi(d_1) &= p(d_1), && (2.6)\\
\phi(d_2) &= p(d_2), && (2.7)\\
\phi(d_1, d_2, s_2) &= p(s_2 | d_1, d_2), && (2.8)\\
\phi(d_1, s_1) &= p(s_1 | d_1), && (2.9)\\
\phi(d_2, s_3) &= p(s_3 | d_2). && (2.10)
\end{aligned}$$

2.1.4 Cluster graphs

In a factor graph each random variable in the PGM occurs exactly once, and two variables can be connected via a factor. Another representation exists where variables and factors are grouped together. Such graphs are called cluster graphs. A specific grouping of factors

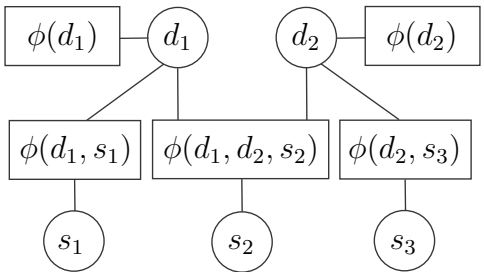


Figure 2.2: An example of a factor graph for the PGM in Figure 2.1.

is referred to as a cluster, while a grouping of random variables is called a sepset. Similar to the factor graph, sepsets are connected via clusters, under the restriction that the variables in a sepset between two clusters must be contained in the scope of the cluster intersection (all variables that occur in both clusters).

Since there are many ways in which to group factors into clusters and variables into sepsets, a cluster graph representation of a factor graph is not unique. In a cluster graph the same variable may occur in multiple sepsets, but those sepsets may be separated by many clusters. For inference, cluster graphs are required to possess the running intersection property, which is defined as follows. If a variable x occurs in cluster C_i and C_j, then there must be a single path between C_i and C_j and x must be contained in all edges of that path.

In Figure 2.3(a) we show an example of a Markov network. This graph is converted to a cluster graph in (b) which does not satisfy the running intersection property. In (c) a cluster graph is shown which does satisfy the running intersection property.

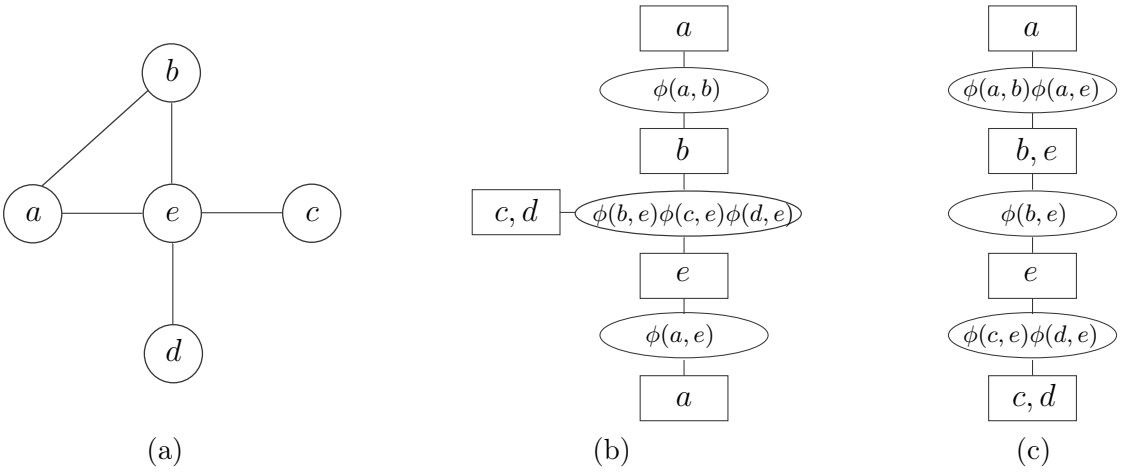


Figure 2.3: (a) A Markov network. (b) An example of a cluster graph that does not satisfy the running intersection property. (c) An example of a cluster graph that satisfies the running intersection property. Since the graph in (c) is a tree, it is also an example of a clique-tree.

2.1.5 Clique trees

A special case of the cluster graph is called a clique tree or junction tree. As the name suggests, these graphs are trees, i.e. contain no loops. Furthermore, all the sepsets are cliques (fully connected) and must satisfy a more specific instance of the running intersection property. This property states that, if a variable x occurs in clique C_i and also in clique C_j, then x must also occur in every clique along the unique path between clique C_i and C_j. The definition for clique trees implies that the sepsets must be equal to all variables in the intersection of C_i and C_j, which is not a requirement for the first definition.

An undirected graph (with or without loops) can be converted to a clique tree if and only if every loop of length 4 or more has a chord (an edge that connects two non-adjacent nodes in a loop). In such cases we say that the graph is triangulated. If the undirected graph is not triangulated, extra links may be added to the chord-less cycles. The Markov network in Figure 2.3(a) has a loop, but the cluster graph in Figure 2.3(c) has a tree structure. Since it satisfies the running intersection property for clique trees, it is also a clique tree.

2.2 Inference

Once a problem is modelled graphically and all the required distributions are known, inference is performed so that meaningful conclusions can be drawn. The type of inference that is performed depends on the type of conclusions that must be drawn from the PGM. One query of interest is the probability distribution over all the random variables in the PGM, given all the evidence. This is referred to as computing the conditional probability query, and is formulated as

$$p\left(X \,|Y = \hat{y}\right), \tag{2.11}$$

where X contains all the random variables in the PGM except the set of observations Y, and $Y = \hat{y}$ is observed. Depending on the application, one may also be interested in a marginal distribution over all or some of the latent variables, or perhaps the normalizing constant [80]. In this work we are interested in conditional probability queries, since we want to draw conclusions about the values of the random variables given observations.

Depending on the problem, one may be interested in the values of the random variables that maximizes the posterior distribution, i.e. finding the set of values x such that

$$\arg\max_{x} \; p\left(X = x \,|Y = \hat{y}\right). \tag{2.12}$$

This problem is known as the maximum a posteriori (MAP) inference problem.

Both queries can only be made once the PGM is calibrated. This occurs when all pairs of adjacent nodes are calibrated, which in turn occurs when the marginal distributions over their sepsets agree. Performing both types of queries can be computationally intractable, since the problem in general is NP-hard [36]. However, inference on many specific PGMs can be performed efficiently using either exact or approximate inference algorithms.

2.2.1 Exact inference

In exact inference a specific ordering exists in which the marginal distributions are computed and the sums, or integrals in the case of continuous variables, can be computed exactly. One example of an exact inference method is variable elimination, where a specific ordering of summations and products over factors are computed depending on the query variables [36]. Consequently, inference has to be re-run for a different set of query variables [80].

The sum product algorithm is a dynamic programming algorithm that shares intermediate computational terms by propagating messages between nodes. In this algorithm, each message is sent exactly once and two adjacent nodes are calibrated after messages in both directions have been sent. This method of inference, which is also called belief propagation, is widely used for PGMs containing discrete and/or Gaussian variables [80]. Inference using belief propagation is, however, exact only when the graph has a tree structure (acyclic). This also applies to clique-trees that are derived from graphs with loops.

2.2.2 Approximate inference

In cases where the PGM is not sufficiently sparse, exact inference becomes intractable or even impossible. Fortunately algorithms exist that enable approximate inference. For cluster graphs with loops, belief propagation can still be performed and is computationally significantly cheaper than exact inference [36]. However, it must be said that the larger the cluster the higher the computational cost and, since different cluster graphs can lead to significantly different answers, a low cost approximation may sometimes be poor.

All messages are initialized to unity, and messages are passed between nodes. After all messages have been propagated once, i.e. messages in both directions have been passed between every pair of adjacent nodes, we are not guaranteed that these nodes will be calibrated. Another round of message passing can be required, and the process may be repeated until convergence. Even though there is no guarantee that convergence will occur, loopy belief propagation (as it is called) remains a popular and useful algorithm for inference [80].

Another approximate inference algorithm preserves the cluster structure, but gains efficiency by approximating the messages themselves. This is achieved by representing complex joint distributions over many variables as a product of small factors, and is called expectation propagation [36]. While this approach may provide a good approximation to the posterior belief [53], it grows in complexity with the size of the graph – similar to belief propagation.

If the factors in the joint distribution specified by the PGM are stationary (i.e. do not depend on the time parameter t), a random walk or traversal of states on the graph can be generated, which forms a Markov chain since the next state in the walk depends only on the current state. A Markov chain can be constructed in such a way that the desired

posterior distribution is the equilibrium distribution, using for example Gibbs sampling. Approximating the posterior distribution in this way is a so-called Markov chain Monte Carlo (MCMC) method [7]. An alternative to MCMC is sequential Monte Carlo (or particle filtering) methods. Particle filters approximate a sequence of target distributions with weighted samples (particles) which represent random hypotheses about the state of the system. MCMC methods have the drawback that it can be difficult to determine whether or not equilibrium has been reached, while sequential Monte Carlo methods suffer from the weight degeneracy issue where a small set of particles can dominate as time progresses [85]. Particle-based methods in general are guaranteed to achieve accurate results at the large sample limit, but the performance with practically reasonable sample sizes is difficult to predict [36].

It turns out that the PGM we design for dynamic object detection consists of discrete and Gaussian variables and, while it should be noted that any of the approximate inference methods mentioned here may be suitable for our PGM, we focus on the sum product algorithm described in the next section.

2.3 Sum product algorithm

In this section we provide details of sum product message passing on factor graphs without loops, which results in exact inference. The procedure for applying the algorithm to a clique tree is fundamentally no different, except that messages are propagated between sepsets and cliques or clusters rather than variable and factor nodes.

In sum product message passing, messages can be propagated asynchronously, since each node can send a message as soon as it is ready. A node is ready to propagate a message to a neighbouring node C_j once it has received messages from all of its neighbours except from C_j. Once a message is sent to a particular node, the belief over the variables at that node is updated. Also, messages are scheduled so that every message is computed and propagated exactly once. Once messages in both directions over an edge are sent, those two adjacent nodes are calibrated. Formally, two adjacent nodes are calibrated if the marginal distributions over the sepset of their updated beliefs agree. It can be shown that the joint distribution over all variables in the graph factorizes as the normalized product of the updated beliefs after calibration [36].

Generally, there are two types of messages: from a variable node to a factor node and from a factor node to a variable node. Examples are shown in Figure 2.4(a) and (b) respectively.

Consider the factor graph shown in Figure 2.4(c). Since variable node y is connected to a single factor node $\phi(x, y)$, it receives no other incoming messages and can immediately propagate a message to $\phi(x, y)$. The same applies to factor nodes $\phi(s, x)$ and $\phi(m, x)$. Such nodes are referred to as leaf nodes. Say we decide to start message passing at leaf node $\phi(s, x)$ by passing a message to x. Since variable node x is also connected to factor nodes $\phi(s, x)$ and $\phi(m, x)$, it cannot propagate any messages until it receives

either the message from $\phi(s,x)$ or $\phi(m,x)$. Since $\phi(m,x)$ is also a leaf-node, it can propagate a message to x. Once this message is received it can propagate a message to $\phi(x,y)$ which, in turn, propagates to y. Since y cannot propagate its message any further, inference in this direction is complete. At this point, however, $\phi(s,x)$ and $\phi(m,x)$ have not yet received any messages, while y has not propagated any messages. Therefore y propagates a message to $\phi(x,y)$ (which is not dependent on the message from $\phi(x,y)$), $\phi(x,y)$ propagates to x and x propagates to both $\phi(s,x)$ and $\phi(m,x)$. This is an example of one possible message passing scheme. We now proceed to describe these messages in more detail.

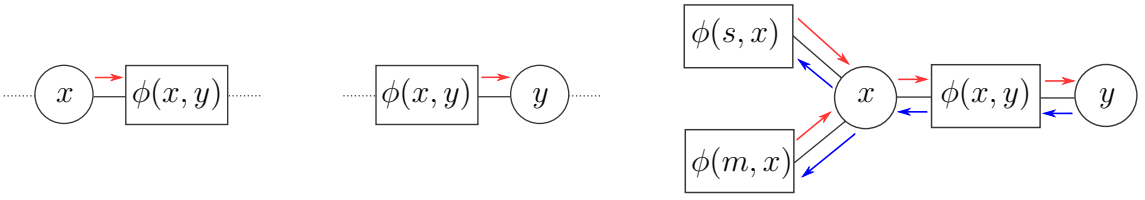
2.3.1 Message passing from a variable node to a factor node

We first discuss message passing from a variable node to a factor node, or sepset to clique in the case of clique trees. Note that, in a factor graph, a specific variable node is only connected to factor nodes and vice versa. Therefore a variable node can only receive messages from factor nodes, and only propagate messages to factor nodes.

In the sum product algorithm all factor leaf nodes have their outgoing messages set to their own factors, while all variable leaf nodes have their outgoing messages initialized to unity. If a variable node receives a message from a single factor node, the outgoing message is equal to the message received from that factor node. If, however, a variable node receives messages from more than one factor node, its outgoing message to another factor node is defined as the product of all incoming messages. In order to formulate this mathematically we require the notation $\mu_{a\to b}(\cdot)$, which indicates that a message is sent from node a to b. Also, we denote factors by $\phi(A)$, where A is the set of all variables that occur in the factor. The message that variable node x sends to factor node $\phi(A)$ is defined as

$$\mu_{x\to\phi(A)}(x) = \prod_{\phi(B)\in\{ne(x)\setminus\phi(A)\}} \mu_{\phi(B)\to x}(x), \tag{2.13}$$

where $x \in A$, $x \in B$ and $ne(x)$ refers to all neighbouring factor nodes directly connected to node x. The \-symbol is the set difference operator. Note that the product is taken over factors (or messages) that are only functions of x.



(a) (b) (c)

Figure 2.4: (a) Isolated message from a variable to a factor. (b) Isolated message from a factor to a variable. (c) An example of a message passing scheme for a factor graph. We start with messages passed in the direction of the red arrows, and follow with messages passed in the direction of the blue arrows.

For the example factor graph in Figure 2.4(c), we can now calculate the message from x to the factor node $\phi(x, y)$ as shown in Figure 2.4(a), as

$$\mu_{x\to\phi(x,y)}(x) = \mu_{\phi(s,x)\to x}(x) \cdot \mu_{\phi(m,x)\to x}(x) = \left(\sum_{s} \phi(s,x)\right)\left(\sum_{m} \phi(m,x)\right). \tag{2.14}$$

The formulation presented in this section holds for exact inference algorithms. In the case where loopy belief propagation is applied, the same definition (2.13) applies, but this message may be propagated many times.

2.3.2 Message passing from a factor node to a variable node

Since factor nodes are only connected to variable nodes in a factor graph (clique nodes to sepset in the case of clique trees), factor nodes can only receive and send messages to variable nodes. Since a factor node has received messages from (possibly) multiple variable nodes, the message received by the factor node is calculated as in (2.13). Next, this message is multiplied by the factor node's own factor, and summed over all variables except the one receiving the message from the factor. Mathematically this is stated as

$$\mu_{\phi(A)\to x}(x) = \sum_{A\setminus x} \phi(A) \prod_{y\in\{ne(\phi(A))\setminus x\}} \mu_{y\to\phi(A)}(y). \tag{2.15}$$

This formulation holds only if the variables are discrete. For continuous variables the summation is replaced by the appropriate integral.

Using the formulation in (2.15) the message from factor node $\phi(x, y)$ to y in Figure 2.4(b) and (c) is given by

$$\mu_{\phi(x,y)\to y}(y) = \sum_{x} \phi(x,y) \cdot \mu_{x\to\phi(x,y)}(x) = \sum_{x} \phi(x,y) \sum_{s} \phi(s,x) \sum_{m} \phi(m,x). \tag{2.16}$$

If the ultimate goal is to perform MAP inference, i.e. to find

$$\arg \max_{x_1,x_2,\ldots,x_n} p(x_1, x_2, \ldots, x_n), \tag{2.17}$$

the max product algorithm may be used instead of the sum product algorithm. The formulation of messages is similar to that of the sum product algorithm, except that the summation in (2.15) is replaced by the maximum operator.

2.3.3 Message passing to and from observed variables

Up to this point in the message passing process no variables were observed, since we propagated probability distributions or factors between all the nodes. If no observations are made, little can be inferred about the values of the latent variables, since the PGM only propagates what we already know about the problem in the form of distributions

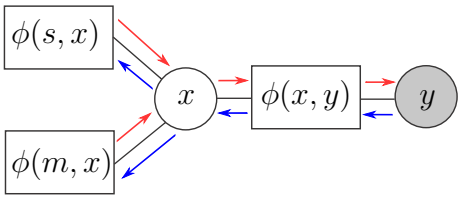


Figure 2.5: The factor graph in Figure 2.4(c), where y is observed.

we specify. Consequently, propagating messages from observations are essential for useful inference.

Observing a variable fixes its value wherever it occurs in the PGM. To illustrate this, consider the same example as in Figure 2.4(c) where a value of $\hat{y}$ is observed for the variable node y. To indicate that a variable is observed, its node is usually shaded as in Figure 2.5. In this example the message from factor node $\phi(x, y)$ to variable node x becomes

$$\mu_{\phi(x,y=\hat{y})\to x}(x) = \phi(x, y = \hat{y}). \tag{2.18}$$

2.3.4 Belief over a variable given all observations

The conditional probability that a variable takes on a particular value given all observations, also known as the belief, can be calculated once the PGM is calibrated. This belief is formulated as $p(x\,|Z)$, the probability distribution of the random variable x given the set of all observations Z. This belief is given simply by the product of all incoming messages after proper normalization.

In Figure 2.5 for example, if $y = \hat{y}$ is observed, the belief over x is

$$p(x\,|y = \hat{y}) \propto \sum_{s,m} \mu_{\phi(s,x)\to x}(x)\ \mu_{\phi(m,x)\to x}(x)\ \mu_{\phi(x,y=\hat{y})\to x}(x) \tag{2.19}$$

$$\propto \sum_{s,m} \phi(s,x)\phi(m,x)\phi(x, y = \hat{y}). \tag{2.20}$$

The expression on the right hand side can be normalized (divided by the sum over all possible x) to arrive at $p(x\,|y = \hat{y})$.

For dynamic object detection we can define a number of latent random variables, such as a binary variable to indicate whether or not a feature is dynamic, or a variable that indicates to which object the feature belongs. From the robot's environment sensors we can also retrieve observations related to these variables. Therefore, we can design a PGM that models this problem. In this case we are interested in the posterior marginal distribution that a feature is dynamic, and to which object it belongs.

Chapter 3

Probabilistic graphical model design

In the previous chapter we provided some background on probabilistic graphical models. Since PGMs are particularly well suited to model latent variables and infer distributions over them from observations, we can use this representation to reach our two main goals: classifying observed features as static or dynamic at every time step (motion segmentation), and clustering dynamic features together that belong to the same object (object segmentation).

This chapter focuses on the modelling choices behind the PGM we propose in order to reach the two goals. Details on how inference is performed over this PGM are given in Chapter 4. We divide the design of our PGM into two interacting components: one related to motion segmentation and one relation to object segmentation. We first describe the components for a single time step and then expand them to incorporate additional time steps consecutively.

3.1 The component related to motion segmentation

The core idea behind the PGM component related to motion segmentation is to link velocity measurements to static or dynamic features. In order to do so, we introduce a binary random variable $x_{i,t}$ to indicate whether or not feature i is moving at time t, where $x_{i,t} = 1$ is the event that the feature is moving (dynamic) and $x_{i,t} = 0$ that it is static. By including this binary random variable we can eventually infer the probability distribution over $x_{i,t}$ from the PGM. Since $x_{i,t}$ has only two possible values, we can apply a threshold to decide if feature i is in fact moving at time t. The same can be applied to all features, for a classification of all features at time t as either static or dynamic. Since $x_{i,t}$ is a latent variable, a distribution over its values has to be inferred from other variables that can be observed.

Note that the discussion here pertains to a single time step, and should be viewed as a first step in the development of our full PGM formulation in which variables are also linked over consecutive time steps. In fact, the distributions given in this section will be used only for the first time step that a particular feature is observed, where information from previous time steps is not available.

3.1.1 Incorporating velocity measurements

Suppose that feature i is moving at time t with velocity $\mathbf{v}_{i,t}$, which can be zero if the feature is static. The variable $\mathbf{v}_{i,t}$ is a continuous random variable in $\mathbb{R}^3$. Physical constraints may limit its range, depending on the types of dynamic objects in a particular environment.

The true value of $\mathbf{v}_{i,t}$ is hidden, since we can only measure the velocity with some error. We denote the measured velocity by $\hat{\mathbf{v}}_{i,t}$, which is also a continuous random variable. This measurement can be obtained if we can measure the feature positions at time t and time $t-1$. Note that in the absence of any other information, the measured value $\hat{\mathbf{v}}_{i,t}$ will also be our best estimate of $\mathbf{v}_{i,t}$. However, as we shall see, if additional indirect information is available, a more accurate estimate of $\mathbf{v}_{i,t}$ is attainable.

One way to specify a graphical model over the variables mentioned thus far, for a particular time index t, is shown in Figure 3.1. Note that whether or not a feature i is currently dynamic, $x_{i,t}$, influences the current true velocity $\mathbf{v}_{i,t}$, which in turn influences the current velocity measurement $\hat{\mathbf{v}}_{i,t}$. The model shown is a Bayesian network, since there are directed links between the variables. For this type of graphical model the distributions $p\left(\hat{\mathbf{v}}_{i,t}\,|\mathbf{v}_{i,t}\right)$ and $p\left(\mathbf{v}_{i,t}\,|x_{i,t}\right)$ in particular are required for inference ($p\left(x_{i,t}\right)$ is also required, but we discuss it in Section 3.1.2).

In order to choose an appropriate distribution for $p\left(\hat{\mathbf{v}}_{i,t}\,|\mathbf{v}_{i,t}\right)$, which is also called a measurement model, let us first consider how the measurement is obtained. There are many sensors that may be used to obtain velocity measurements, but nearly all will require some pre-processing. If the 3D coordinates of a feature can be measured at every time step, e.g. through the triangulation of stereo features and proper incorporation of the robot's pose, then the difference between consecutive coordinates can act as a measurement of the feature's velocity. It is common to assume that measured 3D coordinates are characterized by a Gaussian distribution [48] and therefore that the velocity measurement is also normally distributed. Details of these calculations, and the transformation from robot pose and measured locations to velocity measurements, follow in Chapter 5.

Thus we assume that

$$p\left(\hat{\mathbf{v}}_t\,|\mathbf{v}_t\right) = \mathcal{N}\left(\hat{\mathbf{v}}_t\,|\mathbf{v}_t, V_{i,t}\right), \tag{3.1}$$

where $\mathcal{N}\left(\cdot\,|\boldsymbol{\mu}, \Sigma\right)$ is the multivariate Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance Σ. The matrix $V_{i,t}$ is the measurement covariance of the velocity of feature i at time t (we discuss its calculation in Section 5.5).

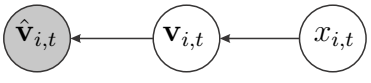


Figure 3.1: The PGM that shows the dependence between the velocity measurement, the true velocity and the current motion state of feature i at time t.

Next we consider the distribution over the true velocity $\mathbf{v}_{i,t}$ given the current motion state $x_{i,t}$. A static feature has zero velocity, so we may assume

$$p\left(\mathbf{v}_{i,t} \left| x_{i,t} = 0 \right.\right) = \delta\left(\mathbf{v}_{i,t}\right), \tag{3.2}$$

with $\delta\left(\cdot\right)$ the dirac-delta function. Without any additional information about the true velocity of dynamic features, a uniform distribution over the possible expected velocities can be assumed. Such expected velocities depend on the robot's environment and the dynamic objects that may be encountered. However, for convenience and tractable inference in our final PGM we choose a Gaussian distribution centred around zero:

$$p\left(\mathbf{v}_{i,t} \left| x_{i,t} = 1 \right.\right) = \mathcal{N}\left(\mathbf{v}_{i,t} \left| \mathbf{0}, C_v \right.\right), \tag{3.3}$$

where C_v is chosen to encompass all realistic velocities of objects in the environment. It may seem unusual to set the mean velocity to zero for dynamic objects, but we do so to account for velocities in any direction. At this point we may be tempted to combine the distributions over $\hat{\mathbf{v}}_{i,t}$ and $\mathbf{v}_{i,t}$ by obtaining the marginal distribution over $x_{i,t}$ but, as we see in Section 3.4.1, all three variables are required when the PGM is expanded to include multiple time steps.

3.1.2 Accommodating semi-static features

Dynamic objects are not necessarily always moving and can be temporarily stationary. To model this, we introduce an additional binary variable $m_{i,t}$ to indicate whether or not a feature i is capable of movement. By $m_{i,t} = 1$ we mean that feature i is capable of movement (and can currently be either moving or stationary), while $m_{i,t} = 0$ means that the feature is always stationary. Since a feature's capability of movement should not change over time, $m_{i,t}$ also cannot change over time. We include the time subscript simply for convenience. The motivation behind including both x- and m- variables, rather than only one of the two, becomes clear in Section 3.2.1.

Since knowing whether or not a feature is capable of movement tells us something about the likelihood of it currently moving, there exists a dependency between the x- and m-variables. We model this dependency as shown in the partial PGM in Figure 3.2, where the dotted line indicates previously defined connections. From this figure it follows that the distributions $p\left(x_{i,t} \left| m_{i,t} \right.\right)$ and $p\left(m_{i,t}\right)$ are required. We note that $x_{i,t} = 1$ is impossible when $m_{i,t} = 0$, and set the other entries of the conditional probability distribution table for the discrete distribution $p\left(x_{i,t} \left| m_{i,t} \right.\right)$ as in Table 3.1. Here we include a probability $a_{x|m}$ that the feature is currently dynamic if it is capable of movement. One can argue

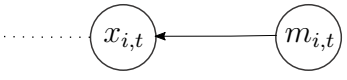


Figure 3.2: The partial PGM that shows the relationship between $x_{i,t}$ and $m_{i,t}$. The dotted line indicates previously defined connections.

$x_{i,t}$	$m_{i,t}$	$p\left(x_{i,t}\,\middle\vert\, m_{i,t}\right)$
0	0	1
1	0	0
0	1	$\left(1 - a_{x\vert m}\right)$
1	1	$a_{x\vert m}$

Table 3.1: Conditional distribution table for $p\left(x_{i,t}\,\middle\vert\, m_{i,t}\right)$. Here $a_{x|m}$ represents the probability that a feature capable of movement is currently in motion.

that if a feature can be dynamic (without knowing the type of object on which it is located), it will be more likely that it is currently dynamic, i.e. $0.5 < a_{x|m} \leq 1$. Without prior information about the environment we may assume $p\left(m_{i,t} = 1\right) = 0.5$.

The partial PGM that contains only the variables $\hat{\mathbf{v}}_{i,t}$, $\mathbf{v}_{i,t}$, $x_{i,t}$ and $m_{i,t}$ forms the component related to motion segmentation. We now proceed to the partial PGM related to object segmentation.

3.2 The component related to object segmentation

Grouping observed features into different objects is essentially a clustering problem, and has been thoroughly researched. Blatt et al. [9] propose the notion of computing the typical cut, which is a variation of graph cuts, where edges between neighbouring nodes are weighed according to their distance. Shental et al. [68] show that the typical cut framework is equivalent to performing inference on an undirected graphical model, where a binary random variable between a pair of neighbouring nodes indicates whether or not they belong to the same cluster.

In this work we use the formulation of Shental et al. [68] as the starting point for the object segmentation part of our PGM. Specifically, the goal is to cluster all static features together while also clustering dynamic features that belong to the same objects. The core idea behind the object segmentation component is to introduce a binary variable that connects pairs of features, while using a measurement of the relative distance between two features as an indication of whether or not they belong to the same object. We also discuss a method to enforce a consistent segmentation boundary between objects.

3.2.1 Modelling the relationship between features

Consider two features i and j where i is currently static while j is currently dynamic. These features cannot be part of the same rigid object, and the same holds if one is capable of movement ($m_{i,t} = 1$) while the other is not ($m_{j,t} = 0$). To model such relationships between features we introduce a latent binary variable $s_{ij,t}$ to indicate whether or not features i and j belong to the same object ($s_{ij,t} = 1$) or not ($s_{ij,t} = 0$). Figure 3.3 graphically depicts the relationship between $x_{i,t}$, $x_{j,t}$, $m_{i,t}$, $m_{j,t}$ and $s_{ij,t}$. An exception occurs if one of the features is located at the instantaneous centre of rotation of a particular object, where it will be (momentarily) static. It is unlikely, however, that the instantaneous centre of

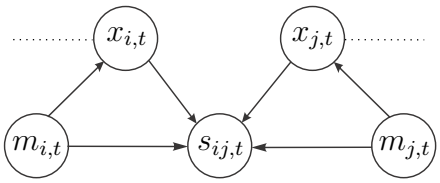


Figure 3.3: The partial PGM that shows the relationship between features i and j. The dotted line represents previously defined relationships.

rotation will be on the surface of an object where observable features are located. One could also argue that the velocity variables $\mathbf{v}_{i,t}$ and $\mathbf{v}_{j,t}$ should be directly connected to $s_{ij,t}$. However, since they are indirectly connected, information can still flow from $\mathbf{v}_{i,t}$ and $\mathbf{v}_{j,t}$ to $s_{ij,t}$, while lowering the complexity of message passing (which turns out to be important in Chapter 4).

The graphical model in Figure 3.3 can be expanded to accommodate N features by repeating it for every possible pair of features. This results in

$$2N + \binom{N}{2} = \frac{N(N+3)}{2} \tag{3.4}$$

random variables, which can lead to intractable inference for large N. In order to avoid this, only features that are spatially close are connected. One way to connect features is to perform a Delaunay triangulation [19] on the 3D feature locations, to obtain a set of tetrahedra connecting features that are spatially close. Section 3.2.3 describes how such a triangular structure can also be exploited to enforce continuous boundaries.

By including a random variable that indicates whether two features belong to the same object or not, we can arrive at a segmentation without specifying the number of distinct objects beforehand. Also, it is not required that all points on a specific object be connected via tetrahedra. Theoretically, the PGM can express the fact that two points form part of the same object if they are connected with a chain of edges [68]. This means that by connecting only spatially close features little, if any, expressiveness is lost.

Since the purpose of segmenting dynamic features into separate objects is for dynamic object detection, it is unnecessary to segment the static features into different objects. For this reason we view all static features as part of one super-object.

Now that the relationship between features has been specified as a partial PGM in Figure 3.3, it follows that we require the distribution $p\left(s_{ij,t} \,|x_{i,t}, x_{j,t}, m_{i,t}, m_{j,t}\right)$. The nonzero entries of an unnormalized version of this distribution table are shown in Table 3.2, where we use the shorthand notation 'x' to mean either 0 or 1. The first two entries of the table represent configurations where the feature is currently moving, yet the feature is incapable of movement. In such cases we choose the probability that the features belongs to the same object as 0.5, since the way we set up other distributions will eliminate these scenarios once inference is performed. We also see an advantage of including both x-

$s_{ij,t}$	$x_{i,t}$	$x_{j,t}$	$m_{i,t}$	$m_{j,t}$	$\tilde{p}\left(s_{ij,t}\,\middle\vert\, x_{i,t}, x_{j,t}, m_{i,t}, m_{j,t}\right)$
x	1	x	0	x	0.5
x	x	1	x	0	0.5
1	0	0	0	0	1
0	0	0	1	0	1
0	0	0	0	1	1
0	1	0	1	0	1
0	0	1	0	1	1
0	1	0	1	1	1
0	0	1	1	1	1
x	0	0	1	1	0.5
x	1	1	1	1	0.5

Table 3.2: Conditional unnormalized distribution table for $\tilde{p}\left(s_{ij,t}\,\middle\vert\, x_{i,t}, x_{j,t}, m_{i,t}, m_{j,t}\right)$. Here we use the shorthand notation 'x' to indicate either 0 or 1.

variables and m-variables in the PGM: if two features have the same x-value but different m-values, they cannot be part of the same object.

The last two rows of Table 3.2 imply that if both features are semi-static and they currently have the same x-value, we need additional information to ascertain whether or not they belong to the same object. We address this issue in the next section by incorporating a similarity measurement.

3.2.2 Incorporating a feature similarity measurement

Consider two features i and j belonging to the same object. Assuming that the object is rigid, we know that the distance between the two remains constant regardless of object motion. Consequently we introduce the change in relative distance between two features as an indication of whether or not they belong to the same object. This measurement is calculated by taking the distance between two points at time $t-1$ and subtracting it from the distance between the same two points at time t. More details on this calculation follow in Chapter 5.

The incorporation of the similarity measurement into the PGM is shown in Figure 3.4, where $\hat{r}_{ij,t}$ and $r_{ij,t}$ denote the measured and true value of the change in relative distance between features i and j. An advantage of calculating the change in relative distance between two features is that the effect of pose uncertainty is eliminated in this process. Depending on the sensors involved, one may decide to incorporate other feature similarity measurements such as colour, difference in velocity vectors, etc. This is achieved simply by copying the structure in Figure 3.4 for each such measurement.

From the figure we can see that the measurement distributions $p\left(\hat{r}_{ij,t}\,\middle\vert\, r_{ij,t}\right)$ and $p\left(r_{ij,t}\,\middle\vert\, s_{ij,t}\right)$ are required. We assume that the measured values are normally distributed around the true values, i.e.

$$p\left(\hat{r}_{ij,t}\,\middle\vert\, r_{ij,t}\right) = \mathcal{N}\left(\hat{r}_{ij,t}\,\middle\vert\, r_{ij,t}, R_{ij,t}\right). \tag{3.5}$$

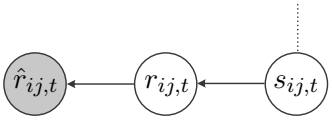


Figure 3.4: The partial PGM that shows the addition of a change in relative distance measurement between features i and j at time t. The dotted line represents previously defined relationships.

The process of estimating the covariance $R_{ij,t}$ is discussed in Chapter 5.

For the remaining distribution we assume

$$p\left(r_{ij,t}\,|s_{ij,t}=0\right)=\delta\left(r_{ij,t}\right), \tag{3.6}$$
$$p\left(r_{ij,t}\,|s_{ij,t}=1\right)=\mathcal{N}\left(r_{ij,t}\,|0,C_r\right). \tag{3.7}$$

That is, we expect the change in relative distance to be zero if the two features are on the same object, and normally distributed around zero if they are not. Again, we might be tempted to combine the distributions over $\hat{r}_{ij,t}$, $r_{ji,t}$ into a marginal distribution over $s_{ij,t}$. Essentially this is what happens when the messages from the continuous variables to the discrete ones are computed (as explained in Chapter 4).

3.2.3 Enforcing consistent object segmentation

Incorporating similarity measures between pairs of features does not guarantee a consistent final segmentation. Consider three features i, j and k connected in a triangle. If $s_{ij,t}=0$, which implies that i and j belong to different objects, but $s_{ik}=1$ and $s_{jk}=1$, we have an inconsistency. Note that here, and in the rest of this section, we omit time indices from the s-variables for neatness.

An example of such a situation is shown in Figure 3.5. In (a) a segmentation boundary (which corresponds to $s_{ij}=0$) enters a triangle formed between three feature locations and one connection is broken (in the sense that its adjacent vertices are believed to belong to different objects). It is still possible for the neighbouring edge to be intact if there is not enough evidence to support a split between those two features. However, the situation in (a) is impossible because a segmentation boundary cannot enter a triangle without leaving it, but both the situations in (b) and (c) are possible. Note that although (c) might not be the correct segmentation, it does not violate the consistent segmentation boundary requirement.

From this observation it follows that, in order to enforce a consistent segmentation, we must require for every triangle over three neighbouring features that either none of the s-variables are zero, precisely two are zero, or precisely three are zero. Consider a factor defined over the three connections in a single triangle such as the one in Table 3.3. Here the factor over the three triangle connections is shown. Note that all configurations with a single broken connection have zero entries while all others are nonzero and equal.

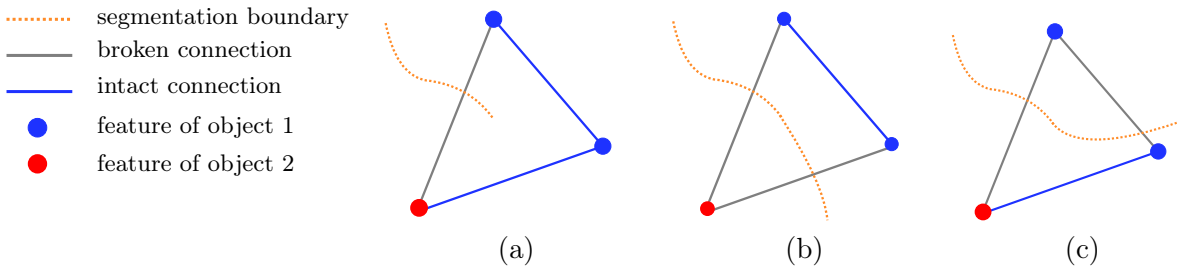


Figure 3.5: In order to obtain a consistent segmentation, a segmentation boundary entering a triangle must also leave it. Consequently the segmentation in (a) is invalid, as opposed to (b) and (c) which are valid segmentations (though not necessarily correct).

s_{ij}	s_{ik}	s_{jk}	$\phi(s_{ij}, s_{ik}, s_{jk})$
0	0	0	1
1	0	0	1
0	1	0	1
0	0	1	1
1	1	0	0
1	0	1	0
0	1	1	0
1	1	1	1

Table 3.3: Factor over s_{ij}, s_{ik}, and s_{jk}.

In 3D the Delaunay triangulation provides a set of tetrahedra that connects features. Since a tetrahedron consists of four triangles defined by four points, say features i, j, k and ℓ, each of the triangles must adhere to the same constraints specified above. If three additional factors such as the one in Table 3.3 are specified over s_{jk}, $s_{j\ell}$, $s_{k\ell}$ and s_{ij}, $s_{i\ell}$, $s_{j\ell}$ and $s_{i\ell}$, s_{ik}, $s_{k\ell}$ respectively, all four factors can be multiplied to form a factor over all six connections in the tetrahedron. The entries which, after multiplication, have nonzero probabilities are shown in Table 3.4.

By applying the restrictions above to all triangular faces of a tetrahedron over four neighbouring points, the five possible configurations in Figure 3.6 arise (up to graph isomorphism). It is easily verified that each of these configurations results in a consistent segmentation.

We expand our PGM by incorporating an additional factor over every tetrahedron as in Table 3.4. If a tetrahedron is formed over features i, j, k and ℓ, the six variables s_{ij}, s_{ik}, $s_{i\ell}$, s_{jk}, $s_{j\ell}$ and $s_{k\ell}$ are connected as shown in Figure 3.7(b). Here we depict a Markov network, since in this case it is more natural to specify a joint factor over all the variables than to specify conditional distributions in a Bayesian network.

s_{ij}	s_{ik}	$s_{i\ell}$	s_{jk}	$s_{j\ell}$	$s_{k\ell}$
0	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1
1	1	0	1	0	0
0	1	1	0	0	1
0	0	0	1	1	1
1	0	1	0	1	0
1	0	0	0	0	1
0	0	1	1	0	0
0	1	0	0	1	0
1	1	1	1	1	1

Table 3.4: Entries with nonzero probabilities in the factor over all six connections in a tetrahedron, where the continuous segmenation boundary constraints on each of the four triangles is enforced. These entries have equal probability, while all other entries have zero probability.

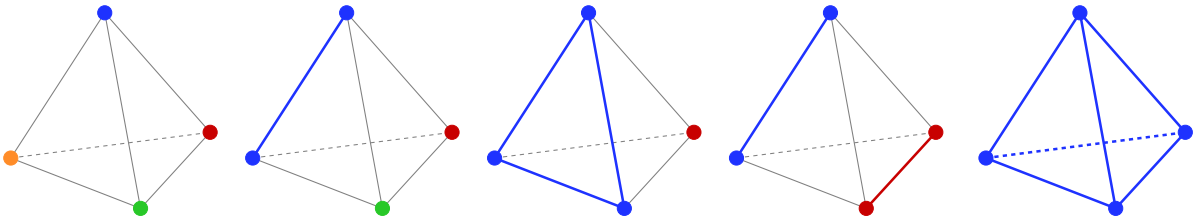


Figure 3.6: Five configurations of boundaries arising from Table 3.4 for a tetrahedron. Broken connections are indicated in grey while blue and red lines indicate intact connections between different features believed to be part of the same object.

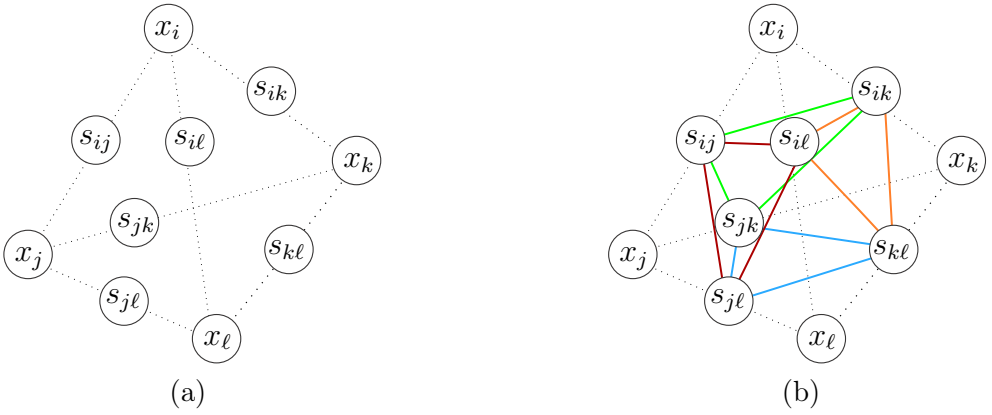
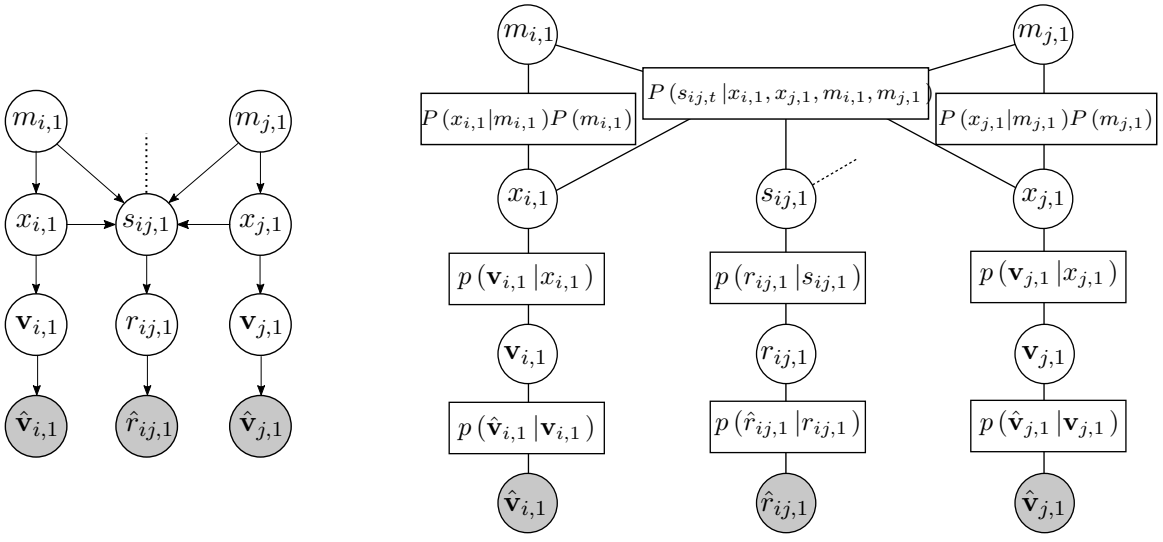


Figure 3.7: A Markov network representation of our model (a) before and (b) after adding factors to enforce consistency in the segmentation on a tetrahedron connecting four features (different colours indicate different factors).

3.3 The PGM for $t = 1$

In this work we follow the convention that $t = 1$ is the first time step where feature velocity measurements are available (in other words the second time step for which feature position estimates are available). In Figure 3.8(a) we show the entire Bayesian network for two features i and j at time step 1. The dotted line represents connections to other s-variables as a result of the continuous boundary enforcement factors, not shown to reduce clutter. The corresponding factor graph is shown in Figure 3.8(b), with factor potentials as specified in Sections 3.1 and 3.2.

Note that we do not include direct links between $\hat{\mathbf{v}}_{i,1}$, $\hat{r}_{ij,1}$ and $\hat{\mathbf{v}}_{j,1}$, even though these are dependent since the measurements are all calculated, in some way or another, from the same set of 3D feature positions. For tractable inference, however, we assume that these different measurements are mutually independent.



(a) (b)

Figure 3.8: (a) The entire Bayesian network for two features i and j at time step $t = 1$. (b) The corresponding factor graph (the dotted line represents connections to other s-variables as a result of the continuous boundary enforcement factors, not shown here to reduce clutter).

3.4 Expanding the PGM temporally

With the PGM described for time step $t = 1$, we proceed to show how more time steps can be added. Once again we split the development into two parts: motion segmentation and object segmentation.

3.4.1 Expanding the motion segmentation component

For the part of the model that relates to motion segmentation we simply duplicate the PGM from $t = 1$ for the number of time steps required. We add connections between

relevant variables across time, as is indicated in Figure 3.9. Here we assume that measurements of the velocity over time are conditionally independent given the true velocities. By linking the velocities across time steps, the PGM can model smooth feature motion and possibly obtain better performance if a feature is observed for multiple time steps. Adding such links adds complexity to the message passing algorithm, but enables more accurate motion segmentation and allows for the handling of semi-static objects.

We also assume that a previous feature state $x_{i,t-1}$ influences the current state $x_{i,t}$, allowing the PGM to express that a feature is more likely to remain in a particular state, if required. Finally, we add links between m-variables over time simply because they are all really the same variable (time subscripts are used only for convenience).

This temporal expansion requires the specification of some additional probability distributions. The distribution $p\left(m_{i,t}\left|m_{i,t-1}\right.\right)$ is trivial, since the two variables are actually the same. It remains to specify $p\left(x_{i,t}\left|x_{i,t-1}, m_{i,t}\right.\right)$ and $p\left(\mathbf{v}_{i,t}\left|\mathbf{v}_{i,t-1}, x_{i,t}\right.\right)$.

We specify $p\left(x_{i,t}\left|x_{i,t-1}, m_{i,t}\right.\right)$ as shown in Table 3.5, where a_s is the probability that a feature is currently static if it was previously static, yet has the ability to move. Similarly a_d is the probability that a feature is currently dynamic if it was previously dynamic and is capable of movement. Both these probabilities can be set to high values.

Specifying $p\left(\mathbf{v}_{i,t}\left|\mathbf{v}_{i,t-1}, x_{i,t}\right.\right)$ is equivalent to specifying motion models for $x_{i,t} = 0$ and $x_{i,t} = 1$. If the feature is static, $\mathbf{v}_{i,t}$ must be zero regardless of the value of $\mathbf{v}_{i,t-1}$, i.e.

$$p\left(\mathbf{v}_{i,t}\left|\mathbf{v}_{i,t-1}, x_{i,t} = 0\right.\right) = \delta\left(\mathbf{v}_{i,t}\right). \tag{3.8}$$

If the feature is dynamic, we adopt the widely used constant velocity motion model

$$p\left(\mathbf{v}_{i,t}\left|\mathbf{v}_{i,t-1}, x_{i,t} = 1\right.\right) = \mathcal{N}\left(\mathbf{v}_{i,t}\left|\mathbf{v}_{i,t-1}, C_v\right.\right). \tag{3.9}$$

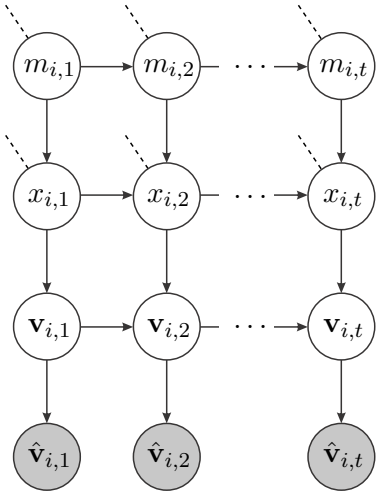


Figure 3.9: Temporal expansion of the PGM related to motion segmentation, where dotted diagonal lines represent additional relationships.

$x_{i,t}$	$x_{i,t-1}$	$m_{i,t}$	$p(x_{i,t} \mid x_{i,t-1}, m_{i,t})$
0	0	0	1
1	0	0	0
0	1	0	0.5
1	1	0	0.5
0	0	1	a_s
1	0	1	$1 - a_s$
0	1	1	$1 - a_d$
1	1	1	a_d

Table 3.5: Conditional distribution table for $p(x_{i,t} \mid x_{i,t-1}, m_{i,t})$. Here a_s and a_d are high probabilities.

A constant acceleration model is another option, but for tractability we require the model to be linear in $\mathbf{v}_{i,t}$.

Incidentally, if $\mathbf{v}_{i,t} = \mathbf{v}_{i,t-1} + \mathbf{q}_{i,t}$ and $\hat{\mathbf{v}}_{i,t} = \mathbf{v}_{i,t} + \mathbf{k}_{i,t}$, where $\mathbf{q}_{i,t}$ and $\mathbf{k}_{i,t}$ are noise vectors, our approach is similar to a Kalman filter with measurement model and prediction model given by(3.1) and (3.9) respectively. Given this similarity one can argue that our model, in some sense, also tracks features.

3.4.2 Expanding the object segmentation component

We temporally expand the part of the PGM related to object segmentation as shown in Figure 3.10. Connections from the s-variables to the x- and m-variables are not shown, but are similar to those shown in Figure 3.3. In this graph we do not include links over time between r-variables, since it adds unnecessary complexity to the message passing algorithm.

The only distribution to specify is $p(s_{ij,t} \mid s_{ij,t-1})$, which is trivial since $s_{ij,t}$ and $s_{ij,t-1}$ are the same variable (we add time indices simply for convenience).

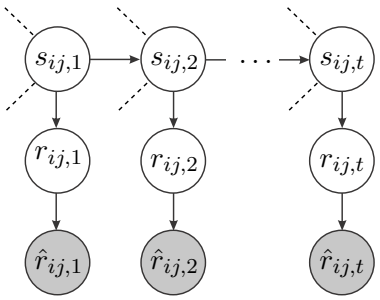


Figure 3.10: Temporal expansion of the PGM related to object segmentation, where dotted diagonal lines represent additional relationships.

The PGM proposed in this chapter consists of latent variables $\mathbf{v}_{i,t}, x_{i,t}, m_{i,t}, s_{ij,t}$ and $r_{ij,t}$, as well as observations $\hat{\mathbf{v}}_{i,t}$ and $\hat{r}_{ij,t}$. In this case all the leaf nodes ($\hat{\mathbf{v}}_{i,t}$ and $\hat{r}_{ij,t}$), which are singly-connected variables, are observations connected to continuous variables ($\mathbf{v}_{i,t}$ and $r_{ij,t}$). These continuous variables are, in particular, connected to discrete variables ($x_{i,t}$ and $s_{ij,t}$), and the discrete variables are densely inter-connected. The next chapter contains details on how to perform inference on our PGM, with particular attention to the fact that it contains both discrete and continuous random variables.

Chapter 4

Inference

In the previous chapter we described the modelling aspect of the proposed PGM. Once observations are made, inference can be performed and conclusions can be drawn about latent variables. Details on how to obtain and pre-process the measurements (or observations) are given in Chapter 5, and in this chapter we discuss how inference is performed on the PGM. Typically, message passing is performed by running a belief propagation algorithm. However, when a PGM contains both discrete and continuous random variables, as it does in our case, performing message passing is no longer straightforward. We pay particular attention to this issue, and propose a solution for our PGM.

4.1 Inference on hybrid PGMs

A hybrid PGM is a model that contains both discrete and continuous random variables, and message passing in such a PGM can be much more challenging than in a purely discrete one [36]. Challenges arise from the fact that there is no universal representation of a factor over continuous variables. Even if the same parametric family is chosen to represent each continuous distribution, operations like multiplication during message passing may result in distributions that are no longer in that same family. Since marginalization over continuous variables requires integration rather than summation, difficulties can also arise when the function is not integrable or does not have a closed-form integral.

The first exact-inference hybrid Bayesian algorithm was developed under the assumption that every continuous node has discrete parents and that each conditional probability distribution is a multivariate Gaussian [41]. The marginal distribution of each continuous variable is then assumed to be a mixture of Gaussians. Later, Koller et al. [37] proposed to model the distributions of discrete nodes with continuous parents by a mixture of exponentials, and to use Monte Carlo methods for approximate inference. In expectation propagation [53] assumed density filtering (ADF) , which is an extension of the Kalman filter, and traditional loopy belief propagation are consolidated into an accurate iterative message passing algorithm. While these methods enable inference in hybrid PGMs, they can become intractable for large networks since complexity generally grows with the size of the graph.

If the PGM consists of Gaussian random variables and discrete random variables, as our PGM does, the continuous distributions may be modelled as mixtures of Gaussians. However, the complexity of these distributions and message passing between the variables is prohibitive, and an alternative solution is required.

Our solution involves fixing the belief over all variables at time $t-1$ before performing message passing at time t. This means that the variables in the PGM are updated incrementally with each new time step, and messages are not propagated back in time. While such an approximation may seem like a limitation, three points should be noted.

The first point is that, at time step $t-1$, an appropriate action must be taken to reach time step t collision-free. Therefore, once time step $t-1$ has passed, we are no longer interested in those variables. In general loopy belief propagation, however, variables at time step $t-1$ can propagate messages to time step t multiple times, thereby altering the beliefs over variables in which we are interested (variables at time t). However, an argument can be made that messages propagated back and forth over time will not change significantly. In Chapter 6 we show that the PGM's performance increases over time, which justifies this assumption.

Secondly, by solving the inference problem incrementally, we only perform message passing at time t between variables related to those observed at time t. This means that the PGM "forgets" variables assumed to be irrelevant at time t, which has the advantage that the number of variables in the PGM does not really grow over time and complexity remains more or less the same for each time step.

The third point to note is that, by fixing the beliefs in this way, we are able to calculate closed-form expressions for the messages sent from the continuous variables (which are all leaf-node adjacent) to the discrete variables, thereby separating the PGM into continuous and discrete parts. In doing so, we can perform loopy belief propagation on only the densely inter-connected discrete variables and, after calibration, propagate messages back to the continuous variables in order to update their beliefs. Following this approach we avoid propagating messages that grow exponentially in complexity during calibration and thus enable tractable inference.

Interestingly, our approach exhibits some similarities to assumed density filtering in that observations are processed one at a time and beliefs from previous observations are not revisited or updated. During the ADF projection step, messages are projected down onto a tractable exponential family distribution by matching moments. In our approach we do not perform this reduction but keep messages in their original form.

Since there are two types of continuous random variables in our PGM, the change in relative distance between two features $r_{ij,t}$ and the velocity of a feature $\mathbf{v}_{i,t}$, we discuss the messages sent by each of these to their discrete neighbours separately, starting with the change in relative distance variable $r_{ij,t}$.

4.2 Messages passed between $r_{ij,t}$ and $s_{ij,t}$

Consider the factor graph in Figure 4.1(a). The change in relative distance measurement $\hat{r}_{ij,t}$ of the continuous latent random variable $r_{ij,t}$, and the discrete variable $s_{ij,t}$ that indicates whether or not feature i and j belong to the same object, are connected in a chain structure. Message passing on chains results in exact inference. Consequently, the message from $\hat{r}_{ij,t}$ to $r_{ij,t}$ and from $r_{ij,t}$ to $s_{ij,t}$ can be computed exactly. The message from $\hat{r}_{ij,t}$ across the factor $p(\hat{r}_{ij,t}|r_{ij,t})$ to $r_{ij,t}$ is given by

$$\mu_{\hat{r}_{ij,t}\to r_{ij,t}}(r_{ij,t}) = p(\hat{r}_{ij,t}|r_{ij,t}). \tag{4.1}$$

After receiving this message, $r_{ij,t}$ can pass the message

$$\mu_{r_{ij,t}\to s_{ij,t}}(s_{ij,t}) = \int \mu_{\hat{r}_{ij,t}\to r_{ij,t}}(r_{ij,t})\, p(r_{ij,t}|s_{ij,t})\, d\, r_{ij,t} \tag{4.2}$$

$$= \int p(\hat{r}_{ij,t}|r_{ij,t})\, p(r_{ij,t}|s_{ij,t})\, d\, r_{ij,t} \tag{4.3}$$

to $s_{ij,t}$. By substituting the appropriate conditional distributions given in Chapter 3, we obtain this message in closed-form:

$$\mu_{r_{ij,t}\to s_{ij,t}}(s_{ij,t}=0) = \mathcal{N}(\hat{r}_{ij,t}|0, R_{ij,t}+C_r), \tag{4.4}$$

$$\mu_{r_{ij,t}\to s_{ij,t}}(s_{ij,t}=1) = \mathcal{N}(\hat{r}_{ij,t}|0, R_{ij,t}). \tag{4.5}$$

Note that it can be computed directly from the measurement value $\hat{r}_{ij,t}$ and the measurement covariance $R_{ij,t}$ (C_r is a constant matrix).

Illustrations of unnormalized versions of (4.4) and (4.5) are shown in Figure 4.2. If a measurement $\hat{r}_{ij,t} = 0.9$ is received, we see that the blue curve (which corresponds to

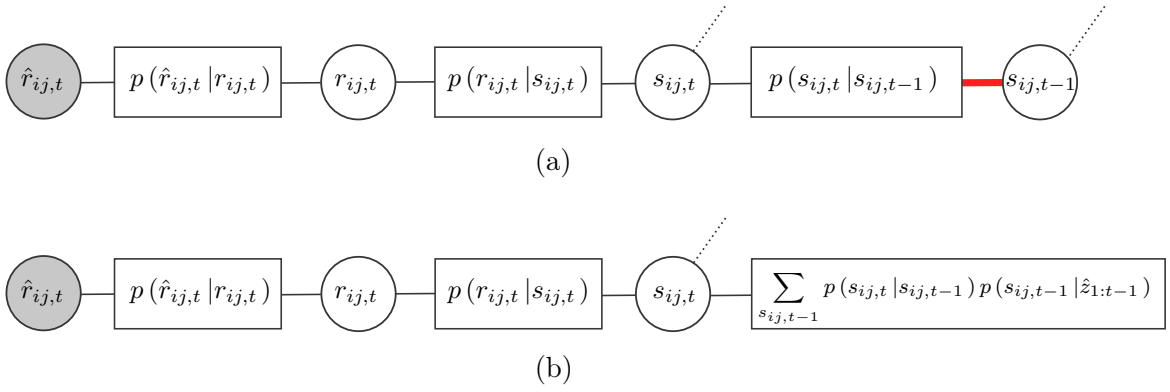


Figure 4.1: (a) Factor graph of the object segmentation component for time steps $t-1$ and t. In (b) we approximate (a) by fixing the belief over $s_{ij,t}$ and only propagating a message from this variable over the connection shown in red (not back to $s_{ij,t-1}$). Here $\hat{z}_{1:t-1}$ denotes all measurements received by the PGM up to time $t-1$ and dotted lines represent connections to other parts of the PGM (not shown for neatness).

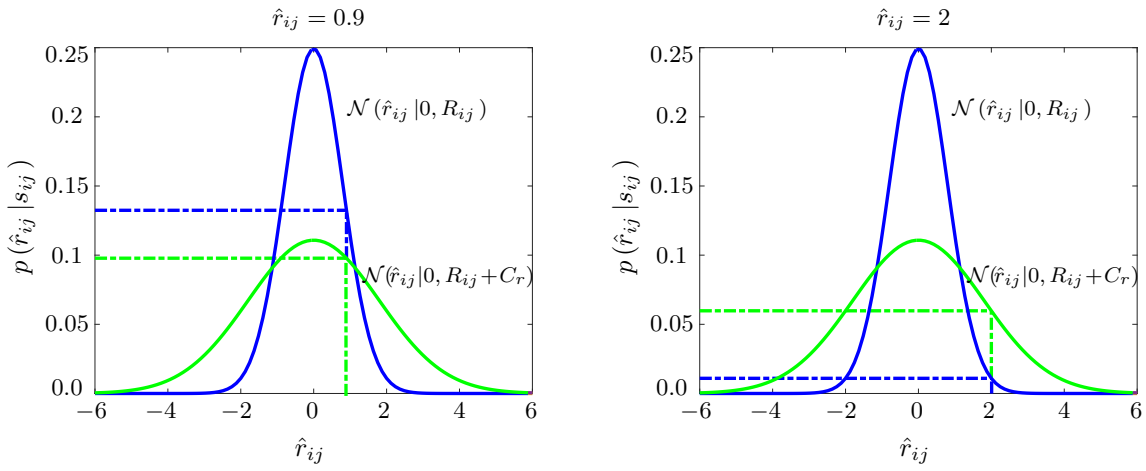


Figure 4.2: Illustrations of the messages passed to s_{ij} for two different measurements of r_{ij}. On the left the blue curve has a higher value at $\hat{r}_{ij} = 0.9$ than the green curve, which means that in this case it is more likely that $s_{ij} = 1$. On the right $\hat{r}_{ij} = 2$, and the opposite situation occurs. (Time indices are omitted from all variables to reduce clutter.)

$s_{ij,t} = 1$) has a higher value than the green curve (which corresponds to $s_{ij,t} = 0$). These two values are then normalized to form a probability distribution, which means that, in this case, it is more likely that $s_{ij,t} = 1$. On the other hand, if $\hat{r}_{ij,t} = 2$, we see that the green curve has a higher value than the blue one, which implies a higher probability for $s_{ij,t} = 0$.

The variable $s_{ij,t}$ first receives the message from $s_{ij,t-1}$ before it propagates its message to the rest of the PGM. Since $s_{ij,t}$ is connected to $x_{i,t}$ and $s_{ij,t-1}$ is connected to $x_{i,t-1}$, the two s-variables form part of a loop and thus the message from $s_{ij,t-1}$ cannot be pre-computed. However, for tractability we assume that this message is fixed in the sense that future messages cannot alter it. This implies that we fix the belief over $s_{ij,t-1}$, which can be obtained from a PGM query as $p(s_{ij,t-1}|\hat{z}_{1:t-1})$ where $\hat{z}_{1:t-1}$ denotes all measurements received by the PGM up to time $t-1$. In doing so the connection between $s_{ij,t}$ and $s_{ij,t-1}$ is replaced by the factor $p(s_{ij,t}|s_{ij,t-1})\,p(s_{ij,t-1}|\hat{z}_{1:t-1})$ after marginalization over $s_{ij,t-1}$, as shown in Figure 4.1(b). The message passed to $s_{ij,t}$ may then be calculated as

$$\mu_{s_{ij,t-1}\to s_{ij,t}}(s_{ij,t} = 0) = p(s_{ij,t-1} = 0\,|\hat{z}_{1:t-1}), \tag{4.6}$$

$$\mu_{s_{ij,t-1}\to s_{ij,t}}(s_{ij,t} = 1) = p(s_{ij,t-1} = 1\,|\hat{z}_{1:t-1}). \tag{4.7}$$

In standard belief propagation messages are also passed back to $r_{ij,t}$ but, since knowledge of the belief over $r_{ij,t}$ is not necessary for our goals and $r_{ij,t}$ does not influence any other variables, we omit this step.

Both $\mu_{\hat{r}_{ij,t}\to s_{ij,t}}(s_{ij,t})$ and $\mu_{s_{ij,t-1}\to s_{ij,t}}(s_{ij,t})$ can be viewed as factors over $s_{ij,t}$, and are supplied to the sum product algorithm at time t before inference is performed. In fact, by calculating these messages directly from the PGM at time $t-1$ and the measurement $\hat{r}_{ij,t}$, we eliminate the need to perform inference on the continuous variables and

loopy belief propagation can be performed over the discrete variables only. This means that, once observations are made, we calculate the messages propagated from $\hat{r}$-variables to s-variables using (4.4) and (4.5). Since s-variables are only connected to other discrete variables, we eliminate the need for performing inference over continuous variables from the object segmentation component of the PGM. The same applies to the motion segmentation component, as we see in the next section.

4.3 Messages passed between $\mathbf{v}_{i,t}$ and $x_{i,t}$

Consider the connections in Figure 3.9(a). Deriving expressions for the messages passed to $x_{i,t}$ is not as simple as for $s_{ij,t}$, since the latent continuous variable $\mathbf{v}_{i,t}$ depends on $\mathbf{v}_{i,t-1}$ and influences $\mathbf{v}_{i,t+1}$. However, it turns out that, by fixing the beliefs over variables at time $t-1$, we can find closed-form expressions for the messages passed from the continuous variables to the discrete ones analytically, so that loopy belief propagation can be performed on the discrete variables only.

We first calculate the message passed from $\mathbf{v}_{i,t}$ to $x_{i,t}$ along with any additional factors that may arise over the discrete variables $x_{i,t}$ and $m_{i,t}$. With these messages available, we only need to perform belief propagation between the discrete variables. Once the PGM is calibrated, we calculate the belief over $\mathbf{v}_{i,t}$ given all the measurements received up to time t. We then assume that the messages passed from variables that exist at time t to variables that exist at time $t+1$ are fixed, i.e. that no updates are sent back to variables at time t after inference is performed at time $t+1$. This allows for the beliefs over $\mathbf{v}_{i,t}$, $x_{i,t}$ and $m_{i,t}$ to be fixed when propagated to time step $t+1$. In the rest of this section we consider a particular feature, and for neatness omit feature subscripts from all variables (e.g. $x_{i,t}$ is written as x_t).

4.3.1 Message passing at $t = 1$

Consider the factor graph for the first time step shown in Figure 4.3(a). The message passed from the continuous variable $\mathbf{v}_1$ to the discrete variable x_1 is the integral of $p(\hat{\mathbf{v}}_1 | \mathbf{v}_1) p(\mathbf{v}_1 | x_1)$ over $\mathbf{v}_1$. Substituting the measurement model $p(\hat{\mathbf{v}}_1 | \mathbf{v}_1)$ and the motion model $p(\mathbf{v}_1 | x_1)$ from Chapter 3 yields

$$\mu_{\mathbf{v}_1 \to x_1}(x_1 = 0) = \mathcal{N}(\hat{\mathbf{v}}_1 | \mathbf{0}, V_1), \tag{4.8}$$

$$\mu_{\mathbf{v}_1 \to x_1}(x_1 = 1) = \mathcal{N}(\hat{\mathbf{v}}_1 | \mathbf{0}, V_1 + C_v), \tag{4.9}$$

after application of the product of Gaussians identity (see Appendix A). The message, which can be viewed as a factor over x_1, can be computed directly from the measurement $\hat{\mathbf{v}}_1$ and its covariance V_1 (C_v is a constant matrix). The message is supplied to the sum product algorithm, along with the factors $p(x_1 | m_1)$ and $p(m_1)$ as defined in Chapter 3. By doing so the continuous part of the PGM is replaced by a discrete factor, which is equivalent to the original PGM as viewed from the discrete part. Recall from Chapter 3 that x_1 and m_1 are connected to the discrete s-variables, which indicate whether or not two features belong to the same object.

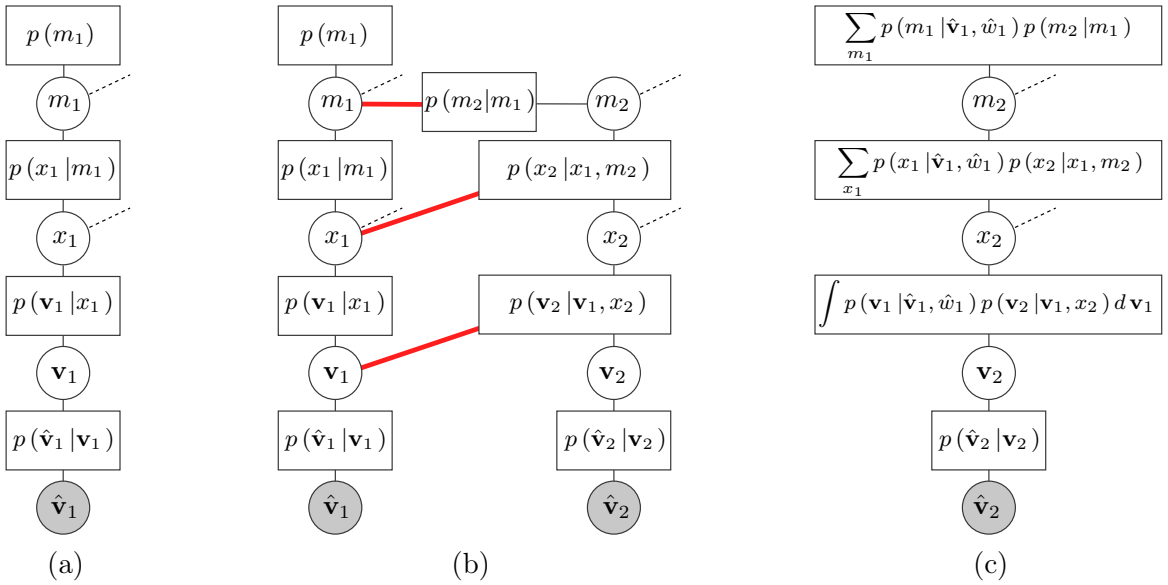


Figure 4.3: Factor graphs of the motion segmentation component for (a) $t = 1$, (b) $t = \{1, 2\}$ and (c) $t = 2$ (connections shown in bold red indicate that those messages are passed only from time step $t = 1$ to $t = 2$ and we assume them to be fixed).

After message passing between the discrete variables, a message is passed back from the discrete variable x_1 to the continuous variable $\mathbf{v}_1$. At this point the belief over x_1 can be inferred by querying the PGM, and is given by $p(x_1|\hat{\mathbf{v}}_1, \hat{w}_1)$ where $\hat{w}_1$ denotes all measurements received by the PGM excluding $\hat{\mathbf{v}}_1$. The belief over x_1 consists of the product of all incoming messages. Consequently, the messages from x_1 to the factor $p(\mathbf{v}_1|x_1)$ and from that factor to $\mathbf{v}_1$ are

$$\mu_{x_1 \to p(\mathbf{v}_1|x_1)}(\mathbf{v}_1, x_1) = \frac{p(x_1|\hat{\mathbf{v}}_1, \hat{w}_1)}{\mu_{\mathbf{v}_1 \to x_1}(x_1)}, \tag{4.10}$$

$$\mu_{p(\mathbf{v}_1|x_1) \to \mathbf{v}_1}(\mathbf{v}_1) = \sum_{x_1} \frac{p(x_1|\hat{\mathbf{v}}_1, \hat{w}_1)}{\mu_{\mathbf{v}_1 \to x_1}(x_1)} p(\mathbf{v}_1|x_1). \tag{4.11}$$

The belief over $\mathbf{v}_1$ is therefore

$$\begin{aligned} p(\mathbf{v}_1|\hat{\mathbf{v}}_1, \hat{w}_1) &= \mu_{\hat{\mathbf{v}}_1 \to \mathbf{v}_1}(\mathbf{v}_1)\ \mu_{p(\mathbf{v}_1|x_1) \to \mathbf{v}_1}(\mathbf{v}_1) \\ &= p(\hat{\mathbf{v}}_1|\mathbf{v}_1) \sum_{x_1} \frac{p(x_1|\hat{\mathbf{v}}_1, \hat{w}_1)}{\mu_{\mathbf{v}_1 \to x_1}(x_1)} p(\mathbf{v}_1|x_1) \\ &= \mathcal{N}(\hat{\mathbf{v}}_1|\mathbf{v}_1, V_1) \left[\frac{\ell_0^{(1)} \delta(\mathbf{v}_1)}{\mathcal{N}(\hat{\mathbf{v}}_1|\mathbf{0}, V_1)} + \frac{\ell_1^{(1)} \mathcal{N}(\mathbf{v}_1|\mathbf{0}, C_v)}{\mathcal{N}(\hat{\mathbf{v}}_1|\mathbf{0}, V_1 + C_v)} \right] \\ &= \ell_0^{(1)} \delta(\mathbf{v}_1) + \ell_1^{(1)} \mathcal{N}\left(\mathbf{v}_1 \middle| \boldsymbol{\mu}_1^{(1)}, \Sigma_1^{(1)}\right), \end{aligned} \tag{4.12}$$

where $\ell_u^{(t)} = p(x_t = u|\hat{\mathbf{v}}_{1:t}, \hat{w}_{1:t})$, $\Sigma_1^{(t)} = \left(C_v^{-1} + V_t^{-1}\right)^{-1}$ and $\boldsymbol{\mu}_1^{(t)} = \Sigma_1^{(t)} \left(V_t^{-1} \hat{\mathbf{v}}_t\right)$.

4.3.2 Message passing at $t = 2$

We are constructing an incremental message passing algorithm, and next we consider the factor graph for the first two time steps, shown in Figure 4.3(b). As mentioned earlier, we fix the beliefs over m_1, x_1 and $\mathbf{v}_1$, which are given by $p(m_1 | \hat{\mathbf{v}}_1, \hat{w}_1)$, $p(x_1 | \hat{\mathbf{v}}_1, \hat{w}_1)$ and $p(\mathbf{v}_1 | \hat{\mathbf{v}}_1, \hat{w}_1)$ respectively. As for $s_{ij,t-1}$ we replace connections between fixed-belief variables and variables at time $t = 2$ by appropriate factors, as shown in Figure 4.3(c), thereby decoupling the two time steps.

The message from $\mathbf{v}_2$ to x_2 is

$$\mu_{\mathbf{v}_2 \to x_2}(x_2) = \int p(\hat{\mathbf{v}}_2 | \mathbf{v}_2) \int p(\mathbf{v}_1 | \hat{\mathbf{v}}_1, \hat{w}_1) \, p(\mathbf{v}_2 | \mathbf{v}_1, x_2) \, d\mathbf{v}_1 \, d\mathbf{v}_2, \tag{4.13}$$

and by substituting the conditional distributions in (3.1), (3.8) and (3.9),

$$\mu_{\mathbf{v}_2 \to x_2}(x_2 = 0) = (\ell_0^{(1)} + \ell_1^{(1)}) \mathcal{N}(\hat{\mathbf{v}}_2 | \mathbf{0}, V_2), \tag{4.14}$$

$$\mu_{\mathbf{v}_2 \to x_2}(x_2 = 1) = \ell_0^{(1)} \mathcal{N}(\hat{\mathbf{v}}_2 | \mathbf{0}, V_2 + C_v) + \ell_1^{(1)} \mathcal{N}(\hat{\mathbf{v}}_2 | \boldsymbol{\mu}_1, V_2 + C_v + \Sigma_1). \tag{4.15}$$

The factors in Figure 4.3(c) that remain are

$$\phi(x_2, m_2) = \sum_{x_1} p(x_1 | \hat{\mathbf{v}}_1, \hat{w}_1) \, p(x_2 | x_1, m_2), \tag{4.16}$$

$$\phi(m_2) = \sum_{m_1} p(m_1 | \hat{\mathbf{v}}_1, \hat{w}_1) \, p(m_2 | m_1), \tag{4.17}$$

which, after substitution of the appropriate conditional distributions, can be computed as

$$\phi(x_2 = 0, m_2 = 0) = \ell_0^{(1)} + \tfrac{1}{2}\ell_1^{(1)}, \tag{4.18}$$

$$\phi(x_2 = 0, m_2 = 1) = \ell_0^{(1)}(a_s) + \ell_1^{(1)}(1 - a_d), \tag{4.19}$$

$$\phi(x_2 = 1, m_2 = 0) = 0, \tag{4.20}$$

$$\phi(x_2 = 1, m_2 = 1) = \ell_0^{(1)}(1 - a_s) + \ell_1^{(1)}(a_d), \tag{4.21}$$

$$\phi(m_2 = 0) = p(m_1 = 0 | \hat{\mathbf{v}}_1, \hat{w}_1), \tag{4.22}$$

$$\phi(m_2 = 1) = p(m_1 = 1 | \hat{\mathbf{v}}_1, \hat{w}_1). \tag{4.23}$$

These two factors together with $\mu_{\mathbf{v}_2 \to x_2}(x_2)$ are supplied to the sum product algorithm at time $t = 2$ before inference is performed. All three can be calculated from the measurement $\hat{\mathbf{v}}_2$ and covariance V_2 as well as the beliefs calculated at $t = 1$.

After inference, the belief over $\mathbf{v}_2$ is updated (as before):

$$\begin{aligned} p(\mathbf{v}_2 | \hat{\mathbf{v}}_{1:2}, \hat{w}_{1:2}) &= \mu_{\hat{\mathbf{v}}_2 \to \mathbf{v}_2}(\mathbf{v}_2) \; \mu_{x_2 \to \mathbf{v}_2}(\mathbf{v}_2) \\ &= p(\hat{\mathbf{v}}_2 | \mathbf{v}_2) \sum_{x_2} \frac{p(x_2 | \hat{\mathbf{v}}_{1:2}, \hat{w}_{1:2}) \int p(\mathbf{v}_1 | \hat{\mathbf{v}}_1, \hat{w}_1) \, p(\mathbf{v}_2 | \mathbf{v}_1, x_2) \, d\mathbf{v}_1}{\mu_{\mathbf{v}_2 \to x_2}(x_2)} \\ &= \ell_0^{(2)} \delta(\mathbf{v}_2) + C_1^{(2)} \mathcal{N}\left(\mathbf{v}_2 \middle| \boldsymbol{\mu}_1^{(2)}, \Sigma_1^{(2)}\right) + C_2^{(2)} \mathcal{N}\left(\mathbf{v}_2 \middle| \boldsymbol{\mu}_2^{(2)}, \Sigma_2^{(2)}\right), \end{aligned} \tag{4.24}$$

where we define

$$C_1^{(2)} = \frac{\ell_1^{(2)}\ell_0^{(1)}\mathcal{N}\left(\hat{\mathbf{v}}_2\,|\mathbf{0}, V_2 + C_v\right)}{\mu_{\mathbf{v}_2\to x_2}\left(x_2=1\right)}, \tag{4.25}$$

$$C_2^{(2)} = \frac{\ell_1^{(2)}\ell_1^{(1)}\mathcal{N}\left(\hat{\mathbf{v}}_2\,\middle|\boldsymbol{\mu}_1^{(1)}, \Sigma_1^{(1)} + C_v + V_2\right)}{\mu_{\mathbf{v}_2\to x_2}\left(x_2=1\right)}, \tag{4.26}$$

$$\Sigma_i^{(t)} = \left(\left(C_v + \Sigma_{i-1}^{(t-1)}\right)^{-1} + V_t^{-1}\right)^{-1}, \quad i \in \{2,\ldots,t\}, \tag{4.27}$$

$$\boldsymbol{\mu}_i^{(t)} = \Sigma_1^{(t)}\left(\left(C_v + \Sigma_{i-1}^{(t-1)}\right)^{-1}\boldsymbol{\mu}_{i-1}^{(t-1)} + V_t^{-1}\hat{\mathbf{v}}_t\right), \quad i \in \{2,\ldots,t\}. \tag{4.28}$$

To summarize, at the end of time step $t = 2$, the velocity distribution is a mixture of Gaussians, and the PGM only consists of variables at time step 2, which means that the PGM has not grown in size. Also, we have decoupled time steps 1 and 2.

4.3.3 Message passing at $t > 2$

The described scheme can be generalized to any two consecutive time steps. To this end, suppose the belief over $\mathbf{v}_{t-1}$ is given by

$$p\left(\mathbf{v}_{t-1}\,|\hat{\mathbf{v}}_{1:t-1}, \hat{w}_{1:t-1}\right) = \ell_0^{(t-1)}\delta(\mathbf{v}_{t-1}) + \sum_{i=1}^{t-1} C_i^{(t-1)}\mathcal{N}\left(\mathbf{v}_{t-1}\,\middle|\boldsymbol{\mu}_i^{(t-1)}, \Sigma_i^{(t-1)}\right), \tag{4.29}$$

which we know holds for $t = 3$ (compare (4.24) and (4.29)). Following similar steps as before, it can be shown that by computing

$$\mu_{\mathbf{v}_t\to x_t}\left(x_t\right) = \int p\left(\hat{\mathbf{v}}_t\,|\mathbf{v}_t\right)\int p\left(\mathbf{v}_{t-1}\,|\hat{\mathbf{v}}_{1:t-1}, \hat{w}_{1:t-1}\right)p\left(\mathbf{v}_t\,|\mathbf{v}_{t-1}, x_t\right)d\mathbf{v}_{t-1}\,d\mathbf{v}_t, \tag{4.30}$$

the message passed from $\mathbf{v}_t$ to x_t is given by

$$\mu_{\mathbf{v}_t\to x_t}\left(x_t=0\right) = \left(\ell_0^{(t-1)} + \sum_{i=1}^{t-1} C_i^{(t-1)}\right)\mathcal{N}\left(\hat{\mathbf{v}}_t\,|\mathbf{0}, V_t\right), \tag{4.31}$$

$$\mu_{\mathbf{v}_t\to x_t}\left(x_t=1\right) = \ell_0^{(t-1)}\mathcal{N}\left(\hat{\mathbf{v}}_t\,|\mathbf{0}, V_t+C_v\right) + \sum_{i=1}^{t-1} C_i^{(t-1)}\mathcal{N}\left(\hat{\mathbf{v}}_t\,\middle|\boldsymbol{\mu}_i^{(t-1)}, \Sigma_i^{(t-1)}+V_t+C_v\right). \tag{4.32}$$

The factors $\phi\left(x_t, m_t\right)$ and $\phi\left(m_t\right)$ turn out to be

$$\phi\left(x_t=0, m_t=0\right) = \ell_0^{(t-1)} + \tfrac{1}{2}\ell_1^{(t-1)}, \tag{4.33}$$

$$\phi\left(x_t=0, m_t=1\right) = \ell_0^{(t-1)}\left(a_s\right) + \ell_1^{(t-1)}\left(1-a_d\right), \tag{4.34}$$

$$\phi\left(x_t=1, m_t=0\right) = 0, \tag{4.35}$$

$$\phi\left(x_t=1, m_t=1\right) = \ell_0^{(t-1)}\left(1-a_s\right) + \ell_1^{(t-1)}\left(a_d\right), \tag{4.36}$$

$$\phi\left(m_t=0\right) = p\left(m_{t-1}=0\,|\hat{\mathbf{v}}_{1:t-1}, \hat{w}_{1:t-1}\right), \tag{4.37}$$

$$\phi\left(m_t=1\right) = p\left(m_{t-1}=1\,|\hat{\mathbf{v}}_{1:t-1}, \hat{w}_{1:t-1}\right), \tag{4.38}$$

and can be calculated from the measurement $\hat{\mathbf{v}}_t$ with covariance V_t and the beliefs from time $t-1$. Inference is performed, whereafter we calculate the belief over $\mathbf{v}_t$ as

$$\begin{aligned} & p\left(\mathbf{v}_t \left| \hat{\mathbf{v}}_{1:t}, \hat{w}_{1:t} \right.\right) \\ & \quad \propto \mu_{\hat{\mathbf{v}}_t \to \mathbf{v}_t}\left(\mathbf{v}_t\right)\ \mu_{x_t \to \mathbf{v}_t}\left(\mathbf{v}_t\right) \\ & \quad \propto p\left(\hat{\mathbf{v}}_t \left| \mathbf{v}_t \right.\right) \sum_{x_t} \frac{p\left(x_t \left| \hat{\mathbf{v}}_{1:t}, \hat{w}_{1:t} \right.\right)}{\mu_{\mathbf{v}_t \to x_t}\left(x_t\right)} \int p\left(\mathbf{v}_{t-1} \left| \hat{\mathbf{v}}_{1:t-1}, \hat{w}_{1:t-1} \right.\right) p\left(\mathbf{v}_t \left| \mathbf{v}_{t-1}, x_t \right.\right) d\,\mathbf{v}_{t-1} \\ & \quad \propto \ell_0^{(t)} \delta(\mathbf{v}_t) + \sum_{i=1}^{t} C_i^{(t)} \mathcal{N}\left(\mathbf{v}_t \left| \boldsymbol{\mu}_i^{(t)}, \Sigma_i^{(t)} \right.\right), \end{aligned} \tag{4.39}$$

where

$$C_1^{(t)} = \frac{\ell_1^{(t)} \ell_0^{(t-1)} \mathcal{N}\left(\hat{\mathbf{v}}_t \left| \mathbf{0}, V_t + C_v \right.\right)}{\mu_{\mathbf{v}_t \to x_t}\left(x_t = 1\right)}, \tag{4.40}$$

$$C_i^{(t)} = \frac{C_{i-1}^{(t-1)} \ell_1^{(t)} \mathcal{N}\left(\hat{\mathbf{v}}_t \left| \boldsymbol{\mu}_{i-1}^{(t-1)}, \Sigma_{i-1}^{(t-1)} + C_v + V_t \right.\right)}{\mu_{\mathbf{v}_t \to x_t}\left(x_t = 1\right)}, \quad i \in \{2, \ldots, t\}. \tag{4.41}$$

The belief has the same form as (4.29), and by induction we now know that it holds for any $t > 2$. Consequently, the message $\mu_{\mathbf{v}_t \to x_t}\left(x_t\right)$ can be calculated according to (4.31) and (4.32) using the C_i-, $\boldsymbol{\mu}_i$- and Σ_i-constants calculated at time $t-1$, the belief over x_{t-1} obtained after inference at time $t-1$, and the current velocity measurement $\hat{\mathbf{v}}_t$ with covariance V_t. Note that the number of components in the mixture of Gaussians in (4.39) grow by one component with each time step. If the robot is moving, this may not be a problem since a feature is typically observed only for a short period of time. One may also decide to discard components by approximating the relevant term with a single normal distribution, if necessary.

4.4 Summary

With the full structure of our PGM in place, as well as a means to compute messages between the continuous and discrete variables, we can now summarize the complete operation of the system. At a particular time step the following steps are performed.

1. A set of 3D coordinates of features is obtained. This set contains only features successfully matched with the previous time step (if a feature disappears for a frame and later reappears, it will be handled as if seen for the first time). Distributions over velocity measurements ($\hat{\mathbf{v}}_{i,t}$) are generated from the 3D feature coordinates, as explained in Chapter 5, with due incorporation of uncertainty in the measurements as well as the uncertainty in the estimate of the robot's current pose. Measurement distributions of relative change in distance between two features ($\hat{r}_{ij,t}$) are also generated for pairs of features, as is also explained in Chapter 5. For features already under consideration, random variables are connected in the PGM to those at the previous time step, while for new features random variables are initialized in a way similar to what we do for features at the first time step (Section 3.3).

2. For tractable inference in our PGM, we pre-compute messages from the continuous variables to the discrete ones before we perform belief propagation in the next step. More specifically, for every feature i we compute the message $\mu_{\mathbf{v}_{i,t} \to x_{i,t}}(x_{i,t})$ according to (4.31) and (4.32), as well as the factors $\phi(x_t, m_t)$ and $\phi(m_t)$ given in (4.38), and all three are supplied to the sum product algorithm. For every possible pair of features (as dictated by a Delaunay triangulation) we also compute a message sent to s_{ij} (the variable that indicates whether or not features i and j belong to the same object) as described in Section 4.2.

3. Finally we run loopy belief propagation over the discrete variables, to infer posterior distributions over those variables given the measurements and messages from the continuous variables. After belief propagation we infer the belief over $\mathbf{v}_{i,t}$ according to (4.39).

Once these steps are completed, marginal distributions over variables of interest can be inferred. From these marginal distributions we can apply a threshold to the feature state probabilities $x_{i,t}$ at a particular time step, for a motion segmentation of all the features observed at time t. Similarly, a grouping of all features for which the belief over $s_{ij,t}$ is higher than a chosen threshold provides an object segmentation of the dynamic features observed at time t.

Contrary to traditional belief propagation algorithms, the message passing algorithm proposed in this chapter is incremental in the sense that time step $t-1$ is decoupled from time step t. Consequently, the PGM does not grow in size over time, since the only variables included in message passing at time t are those related to the observations made at time t. We also avoid propagating messages that grow exponentially in complexity during calibration and thus enable tractable inference.

Chapter 5

Measurement pre-processing

Our PGM approach to solving the dynamic object detection problem is compatible with various types of sensors and environments. In order to evaluate performance we consider the KITTI datasets [23]. They are well suited for our approach since they provide a sequence of stereo images captured from a platform driving through urban environments. Environment and object types vary across the sets, which allows for testing in various scenarios. These datasets have also become a popular benchmark against which the performance of different algorithms can be compared [79]. Our approach requires feature velocities as well as change in relative distance measurements, and both must be characterized with Gaussian distributions. In this chapter we discuss how to extract these requirements from the KITTI datasets. We stress that our method for detecting dynamic objects is not limited to these sets. Even though the discussion here may be viewed as an example of measurement pre-processing, most practical systems may require similar calculations.

We start by discussing the KITTI sensor set-up, followed by how a position estimate relative to the stereo rig for a particular feature is obtained. The KITTI vehicle is also equipped with an IMU/GPS unit for pose estimation. To calculate the feature position estimates in a global coordinate system, we have to understand the raw output from the IMU/GPS unit and describe how these pose measurements are applied to transform camera coordinates to global coordinates. Sections 5.1 to 5.4 contain details on how to calculate distributions over global 3D feature positions. Sections 5.5 and 5.6 describe the general procedure to obtain feature velocity measurements and change in relative distance measurements.

5.1 Sensor set-up

In Figure 5.1 a schematic representation of the sensor set-up is shown. In particular, the KITTI datasets provide sequences of stereo images after stereo rectification and distortion removal [26]. The vehicle is also equipped with an IMU and GPS for localization measurements. Measurements from a Velodyne laser range scanner are also available, but we do not use those. It is worth mentioning, however, because the KITTI development kit provides homogeneous coordinate transformations between the IMU and the range

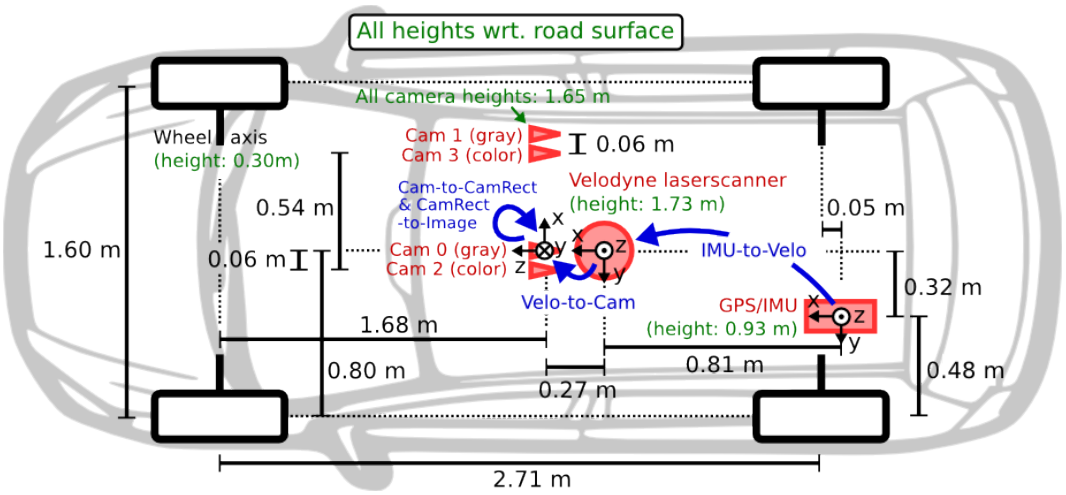


Figure 5.1: Schematic of the KITTI vehicle sensor set-up. Image reproduced with permission from www.cvlibs.net.

scanner, and between the range scanner and the stereo rig. Coordinate conversion between the stereo rig and the IMU therefore occurs via the range scanner, as we explain in Section 5.3.

5.2 Feature positions in the camera coordinate system

Our PGM assumes that all measurement models are described by Gaussian error distributions, i.e. that each is described by a mean and covariance where the mean is the measured value. Recall from Chapter 3 that we require two types of measurement distributions: one over the velocity of a feature, and one over the change in relative distance between two features. For the first of these we require a distribution over every global 3D feature position and for the second we require a distribution over the 3D position of a feature in a body-fixed coordinate system. Sections 5.5 and 5.6 deal with how these distributions can be calculated. In this section we focus on how to obtain a distribution over every 3D feature position in the camera body-fixed system.

In order to obtain a distribution over position, a particular feature under consideration must be observed from at least two spatially separate viewpoints. The process of determining the 3D position from image coordinates in two images is known as triangulation, and a schematic representation is shown in Figure 5.2. In reality the 3D position $\boldsymbol{X}_{\text{cam}}$ of a feature in the camera coordinate system is projected onto the left and right camera coordinates $\boldsymbol{x}_L$ and $\boldsymbol{x}_R$ respectively. From these image coordinates the 3D position $\boldsymbol{X}_{\text{cam}}$ can be recovered, assuming that the two cameras in the stereo rig are calibrated [26].

5.2.1 Feature extraction

The first step towards triangulation is to detect and match a set of features that occur in both the left and right camera images at a particular time t. In order to do so, interest

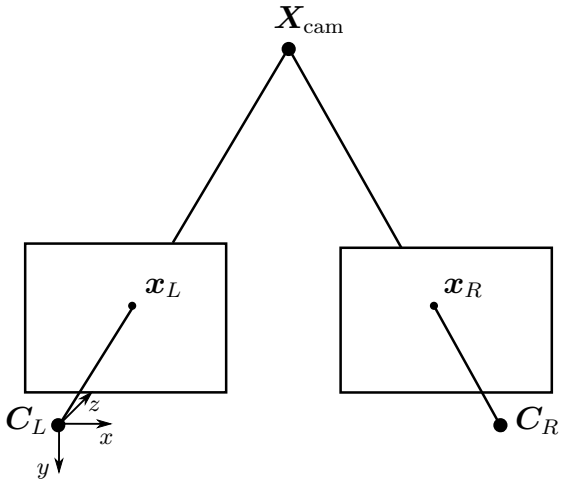


Figure 5.2: Schematic representation of the triangulation process. $\boldsymbol{C}_L$ and $\boldsymbol{C}_R$ are the left and right camera centres respectively, while $\boldsymbol{x}_L$ and $\boldsymbol{x}_R$ are the image coordinates of the projections of point $\boldsymbol{X}_{\text{cam}}$ (after stereo rectification) in the left and right images. In the KITTI set-up $\boldsymbol{C}_L$ is chosen as the origin of the camera coordinate system, with axes as shown.

points are detected separately in each image using a feature extractor such as Good Features to Track [69], SIFT [44], SURF [5] or ORB [66]. Next, the descriptor vectors of features are matched across the left and right images exhaustively or by means of an optimized nearest neighbour search [55]. The same process can be used to match features over consecutive time steps. If a match cannot be found for a feature in the left or right image, or if the feature cannot be matched with a feature from the previous time step, that feature is not included in the PGM. This implies that the number of features that can be matched depends largely on the speed of the robot. If it is moving slowly the likelihood of observing the same feature in consecutive time steps is higher than when the robot is moving at a high speed. On the other hand, features may reappear after being obscured from view. Unless such features are compared to all previously observed features, which is computationally intensive, they will be regarded as newly observed features.

Suppose the left and right image coordinates of a particular feature at time t are

$$\boldsymbol{x}_{L,t} = \begin{bmatrix} x_{L,t} \\ y_{L,t} \end{bmatrix}, \qquad \boldsymbol{x}_{R,t} = \begin{bmatrix} x_{R,t} \\ y_{R,t} \end{bmatrix}, \tag{5.1}$$

respectively. If the images are rectified [26], we expect $y_{L,t} = y_{R,t}$. Consequently we discard any features for which $|y_{L,t} - y_{R,t}| > 1$, i.e. where their y-image coordinates differ by more than a single pixel. This restriction rejects some outliers, and we compute a new y-image coordinate $y_t = \frac{1}{2}(y_{L,t} + y_{R,t})$ for every remaining feature.

The process of locating feature points in a discrete pixel space can lead to uncertainty. It is common practice to assume that such uncertainties can be modelled effectively by a Gaussian distribution [48] centred around the detected coordinates. In our experiments we determined that a variance of 0.04 pixels in the coordinates $x_{L,t}$ and $x_{R,t}$ is suitable, and a variance of 0.02 in y_t.

5.2.2 Triangulation

From the feature coordinates in the left and right image we can compute 3D coordinates relative to the current camera position at time t. This is achieved through the transformation

$$\boldsymbol{X}_{\text{cam},t} = \begin{bmatrix} \frac{(x_{L,t}-p_x)B}{x_{L,t}-x_{R,t}} \\ \frac{(y_t-p_y)B}{x_{L,t}-x_{R,t}} \\ \frac{fB}{x_{L,t}-x_{R,t}} \end{bmatrix}, \tag{5.2}$$

where B is the distance between the centres of the two cameras (the baseline), f is the focal length and (p_x, p_y) is the principal point [26]. Axis definitions are shown in Figure 5.2, and correspond to the KITTI axis definitions.

We require Gaussian distributions over all measurements. We follow standard practice by assuming a Gaussian distribution over the image coordinate measurements, but note that the transformation in (5.2) is nonlinear in the vector

$$\boldsymbol{x}' = [\ x_{L,t},\ x_{R,t},\ y_t\ ]^T. \tag{5.3}$$

Therefore, a linearization process is required to ensure that the distribution over $\boldsymbol{X}_{\text{cam}}$ is also Gaussian, and we proceed to compute the accompanying covariance matrix $\Sigma_{\text{cam},t}$. For stereo triangulation it is common practice to perform the linearization using the Taylor approximation (as described in Appendix B) [10]. This method requires the Jacobian matrix which we compute as

$$J_t = \begin{bmatrix} \frac{B(p_x-x_{R,t})}{d^2} & \frac{B(x_{L,t}-p_x)}{d^2} & 0 \\ \frac{B(p_y-y_t)}{d^2} & \frac{B(y_t-p_y)}{d^2} & \frac{B}{d} \\ -\frac{Bf}{d^2} & \frac{Bf}{d^2} & 0 \end{bmatrix}, \tag{5.4}$$

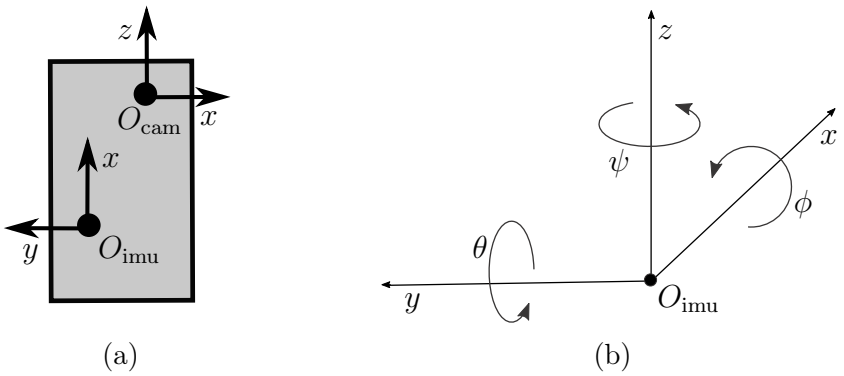
where $d = x_{L,t} - x_{R,t}$. The resulting covariance matrix is computed as

$$\Sigma_{\text{cam},t} = J_t \begin{bmatrix} 0.04 & 0 & 0 \\ 0 & 0.04 & 0 \\ 0 & 0 & 0.02 \end{bmatrix} J_t^T. \tag{5.5}$$

With both $\boldsymbol{X}_{\text{cam},t}$ and $\Sigma_{\text{cam},t}$ known, the Gaussian distribution over the 3D feature position relative to the stereo rig from which it is observed, is described.

5.3 Feature positions in the IMU coordinate system

Since $\boldsymbol{X}_{\text{cam},t}$ is calculated relative to the stereo rig and, since pose estimation is performed by the IMU, we first have to transform $\boldsymbol{X}_{\text{cam},t}$ to the body-fixed coordinate system of the IMU. This transformation is fixed, and supplied in the KITTI development kit. A schematic of the two coordinate systems is given in Figure 5.3(a).



(a) (b)

Figure 5.3: (a) Definition of the right-handed camera and IMU coordinate systems as viewed from above the vehicle. (b) Definition of the roll (ϕ), pitch (θ) and yaw (ψ) angles in the IMU/GPS coordinate system.

The transformation that maps a feature position in camera coordinates to the IMU coordinate system is given by

$$\begin{bmatrix} \boldsymbol{X}_{\text{imu},t} \\ 1 \end{bmatrix} = T^{-1}_{\text{imu}\to\text{velo}} T^{-1}_{\text{velo}\to\text{cam}} \begin{bmatrix} \boldsymbol{X}_{\text{cam},t} \\ 1 \end{bmatrix}, \tag{5.6}$$

where $T_{\text{imu}\to\text{velo}}$ and $T_{\text{velo}\to\text{cam}}$ are homogeneous coordinate transformations from the IMU to the range scanner, and from the range scanner to the stereo rig respectively. Since the transformation in (5.6) is linear, the resulting covariance matrix is given by

$$\Sigma_{\text{imu},t} = A\Sigma_{\text{cam},t}A^T, \tag{5.7}$$

where $A = T^{-1}_{\text{imu}\to\text{velo}} T^{-1}_{\text{velo}\to\text{cam}}$.

5.4 Feature positions in the global coordinate system

Next we require the transformation from vehicle coordinates to world coordinates. This is achieved by considering the measurements returned by the IMU. Before we calculate this transformation and specify exactly how the world coordinate system is defined, we require some preliminaries on the raw measurements returned by the vehicle's pose estimator.

Here we assume independence between raw measurements taken at different time steps and that the true values are normally distributed around these measurements. In the next two sections we discuss the raw measurements that describe the vehicle's position and orientation, following the notation of the IMU reference manual [60] as well as the KITTI development kit.

5.4.1 Raw position measurements

The differential GPS on the KITTI vehicle provides an absolute location estimate at each time step. This estimate is provided in the form of a global latitude, longitude and altitude above sea-level, where latitude and longitude are measured in degrees, and

altitude in metres. We represent the uncertainty in these measurements by the covariance matrix

$$\Sigma_t^{(\text{raw\_pos})} = \begin{bmatrix} 10^{-12} & 0 & 0 \\ 0 & 10^{-12} & 0 \\ 0 & 0 & 0.0004 \end{bmatrix}, \tag{5.8}$$

which corresponds to a standard deviation of 2 cm in altitude and roughly 7 cm in horizontal and vertical vehicle position (as viewed from above), and are of the same order as the estimated accuracy in the IMU reference manual [60]. Here we assume that the latitude, longitude and altitude measurements are uncorrelated, and that the error is normally distributed around a measured value.

5.4.2 Raw orientation measurements

The vehicle's orientation can be described using three angles: roll (ϕ), pitch (θ) and yaw (ψ), which are rotations about the axes shown in Figure 5.3(b). Here the axis definitions are in accordance with the KITTI set-up in Figure 5.1, where the positive x-axis points toward the front of the vehicle and the origin is at the centre of the IMU/GPS unit. The unit outputs these three angles in degrees at each time step, and we convert them to radians.

Following the IMU reference manual [60], we assume Gaussian noise with a standard deviation of 0.1° in each of these angles, so that the covariance matrix is

$$\Sigma_t^{(\text{raw\_orient})} = \left(\frac{0.1\pi}{180}\right)^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.9}$$

5.4.3 Global vehicle position

From the raw position and orientation measurements a distribution over the absolute vehicle position and orientation can be computed.

The altitude above sea-level estimate alt_t at time t is simply the z-coordinate of the vehicle, $z_{\text{veh},t}$, but the latitude and longitude require some processing before they can be viewed as x- and y-coordinates. We employ the well-known Mercator projection [73], which we state here for completeness. The surface of the earth is mapped to a cylinder such that latitudes remain horizontal, and the cylinder is cut vertically and unrolled. In this coordinate system the origin is at the meeting point of the central meridian and the equator, the x-axis along the equator and the scale in metres. The underlying assumption in this mapping is that the coordinate system is inertial, which is not entirely true because of the rotation of the earth, and also Cartesian, which it is not because of the curvature of the earth. However, for movement on a small part of the earth's surface, these coordinates are close to inertial and locally Cartesian. The equations to convert the raw latitude and

longitude measurements are given by

$$x_{\text{veh},t} = \frac{\pi e_r \text{lat}_t s(\text{lat}_1)}{180}, \tag{5.10}$$

$$y_{\text{veh},t} = e_r \, s(\text{lat}_1) \, \ln\left(\tan\left(90 + \text{long}_t\right) \frac{\pi}{360}\right), \tag{5.11}$$

where $e_r = 6,378,137$ is the average radius of the earth in metres and s is computed as

$$s(\text{lat}_1) = \cos\left(\frac{\text{lat}_1 \pi}{180}\right). \tag{5.12}$$

5.4.4 Global vehicle orientation

From the raw angle measurements ϕ, θ and ψ, a rotation matrix can be composed that rotates a vector in the IMU coordinate system to the absolute coordinate system. According to the KITTI development kit and the IMU reference manual [60], this rotation matrix is given by

$$R_t = R_{\psi,t} \, R_{\theta,t} \, R_{\phi,t}, \tag{5.13}$$

where

$$R_{\phi,t} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi_t) & -\sin(\phi_t) \\ 0 & \sin(\phi_t) & \cos(\phi_t) \end{bmatrix}, \qquad R_{\theta,t} = \begin{bmatrix} \cos(\theta_t) & 0 & \sin(\theta_t) \\ 0 & 1 & 0 \\ -\sin(\theta_t) & 0 & \cos(\theta_t) \end{bmatrix}, \tag{5.14}$$

and

$$R_{\psi,t} = \begin{bmatrix} \cos(\psi_t) & -\sin(\psi_t) & 0 \\ \sin(\psi_t) & \cos(\psi_t) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.15}$$

5.4.5 Global vehicle pose

Since the mapping that takes a point in the IMU coordinate system to global coordinates involves a rotation and translation, a pose matrix that combines the two can be computed as

$$\hat{P}_t = \begin{bmatrix} R_t & \boldsymbol{x}_{\text{veh,t}} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{5.16}$$

where

$$\boldsymbol{x}_{\text{veh,t}} = \left[x_{\text{veh},t}, \, y_{\text{veh},t}, \, z_{\text{veh},t} \right]^T \tag{5.17}$$

is the translation vector. When this transformation is applied to a vector in the IMU coordinate system, the resulting vector will be in the global coordinate system.

Following the KITTI development kit, we choose the origin of the global coordinate system to align with the first pose of the sequence, and compute all poses relative to the first. This means that the global vehicle pose in our world coordinate system is given by

$$P_t = \left(\hat{P}_1\right)^{-1} \hat{P}_t. \tag{5.18}$$

5.4.6 Vehicle pose distribution

Now that we have described how the global vehicle pose is obtained from the raw IMU measurements, we proceed to calculate the resulting vehicle pose distribution. If we consider the final pose matrix that represents the pose at time t in (5.18), we see that it depends on raw measurements at time t, as well as time $t = 1$. Also, the transformation should take a point in IMU coordinates $\boldsymbol{X}_{\text{imu},t}$ and transform it to world coordinates. Accordingly, we define the Gaussian random variable

$$\hat{\boldsymbol{x}}_t = \left[\, \text{lat}_1,\, \text{long}_1,\, \text{alt}_1,\, \phi_1,\, \theta_1,\, \psi_1,\, \text{lat}_t,\, \text{long}_t,\, \text{alt}_t,\, \phi_t,\, \theta_t,\, \psi_t,\, \boldsymbol{X}_{\text{imu},t}^T \,\right]^T, \tag{5.19}$$

where we combine the covariance matrices from (5.8), (5.9) and (5.7) into the final covariance matrix

$$\hat{\Sigma}_t = \begin{bmatrix} \Sigma_1^{(\text{raw\_pos})} & & & & \cdots & 0 \\ & \Sigma_1^{(\text{raw\_orient})} & & & & \vdots \\ & & \Sigma_t^{(\text{raw\_pos})} & & & \\ \vdots & & & \Sigma_t^{(\text{raw\_orient})} & & \\ 0 & \cdots & & & & \Sigma_{\text{imu},t} \end{bmatrix}. \tag{5.20}$$

In Algorithm 1 we define the transformation function $g(\boldsymbol{x})$ that takes the raw pose measurements and the point in IMU coordinates (all packed into a vector $\boldsymbol{x}$ as in (5.19)) to the world coordinate system.

Algorithm 1 Transformation function to map point in IMU coordinates to world coordinates. $\boldsymbol{x}[i]$ denotes the i-th entry of $\boldsymbol{x}$.

function $g(\boldsymbol{x})$
  compute $s = \cos\left(\frac{\pi \boldsymbol{x}[1]}{180}\right)$
  compute $\boldsymbol{t}_1 = \begin{bmatrix} \frac{\pi\, e_r \boldsymbol{x}[1]}{180} \\ e_r\, s\, \ln\left(\tan(90 + \boldsymbol{x}[2])\frac{\pi}{360}\right) \\ \boldsymbol{x}[3] \end{bmatrix}$

  compute $\boldsymbol{t}_t = \begin{bmatrix} \frac{\pi\, e_r \boldsymbol{x}[7]}{180} \\ e_r\, s\, \ln\left(\tan(90 + \boldsymbol{x}[8])\frac{\pi}{360}\right) \\ \boldsymbol{x}[9] \end{bmatrix}$
  compute R_1 using $\boldsymbol{x}[4], \boldsymbol{x}[5], \boldsymbol{x}[6]$ according to (5.13)
  compute R_t using $\boldsymbol{x}[10], \boldsymbol{x}[11], \boldsymbol{x}[12]$ according to (5.13)
  compute $\hat{P}_1 = \begin{bmatrix} R_1 & \boldsymbol{t}_1 \\ \boldsymbol{0}^T & 1 \end{bmatrix}$
  compute $\hat{P}_t = \begin{bmatrix} R_t & \boldsymbol{t}_t \\ \boldsymbol{0}^T & 1 \end{bmatrix}$
  compute $P = \hat{P}_1^{-1} \hat{P}_t$
  let $\boldsymbol{X} = [\, \boldsymbol{x}[13], \boldsymbol{x}[14], \boldsymbol{x}[15], 1 \,]^T$
  compute $\boldsymbol{y} = P\boldsymbol{X}$
  return $[\, \boldsymbol{y}[1], \boldsymbol{y}[2], \boldsymbol{y}[3] \,]^T$

Since the function $g(\boldsymbol{x})$ is nonlinear, the transformed distribution is not Gaussian. It can, however, be approximated by a Gaussian distribution. One option is use a Taylor expansion, as in Section 5.2.2. However, because of the specific nonlinearities in the function, we found this method to be inadequate. The unscented transform (see Appendix B), which is an alternative linearization method, samples points from the original distribution and fits a Gaussian distribution over the transformed points. We apply this transform with $\kappa = 0.5$ and $\alpha = 0.55$, in order to obtain the distribution over the global 3D feature position in the world coordinate system as

$$p(\boldsymbol{X_t}) \approx \mathcal{N}\left(\boldsymbol{X_t} \,|\boldsymbol{\mu_t}, \Sigma_t\right). \tag{5.21}$$

Since both $\boldsymbol{\mu}_t$ and Σ_t are known at a particular time, and describe a distribution over the global 3D feature position, we can proceed to calculate the measurements required as observations for our PGM.

5.5 Velocity measurement

For a velocity measurement we require data from two time steps. Suppose a particular feature i has world coordinates $\boldsymbol{X}_{i,t-1}$ at time $t-1$ and $\boldsymbol{X}_{i,t}$ one time step later. The velocity measurement $\hat{\mathbf{v}}_{i,t}$ is simply calculated as

$$\hat{\mathbf{v}}_{i,t} = \boldsymbol{X}_{i,t} - \boldsymbol{X}_{i,t-1}. \tag{5.22}$$

Since we subtract two consecutive feature positions, the velocity is given in metres per time step. This measurement can be converted to metres per second but, if the same calculation is used throughout, this step can be omitted.

If $\boldsymbol{X}_{i,t}$ and $\boldsymbol{X}_{i,t-1}$ have covariance matrices $\Sigma_{i,t}$ and $\Sigma_{i,t-1}$ respectively, and since the velocity measurement is a linear transformation, the velocity measurement covariance matrix (occuring in (3.1)) is given by

$$V_{i,t} = \Sigma_{i,t} + \Sigma_{i,t-1}. \tag{5.23}$$

Here we must assume that consecutive position measurements $\boldsymbol{X}_{i,t}$ and $\boldsymbol{X}_{i,t-1}$ of the same feature are independent.

5.6 Change in relative distance measurement

The change in relative distance measurement is calculated by taking the distance between two points at time $t-1$ and subtracting it from the distance between the same two points at time t. Since we are only interested in the change in relative distance, these coordinates need not be in the global coordinate system when subtracted from each other.

If the measured 3D positions of features i and j at time $t-1$ relative to the stereo cameras are $\boldsymbol{X}^{(i)}_{\text{cam},t-1}$ and $\boldsymbol{X}^{(j)}_{\text{cam},t-1}$, as calculated in (5.2), the distance between these two features

at time $t-1$ is given by

$$d_{ij,t-1} = \left\| \boldsymbol{X}^{(i)}_{\text{cam},t-1} - \boldsymbol{X}^{(j)}_{\text{cam},t-1} \right\|, \tag{5.24}$$

where $\|\cdot\|$ denotes the Euclidean norm. Similarly, at time t we have

$$d_{ij,t} = \left\| \boldsymbol{X}^{(i)}_{\text{cam},t} - \boldsymbol{X}^{(j)}_{\text{cam},t} \right\|, \tag{5.25}$$

so that the change in relative distance measurement is

$$\hat{r}_{ij,t} = d_{ij,t} - d_{ij,t-1}. \tag{5.26}$$

If we define a new Gaussian random variable $\boldsymbol{d}_t$ as

$$\boldsymbol{d}_t = \left[\boldsymbol{X}^{(i)}_{\text{cam},t-1}, \boldsymbol{X}^{(j)}_{\text{cam},t-1}, \boldsymbol{X}^{(i)}_{\text{cam},t}, \boldsymbol{X}^{(j)}_{\text{cam},t} \right]^T, \tag{5.27}$$

with accompanying covariance matrix

$$\Sigma_{d,t} = \begin{bmatrix} \Sigma^{(i)}_{\text{cam},t-1} & & \cdots & 0 \\ & \Sigma^{(j)}_{\text{cam}t-1} & & \vdots \\ \vdots & & \Sigma^{(i)}_{\text{cam},t} & \\ 0 & \cdots & & \Sigma^{(j)}_{\text{cam},t} \end{bmatrix}, \tag{5.28}$$

we can define the transformation function $h(\boldsymbol{d})$ in Algorithm 2 that provides the change in relative distance from the vector $\boldsymbol{d}$.

Algorithm 2 Transformation function to calculate the change in relative distance measurement from the vector $\boldsymbol{d}_t$. $\boldsymbol{d}[i]$ denotes the i-th entry of $\boldsymbol{d}$.

1: **function** $h(\boldsymbol{d})$
2: compute r_1 using $\boldsymbol{d}[1], \boldsymbol{d}[2], \boldsymbol{d}[3]$ and $\boldsymbol{d}[4], \boldsymbol{d}[5], \boldsymbol{d}[6]$ according to (5.24)
3: compute r_2 using $\boldsymbol{d}[7], \boldsymbol{d}[8], \boldsymbol{d}[9]$ and $\boldsymbol{d}[10], \boldsymbol{d}[11], \boldsymbol{d}[12]$ according to (5.25)
4: compute $\hat{r} = r_2 - r_1$
return $\hat{r}$

The function $h(\boldsymbol{d})$ is nonlinear in $\boldsymbol{d}$, which means that the distribution over the change in relative distance measurement can only be approximated by a Gaussian distribution. Again, we apply the unscented transform with $\kappa = 0.5$ and $\alpha = 0.55$, in order to obtain

$$p(\hat{r}_{ij,t} \,|r_{ij,t}) \approx \mathcal{N}(\hat{r}_{ij,t} \,|r_{ij,t}, R_{ij,t}). \tag{5.29}$$

To summarize, there are two types of measurements required for inference on our PGM at each time step: velocity measurements and change in relative distance measurements. For the first we require a global 3D feature position distribution which, in the case of the KITTI datasets, can be computed as described in Sections 5.2 to 5.4. The second type of measurement can be calculated using a body-fixed 3D feature position distribution, such as the one described in Section 5.2.2. Once the required distributions are obtained, the two types of measurements can be computed as described in Sections 5.5 and 5.6.

Chapter 6

Experimental results

In this chapter we describe results from a number of experiments on several datasets from the KITTI benchmark suite [23] which cover rural, suburban and city traffic environments. Details on the specific sets we chose to consider and motivation behind these choices are provided in Section 6.1, and in Section 6.2 we state some preliminaries.

We compare the performance of our PGM approach to triTrack [43] which, to our knowledge, is the most recent publicly available feature-based motion segmentation system. For the purposes of this comparison we use their feature detector, Libviso [24], although our approach is compatible with any good feature detector. We first compare the overall performance of both methods on all the datasets (Section 6.3). After that we investigate the ability of our method to perform motion segmentation over time (Section 6.4) as well as to handle semi-static objects (Section 6.5). Finally we provide a qualitative analysis of the performance on our method and that of triTrack (Section 6.6).

6.1 Datasets

The KITTI benchmark suite contains about 50 datasets in total in their city, residential and road categories. Every dataset typically contains a few hundred stereo frames, and within each typical frame around 1,000 features are detected. We decided to manually annotate the features for ground truth in our quantitative evaluations, and label each one as either "static" ($x_{i,t} = 0$, $m_i = 0$), "semi-static and currently stationary" ($x_{i,t} = 0$, $m_i = 1$), "semi-static and currently moving" ($x_{i,t} = 1$, $m_i = 1$), or "unclear". Annotating all the features in all the datasets (which would be about 5 million feature measurements in total) was not feasible, and we resorted to choosing sets representative of the type of scenarios that a mobile robot may encounter in practice. Each dataset is chosen for a specific reason and tests the algorithm's performance in a specific scenario.

Every set includes pose estimates from an IMU as well as rectified stereo images captured at 10 Hz. We lowered the frame rate, by processing every third frame, as a compromise between having more information (from a high frame rate) and more movement between frames (from a lower frame rate).

Recall that only features observed at time t are included in the PGM at time t, which means that the PGM does not increase in complexity over time. Also, a particular dynamic feature is typically not observed for more than 7 or 8 frames. Since every feature had to be manually annotated, it was decided to limit the length of each set to about 15 frames (or time steps). This means that we have about 6,700 unique features per sequence (out of approximately 15,000 feature measurements in total per dataset).

The first three frames of every chosen dataset are shown in Figure 6.1. In the first dataset (`2011_09_26_drive_0013`) the KITTI vehicle is driving at a medium speed along a straight road, with other vehicles driving at a similar speed in the same direction. Tests on this dataset will indicate how well our method performs while following dynamic objects.

(a) `2011_09_26_drive_0013`

(b) `2011_09_26_drive_0005`

(c) `2011_09_26_drive_0106`

(d) `2011_09_26_drive_0057`

(e) `2011_09_26_drive_0014`

(f) `2011_09_26_drive_0011`

Figure 6.1: Sample frames from the six KITTI datasets chosen for testing.

The second dataset (`2011_09_26_drive_0005`) contains a scene where the KITTI vehicle is driving behind a cyclist and a minivan, while turning a corner. This dataset will assess how well our method identifies different types of dynamic objects while turning, when pose estimation may be more inaccurate.

In the third dataset (`2011_09_26_drive_0106`) the KITTI vehicle is driving along a straight road with slow-moving approaching vehicles, and this set will assess how well our method handles approaching traffic and slow-moving objects.

The fourth dataset (`2011_09_26_drive_0057`) contains a semi-static object. In this sequence the KITTI vehicle stops behind another vehicle at a traffic light. After a while the traffic light turns green and the vehicle in front starts to drive.

In the first few frames of the fifth dataset (`2011_09_26_drive_0014`), the KITTI vehicle is stationary. After a while it makes a right turn and encounters fast-moving approaching vehicles.

The sixth dataset (`2011_09_26_drive_0011`) contains frames where the KITTI vehicle approaches an intersection, while other vehicles are moving across its field of view. Theoretically, this case should be the simplest for dynamic object detection, since the apparent movement in the image is large.

By testing our algorithm on these six datasets we cover a wide variety of situations: slow moving objects, fast moving objects, different types of dynamic objects, objects moving in the same direction, objects moving in the opposite direction and objects moving in a direction perpendicular to the KITTI vehicle.

6.2 Preliminaries

Our method requires values for a number of hyperparameters. In this work we determined those values through manual tuning while visualizing the results on a dataset different from the six chosen for further tests. This tuning process was purposefully cursory and, once set, the parameters were kept constant in all our experiments.

The three parameters required in Tables 3.1 and 3.5, namely the probabilities $a_{x|m}$, a_s and a_d, are all set to be 0.9. From a logical standpoint these values should be high. Through some experimentation we found that the performance of our method is not sensitive to these values, and remains high with values as low as 0.7.

Appropriate values for the covariance matrices C_v and C_r were determined as

$$C_v = \begin{bmatrix} 0.5^2 & 0 & 0 \\ 0 & 0.5^2 & 0 \\ 0 & 0 & 0.5^2 \end{bmatrix}, \quad C_r = 1^2. \tag{6.1}$$

We also found our method to be fairly insensitive to the values in these matrices.

The unscented transform parameters $\alpha = 0.55$ and $\kappa = 0.5$ used in Chapter 5 were determined by visualizing error ellipses around 3D feature positions and choosing values that seemed to reflect the fact that features close to the vehicle should have a standard deviation in the order of centimetres. Of all the required hyperparameters, our method seems to be most sensitive to α. However, by the cursory manner in which we chose a value for α it is unlikely that the optimal value was chosen, which does indicate some insensitivity to this parameter.

If suitable data is available, the hyperparameters may also be learned through expectation maximization or a Markov chain Monte Carlo method.

As mentioned, features are extracted from the stereo images and matched over time using the triTrack dependency Libviso, where features are detected by a corner and blob filter and matched using visual odometry. No further outlier rejection is performed after this step. Our method is of course not limited to the Libviso features, and may be used in conjunction with any good feature extractor and matching algorithm.

6.3 Overall performance

The motion segmentation result from triTrack is a labelling of all features as either "static", "dynamic" or "unknown". With our manually annotated ground truth feature labels for the datasets listed in Section 6.1, we can compare our method's motion segmentation capability to triTrack's in a quantitative manner. Following the steps set out in Section 4.4, we can infer the marginal distribution over $x_{i,t}$ for each feature after message passing, apply a threshold and compare it to the annotated label of that feature. Processing a single stereo frame takes about 5 to 6 seconds, depending on the number of features detected in the image. The feature extraction and matching typically takes less than a second, while the bulk of the computation time is dedicated to loopy belief propagation.

Results measured over the six datasets are shown in Table 6.3, where the percentage of each error type is calculated for every frame and averaged over the entire dataset. For our method we applied a threshold of 0.5 to the obtained feature state probabilities to arrive at classifications. In the table we show both the type I and type II errors made by each method. Here a type I error occurs when a stationary feature is classified as dynamic, and a type II error occurs when a dynamic feature is classified as stationary. We express these two errors as percentages of all features classified as dynamic and of all features classified as static, respectively. For comparisons with triTrack, the label "semi-static and currently moving" is grouped with "dynamic", and "semi-static and currently stationary" with "static", since their method does not include a label for semi-static features.

The results indicate that our method significantly outperforms triTrack. The weaker performance of our method regarding type II error for the third dataset can be explained by the apparent motion of some features being too small compared to the associated measurement uncertainties (this set contains slow-moving approaching vehicles). We do note that the type II error made by our approach is comparable to that of triTrack.

	triTrack [43] error %		our PGM approach error %	
dataset	type I	type II	type I	type II
2011_09_26_drive_0013	7.2	17.7	**2.3**	**1.2**
2011_09_26_drive_0005	11.9	23.5	**7.3**	**3.9**
2011_09_26_drive_0106	6.7	**25.4**	**1.3**	29.1
2011_09_26_drive_0057	6.6	18.9	**3.4**	**7.7**
2011_09_26_drive_0014	12.9	22.9	**7.6**	**3.5**
2011_09_26_drive_0011	3.7	5.9	**3.4**	**4.0**
average	8.2	17.8	**4.2**	**8.2**

Table 6.1: Motion segmentation error percentages of triTrack and our method, measured over six KITTI datasets against manually annotated ground truth.

If we consider the fourth dataset, which contains the semi-static object, we see that the type II error of our method is significantly lower than that of triTrack. This may be attributed to our explicit handling of semi-static objects, which enables our method to alter its belief quickly if a stationary feature starts to move.

6.4 Motion segmentation accuracy over time

We now proceed to investigate certain aspects of our method's performance by analysing specific sections of the datasets. We start by assessing the motion segmentation performance for features that are observed over a number of frames.

For this assessment we take the first dataset (2011_09_26_drive_0013), and consider only the features that are observed for at least four consecutive time steps. Results are plotted as receiver operating characteristic (ROC) curves in Figure 6.2, where "dynamic" is taken as the positive label and "static" as the negative label.

The advantage of explicitly modelling time dependency between feature states and velocities is quite clear, since the motion segmentation accuracy tends to improve over time. This means that, the longer we are able to observe a feature, the clearer it becomes whether the feature is static or dynamic. Similar experiments on the other five datasets support this tendency.

6.5 Handling of semi-static objects

Next we consider the ability of our PGM to deal with semi-static objects for which the states of features may change over time. We consider a particular subset of frames from the dataset 2001_09_26_drive_0057, where a vehicle directly in front of the KITTI vehicle is stationary for about four frames before it starts moving forward.

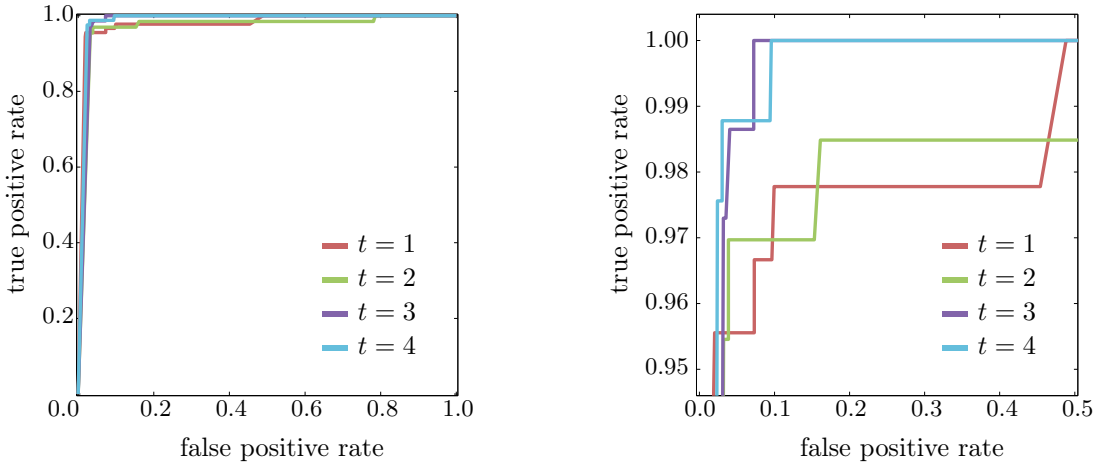


Figure 6.2: ROC curves showing the motion segmentation accuracy for a group of features over time, and a zoom-in on the right (we take "dynamic" to be the positive label).

Figure 6.3(a) shows a number of ROC curves that indicate how accurately our system estimates the current states of features on the vehicle, from the moment that it starts to move. We see weak performance at time $t = 5$ (when the vehicle starts to move), because the PGM has strong reason to believe that most of the features are stationary. However, as more observations become available, the prior belief is corrected and performance increases quickly and dramatically with each additional time step, until near perfect classification is reached at $t = 9$. Our method is able to do this because of the m-variables that make provision specifically for semi-static features. Snapshots of the feature state probabilities are depicted in Figure 6.4, for the same time steps shown in Figure 6.3.

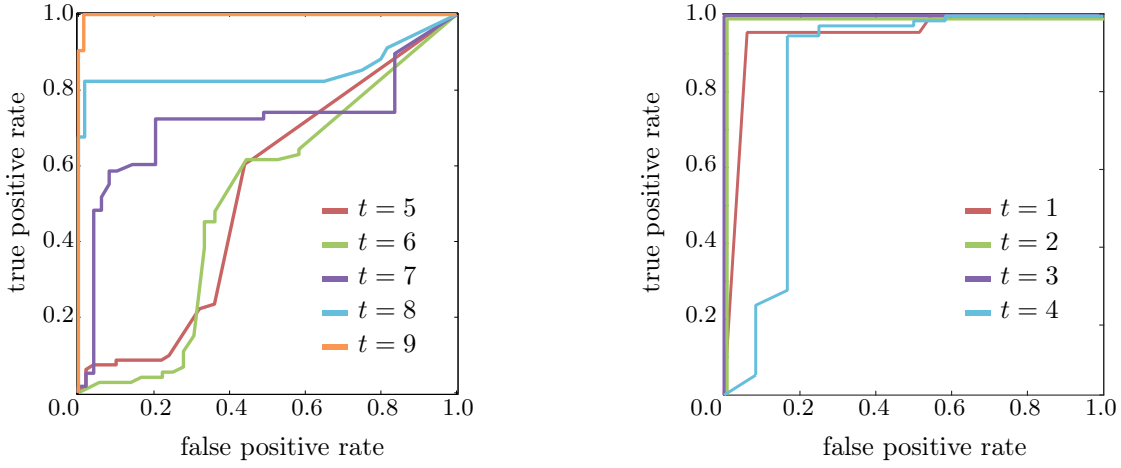


Figure 6.3: (a) ROC curves showing the motion segmentation accuracy for a group of features on a semi-static object over time (here the object was stationary for $t < 5$ and dynamic for $t \geq 5$). (b) ROC curves showing the motion segmentation accuracy for a group of features on another semi-static object over time (here the object was dynamic for $t \leq 4$ and static for $t > 4$).

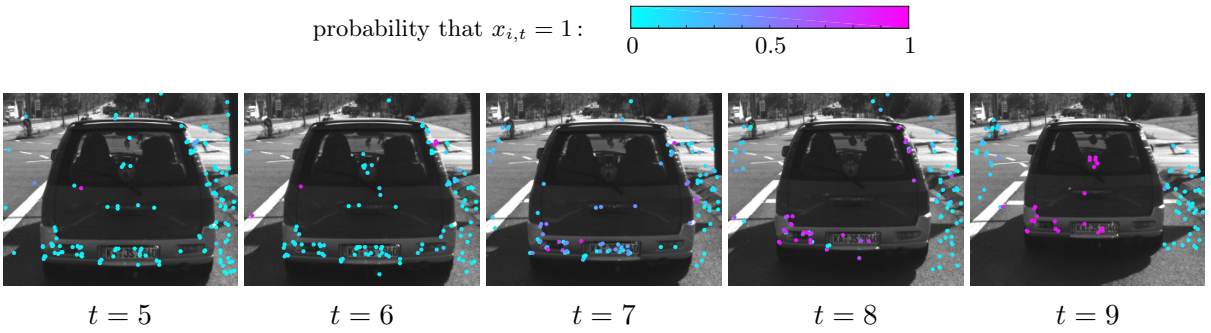


Figure 6.4: A depiction of the probabilities that features are dynamic, as returned by our PGM approach, for the same semi-static test case as in Figure 6.3(a).

Figure 6.3(b) shows ROC curves for a scenario involving the opposite case, where a dynamic object moves for four frames and then becomes stationary. We see that classification accuracy initially increases, and then decreases slightly. Unfortunately ROC curves cannot be drawn for $t > 4$, because all features are stationary and thus there are no positives. However, the PGM also classifies all features as stationary at $t = 5$ (after applying a threshold of 0.5 to the feature state probabilities).

6.6 Qualitative analysis

We now proceed with a qualitative assessment of the performance of our method in comparison to that of triTrack. In this section we show snapshots to illustrate certain behaviour of these two methods.

6.6.1 Motion segmentation

We first show results from the KITTI dataset `2011_09_26_drive_0005`, where the KITTI vehicle is taking a turn while observing another vehicle and a cyclist. In Figure 6.5 we show a sequence of motion segmentation results from both triTrack and our approach.

TriTrack provides a hard classification of the features at every time step, as indicated in the figure, and we see that many static features are misclassified as dynamic. This misclassification may be a result of distant measurements from stereo vision carrying significantly larger uncertainty.

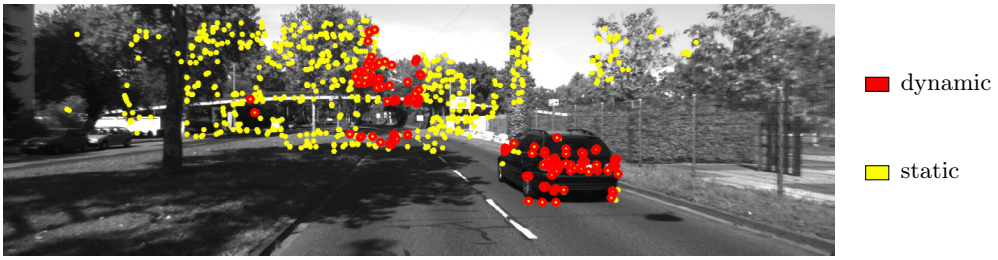
For our approach we see that features on the dynamic objects are given high probabilities of being dynamic, while the probabilities of the rest of the features vary from 0 to 0.5. This indicates that the model is quite certain about which features are currently dynamic, while it is less certain whether or not stationary features are in fact stationary. Apart from a few features in the second time step, our method appears to label the features correctly as static or dynamic.

Figure 6.5: A sequence of motion segmentation results over time for KITTI dataset `2011_09_26_drive_0005`, from triTrack on the left and our approach on the right.

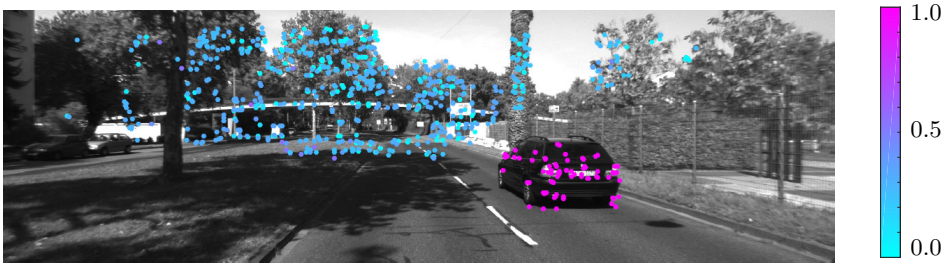
This example illuminates the benefits of including both pose and measurement uncertainties when inferring feature states. Keep in mind that the robot is moving through the environment as frames are captured, so most (if not all) of the features appear to be moving from the robot's point of view. Static features measured with large uncertainty may easily be misclassified as dynamic if those uncertainties are not properly included in the model.

In Figure 6.6(a) and (b) we show snapshots of the motion segmentation results obtained on dataset `2011_09_26_drive_0013` with triTrack, and the feature state probabilities obtained with our PGM approach. Similar to the previous example, triTrack misclassifies a handful of static features while our approach appears to be very successful in separating the dynamic features from static ones.

Next we show an example where both methods do not perform that well. Figure 6.7 shows a snapshot of results from dataset `2011_09_26_drive_0106`, where cars in the left lane (on the left side of the frame) are all slowly moving towards the vehicle. Our method fails to detect the motion of the white minivan as well as a number of features on the car behind it, because those objects are moving very slowly. It seems as if triTrack performs slightly better, although a few features on the car are incorrectly labelled as static.

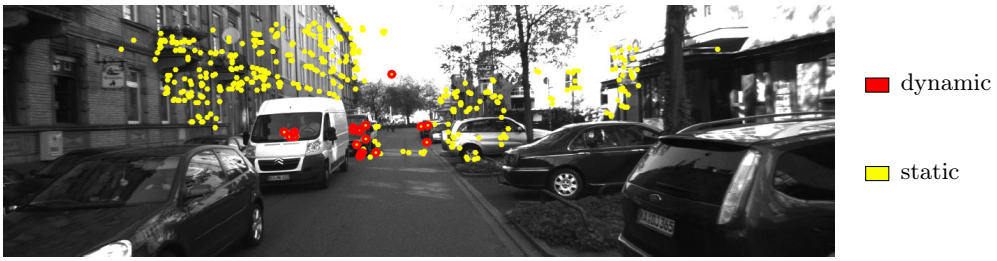


(a)



(b)

Figure 6.6: Example motion segmentation results from (a) triTrack and (b) our PGM approach on KITTI dataset `2011_09_26_drive_0013`.



(a)

(b)

Figure 6.7: Example motion segmentation results from (a) triTrack and (b) our PGM approach on KITTI dataset `2011_09_26_drive_0106`.

6.6.2 Object segmentation

We now proceed to analyse the object segmentation performance.

The triTrack method first clusters moving points into objects, which are then associated with tracklets. If a tracklet is observed for a sufficient period of time, the associated object is identified for tracking and is represented by a bounding box in the image. Our segmentation is obtained by applying a threshold of 0.6 to the posterior distributions over all s-variables. That is to say, if the posterior probability that $s_{ij} = 1$ is greater than 0.6, features i and j are taken to be part of the same object. Separate objects are then identified as groups of connected features.

In Figure 6.8 we show a sequence of snapshots from dataset `2011_09_26_drive_0005` and results from both triTrack and our approach. Once triTrack has identified an object for tracking, it is displayed with a number and a bounding box, as can be seen for time steps $t = 5,\ 6,\ 7$. We see that for the first four time steps, no objects were identified for tracking, and the cyclist is never detected in these seven time steps.

For our method, we see that an incorrect segmentation occurs at time $t = 1$, where the minivan and the cyclist have approximately the same instantaneous velocities, as well as at time $t = 2$, where three stationary points are incorrectly grouped as a dynamic object. However, since we introduce a time-dependency between variables into our PGM, those errors are eliminated in subsequent time steps.

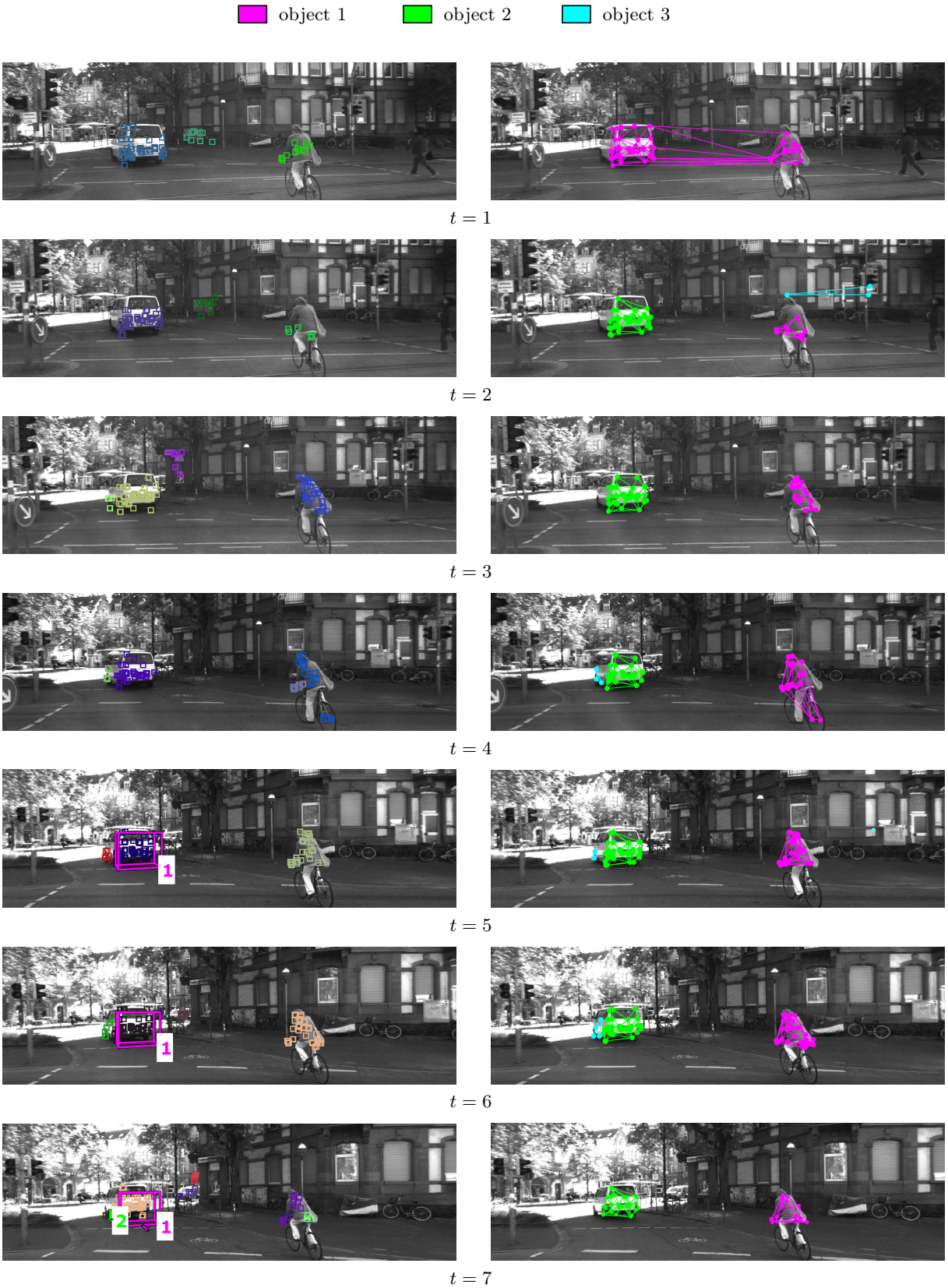
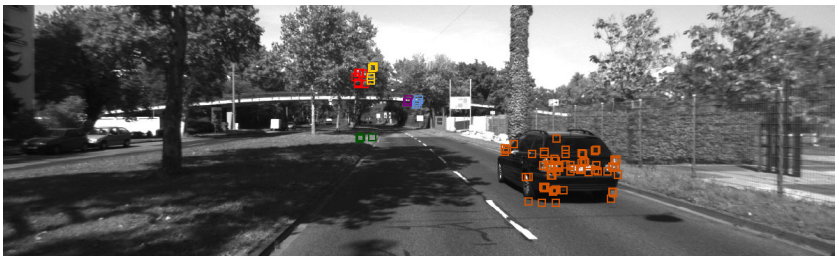


Figure 6.8: A sequence of object segmentation results over time for KITTI dataset `2011_09_26_drive_0005`, from triTrack on the left and our approach on the right.

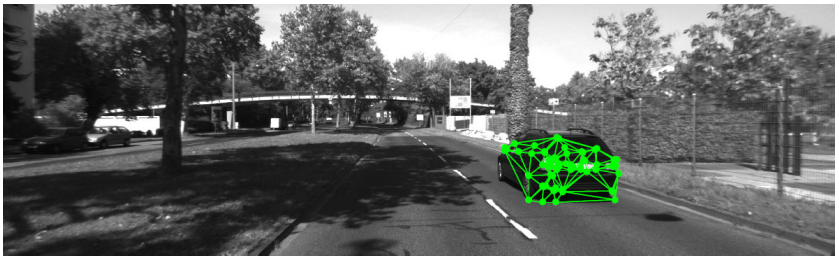
In time steps $t = 5$ and $t = 6$ we see that the minivan has been over-segmented by both methods (and also by our method at $t = 4$). This may be due to the fact that the wheels appear to be moving at a velocity different from the rest of the minivan.

A further object segmentation result is shown in Figure 6.9. In this example the dynamic object is travelling at roughly the same speed as the KITTI vehicle. At the point in time shown, the vehicle has been in view for at least seven frames yet triTrack fails to identify it for tracking. Our method on the other hand correctly identifies the vehicle as the only moving object in view.

From this evaluation we can conclude that triTrack seems to require quite a number of time steps to identify objects for tracking, and in some cases misses them completely. Our method is able to identify objects quickly, though it may produce an over-segmentation from time to time.



(a)



(b)

Figure 6.9: Example object segmentation results from (a) triTrack and (b) our PGM approach on KITTI dataset `2011_09_26_drive_0013`.

The results in this chapter show that our method frequently outperforms triTrack, which is a state-of-the art feature-based method for dynamic object detection. The results also show that the motion segmentation from our method typically improves over time and is able to handle semi-static objects effectively. The results further demonstrate that our PGM can struggle to segment the features correctly in the presence of slow moving objects, and is sometimes prone to over-segment.

Chapter 7

Conclusion

In this work we set out to develop a technique that can detect dynamic objects for a mobile robot navigating through a dynamic environment. This is a challenging problem since measurements of the environment returned by on-board sensors may indicate that everything in the environment is moving, while much of it may be caused by the robot's own motion.

A literature review revealed that almost none of the currently available methods incorporate pose uncertainty to compensate for this, and rather assume an exact known pose even though pose estimates and environment measurements may carry significant uncertainty. Furthermore, most methods assume that the robot's ego-motion can be retrieved from the environment measurements and, in doing so, that the environment is predominantly static.

The majority of approaches operate on the image plane, where information is lost during the projection from 3D to 2D. Furthermore, most methods consider all the pixels in an image, which leads to high dimensionality and computational cost. Alternatively, features can be detected in the images, which enables fast tracking because of a lowered complexity. Surprisingly, there exist few feature-based motion segmentation algorithms.

Moreover, despite the ability of PGMs to model a problem with many latent variables whose values can be inferred after observing related variables, PGMs have not been a popular approach to the motion and object segmentation problems. This may be partly due to the fact that modelling the problem is challenging, with no guarantees of success. Inference in large networks can also be computationally complex, which creates a trade-off between realistic problem modelling and tractable inference.

7.1 Proposed solution

In order to solve the dynamic object detection problem, we proposed a novel feature-based method for probabilistic motion segmentation that achieves two main goals. Firstly, observed features are classified as static or dynamic and secondly, dynamic features are grouped into separate objects. We presented a novel application of probabilistic graphical

models that can, accordingly, be divided into two interacting components. The first is the motion detection component, which is developed from first principles and offers a new way of describing the problem.

Within the first component we assign a time-dependent binary state variable to every feature that indicates whether or not it is currently dynamic (current state variable). Furthermore, we accommodate for semi-static objects by connecting the current state variable to an additional binary discrete variable which indicates whether or not the feature is part of an object that is capable of movement. This allows for stationary features to become dynamic and vice versa, without losing track of which features should be monitored for collision avoidance.

The second component builds on an idea from Shental et al. [68], adapted to interact with the first component as well as to fit into our PGM framework. Within this component we connect each feature pair (that is connected in a Delaunay triangulation) to a binary variable that indicates whether or not those two features are part of the same object.

At every time step we assume access to a belief distribution over the robot's pose, obtained from some type of localization estimator. We also assume a set of known feature correspondences between the current time step and the previous one, as well as a measurement model over the individual features in order to compute feature velocities. We pre-process the feature image coordinates into observations of a feature's velocity and we also calculate the change in relative distance between two features. The first observation is a direct indication of whether the feature is currently static or dynamic, while the second is an indication of whether or not two features belong to the same object. Both these types of observations are connected to latent random variables that represent their true respective values (i.e. the true feature velocity and the true change in relative distance).

By modelling the problem in this way the PGM contains both continuous and discrete random variables. Tractable inference in such PGMs is challenging and, in some cases, even impossible. In our case, however, a few simplifying assumptions allow for closed-form computation of the messages sent from continuous variables to discrete ones, thereby decoupling the continuous and discrete variables. Loopy belief propagation can then be performed on the discrete variables only, which enables tractable inference. After calibration, messages are propagated back to the continuous variables in order to update their beliefs.

7.2 Performance analysis

We tested our PGM approach on several datasets from the KITTI benchmark suite [23]. The chosen datasets cover urban and residential environments, scenarios where the dynamic objects are approaching, departing and crossing the road, scenarios where the KITTI vehicle drives along a straight road and drives around a corner. It also contains vehicles, cyclists and semi-static objects. We compared our method to an existing state-of-the-art feature-based method called triTrack [43].

Experimental results indicate that our PGM approach performs well overall, and outperforms triTrack significantly on type I and II error percentages on five out of the six datasets. It also does not perform significantly worse on the sixth set. An investigation of the performance for features seen over multiple time steps revealed that accuracy increases as more observations become available.

Our PGM is also able to handle semi-static objects in the sense that, if a stationary object starts to move, the PGM is able to identify the transition within a small number of frames. Similarly for dynamic objects that become stationary, the PGM is able to classify such features as stationary within one or two frames, even though the object may have been dynamic for longer than that.

A qualitative comparison with triTrack revealed that the latter is prone to label distant stationary features as dynamic, possibly due to the fact that the measurements of such features carry significant uncertainty or because pose uncertainty can affect distant features more severely. Since our approach accommodates for measurement and pose uncertainty, such errors are markedly reduced. On the other hand, we also identified scenarios in which both methods do not perform that well. When the KITTI vehicle encounters slow moving objects, their movement is overwhelmed by the measurement uncertainty and our method fails to detect them. TriTrack suffers from a similar problem, but seemingly to a lesser degree.

Finally, we saw that our method is able to detect the dynamic objects in the environment correctly after about three frames, while triTrack struggles to identify some of these objects after as many as seven frames. Both methods are prone to over-segmentation. For our method this can be explained by the rigid object assumption.

7.3 Future work

Possible future work may include an investigation into further similarity metrics, such as colour, shape or other existing similarities, to improve object segmentation. Another route is to investigate a relaxation of the rigid body assumption, which may solve the over-segmentation problem.

In our current solution there are a few parameters that need to be fixed, namely the covariances C_v, C_b and C_r as well as the probabilities $a_{x|m}$, a_s and a_d. Learning these values in a training phase may increase the generality of the technique. Taking this idea one step further may be to learn the entire structure of the PGM.

Other ways in which to enable even faster inference presents another research direction, since our current implementation is not quite fast enough for real-time applications (processing a stereo frame typically takes about 5 seconds). Additionally, ways to handle slow-moving objects can be researched, and the tracking component of the PGM can be refined to relax the known correspondence assumption. With all these components in place, dynamic object trajectory prediction may become a promising research topic.

7.4 Contributions and significance

By modelling the problem as a PGM, we developed a novel application for PGMs in a research area where it has not been particularly popular. An important advantage of PGMs is that they can be extended fairly effortlessly, in the sense that any additional information can simply be added to the graph structure. The output of the PGM is probabilistic, which means that it provides a soft classification, and may be more advantageous than hard classifications for subsequent modules in the autonomous navigation process that also reason under uncertainty.

The PGM design process of specifying random variables, relationships and distributions entailed a number of decisions, which relied on an in-depth understanding of the problem while trading off assumptions and tractability with accuracy. Some of these assumptions include independence between the change in relative distance and velocity measurements, Gaussian distributions over all continuous random variables, and that variables at time t cannot propagate messages back to variables at time $t-1$. These assumptions enabled us to develop an incremental inference algorithm in which we decouple the PGM at time $t-1$ from the one at time t, and imply that the number of variables in the PGM does not grow over time. As a consequence, inference becomes tractable, even though we have both continuous and discrete random variables in the PGM.

On the other hand, we avoided some common assumptions typically made to solve the motion segmentation problem. Our method does not require the exact pose of the robot to be known at every time step. In fact, we accommodate for both pose and measurement uncertainty in our model. By including a distribution over the pose in the pre-processing of the velocity measurements, we also avoid the requirement that the scene must be predominantly stationary. This means that the system is able to function in highly dynamic environments without the need for an expensive localization sensor.

For the object segmentation component of the PGM we designed the similarity measurements such that they are unaffected by pose uncertainty. In this component we have made the assumption that dynamic objects are rigid, which can lead to over-segmentation. However, the chosen PGM structure does not require the number of dynamic object beforehand, contrary to many other clustering algorithms.

Our PGM has also been specifically designed to also handle semi-static objects. Consequently, if the robot is in an environment where dynamic objects may stop and start moving again, such as busy urban intersections, the robot will not lose track of the objects capable of movement and be able to plan its own set of actions accordingly.

Even though we have used the example of urban environments, our method is not limited to that. By not training on environment-specific data, our algorithm may, with a few simple adaptations, be amenable to underwater or aerial vehicles. Also, even though the method was tested using stereo cameras, it can be employed using different sensors, as long as 3D feature positions can be estimated at every time step, since we have avoided operating on the image plane.

Overall we have avoided some common limiting assumptions and, even though we introduced a few of our own assumptions, our method outperforms the state-of-the-art. We believe that modelling the dynamic object detection problem as a PGM and following the design choices made in this work, significant progress has been made towards the goal of having autonomous vehicles assist humans in everyday life.

Bibliography

[1] M. Agrawal, K. Konolige, L. Iocchi, *Real-time detection of independent motion using stereo*, IEEE Workshop on Applications of Computer Vision, pp. 207–214, 2005.

[2] A. Argyros, *Visual detection of independent 3D motion by a moving observer*, PhD thesis, University of Crete, Greece, 1996.

[3] S. Avidan, *Ensemble tracking*, IEEE Transactions on Pattern Analyis and Machine Intelligence, vol. 29, no. 2, pp. 261–271, 2007.

[4] D. Barber, *Bayesian Reasoning and Machine Learning*, Cambridge University Press, 2012.

[5] H. Bay, T. Tuytelaars, L. van Gool, *Speeded-up robust features (SURF)*, Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346–359, 2008.

[6] A. Behrad, A. Shahrokni, S. Motamedi, *A robust vision-based moving target detection and tracking system*, Image and Vision Computing Conference, pp. 627–630, 2001.

[7] B. Berg, A. Billoire, *Markov Chain Monte Carlo Simulations*, John Wiley & Sons, 2008.

[8] M. Black, D. Fleet, Y. Yacoob, *Robustly estimating changes in image appearance*, Computer Vision and Image Understanding, vol. 78, no. 1, pp. 8–31, 2000.

[9] M. Blatt, S. Wiseman, E. Domany, *Data clustering using a model granular magnet*, Neural Computation, vol. 9, no. 8, pp. 1805–1842, 1997.

[10] S. Blostein, T. Huang, *Error analysis in stereo determination of 3-D point positions*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 6, pp. 752–765, 1989.

[11] Y. Boykov, O. Veksler, R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pp. 1222–1239, 2001.

[12] A. Bugeau, P. Pérez, *Detection and segmentation of moving objects in complex scenes*, Computer Vision and Image Understanding, vol. 113, no. 4, pp. 459–476, 2009.

[13] I. Cameron, K. Hangos, *Process Modelling and Model Analysis*, Academic Press, 2001.

[14] H. Chen, P. Meer, *Robust fusion of uncertain information*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 35, no. 3, pp. 578–586, 2005.

[15] C. Choi, H. Christensen, *3D textureless object detection and tracking: an edge-based approach*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3877–3884, 2012.

[16] C. Ciliberto, S. Fanello, L. Natale, G. Metta, *A heteroscedastic approach to independent motion detection for actuated visual sensors*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3907–3913, 2012.

[17] C. Ciliberto, U. Pattacini, L. Natale, F. Nori, G. Metta, *Reexamining Lucas-Kanade method for real-time independent motion detection: application to the iCub humanoid robot*, IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4154–4160, 2011.

[18] T. Darrell, A. Pentland, *Robust estimation of a multi-layered motion representation*, IEEE Workshop on Visual Motion, pp. 173–178, 1991.

[19] B. Delaunay, *Sur la sphère vide*, Bulletin de l'Académie des Sciences de l'URSS, Classe des sciences mathématiques et naturelles, vol. 6, pp. 793–800, 1934.

[20] B. Douillard, D. Fox, F. Ramos, H. Durrant-Whyte, *Classification and semantic mapping of urban environments*, International Journal of Robotics Research, vol. 30, no. 1, pp. 5–32, 2011.

[21] H. Durrant-Whyte, T. Bailey, *Simultaneous localization and mapping: part I*, IEEE Robotics and Automation Magazine, vol. 13, no. 2, pp. 99–110, 2006.

[22] H. Durrant-Whyte, T. Bailey, *Simultaneous localization and mapping: part II*, IEEE Robotics and Automation Magazine, vol. 13, no. 3, pp. 108–117, 2006.

[23] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, *Vision meets robotics: the KITTI dataset*, International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.

[24] A. Geiger, J. Ziegler, C. Stiller, *StereoScan: Dense 3D reconstruction in real-time*, Intelligent Vehicles Symposium, pp. 963–968, 2011.

[25] J. Hartigan, M. Wong, *Algorithm AS 136: a k-means clustering algorithm*, Journal of the Royal Statistical Society, vol. 28, no. 1, pp. 100–108, 1979.

[26] R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2003.

[27] S. Heinz, *Mathematical Modeling*, Springer-Verlag, 2011.

[28] B. Horn, B. Schunck, *Determining optical flow*, Artificial Intelligence, vol. 17, pp. 185–203, 1981.

[29] G. Huang, A. Rad, Y. Wong, *A new solution to map dynamic indoor environments*, International Journal of Advanced Robotic Systems, vol. 3, no. 3, pp. 199–210, 2006.

[30] A. Jain, M. Murty, P. Flynn, *Data clustering: a review*, ACM Computing Surveys, vol. 31, no. 3, pp. 264–323, 1999.

[31] R. Jain, H. Nagel, *On the analysis of accumulative difference pictures from image sequences of real world scenes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 1, no. 2, pp. 206–214, 1979.

[32] S. Julier, *The scaled unscented transform*, American Control Conference, pp. 4555–4559, 2002.

[33] R. Kaestner, J. Maye, Y. Pilat, R. Siegwart, *Generative object detection and tracking in 3D range data*, IEEE International Conference on Robotics and Automation, pp. 3075–3081, 2012.

[34] R. Kalman, *A new approach to linear filtering and prediction problems*, Journal of Basic Engineering, vol. 82, pp. 34–45, 1960.

[35] J. Klappstein, T. Vaudrey, C. Rabe, A. Wedel, R. Klette, *Moving object segmentation using optical flow and depth information*, Advances in Image and Video Technology, vol. 5414, pp. 611–623, 2009.

[36] D. Koller, N. Friedman, *Probabilistic Graphical Models*, MIT Press, 2009.

[37] D. Koller, U. Lerner, D. Angelov, *A general algorithm for approximate inference and its application to hybrid Bayes nets*, Conference on Uncertainty in Artificial Intelligence, pp. 324–333, 1999.

[38] M. Kong, J. Leduc, B. Ghosh, V. Wickerhauser, *Spatio-temporal continuous wavelet transforms for motion-based segmentation in real image sequences*, IEEE International Conference on Image Processing, pp. 662–666, 1998.

[39] S. Kumar, F. Odone, N. Noceti, L. Natale, *Object segmentation using independent motion detection*, IEEE-RAS International Conferences on Humanoid Robots, pp. 94–100, 2015.

[40] M. Kumar, P. Torr, A. Zisserman, *Learning layered motion segmentations of video*, International Journal of Computer Vision, vol. 76, no. 3, pp. 301–319, 2008.

[41] S. Lauritzen, *Propagation of probabilities, means and variances in mixed graphical association models*, Journal of the American Statistical Association, vol. 87, no. 420, pp. 1098–1108, 1992.

[42] D. Lee, P. Merrel, Z. Wei, B. Nelson, *Two-frame structure from motion using optical flow probability distributions for unmanned air vehicle obstacle avoidance*, Machine Vision and Applications, vol. 21. no. 3, pp. 229–240, 2010.

[43] P. Lenz, J. Ziegler, A. Geiger, M. Roser, *Sparse scene flow segmentation for moving object detection in urban environmnets*, Intelligent Vehicles Symposium, pp. 926–932, 2011.

[44] D. Lowe, *Object recognition from local scale-invariant features*, International Conference on Computer Vision, pp. 1150–1157, 1999.

[45] M. Luber, G. Tipaldi, K. Arras, *Place-dependent people tracking*, The International Journal of Robotics Research, vol. 30, no. 3, pp. 280–293, 2011.

[46] B. Lucas, T. Kanade, *An iterative image registration technique with an application to stereo vision*, International Joint Conference on Artificial Intelligence, pp. 674–679, 1981.

[47] S. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 674–693, 1989.

[48] L. Matthies, S. Shafer, *Error modeling in stereo navigation*, IEEE Journal of Robotics and Automation, vol. 3, no. 3, pp. 239–250, 1987.

[49] C. McCarthy, N. Barnes, *Performance of optical flow techniques for indoor navigation with a mobile robot*, IEEE International Conference on Robotics and Automation, pp. 5093–5098, 2004.

[50] G. Medioni, M. Lee, C. Tang, *A Computational Framework for Segmentation and Grouping*, Elsevier Science, 2000.

[51] S. Mills, K. Novins, *Motion segmentation in long image sequences*, British Machine Vision Conference, pp. 162–171, 2000.

[52] C. Min, G. Medioni, *Inferring segmented dense motion layers using 5D tensor voting*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 9, pp. 1589–1602, 2008.

[53] T. Minka, *Expectation propagation for approximate Bayesian inference*, Conference on Uncertainty in Artificial Intelligence, pp. 362–369, 2001.

[54] T. Moon, *The expectation-maximization algorithm*, IEEE Signal Processing Magazine, vol. 13, no. 6, pp. 47–60, 1996.

[55] M. Muja, D. Lowe, *Scalable nearest neighbor algorithms for high dimensional data*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 11, pp. 2227–2240, 2014.

[56] D. Müller, M. Meuter, S. Park, *Motion segmentation using interest points*, Intelligent Vehicles Symposium, pp. 19–24, 2008.

[57] I. Myung, *Tutorial on maximum likelihood estimation*, Journal of Mathematical Psychology, vol. 47, no. 1, pp. 90–100, 2003.

[58] J. Odobez, P. Bouthemy, *Separation of moving regions from background in an image sequence acquired with a mobile camera*, Video Data Compression for Multimedia Computing, pp. 283–311, 1997.

[59] S. Ogale, Y. Aloimonos, *Shape and the stereo correspondence problem*, International Journal of Computer Vision, vol. 65, no. 3, pp. 147–162, 2005.

[60] *OXTS RT GNSS-aided IMU user manual*, available online at www.oxts.com/Downloads/Products/RT3000/rtman.pdf, last accessed June 2016.

[61] N. Paragios, G. Tziritas, *Adaptive detection and localization of moving objects in image sequences*, Signal Processing: Image Communication, vol. 14, pp. 277-296, 1999.

[62] S. Pundlik, S. Birchfield, *Motion segmentation at any speed*, British Machine Vision Conference, pp. 427–436, 2006.

[63] R. Radke, S. Andra, O. Al-Kofahi, B. Roysam, *Image change detection algorithms: a systematic survey*, IEEE Transactions on Image Processing, vol. 14, no. 3, pp. 294–307, 2005.

[64] Y. Ren, C. Chua, Y. Ho, *Statistical background modeling for non-stationary camera*, Pattern Recognition Letters, vol. 24, no. 1, pp. 183–196, 2003.

[65] D. Rizzini, F. Oleari, A. Atti, J. Aleotti, S. Caselli, *Unsupervised range image segmentation and object recognition using feature proximity and Markov Random Field*, Intelligent Autonomous Systems, vol. 13, pp. 807–820.

[66] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, *ORB: An efficient alternative to SIFT or SURF*, IEEE International conference on computer vision, pp. 2564–2571, 2011.

[67] T. Sheikh, M. Shah, *Bayesian modeling of dynamic scenes for object detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 11, pp. 1778–1792, 2005.

[68] N. Shental, A. Zomet, T. Hertz, Y. Weiss, *Pairwise clustering and graphical models*, Advances in Neural Information Processing Systems, pp. 185–192, 2004.

[69] J. Shi, C. Tomasi, *Good features to track*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 593–600, 1994.

[70] S. Shimojo, G. Silverman, K. Nakayama, *Occlusion and the solution to the aperture problem for motion*, Vision Research, vol. 29, no. 5, pp. 619–626, 1989.

[71] B. Silverman, *Density Estimation for Statistics and Data Analysis*, CRC Press, 1986.

[72] S. Smith, *Integrated real-time motion segmentation and 3D interpretation*, International Conference on Pattern Recognition, pp. 49–55, 1996.

[73] J. Snyder, *Flattening the earth*, The University of Chicago Press, 1997.

[74] Z. Sun, G. Bebis, R. Miller, *On-road vehicle detection using optical sensors: a review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 5, pp. 694–711, 2006.

[75] D. Sun, E. Sudderth, M. Black, *Layered segmentation and optical flow estimation over time*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 1768–1775, 2012.

[76] A. Tavakkoli, M. Nicolescu, G. Bebis, M. Nicolescu *Non-parametric statistical background modeling for efficient foreground region detection*, Machine Vision and Applications, vol. 20, no. 6, pp. 395–409, 2009.

[77] G. Terrel, D. Scott, *Variable kernel density estimation*, The Annals of Statistics, vol. 20, no. 3, pp. 1236–1265, 1992.

[78] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[79] V. Usenko, J. Engel, J. Stückler, D. Cremers, *Reconstructing street-scenes in real-time from a driving car*, IEEE International Conference on 3D Vision, pp. 607–614, 2015.

[80] M. Wainwright, M. Jordan, *Graphical models, exponential families, and variational inference*, Foundations and Trends in Machine Learning, vol. 1–2, pp. 1–305, 2008.

[81] C. Wang, C. Thorpe, S. Thrun, *Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas*, IEEE International Conference on Robotics and Automation, pp. 842–849, 2003.

[82] Y. Weiss, *Smoothness in layers: motion segmentation using nonparametric mixture estimation*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 520–526, 1997.

[83] L. Wiskott, *Segmentation from motion: combining Gabor- and Mallat-wavelets to overcome the aperture and correspondence problems*, Pattern Recognition, vol. 32, pp. 1751–1766, 1999.

[84] C. Wojek, S. Walk, S. Roth, K. Schindler, B. Schiele, *Monocular visual scene understanding: understanding multi-object traffic scenes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 4, pp. 882–897, 2013.

[85] Y. Yang, S. Dunson, *Sequential Markov Chain Monte Carlo*, Cornell University Library, arXiv preprint arXiv:1308.3861, 2013.

[86] L. Zappella, X. Lladó, J. Salvi, *New trends in motion segmentation*, INTECH Pattern Recognition, pp. 31–46, 2009.

Appendix A

Probability Theory

In this appendix we provide a very brief overview of some concepts from Probability Theory employed in this dissertation. We start with a number of definitions in Section A.1 and then state a few useful identities in Section A.2.

A.1 Definitions

Here we define the concepts of a random variable, probability functions, joint and conditional probabilities, marginalization and independence. We end the section with quick mention of the dirac-delta function and Gaussian distributions.

A.1.1 Random variable

A random variable X is a variable that can take on a set of possible values, each with an associated probability. If the set is discrete, X is said to be a discrete random variable, and if X can assume only two possible values (typically denoted as 0 and 1), it is called a binary random variable. If the set of possible values is continuous, X is a continuous random variable.

As an example, consider the statistical experiment of throwing a die and let X denote the outcome. The possible events are $\{X = 1,\ X = 2, \ldots, X = 6\}$. In this case X is a discrete random variable. An example of a continuous variable is one that denotes the outcome of measuring the time it takes to complete a particular task.

A.1.2 Probability functions

A probability function is used to describe the relative likelihood that a random variable takes on a particular value. Such a function is nonnegative and sums to one.

If the random variable is discrete, its probability function is called a probability mass function. For the die example above, the probability mass function might be

$$p\left(X = i\right) = \frac{1}{6}, \qquad i \in \{1, 2, 3, 4, 5, 6\}. \tag{A.1}$$

For continuous random variables the probability function is called the probability density function. An example of such a function is the uniform probability density function, which is defined as

$$p(X = x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b, \\ 0 & \text{otherwise.} \end{cases} \tag{A.2}$$

A.1.3 Joint probability

We might be interested in the probability function over the combined possible values of two or more random variables. We define that as the joint probability. The joint probability of X and Y is written as $p(X, Y)$.

A.1.4 Conditional probability

We use a conditional probability when we are interested in the probability of some variable X given that we observe the value of another variable Y. This probability is defined as

$$p(X \,|Y) = \frac{p(X, Y)}{p(Y)}. \tag{A.3}$$

A.1.5 Marginalization

From the joint probability over a set of variables $\{X, Y\}$, we can obtain the probability for a subset X through a process called marginalization. It is defined as

$$p(X) = \sum_{Y} p(X, Y) \tag{A.4}$$

for discrete random variables, and

$$p(X) = \int p(X, Y)\, dY \tag{A.5}$$

for continuous random variables. Note that we sum or integrate over all possible values of Y.

A.1.6 Statistical independence

Two random variables X and Y are said to be statistically independent if an observation of the one does not influence the probability of the other. More formally, X and Y are statistically independent if and only if

$$p(X \,|Y) = p(X), \tag{A.6}$$

which, according to A.3, is equivalent to

$$p(X, Y) = p(X)\, p(Y). \tag{A.7}$$

Furthermore, two random variables X and Y are said to be conditionally independent given a third random variable Z if and only if

$$p(X, Y \,|Z) = p(X \,|Z)\, p(Y \,|Z), \tag{A.8}$$

It means that if Z is observed, then observing X will not influence the probability of Y.

A.1.7 Dirac-delta function

The dirac-delta function is a distribution function defined on real numbers, and is zero everywhere except at zero. That is,

$$\delta(x) = \begin{cases} \infty, & x = 0, \\ 0, & x \neq 0. \end{cases} \tag{A.9}$$

A property of the dirac-delta function used throughout this work is

$$\int_{-\infty}^{\infty} f(x)\,\delta(x)\,dx = f(0)\,. \tag{A.10}$$

A.1.8 Gaussian distribution

The Gaussian (or normal) distribution is commonly encountered in statistics, and also throughout this dissertation. For a one-dimensional continuous random variable x it is defined as

$$\mathcal{N}\left(x\,|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{A.11}$$

This distribution is completely described by the mean μ and standard deviation σ. Some examples of this distribution are shown in Figure A.1 (left).

If the random variable is k-dimensional, we may define the multivariate Gaussian probability density function as

$$\mathcal{N}\left(\boldsymbol{x}\,|\boldsymbol{\mu},\Sigma\right) = \frac{1}{\sqrt{(2\pi)^k\,|\Sigma|}}\exp\left(-(\boldsymbol{x}-\boldsymbol{\mu})^T\,\Sigma^{-1}\,(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{A.12}$$

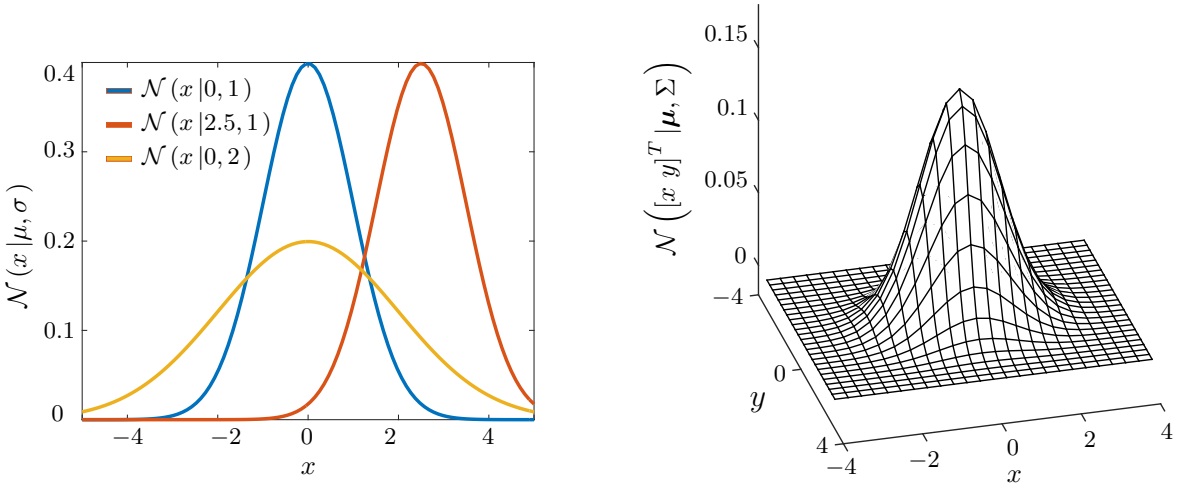


Figure A.1: Examples of one-dimensional Gaussian distribution functions on the left, and a two-dimensional Gaussian distribution with $\boldsymbol{\mu} = \mathbf{0}$ and Σ equal to the 2×2 identity matrix on the right.

where $|*|$ denotes the determinant of a matrix. This distribution can also be uniquely specified by the mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. Figure A.1 (right) shows an example of a two-dimensional normal distribution.

A.2 Useful identities

Here we state some useful identities, including the chain rule, Bayes' rule, and also an identity involving the product of two Gaussian distributions.

A.2.1 Chain rule

The chain rule allows us to express the joint probability over many variables as a product of conditional probabilities. It is given by

$$p\left(x_1, \ldots, x_N\right) = \prod_{i=1}^{N} p\left(x_i \left| x_1, \ldots, x_{i-1}\right.\right), \tag{A.13}$$

and also applies to conditional probabilities, i.e.

$$p\left(x_1, \ldots, x_N \left| z\right.\right) = \prod_{i=1}^{N} p\left(x_i \left| x_1, \ldots, x_{i-1}, z\right.\right). \tag{A.14}$$

A.2.2 Bayes' rule

Bayes' rule is a combination of the definition of conditional probability and the chain rule. It can be stated as

$$p\left(a \left| b\right.\right) = \frac{p\left(b \left| a\right.\right) p\left(a\right)}{p\left(b\right)}. \tag{A.15}$$

A.2.3 Product of two Gaussian distributions

The product of two Gaussian distributions over a common variable is also a Gaussian distribution.

Mathematically this is expressed as

$$\mathcal{N}\left(\boldsymbol{x} \left| \boldsymbol{a}, A\right.\right) \mathcal{N}\left(\boldsymbol{x} \left| \boldsymbol{b}, B\right.\right) = \mathcal{N}\left(\boldsymbol{a} \left| \boldsymbol{b}, A + B\right.\right) \mathcal{N}\left(\boldsymbol{x} \left| \boldsymbol{c}, C\right.\right), \tag{A.16}$$

where

$$C = \left(A^{-1} + B^{-1}\right)^{-1}, \tag{A.17}$$

$$\boldsymbol{c} = C\left(A^{-1}\boldsymbol{a} + B^{-1}\boldsymbol{b}\right). \tag{A.18}$$

Appendix B

Transformation of random variables

In this appendix we provide details of how a distribution resulting from the transformation of random variables can be calculated. There are many types of transformations and ways to calculate the resulting distribution, but we limit the discussion to techniques relevant to the work in this dissertation.

Suppose the transformation $g(\cdot)$ maps the continuous random variable $\boldsymbol{X}$ to the continuous variable $\boldsymbol{Y}$, i.e. $\boldsymbol{Y} = g(\boldsymbol{X})$. Suppose further that $\boldsymbol{X}$ has a distribution function $f_X(\boldsymbol{x})$. The goal is to find the distribution function over $\boldsymbol{Y}$, that is $f_Y(\boldsymbol{y})$.

In this work we consider random variables with a Gaussian distribution, so

$$f_X(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}\,|\boldsymbol{\mu}_X, \Sigma_X)\,. \tag{B.1}$$

B.1 Sum of two independent random variables

Suppose we are interested in the distribution resulting from the sum of two random variables $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$. If $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ have distribution functions $f_{X_1}(\boldsymbol{x_1})$ and $f_{X_2}(\boldsymbol{x_2})$ respectively, the distribution over $\boldsymbol{Y}$ resulting from the transformation $\boldsymbol{Y} = g(\boldsymbol{X_1}, \boldsymbol{X_2}) = \boldsymbol{X_1} + \boldsymbol{X_2}$ is given by

$$f_Y(\boldsymbol{y}) = \int_{-\infty}^{\infty} f_{X_1}(\boldsymbol{x}) f_{X_2}(\boldsymbol{y} - \boldsymbol{x}) d\,\boldsymbol{x}, \tag{B.2}$$

which is the convolution of the two distributions f_{X_1} and f_{X_2} [27].

If $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$ are both Gaussian random variables, i.e. $f_{\boldsymbol{X_1}} = \mathcal{N}(\boldsymbol{x}_1\,|\boldsymbol{\mu}_1, \Sigma_1)$ and $f_{\boldsymbol{X_2}} = \mathcal{N}(\boldsymbol{x}_2\,|\boldsymbol{\mu}_2, \Sigma_2)$, we have

$$f_Y(\boldsymbol{y}) = \int_{-\infty}^{\infty} \mathcal{N}(\boldsymbol{x}\,|\boldsymbol{\mu_1}, \Sigma_1)\,\mathcal{N}(\boldsymbol{y} - \boldsymbol{x}\,|\boldsymbol{\mu_2}, \Sigma_2)\,d\,\boldsymbol{x} = \mathcal{N}(\boldsymbol{y}\,|\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \Sigma_1 + \Sigma_2)\,, \tag{B.3}$$

which indicates that $\boldsymbol{Y}$ is also normally distributed with mean $\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2$ and covariance $\Sigma_1 + \Sigma_2$ [27].

B.2 The transformation $\boldsymbol{AX + b}$

Next we consider the linear transformation, where the Gaussian random variable $\boldsymbol{X}$ is multiplied by the matrix A and the vector $\boldsymbol{b}$ is added to the result.

If $\boldsymbol{Y} = g(\boldsymbol{x}) = A\boldsymbol{X} + b$, where $\boldsymbol{X}$ is Gaussian with $f_X(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}\,|\boldsymbol{\mu}_X, \Sigma_X)$, then

$$f_Y(\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{y}\,\middle|A\boldsymbol{\mu}_X + \boldsymbol{b}, A\Sigma_X A^T\right). \tag{B.4}$$

B.3 Nonlinear transformations

In Sections B.1 and B.2 two linear transformations are considered. If the transformation $\boldsymbol{y} = g(\boldsymbol{x})$ is nonlinear in $\boldsymbol{x}$, $f_Y(\boldsymbol{y})$ is not necessarily a Gaussian distribution. In many cases, however, it can be approximated by a Gaussian distribution. We discuss two methods to perform such an approximation.

B.3.1 The Taylor approximation

A popular method to approximate the density function f_Y when the transformation is nonlinear, is to compute the first-order Taylor approximation of the function $g(\boldsymbol{X})$. This results in a linear transformation of the form $A\boldsymbol{X} + \boldsymbol{b}$. The method is also called the delta method, and lays the foundation for the extended Kalman filter (EKF) [78].

The first step is to perform the first-order Taylor approximation of the function

$$g(\boldsymbol{X}) \approx g(\boldsymbol{\mu}_X) + g'(\boldsymbol{\mu}_X)(\boldsymbol{X} - \boldsymbol{\mu}_X). \tag{B.5}$$

If we let $g'(\boldsymbol{\mu}_X) = \frac{\delta Y}{\delta X}\big|_{\boldsymbol{\mu}_X} = J$, which is the Jacobian matrix, we have

$$g(\boldsymbol{X}) \approx J\boldsymbol{X} + \boldsymbol{b}, \tag{B.6}$$

where $\boldsymbol{b} = g(\boldsymbol{\mu}_X) - g'(\boldsymbol{\mu}_X)\boldsymbol{\mu}_X$. If $g(\boldsymbol{X})$ is a nonlinear function approximated thusly, and if $\boldsymbol{X}$ is a Gaussian random variable with density function $f_X(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}\,|\boldsymbol{\mu}_{\boldsymbol{X}}, \Sigma_X)$, then it can be shown that

$$f_Y(\boldsymbol{y}) \approx \mathcal{N}\left(\boldsymbol{y}\,\middle|g(\boldsymbol{\mu}_X), J\Sigma_X J^T\right). \tag{B.7}$$

The Taylor approximation approach generally works well when the transformed mean $g(\boldsymbol{\mu}_x)$ is close to the true mean $\boldsymbol{\mu}_{\boldsymbol{Y}}$.

B.3.2 The unscented transform

The unscented transform [32] is another way to linearize nonlinear transformations of random variables. The general idea is to identify points in the original distribution f_X, called sigma points, to encode the mean and covariance information of f_X. The sigma points are then transformed using the nonlinear function $g(\boldsymbol{X})$, and a Gaussian distribution is fitted over the transformed sigma points in order to approximate $f_Y(\boldsymbol{Y})$. This method lays the foundation for the unscented Kalman filter (UKF).

Formally, if $g(\boldsymbol{X})$ is a nonlinear function and $\boldsymbol{X}$ an n-dimensional Gaussian random variable with density function $f_X(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}\,|\boldsymbol{\mu_X}, \Sigma_X)$, then $f_Y(\boldsymbol{y})$ can be approximated as

$$f_Y(\boldsymbol{y}) \approx \mathcal{N}\left(\boldsymbol{y}\,\middle|\hat{\boldsymbol{\mu}}_Y, \hat{\Sigma}_Y\right). \tag{B.8}$$

By using the unscented transform, the mean and covariance are given by

$$\hat{\boldsymbol{\mu}}_Y = \sum_{i=0}^{2n} w_m^{(i)} g(\boldsymbol{X}^{(i)}), \quad \hat{\Sigma}_Y = \sum_{i=0}^{2n} w_c^{(i)} \left(g(\boldsymbol{X}^{(i)}) - \hat{\boldsymbol{\mu}}_Y\right)\left(g(\boldsymbol{X}^{(i)}) - \hat{\boldsymbol{\mu}}_Y\right)^T, \tag{B.9}$$

where

$$\begin{aligned}
w_m^{(0)} &= \frac{\lambda}{n+\lambda}, && \text{(B.10)}\\
w_m^{(0)} &= w_m^{(0)} + (1 - \alpha^2 + \beta), && \text{(B.11)}\\
w_m^{(i)} &= w_c^{(i)} = \frac{1}{2(n+\lambda)}, \quad i \in \{1, \ldots, 2n\}, && \text{(B.12)}\\
\boldsymbol{X}^{(0)} &= \boldsymbol{\mu}_X, && \text{(B.13)}\\
\boldsymbol{X}^{(i)} &= \boldsymbol{\mu}_X + \left[\sqrt{(n+\lambda)\Sigma_X}\right]_i, \quad i \in \{1, \ldots, n\}, && \text{(B.14)}\\
\boldsymbol{X}^{(i)} &= \boldsymbol{\mu}_X - \left[\sqrt{(n+\lambda)\Sigma_X}\right]_i, \quad i \in \{n+1, \ldots, 2n\}. && \text{(B.15)}
\end{aligned}$$

Here $\alpha \in (0, 1]$, $\beta = 2$, $\lambda = \alpha^2(n + \kappa) - n$, $\kappa \geq 0$, and $\left[\sqrt{\cdot}\right]_i$ denotes the i-th column of the matrix square root.

The unscented transform is usually preferred over the Taylor approach when the transformed mean $g(\boldsymbol{\mu}_X)$ is far from the true mean $\boldsymbol{\mu}_Y$.