Multi-target Tracking Using on-line Viterbi Optimisation and Stochastic Modelling

Håkan Ardö

Lund Institute of Technology
Lund University

Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Doctoral Theses in Mathematical Sciences 2008:10
ISSN 1404-0034

ISBN 978-628-7685-2
LUTFMA-1034-2008

Printed in SWEDEN by MediaTryck, Lund 2009

Preface

The contents of the thesis is based on the following papers,

Main papers

- Ardö, Håkan and Åström, Kalle and Berthilsson, Rikard, "Multi-target Tracking Using on-line Viterbi Optimisation" *To be submitted.*, 2009.
- Ardö, Håkan and Åström, Kalle and Berthilsson, Rikard, "Bayesian Formulation of Image Patch Matching Using Cross-correlation" *To be submitted.*, 2009.
- Laureshyn, Aliaksie and Ardö, Håkan and Thomas Jonsson and Åse Svensson, "Application of automated video analysis for behavioural studies: concept and experience", *10th International Conference on Application of Advanced Technologies in Transportation*, 2008
- Ardö, Håkan and Åström, Kalle, "Multi Sensor Loitering Detection Using Online Viterbi", *Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2007.
- Ardö, Håkan and Berthilsson, Rikard and Åström, Kalle, "Real Time Viterbi Optimization of Hidden Markov Models for Multi Target Tracking", *IEEE Workshop on Motion and Video Computing*, 2007
- Laureshyn, Aliaksie and Ardö, Håkan, Straight to Video? Automated video Analysis as a tool for Analyzing road user behavior *Traffic Technology International, Annual*, 2007
- Laureshyn, Aliaksie and Ardö, Håkan, "Automated video analysis as a tool for analysing road user behaviour", *13th World Congress on Intelligent Transport Systems and Services*, 2006
- Ardö, Håkan and Berthilsson, Rikard, "Adaptive Background Estimation using Intensity Independent Features", *17th British Machine Vision Conference*, 2006

Subsidiary papers

- Jiang, Hongtu and Ardö, Håkan and Owall, Viktor, "Hardware Architecture for Real-Time Video Segmentation Utilizing Memory Reduction Techniques", Accepted for publication in *IEEE Transactions on Circuits and Systems for Video Technology*,
- Jiang, Hongtu and Owall, Viktor and Ardö, Håkan, "Real-Time Video Segmentation with VGA Resolution and Memory Bandwidth Reduction", *IEEE International Conference on Video and Signal Based Surveillance*, 2006.
- Jiang, Hongtu and Ardö, Håkan and Owall, Viktor, "Hardware Accelerator Design for Video Segmentation with Multi-modal Background Modelling", *IEEE International Symposium on Circuits and Systems*, 2005
- Ardö, Håkan, "Learning based system for detection and tracking of vehicles", *14th Scandinavian Conference on Image Analysis*, 2005

In collaboration with the department of Electroscience a hardware implementation of Stauffer and Grimson's [75] background foreground segmentation algorithm were constructed. The main bottleneck of the algorithm were concluded to be the memory bandwidth required. This was reduced by utilising the fact that the background distribution of neighbouring pixels often is very similar. By detecting when it is similar enough to be approximated as identical the entire background model can be run-length encoded and thereby the memory bandwidth is lowered. The main work of this collaboration were performed by Hongtu and is documented in the references above and will not be discussed further in this thesis.

The last reference above documents some early, initial work on automated camera calibration. The idea is to build an automated system where all the operator has to do is to mount the camera and point it at an intersection or road segment of interest. The system will then analyse the scene and by tracking moving blobs, estimate the camera parameters, find the ground plane and locate the different lanes. As the system learns more and more about the scene it is viewing it will be able to make more and more advanced analyses of it. Once the geometry is learnt the blobs tracked can be classified into buses, cars, pedestrians and bicycles based on their size and speed. Then, typical trajectories used to pass the intersection can be learnt and abnormalities detected. Once the different lanes are detected it can be decided whether the scene is a road segment, an intersection, a T-junction or a roundabout and then different more specific analyses can be applied based on what kind of scene is viewed. While this is still an overall goal it has fallen outside the scope of this thesis.

Acknowledgements

First of all, let me thank my supervisors Kalle Åström and Rikard Berthilsson for their constant enthusiasm and many fruitful discussions about these subjects. They have also provided lots of constructive feedback on this manuscript which has improved it's quality immensely.

During the time this theses was developed I've had the privilege to work at the Center for Mathematical Sciences at Lund University. I would like to thank all colleagues and coworkers there. No one mentioned and none forgotten. The main funding of the project have been provided by Vägverket and Vinova for which we all are grateful.

A significant part of this work have been done in collaboration with Aliaksie Laureshyn and Åse Svensson at the Department of Technology and Society, Lund University. A special thanks goes to Aliaksie for putting up with my sometimes buggy implementations that's seldom ready when promised and not always do what they are supposed to do.

Finally, a great thanks to my friends and family for their general support in all aspects of life and for providing lots of opportunities to not work on mathematical problems.

Contents

Chapter 1

Introduction

There is an increased need for automatic analysis of traffic user behaviour and safety. This project is initiated by the needs of governmental organisations such as 'Väg och trafik Institutet' but also department of public works in cities. One of the primary motives here is the possibilities to assess traffic safety. Often safety is neglected when constructing new roads and intersections. If safety is considered there are very few tools to actually measure and monitor traffic safety. One possibility is to study how many accidents that are reported to the police at a certain section of a road or at a certain intersection during a year. Recent research [29] has, however, shown that it is possible to predict how many such accidents there is by manually observing certain events during a shorter time interval, e.g. 2-3 days. These traffic studies are however very costly and time-consuming and requires that trained personnel study videos of traffic.

The problem of tracking moving objects has been studied for a long time, see for example [70, 26]. Two main approaches are commonly used. Either a set of hypothesis are generated and tested against the observed image [30, 20, 33, 8], or methods for detecting objects in single frames, e.g. using edge information, templates or machine learning, are combined with dynamic models in order to gain robustness [65, 90, 76, 12].

Tracking can be used in automatic traffic analysis systems, where the goal may be to increase traffic safety or increase traffic flow by reducing risk for congestion. Concerning traffic safety, an intelligent traffic monitoring system can be used to gather statistics from traffic scenes that can form the basis, for example, for redesigning dangerous street crossings. As stated above, such statistics is today gathered through costly and manual ocular inspection during several days. Furthermore, by having access to instant and accurate traffic data throughout a road net, it is possible to reduce the risk for traffic congestion and optimising traffic flow, see [56, 63].

1.1 Notations

Scalars	a, b, c
Vectors	$\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$
Matrices	$\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$
Transpose of $\mathbf{A}$	$\mathbf{A}^{\mathrm{T}}$
Identity matrix of size $d \times d$	$\mathbf{I}_d$
Column vector, d-long, all elements one	$\mathbf{1}_d$
Column vector, d-long, all elements zero	$\mathbf{0}_d = 0\mathbf{1}_d$
Matrix, $d \times d$, all elements one	$\mathbf{1}_{d\times d} = \mathbf{1}_d{\mathbf{1}_d}^{\mathrm{T}}$
Matrix, $d \times d$, all elements zero	$\mathbf{0}_{d\times d} = 0\mathbf{1}_{d\times d}$
Scalar valued functions	$g(\cdot)$, $h(\cdot)$
Vector valued functions	$\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$
Sets	$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{S}, \mathcal{X}$
Number of elements in $\mathcal{A}$	$\lvert\mathcal{A}\rvert$
Spaces	$\mathbb{R}$, $\mathbb{N}$, $\mathbb{S}$
Expected Value of x	$\mathcal{E}(x)$
Images (see below)	A, B, C, I
Pixel $\mathbf{a} = (x, y)$ in image I	$I(\mathbf{a})$, $I(x, y)$
Probability distribution of x (see below)	$f(x)$
Probability distribution of x conditioned on y	$f(x \mid y)$
Probability of the event x	$p(x)$
Probability of the event x conditioned on y	$p(x \mid y)$

Gaussian distribution	$\mathcal{N}(\mathbf{x} \mid \mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n \lvert\mathbf{\Sigma}\rvert}} e^{-(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x}-\mu)}$
Gamma function	$\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} \mathrm{d}t$
Incomplete gamma function	$\Gamma(z, a) = \int_a^\infty e^{-t} t^{z-1} \mathrm{d}t$
Error function	$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t$

1.1.1 Images and functions

This thesis primarily deals with sequences of grey scale images. This is mainly due to notational convenience. In many cases generalisation to colour images is straightforward. In other cases it is not. In a few cases some short discussion on how colour images can be treated is also included.

Images will be treated as $\mathbb{R}^2 \to \mathbb{R}$ functions. These functions are constructed from the image data returned by a camera by nearest neighbour interpolation in between integer pixel positions and by setting the image intensity equal to zero outside the real image borders. Video sequences will be treated as a sequences of images, $I_t(\cdot)$, for some discrete time steps $t = 0, 1, \cdots, \tau$. To simplify notation, the function parameter will be dropped when considering strictly pixel wise operations, e.g.

$$\begin{array}{c} D_t = \left(I_t - I_{t-1}\right)^2 \\ \Leftrightarrow \\ D_t\left(\mathbf{a}\right) = \left(I_t\left(\mathbf{a}\right) - I_{t-1}\left(\mathbf{a}\right)\right)^2 \text{ for all } \mathbf{a} \in \mathbb{R}^2 \end{array} . \tag{1.1}$$

When the entire image is considered, the notation $I_t(\cdot)$ will be used in order to distinguish it from the pixel wise notation.

This notation also allows the use integrals instead of sums when summing over images. The reason for this is that it admits the use convenient notation and well known results from integration theory.

1.1.2 Probabilities and distributions

Let $\bar{a}$ be a stochastic variable with distribution function $f_{\bar{a}}\left(a\right)$. This distribution will be denoted $f\left(a\right)$, where the stochastic variable is implicitly specified as the stochastic variable from which the parameter a is a realisation (sample). The notation a will be used both when referring to the stochastic variable, $\bar{a}$, the parameter of the probability distribution function, a, and a realisation of the stochastic variable. When several realisations are considered, they will be denoted $a^{(i)}$ for $i = 1, 2, \cdots$. In the same manner, the conditional distribution of $\bar{a}$ given that the stochastic variable $\bar{b}$ has the realisation b, $f_{\bar{a}}\left(a\left|\bar{b} = b\right.\right)$, is denoted $f\left(a\left|b\right.\right)$.

For a discrete stochastic variable $\bar{a}$, the probability of the event $\bar{a} = a$ will be denoted $p\left(a\right)$ and the probability of the event $\bar{a} = 0$ will be denoted $p\left(a = 0\right)$. In the same manner, the probability of the event $\bar{a} = a$ conditioned on the event $\bar{b} = b$ will be denoted $p\left(a\left|b\right.\right)$.

1.2 Foreground/Background Segmentation

To estimate the background in real time is an important first step for many video surveillance applications. Solving this problem, in a reliable way, allows remaining resources to be devoted to tracking and identifying moving objects and to interpret events that occur in the scene.

The setting is that of a static camera viewing some scene where most part of the scene is (pseudo) static background. Here pseudo static means that a tree swaying in the wind or rippling water often is considered part of the background. On top of this background there are objects moving, and it is those objects, the foreground, that are of interest. The camera viewing the scene produces one image I_t for each time step $t = 1, 2, \cdots$, and the task of the background/foreground segmentation algorithm is to segment out those moving objects from the background in each image.

1.2.1 Difference Image

A classical approach to foreground background segmentation is to look at the absolute difference between input frames, $|I_{t-\tau} - I_t|$ for some, typically small, values of τ. For this to become robust, more advanced features than the intensity levels can to be used. Jain and Nagel [34] for example uses 4×6 patches represented with their mean and variance.

A more recent method is suggested by Mahadevan and Vasconcelos [55]. They use $16\times16\times11$ spatio-temporal patches centred around each pixel. Each such patch is further divided into several overlapping $8\times8\times11$ patches and Dynamic Texture (DT) models are fitted to each of them, and the distribution of those dynamic textures are compared. The mutual information between the DTs in the centre of the patch and along the border of the patch is calculated. This value is thresholded to form a foreground detector. The idea is to locate small objects (smaller than 16x16) whose motion differ from its surrounding motion.

The problem with these methods is that if an objects stand still for a few frames they will directly become part of the background. This happens for example when a car stops for a red light. To solve this background subtraction could be used.

1.2.2 Background Subtraction

In its simplest form a background image, B is assumed to be available. This is an image of the scene when there are no objects present. The segmentation algorithms will study each pixel separately and if its distance to the background is larger than some threshold, σ it will be classified as foreground, otherwise as background. A binary segmentation image, F_t, that is 1 in foreground pixels and 0 in background pixels is then produced according to

$$F_t = \begin{cases} 1 & \text{if } |I_t - B| > \sigma \\ 0 & \text{otherwise} \end{cases} . \tag{1.2}$$

Unfortunately, a background image B is usually not available and for long term surveillance (more than a few minutes) the background is not completely static. In those cases the background image has to be estimated continuously from the video sequence and adapt to slow or more permanent changes in the background. In those cases some background model is assumed and the parameters of this model is updated continuously.

Wren *et al* [89] assumes the background pixels to be independent Gaussian distributed and update the mean and variance of each pixel using a *learning factor*, c. That is a real number, $0 < c < 1$, indicating how fast changes should fade into the background. An estimate of the mean background image B_t and its variance V_t is made pixel wise for each frame,

$$B_t = cB_{t-1} + (1 - c)\, I_t, \tag{1.3}$$

$$V_t = cV_{t-1} + (1 - c)\,(B_t - I_t)^2 . \tag{1.4}$$

With this model the likelihood of each pixel belonging to the background is,

$$f_{\text{bg}}(I_t) = \mathcal{N}(I_t \,|B_t, V_t)\,. \tag{1.5}$$

Thus it is the possible to produce a binary foreground image that defines the foreground as the pixels that deviates more than for example 2.5 standard deviations from the background mean value, i.e.

$$F_t = |I_t - B_t| > 2.5\sqrt{V_t}. \tag{1.6}$$

Kalman filter approaches

Equation 1.3 for updating the background model using a learning factor can be rewritten into the form

$$B_t = B_{t-1} + (1-c)\,(I_t - B_{t-1})\,. \tag{1.7}$$

If the factor $(1-c)$ were to be interpreted as the Kalman gain, this would be the mean updating equation of a Kalman filter with the pixel intensity used as both state and observation and a mean zero velocity dynamic model. See Section 1.3.1 for an introduction to the Kalman filter. That theory dictates how the Kalman gain should be chosen to estimate the mean and the variance of the state variable over time given that the variance of the observations are known. In this case, when the state and the observations are the same, the variation of the observation (e.g the noise levels) is not known but has to be measured from the data.

Karamann and Brandt [41] has suggested to assume known and constant noise levels and only estimate the mean values of the background. They use the learning factor ideas to set the Kalman gain to one of two different constants depending on whether the pixel currently is classified as foreground or as background. That would correspond to having a larger variance for (be more uncertain about) observation classified as foreground than observations classified as background. They also suggest to use a second order model that assumes zero mean acceleration of the pixel intensities instead and estimates the intensity speed (or change rate) as well.

Boninsegna and Bozzoli [6] has extended this approach by using the assumed noise level variance to calculate the Kalman gain according to the Kalman filtering theory and use this gain for pixels classified as background. For pixels classified as foreground, the Kalman gain is set to $\frac{S_{t^*}}{S_{t^*}+(B_t-I_t)^2}$, where S_{t^*} is the variance of the Kalman prediction, Equation 1.58, of the intensity the last time the pixel were classified as background.

Different features

The performance of the segmentation can be increased by not using the intensity values directly, but extracting features from the image and building a background model of the extracted features. Koller *et al* [47] concluded that the images of vehicles driving on a motorway almost entirely consists of uniform regions and horizontal or vertical lines.

To increase the importance of this three kinds of features they suggested to produce a feature vector with three elements at each pixel by applying three filters to the image prior to building the background model. The filters used where a Gaussian filter and two Gaussian derivatives in the horizontal and vertical direction respectively. They used the learning factor approach, or equivalently a first order Kalman filter, with different learning factors for pixels detected as foreground and pixels detected as background, on each of the three filter responses separately. One foreground mask were produced for each filter and the final result were the logical or between the three masks. The use of derivatives as features has also been considered by Pless *et al* [62]. In addition to that, they consider temporal derivatives as well as optical flow.

1.2.3 Mixture of Gaussians

If the background contains for example a swaying tree, the variance estimate, V_t, will become very high, which makes the system more or less blind in those areas. The problem is that the pixels are sometimes green leaves and sometimes brown branches, and a Gaussian distribution can't explain that kind of multi modal variations very well.

Stauffer and Grimson has suggested a method [75] that instead models each pixel as a mixture of Gaussians, and estimates it using an EM-like algorithm. Wayne and Schoonees [86] gave a stricter theoretical interpretation of that algorithm pointing out the various approximations made.

The algorithm is presented for colour images. Here only greyscale images are considered, but the generalisation to colour images is straightforward [86]. Each pixel is treated separately. The scene model states that the pixel at each time is viewing one of n possible surfaces some of which belong to the background such as leaves or branches on a swaying tree and some to the foreground such as a moving pedestrian. Each of those surfaces are assumed to generate intensities that are Gaussian distributed. The observed image I_t will depend on an unknown processes $J_t \in \{1, 2, \cdots, n\}$, specifying which surface is currently observed. The distribution of J_t is estimated as the discrete set of probabilities $W_{t,j} = p(J_t = j)$, and the distribution of the observed image, I_t, is assumed Gaussian with estimated mean $B_{t,j}$ and variance $V_{t,j}$ that depend on J_t, i.e. $f(I_t | J_t = j) = \mathcal{N}(I_t | B_{t,j}, V_{t,j})$. This gives the distribution of the observed pixels,

$$f(I_t) = \sum_{j=1}^{n} W_{t,j} \mathcal{N}(I_t | B_{t,j}, V_{t,j}), \tag{1.8}$$

which is a mixture of both the foreground and the background distributions as some of the components belong to the background and some to the foreground. The parameters, $W_{t,j}$, $B_{t,j}$ and $V_{t,j}$ are continuously estimated from the data. That's why they depend on t and what makes the model adapt to changes in the background. The number of components n has to be specified.

The model parameters are updated each frame using a learning factor, c, as follows. Each pixel, $\mathbf{x}$, in each frame, t, is assigned to the component, $J_t(\mathbf{x})$, it best matches,

$$J_t = \operatorname{argmax}_j \left(W_{t,j} \mathcal{N}\left(I_t \left| B_{t,j}, V_{t,j} \right. \right) \right). \tag{1.9}$$

The match is considered good enough if

$$|I_t - B_{t,J_t}| < 2.5\sqrt{V_{t,J_t}} \tag{1.10}$$

and in that case the matched component is updated using

$$B_{t,J_t} = cB_{t-1,J_t} + (1-c)\, I_t, \tag{1.11}$$

$$V_{t,J_t} = cV_{t-1,J_t} + (1-c)\left(B_{t,J_t} - I_t\right)^2, \tag{1.12}$$

$$W_{t,J_t} = cW_{t-1,J_t} + (1-c)\, 1. \tag{1.13}$$

If the match is not good enough the component with smallest $W_{t,i}/V_{t,i}$ is discarded and replaced with a new component with some default weight and variance while the mean is set to I_t. This newly created component is from here on considered the matched component J_t. All other components, $i \neq J_t$, that was not matched, are updated using

$$B_{t,j} = B_{t-1,j}, \tag{1.14}$$

$$V_{t,j} = V_{t-1,j}, \tag{1.15}$$

$$W_{t,j} = cW_{t-1,j} + (1-c)\, 0. \tag{1.16}$$

To produce the binary foreground image, F_t, some threshold α is used, that specifies the minimum amount of time the background is assumed visible in each pixel. The components are sorted by decreasing $\frac{W_{t,j}}{\sqrt{V_{t,j}}}$ and all components k for which,

$$\sum_{j=1}^{k-1} W_{t,j} < \alpha \tag{1.17}$$

are considered background components and the rest is foreground components. Finally $F_t(x)$ is set to 1 if $I_t(x)$ was matched to a foreground component and it is set to 0 if $I_t(x)$ was matched to a background component.

Friedman and Russel [17] have suggested a very similar algorithm, especially targeted for traffic surveillance. They use $n = 3$ components and assume that they represent road, shadow and vehicle. All observations are assumed to originate form one of those three classes and thus the concept of a good enough match is not needed, e.g. the best matching component is always updated and components are never replaced. Also, the threshold α, that specifies the minimum amount of time the background is assumed visible in each pixel, is not needed. Instead the component with smallest mean is assumed to be the shadow component and among the two other components, the one with smallest variance is assumed to be the road component. This way each pixel is classified into road, shadow or vehicle and a binary foreground mask can be produced by letting road and shadow pixels be classified as background and vehicle pixels as foreground.

To allow the number of components n to vary and be automatically estimated in each frame in Zivkovic [93] has suggested to assume a *Dirichlet* prior with negative coefficients, ρ_j, on the weights, $W_{t,j}$, i.e.

$$f\left(W_{t,1}, W_{t,2}, \cdots, W_{t,n}\right) = \frac{1}{a} \prod_j W_{t,j}^{\rho_j}, \tag{1.18}$$

where a is a normalising constant making sure it integrates to 1 and $\rho_j = -\rho$ is some parameter that controls how much evidence must be available before a component is considered visible. That leads to an additional term being introduced into the the weight update equations. That is Equation 1.13 is replaced by

$$W_{t,j} = cW_{t-1,j} + (1-c)\,1 - (1-c)\,c_T \tag{1.19}$$

and Equation 1.16 is replaced by

$$W_{t,j} = cW_{t-1,j} + (1-c)\,0 - (1-c)\,c_T, \tag{1.20}$$

where c_T is some constant parameter. This update will suppress components that are not supported by recent data and when the weight becomes negative the component is discarded. When a new input pixel cannot be matched to the current set of components a new component is added instead of removing the one with smallest weight. At each iteration the remaining weights has to normalised to sum to one.

Faster initialisation

For typical surveillance situations, the learning factor, c, is chosen fairly large to allow objects to stand still without affecting the background model too much. That means that the system will need a fair amount of initialisation time after it has been started before it delivers decent data. When deploying such a system this might not be a problem, since the system is typically installed and started once and then it will be running continuously for a long time. But there are some situation, for example during the development of the system, when it is restarted more frequently. In those cases it is of interest to shorten the initialisation time. It can be achieved by replacing the learning factor c by a time dependent learning factor $c_t = \min\left(1 - 1/t, c\right)$. This means that during initialisation the model parameters will be estimated as the mean over the frames observed so far, with equal weights, and after long time the system will react exactly as before.

Pan-tilt-zoom Cameras

Hayman and Eklundh [25] extends the mixture model to handle pan-tilt rotations of the camera. This is of interest also for static surveillance cameras as strong wind might cause them to move slightly. They let the background image B_t be a large mosaic image and each new input frame I_t is registered to this mosaic image by identifying feature points. If this registration could be done perfectly, the above algorithm could be applied as it is. But that is not the case as there exists imperfections in the camera model and the camera might undergo subpixel motions.

The background model is still built up using the above registration method, ignoring the errors, which might cause a slightly blurred model. The difference lies in the matching of pixels to background models. Let $\hat{I}_t$ be the registered version of I_t. The observed pixel $\hat{I}_t(\mathbf{x})$ at location $\mathbf{x}$ is considered to be an observation of a linear combination of its neighbouring pixels, $\mathbf{x}_k$, with unknown coefficients a_k,

$$\hat{I}_t(\mathbf{x}) = \sum_k a_k B_t(\mathbf{x}_k). \tag{1.21}$$

The coefficients a_k are assumed to be Dirichlet distributed with known parameters and are eliminated by marginalisation (integrated out). This results in the so called *mixel distribution*. It is approximated with a Gaussian distribution with matching mean and variance, and the rest of the algorithm is used as presented above with the mean and the variance of the Gaussians replaced with the mean and the variance from this mixel distribution. The order of the different components of the neighbouring pixel are assumed to be the same and no mixing between different components are made.

Less influence of from foreground objects

Harville [23] has suggested to use feedback from higher level modules of a system to alter the updating of a Gaussian mixture model. The idea is that after some higher level processing have been done, there might be some pixels that are guaranteed to contain foreground and some that are guaranteed to contain background. The mixture models of pixels guaranteed to contain foreground are not updated at all, while the mixture models of pixels guaranteed to contain background but detected as foreground is updated at an accelerated pace.

Huwer and Niemann [28] has suggested to use frame differentiating to detect areas that are very unlikely to contain background, and don't update the background model in those areas. They use a short time learning factor, c, on $\Delta I_t = |I_t - I_{t-1}|$ to produce $D_t = cD_{t-1} + (1-c)\Delta I_t$. It is then thresholded to detect pixels that have changed a lot during the last few frames and the background model at those pixels are not updated. The output of the segmentation algorithm could have been used in a similar way to only update areas where background were detected. But this kind of feedback might enlarge errors made by the detector. That is, an area erroneously detected as background will be even more likely to be detected as background in the next frame as the background model have been updated in that direction, and similar for erroneously detected foreground. The suggested approach decouples the background updating from its own detections, which means that errors made by the detector will not be enlarged in later frames. However, in traffic surveillance, the main influence of foreground objects on the background model is when a car stops at a red light and remains motionless for a few minutes. In that case this approached will not help.

An offline approach were suggested by Farin *et al* [16], who groups the image into blocks and considers each block separately. A motion estimation based on block matching is performed and blocks detected as moving are declared foreground. For each block $B_t \subset I_t$, a distance matrix, with elements $d_{t_1,t_2} = \frac{1}{|B_{t_1}|}\sum_{\mathbf{x}} |B_{t_1}(\mathbf{x}) - B_{t_2}(\mathbf{x})|$, over all frames is created. The intensity value are assumed to be $0 \leq B_t(\mathbf{x}) \leq 1$. From this matrix the subset of frames, $\mathcal{T}$, where the block is background is found by minimising

$$\sum_{t_1,t_2 \in \mathcal{T}} d_{t_1,t_2} + \sum_{t_1 \notin \mathcal{T} \text{ or } t_2 \notin \mathcal{T}} (1 - d_{t_1,t_2}) \tag{1.22}$$

over $\mathcal{T}$ under the constraint that the blocks detected as moving may not be placed in $\mathcal{T}$. The background model can then be built from the frames $\mathcal{T}$ only.

1.2.4 Kernel density estimation

Kernel density estimation is a technique to estimate a probability distribution function from a set of samples without making assumptions about which class of distribution they belong too. It is described in Section 1.3.1. Elgammal *et al* [15] has suggested to use this idea to estimate the background distribution from a set of n recent observations, I_{s_k} by

$$f_{\text{bg}}\left(I_t \left| I_{s_1}, \cdots, I_{s_n}\right.\right) = \frac{1}{n}\sum_{k=1}^{n} \mathcal{N}\left(I_t \left| I_{s_k}, \sigma^2\right.\right). \tag{1.23}$$

The kernel bandwidth σ has to be estimated and it corresponds to the noise level. This technique applies to colour images as well. In that case the covariance matrix is assumed diagonal and the noise level estimated separately for each channel. The estimate is performed by looking at the median, m, of differences between consecutive frames, $|I_t - I_{t-1}|$. This difference is assumed to be samples from twice the noise level since both I_t and I_{t-1} contain noise. This gives the noise level

$$\sigma = \frac{m}{\mathcal{N}_{\text{cdf}}^{-1}(3/4)\sqrt{2}} \approx \frac{m}{0.9539}, \tag{1.24}$$

where $\mathcal{N}_{\text{cdf}}^{-1}(x)$ is the inverse of the normal cumulative distribution function.

Elgammal *et al* uses two background distributions form in this way. One long term model that is formed over all observed intensity values over a long time frame and one short term model that is formed over recent observations classified as background only. For a pixel to be detected as foreground, its likelihood has to be larger than some threshold in both distributions. Also, pixels detected as foreground in the short term model adjacent to pixel detected as foreground in both models are classified as foreground. A post processing step is used that removes connected segments from the foreground where most pixels in the segment matches the background model of a neighbouring pixel. This allows the system to operate even if the camera is moving slightly.

Another approach were suggested by Sheikh and Shah [69]. They considered each pixel in each frame, $I_t\left(x, y\right)$ a sample from some distribution over the joint colour and position space indexed by (x, y, r, g, b), and estimated a probability distribution over this five dimensional space using kernel density estimation. That results in a system than can handle small motions of the camera. They build a background distribution, f_{bg}, in this way from all pixels from the last n frames. They also build a foreground distribution as a mixture between a uniform distribution and a kernel density estimation formed over all pixels classified as foreground during the last m frames, with $m << n$.

Mittal and Paragios [57] have suggested a background foreground algorithm where the kernel bandwidth depends on the data. To use their approach the features used has to be measure together with some uncertainty of the measure. They use normalised rgb and optical flow as features and derive covariance matrices representing the uncertainty of the measurement uncertainty by assuming a known constant noise level and independence between the r, g, and b channels of the image. The kernel used is a Gaussian kernel with covariance matrix equal to the sum of the covariance matrix from the measurement made in the current frame and covariance matrix of the sample in the background model.

1.2.5 Post processing

Markov random field

When both a foreground distribution, f_{fg}, and a background distribution, f_{bg}, is available, they can be compared to form a binary background foreground segmentation, $F_t(\mathbf{x})$, that is one if $\mathbf{x}$ is classified as foreground and zero if it is background. The Bayesian approach is to estimate the probability of foreground as

$$\begin{aligned} p\left(F_t = 1 \left| I_t \right.\right) = \frac{f\left(I_t \left| F_t = 1\right.\right) p\left(F_t = 1\right)}{f\left(I_t\right)} &= \\ &= \frac{f_{\text{fg}}\left(I_t\right) p\left(F_t = 1\right)}{f_{\text{fg}}\left(I_t\right) p\left(F_t = 1\right) + f_{\text{bg}}\left(I_t\right) p\left(F_t = 0\right)}, \end{aligned} \quad (1.25)$$

and then assume some prior distribution for F_t. A maximum a posterior (MAP) estimate of F_t is found by maximising this probability. If a uniform prior, $p\left(F_t = 1\right) = p\left(F_t = 0\right) = 0.5$, is assumed, it factors out and cancels. If f_{fg} is assumed uniform this MAP estimation will be a thresholding of the background likelihood as above. Here each pixel is treated separated as they are considered independent. The joint distribution over all pixels would be,

$$f\left(F_t(\cdot)\,|I_t(\cdot)\,\right) = \prod_{\mathbf{x}} p\left(F_t(\mathbf{x})\,|I_t(\mathbf{x})\,\right). \tag{1.26}$$

To incorporate a smoothness constraint claiming that neighbouring pixels are more likely to belong to the same class than to different classes, a Markov random filed can be used as prior. Such priors are formed by choosing some neighbouring structure, $\mathcal{N}$, and considering all pairs $(\mathbf{x}, \mathbf{y}) \in \mathcal{N}$ of neighbouring pixels $\mathbf{x}$ and $\mathbf{y}$. For each such pairs there is 4 possible outcomes of the foreground segmentation, $(F_t(\mathbf{x}), F_t(\mathbf{y}))$, namely $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$. By placing a higher likelihood on $(0,0)$ and $(1,1)$ than on $(0,1)$ and $(1,0)$, smoother segmentations that contains few transitions between background and foreground will be preferred over segmentations with a lot of small segments.

In this case the the pixels will no longer be independent and the joint distribution over all pixels has to be considered. This was suggested by Sheikh and Shah [69], who uses a Ising model prior as the smoothness constraint. It has a single parameter λ specifying how regular the foreground objects are expected to be, and is formed by a product over all neighbouring pixels $\mathbf{x}$ and $\mathbf{y}$.

$$f\left(F_t(\cdot)\right) = \frac{1}{k} \prod_{(\mathbf{x},\mathbf{y}) \in \mathcal{N}} e^{\lambda(F_t(\mathbf{x})F_t(\mathbf{y}) - (1-F_t(\mathbf{x}))(1-F_t(\mathbf{y})))}, \tag{1.27}$$

where k is some constant making sure $\sum_{F_t(\cdot)} f\left(F_t(\cdot)\right) = 1$. The sum is taken over all possible segmentations $F_t(\cdot)$. The probability distribution of the observed image can be found by assuming the pixel intensities, I_t, to be conditional independent given the segmentation F_t,

$$f\left(I_t(\cdot)\,|F_t(\cdot)\,\right) = \prod_{\mathbf{x}} f_{\text{fg}}\left(I_t(\mathbf{x})\right)^{F_t(\mathbf{x})} f_{\text{bg}}\left(I_t(\mathbf{x})\right)^{1-F_t(\mathbf{x})}. \tag{1.28}$$

Bayes Law gives the posterior distribution

$$f\left(F_t(\cdot)\,|I_t(\cdot)\right) = \frac{1}{k} f\left(I_t(\cdot)\,|F_t(\cdot)\right) f\left(F_t(\cdot)\right), \tag{1.29}$$

where k is some constant making sure $\sum_{F_t(\cdot)} f\left(F_t(\cdot)\,|I_t(\cdot)\,\right) = 1$. A maximum a posterior estimate of $F_t(\cdot)$ can then be found by optimising this likelihood over all possible segmentations $F_t(\cdot)$, which can be done very efficiently using dynamic graph cuts [45].

Connected Segments

Another approach is to extract connected segments from the binary foreground background segmentation image, and assume that they represent separate objects. Then use some prior knowledge about the objects that are typically observed in the scene to discard connected segments that does not fit this prior knowledge.

One typical feature to use here is the size of the connected segment and discard small connected segments. A more general heuristics were suggested by Javed *et al* [35] that assumed that the image grey level gradient is high at object borders. They use a Gaussian mixture model to model the image intensity in the same way as is presented above, and using it connected segments are extracted. The parameters of this model are estimated as above and from those parameters they calculate the parameters of a Gaussian mixture model describing the image grey level gradients (no extra parameters have to be measured). The gradients mixture model is used to classify the border pixels of each connected segment as foreground or background. If a significant part of a connected segments' border is classified as background by these gradient features it is discarded. By also requiring that the grey level gradient magnitude along the border has to be large they can discard connected segments arising from new background becoming visible when a previously stationary objects starts moving.

1.2.6 Lighting variations

One major problem with these methods is that on cloudy, windy days, the changing cloud cover causes the lighting conditions to vary faster than the algorithm adopts and thus large cloud shadows turn up as foreground objects. Also, shadows cast by the objects themself is typically detected as foreground, which might be undesirable. One way to attack that problem is to build a background model not on the intensity values directly but on some more lighting independent features. For Colour images different colour spaces can be used. One common example [57, 19, 15] is normalised rgb which replaced the r, g and b intensity values with $\frac{r}{r+g+b}$, $\frac{g}{r+g+b}$ and $\frac{b}{r+g+b}$. In some cases removing all intensity information like this is considered too much and the luminance feature $y = r + g + b$ is also introduced, and large enough variations in this parameters is also classified as foreground. Other features that also is intensity independent is optical flow [57, 62] and stereo disparity (depth) [19].

Colour spaces

Kumar *et al* [49] have compared how a background foreground system that estimates a single Gaussian and its variance as background model performs using five different colour spaces. Some heuristics for shadow detection, different for each colour space were also used. The colour spaces compared were RGB, XYZ, YCbCr, HSV and normalised RGB. The first three are related with linear mapping and should theoretically give the same performance if complete covariance matrices are estimated. Often the covariance matrices are assumed to be diagonal though, and this is the case in [49] as well. In that case YCbCr gives the best result because these colours are most independent and thus the diagonal covariance matrix approximation fits the real covariance matrix better.

1.2.7 Multiple different models

Hu *et al* [27] has suggested to use 3 different models to describe different parts of the background with different properties. Each pixel $\mathbf{x}$ in the image is associated with a patch, $P_{\mathbf{x}}$, centred around the pixel. This patch is classified into one of sketchable, textured or flat. The classification is performed by first identifying the sketchable patches. Then among the remaining patches a threshold on the intensity spacial variance over the patch is used to separate the textured (high variance) patches from the flat (low variance).

For the sketchable patches a set of predefined prototypes, Q_i, is used and consists of for example bars, edges and L-junctions rotated into 8 different orientations. They are binary masks consisting of the values -1 and 1. The feature response, r of a prototype on a image patch is defined as $\sum_{\mathbf{y}} Q_i(\mathbf{y}) P_{\mathbf{x}}(\mathbf{y})$. Each sketchable patch is assigned to a prototype and the response r for this combination is assumed laplacian distributed. The parameters of this distribution is continuously updated using a learning factor.

For the flat patches, they are modelled with a single mixture of Gaussians for the entire patch in the same manner as single pixels were modelled above. For the textured patches the histogram, $\mathbf{h}$, over some modified local binary patterns are used as features. The temporal mean $\mathbf{m}$ over those histograms is maintained using a learning factor. The difference between two histograms is defined as

$$d\left(\mathbf{h}, \hat{\mathbf{h}}\right) = \sum_i \min\left(h_i, \hat{h}_i\right). \tag{1.30}$$

A variability measure, s, is maintained using a learning factor on $d(\mathbf{h}, \mathbf{m})$, and the classification into foreground or background is performed by thresholding $d(\mathbf{h}, \mathbf{m})/s$.

1.2.8 Discussion

Manny intensity independent features break down in dark areas. Take for example the normalised rgb. When r, g and b all are small, the denominator becomes close to zero thus the noise is scaled up out of proportion. Gordon *et al* [19] has suggested to ignore normalised rgb features in dark areas and there rely on other features instead. In their case the results from a stereo matching algorithm. Some fix threshold were used to decide if the features were reliable or not. In the same fashion Hu *et al* [27] used 3 different models for background patches with different amount of structures. Also, Wayne and Schoonees [86] suggests to use two thresholds on the background likelihood to classify pixel into background, foreground and unknown depending on how close the background model the current frame is. The unknown pixels are then filled in a morphological post processing step based on their neighbours.

All features are unreliable in some cases, such as when there is no structure or the foreground happens to be fairly close to the background. However many features can be very reliable in other cases when there is a lot of structure or a big difference between the foreground object and the background. This property of sometimes being unreliable and sometimes being very reliable is not a discrete property. It is a property that varies continuously from for example a lot of structure to uniform. This can be utilised much more efficiently by instead of thresholding the features into reliable and not reliable, using a continuous estimate of how reliable they are and weight the different features accordingly.

In this thesis it is suggested to use normalised cross correlation. It is independent both to translations and scaling of the intensity domain, which makes it lighting independent for the ideal camera. The background and foreground distribution for this feature is derived in Chapter 2 and it depends on a single parameter, the signal to noise ratio. Here the signal refers to the amount of structure in the patch. This makes it possible to use this feature for all patches even if the amount of structure is low. In that case the foreground probability will be close to 0.5 and represent an uncertain state. The segmentation will then rely more on other features or on neighbours using Markov random fields. This means that there will be no need to chose between several different distinct models. Instead the signal to noise ratio is measured and a single parametrised model will move continuously from being very certain about highly structured patches to being very unsure about uniform patches.

1.3 Tracking

Many classical tracking algorithms are based on continuous state space methods. In this case a state space, $\mathbb{S}$, is constructed, typically chosen to be $\mathbb{R}^n$. The elements of this space, $\mathbf{q} \in \mathbb{S}$, are vectors representing the state of the object being tracked. The coordinates of those vectors represents properties like the position, speed and orientation of the object. A discrete time stochastic Markov process is constructed by considering the state of the object, $\mathbf{q}_t$, at discrete times, t, and introducing a dynamic model stating that

$$\mathbf{q}_t = \mathbf{h}\left(\mathbf{q}_{t-1}, \mathbf{w}_t\right), \tag{1.31}$$

where $\mathbf{h}\left(\cdot\right)$ can be almost any function and $\mathbf{w}_t \in \mathbb{R}^n$ is the model noise process. At each time step observations, $\mathbf{o}_t \in \mathbb{R}^m$, are made, from some observations function

$$\mathbf{o}_t = \mathbf{o}\left(\mathbf{q}_t, \mathbf{v}_t\right), \tag{1.32}$$

with observation noise $\mathbf{v}_t \in \mathbb{R}^m$. Note that the dimension of the state space does not have to be the same as the dimension of the observation space, e.g. typically $n \neq m$. An equivalent way of formulating this model is to instead of specifying $\mathbf{h}\left(\cdot\right)$ and $\mathbf{o}\left(\cdot\right)$, specifying the probability distribution functions $f\left(\mathbf{q}_t \left|\mathbf{q}_{t-1}\right.\right)$ and $f\left(\mathbf{o}_t \left|\mathbf{q}_t\right.\right)$. The relationship between the two formulations can be expressed using an inverse function $\mathbf{w}_t = \tilde{\mathbf{h}}\left(\mathbf{q}_{t-1}, \mathbf{q}_t\right)$ that together with $\mathbf{q}_t = \mathbf{h}\left(\mathbf{q}_{t-1}, \mathbf{w}_t\right)$ forms a bijective mapping between $\mathbf{w}_t$ and $\mathbf{q}_t$ for any given $\mathbf{q}_{t-1}$. Also, an inverse function $\mathbf{v}_t = \tilde{\mathbf{o}}\left(\mathbf{q}_t, \mathbf{o}_t\right)$ that together with $\mathbf{o}_t = \mathbf{o}\left(\mathbf{q}_t, \mathbf{v}_t\right)$ forms a bijective mapping between $\mathbf{v}_t$ and $\mathbf{o}_t$ for any given

$\mathbf{q}_t$ is needed. If these two bijections exists and are differentiable, the two formulations are related by [74]

$$f\left(\mathbf{q}_t\left|\mathbf{q}_{t-1}\right.\right) = f_w\left(\tilde{\mathbf{h}}\left(\mathbf{q}_{t-1}, \mathbf{q}_t\right)\right)\left|\frac{\mathrm{d}\tilde{\mathbf{h}}}{\mathrm{d}\mathbf{q}_t}\right|, \tag{1.33}$$

$$f\left(\mathbf{o}_t\left|\mathbf{q}_t\right.\right) = f_v\left(\tilde{\mathbf{o}}\left(\mathbf{o}_t, \mathbf{q}_t\right)\right)\left|\frac{\mathrm{d}\tilde{\mathbf{o}}}{\mathrm{d}\mathbf{o}_t}\right|, \tag{1.34}$$

where $f_w(\cdot)$ and $f_v(\cdot)$ are the probability distributions of the noise processes $\mathbf{w}_t$ and $\mathbf{v}_t$ respectively. The tracking problem is then often stated as

Problem 1.1. *Given* **past** *observation from time* 0 *to time* t*, denoted* $\mathcal{O}_{0\cdots t} = \{\mathbf{o}_0, \cdots, \mathbf{o}_t\}$*, find the expected value of the probability distribution of the* **current state***,* $f\left(\mathbf{q}_t\left|\mathcal{O}_{0\cdots t}\right.\right)$*,*

$$\hat{\mathbf{q}}_t = \int_{\mathbb{S}} \mathbf{q} f\left(\mathbf{q}\left|\mathcal{O}_{0\cdots t}\right.\right) d\mathbf{q}. \tag{1.35}$$

In this thesis the tracking problem will instead be formulated in a somewhat more ambitions way.

Problem 1.2. *Given* **all** *(past and future) observations from time* 0 *to time* $\tau \to \infty$*, denoted* $\mathcal{O}_{0\cdots\tau}$*, find the* **state sequence***,* $\mathcal{Q}^*_{0\cdots\tau} = (q^*_0, \ldots, q^*_\tau)$*, that maximises the likelihood*

$$f\left(\mathcal{Q}_{0\cdots\tau}\left|\mathcal{O}_{0\cdots\tau}\right.\right). \tag{1.36}$$

That is, instead of the classical solution of using a filter to smooth the observed data, we want to optimise over all possible sequences of states. This chapter will however begin with an overview of the classical methods.

1.3.1 Single target

Kalman Filter

If $\mathbf{h}(\cdot)$ and $\mathbf{o}(\cdot)$ are linear functions and the noise processes, $\mathbf{w}_t$ and $\mathbf{v}_t$ are Gaussian, the solution can be found algebraically using the Kalman filter [39]. A good introduction to Kalman filters can be found in [87]. The dynamics, $\mathbf{h}(\cdot)$, is in this case a linear transformation of the previous state with coefficient matrix $\mathbf{H}$, and additive Gaussian noise, $\mathbf{w}_t$, with mean 0 and covariance matrix $\mathbf{Q}$, i.e

$$\mathbf{q}_t = \mathbf{H}\mathbf{q}_{t-1} + \mathbf{w}_t, \tag{1.37}$$

$$f_w\left(\mathbf{w}_t\right) = \mathcal{N}\left(\mathbf{w}_t\left|0, \mathbf{Q}\right.\right). \tag{1.38}$$

The observation is defined as a linear transformation of the current state with coordinate matrix $\mathbf{O}$, and additive Gaussian noise, $\mathbf{v}_t$, with mean 0 and covariance matrix $\mathbf{R}$, i.e

$$\mathbf{o}_t = \mathbf{O}\mathbf{q}_t + \mathbf{v}_t, \tag{1.39}$$

$$f_v(\mathbf{v}_t) = \mathcal{N}(\mathbf{v}_t \,|0, \mathbf{R}) . \tag{1.40}$$

Using the equivalent formulation mentioned above, the same model can be expressed as

$$f(\mathbf{q}_t \,|\mathbf{q}_{t-1}) = \mathcal{N}(\mathbf{q}_t \,|\mathbf{H}\mathbf{q}_{t-1}, \mathbf{Q}) , \tag{1.41}$$

$$f(\mathbf{o}_t \,|\mathbf{q}_t) = \mathcal{N}(\mathbf{o}_t \,|\mathbf{O}\mathbf{q}_t, \mathbf{R}) . \tag{1.42}$$

The probability distribution for time $t = 0$ is assumed to be to be Gaussian, $f(\mathbf{q}_0 \,|\mathbf{o}_0) = \mathcal{N}(\mathbf{q}_0 \,|\hat{\mathbf{q}}_0, \mathbf{P}_0)$ with known parameters $\hat{\mathbf{q}}_0$ and $\mathbf{P}_0$.

The algorithm then propagates the previous distribution function, $f(\mathbf{q}_{t-1} \,|\mathcal{O}_{0\cdots t-1}) = \mathcal{N}(\mathbf{q}_{t-1} \,|\hat{\mathbf{q}}_{t-1}, \mathbf{P}_{t-1})$ forward in time using the dynamic model and thereby gets a prediction, $\hat{\mathbf{q}}_t^-$, of where the object is located at the current time t before the current observation have been considered,

$$f(\mathbf{q}_t \,|\mathcal{O}_{0\cdots t-1}) = \mathcal{N}\left(\mathbf{q}_t \,\middle|\hat{\mathbf{q}}_t^-, \mathbf{P}_t^-\right) , \tag{1.43}$$

where

$$\hat{\mathbf{q}}_t^- = \mathbf{H}\hat{\mathbf{q}}_{t-1}, \tag{1.44}$$

$$\mathbf{P}_t^- = \mathbf{H}\mathbf{P}_{t-1}\mathbf{H}^T + \mathbf{Q}. \tag{1.45}$$

This prediction is then compared with the current observation and the *innovation*, $\mathbf{d}_t = \mathbf{o}_t - \mathbf{O}\hat{\mathbf{q}}_t^-$, together with the *Kalman gain*, $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{O}^T\left(\mathbf{O}\mathbf{P}_t^-\mathbf{O}^T + \mathbf{R}\right)^{-1}$, is used to include the additional information of the observation in the current frame, resulting in

$$f(\mathbf{q}_t \,|\mathcal{O}_{0\cdots t}) = \mathcal{N}(\mathbf{q}_t \,|\hat{\mathbf{q}}_t, \mathbf{P}_t) , \tag{1.46}$$

where

$$\hat{\mathbf{q}}_t = \hat{\mathbf{q}}_t^- + \mathbf{K}_t\mathbf{d}_t, \tag{1.47}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{O})\,\mathbf{P}_t^-. \tag{1.48}$$

Extended Kalman Filter

If the dynamics or observations are not linear, approximative solutions can be found by linearising around the current state. This is achieved by letting $\mathbf{H}$ and $\mathbf{O}$ be the jacobians of the corresponding functions, and is called the extended Kalman filter or EKF [36]. $\mathbf{H}$ and $\mathbf{O}$ will now vary over time.

The model would in this case be very general,

$$\mathbf{q}_t = \mathbf{h}(\mathbf{q}_{t-1}, \mathbf{w}_t) , \tag{1.49}$$

$$\mathbf{z}_t = \mathbf{o}(\mathbf{q}_t, \mathbf{v}_t) . \tag{1.50}$$

The function $\mathbf{h}(\cdot)$ is linearised around $(\hat{\mathbf{q}}_{t-1}, \mathbf{0})$, and its jacobian is split into two matrices $\mathbf{H}_t$ and $\mathbf{W}_t$ corresponding to the $\mathbf{q}_{t-1}$ and $\mathbf{w}_t$ variables respectively,

$$\Delta\mathbf{h}\left|_{(\hat{\mathbf{q}}_{t-1},\mathbf{0})}\right. = \begin{pmatrix} \mathbf{H}_t & \mathbf{W}_t \end{pmatrix}. \tag{1.51}$$

The prediction equations becomes

$$\hat{\mathbf{q}}_t^- = \mathbf{h}(\hat{\mathbf{q}}_{t-1}, 0), \tag{1.52}$$

$$\mathbf{P}_t^- = \mathbf{H}_t\mathbf{P}_{t-1}\mathbf{H}_t^T + \mathbf{W}_t\mathbf{Q}_t\mathbf{W}_t^T. \tag{1.53}$$

The function $\mathbf{o}(\cdot)$ can then be linearised around this predicted value, and as before, the jacobian is split into to parts corresponding to the the $\mathbf{q}_t^-$ and $\mathbf{v}_t$ variables respectively,

$$\Delta\mathbf{o}\left|_{(\hat{\mathbf{q}}_{t-1}^-,\mathbf{0})}\right. = \begin{pmatrix} \mathbf{O}_t & \mathbf{V}_t \end{pmatrix}. \tag{1.54}$$

The innovation becomes $\mathbf{d}_t = \mathbf{o}_t - \mathbf{o}\left(\hat{\mathbf{q}}_t^-, \mathbf{0}\right)$, and the Kalman gain

$$\mathbf{K}_t = \mathbf{P}_t^-\mathbf{O}_t^T\left(\mathbf{O}_t\mathbf{P}_t^-\mathbf{O}_t^T + \mathbf{V}_t\mathbf{R}\mathbf{V}_t^T\right)^{-1}. \tag{1.55}$$

Finally, the update equations are the same as before

$$\hat{\mathbf{q}}_t = \hat{\mathbf{q}}_t^- + \mathbf{K}_t\mathbf{d}_t, \tag{1.56}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{O}_t)\mathbf{P}_t^-. \tag{1.57}$$

This only gives approximative solutions and might fail to generate usable results altogether.

Probabilistic Data Association Filter

Using for example the detections from an object detector as the observation in a Kalman filter might be tricky, even if there is only one object present. The detector might generate false detections on the background, often called *clutter*, and/or it might not detect the true object. The later can be handled by only using the prediction step and letting $\hat{\mathbf{q}}_t = \hat{\mathbf{q}}_t^-$ and $\hat{\mathbf{P}}_t = \hat{\mathbf{P}}_t^-$. If there is only a few false detections, it might be possible to remove the false detections by simply choosing the detection closest to the predicted observation, $\mathbf{o}_t^- = \mathbf{O}\mathbf{q}_t^-$. In heavy clutter, e.g. a lot of false detection, this does not work because the probability of choosing one of the false detections instead of the correct detection will be too big.

Instead, all detections sufficiently close to $\mathbf{o}_t^-$ has to be considered. This is done by the Probabilistic Data Association Filter, PDAF [3]. The distribution of the detection, $\mathbf{o}_t$, given the prediction $\mathbf{o}_t^-$ is also Gaussian (or approximated as Gaussian in the non-linear case), with covariance $\mathbf{S}_t = \mathbf{O}\mathbf{P}_t^-\mathbf{O}^T + \mathbf{R}$,

$$f\left(\mathbf{o}_t \left|\mathbf{o}_t^-, \mathbf{S}_t\right.\right) = \mathcal{N}\left(\mathbf{o}_t \left|\mathbf{o}_t^-, \mathbf{S}_t\right.\right). \tag{1.58}$$

The PDAF algorithm uses a constant γ and considers all detections for which $\mathbf{d}_t^T \mathbf{S}_t^{-1} \mathbf{d}_t \leq \gamma$, with $\mathbf{d}_t = \mathbf{o}_t - \mathbf{o}_t^-$ the innovation as before. This criteria forms an ellipse outside which the observations are discarded. This ellipse is often called a *validation gate*. The probability of the true observation, i.e. a sample from the distribution function (1.58), being discarded is $\alpha_1 = F_{\chi^2}(\gamma)$, where $F_{\chi^2}(\cdot)$ is the χ^2 cumulative distribution function. Typically the γ parameter is chosen by calculating it from a desired value of α_1 using this relation.

There will now be n_t observation in frame t. They will be denoted $\mathbf{o}_{t,i}$ for $i = 1, 2, \cdots, n_t$, and the set of all of them is denoted $\mathcal{O}_{t,*} = \{\mathbf{o}_{t,1}, \mathbf{o}_{t,1}, \cdots, \mathbf{o}_{t,n_t}\}$. The corresponding innovations are denoted $\mathbf{d}_{t,i} = \mathbf{o}_{t,i} - \mathbf{o}_t^-$. The probability distribution of those observations given that they originated from the tracked object is a truncated Gaussian distribution, since observation outside the validation gate have been discarded,

$$f\left(\mathbf{o}_{t,i} \left| \mathbf{o}_t^-, \mathbf{S}_t\right.\right) = \begin{cases} \frac{1}{1-\alpha_1} \mathcal{N}\left(\mathbf{o}_{t,i} \left| \mathbf{o}_t^-, \mathbf{S}_t\right.\right) & \mathbf{d}_{t,i}^T \mathbf{S}_t^{-1} \mathbf{d}_{t,i} \leq \gamma \\ 0 & \text{otherwise} \end{cases} . \tag{1.59}$$

The PDAF is derived by assuming that i) the false detections are uniformly distributed over the image with area v, ii) the detector will fail to detect the object with probability α_2, and iii) a single object will never generate more than one detection. Under these assumptions, there is $n_t + 1$ mutually exclusive events, $\mathcal{A}_i$, with $i = 0, 1, \cdots, n_t$. The event $\mathcal{A}_0$ is that the object is not detected at all within the validation gate, and the event $\mathcal{A}_i$, $i > 0$, is that $\mathbf{o}_{t,i}$ is the correct detection. The likelihood of those events, $f\left(\mathcal{A}_i \left| \mathcal{O}_{t,*}, \mathbf{o}_t^-\right.\right)$, is (see [3] for details)

$$\beta_{t,i} = \frac{1}{c} \begin{cases} f\left(\mathbf{o}_{t,i} \left| \mathbf{o}_t^-\right.\right) & i > 0 \\ \frac{n_t}{v} \frac{\alpha_1 + \alpha_2 - \alpha_1 \alpha_2}{(1-\alpha_1)(1-\alpha_2)} & i = 0 \end{cases} , \tag{1.60}$$

where c is a normalising constant making sure that $\sum_{i=0}^{n_t} \beta_{t,i} = 1$. A synthesised innovation is then calculated as the weighted mean over all innovations,

$$d_t = \beta_{t,0} 0 + \sum_{i=1}^{n_t} \beta_{t,i} \mathbf{d}_{t,i}. \tag{1.61}$$

The innovation of the event that the object is not detected is set to 0, which would correspond to using the prediction only. This synthesised innovation is then used by the Kalman filter to update the mean estimate, $\hat{\mathbf{q}}_t$ and its covariance matrix, $\mathbf{P}_t$. Finally the covariance matrix, $\mathbf{P}_t$ is increased to incorporate the additional uncertainty introduced by no longer having a single observation that with probability 1 comes form the observation model. This means that $\mathbf{P}_t$ has to be replaced by $\mathbf{P}_t + \mathbf{P}_t'$, with

$$\mathbf{P}_t' = \mathbf{W}_t \left(\sum_{i=1}^{n_k} \beta_{t,i} \mathbf{d}_{t,i} \mathbf{d}_{t,i}^T - \mathbf{d}_t \mathbf{d}_t^T \right) \mathbf{W}_t^T , \tag{1.62}$$

$$\mathbf{W}_t = \mathbf{P}_t^- \mathbf{O}^T \mathbf{S}_t^{-1}. \tag{1.63}$$

Monte-Carlo Integration

In many situations approximating distributions as Gaussian is too much of an approximation. Especially in cases where there is need to represent multiple hypothesis. In those cases multi modal distributions are needed, and very often the distributions become too complicated to handle analytically. In those cases other approximations are needed. Monte-Carlo integration is a technique to calculate expected values from any distribution it is possible to draw samples from. The expected value of some function $g()$ of a stochastic variable $\mathbf{q}$, denoted $\mathcal{E}(g(\mathbf{q}))$, with probability distribution $f(\mathbf{q})$ is approximated using

$$\mathcal{E}(g(\mathbf{q})) = \int g(\mathbf{q}) f(\mathbf{q}) \, \mathrm{d}q \approx \frac{1}{n} \sum_{i=1}^{n} g(\mathbf{q}_i), \tag{1.64}$$

where $\mathbf{q}_i$ are independent samples from $f(\mathbf{q})$. This approximation will converge to the expected value in mean square error sense as $n \to \infty$.

Importance Sampling

If the distribution of interest, $f(\mathbf{q})$ is too complicated to draw samples from or if the convergence rate of the Monte-Carlo integration is too slow, importance sampling can be used. The idea is to sample from some other distribution, $\hat{f}(\mathbf{q})$, instead and then use the same approximation on

$$\int g(\mathbf{q}) f(\mathbf{q}) \, \mathrm{d}q = \int \frac{g(\mathbf{q}) f(\mathbf{q})}{\hat{f}(\mathbf{q})} \hat{f}(\mathbf{q}) \, \mathrm{d}q \approx \frac{1}{n} \sum_{i=1}^{n} \frac{f(\mathbf{q}_i)}{\hat{f}(\mathbf{q}_i)} g(\mathbf{q}_i). \tag{1.65}$$

Here $\mathbf{q}_i$ are independent samples from $\hat{f}(\mathbf{q})$ instead. The probability distribution $\hat{f}(\mathbf{q})$ is often called the *importance function* and can be almost any distribution as long as the integrand above stays bounded, which means that $\hat{f}(\mathbf{q}) > 0$ for all $\mathbf{q}$. The convergence time of the importance sampling will be significantly faster if $\hat{f}(\mathbf{q})$ is chosen as close as possible to $g(\mathbf{q}) f(\mathbf{q})$. That will concentrate the samples to the areas where the integrand is large and thus the information gained from each sample is maximised.

Markov Chain Monte-Carlo

The above methods requires independent samples from some distribution $f_s(\mathbf{q})$. In many cases that distribution is quite complex and it might be hard to generate independent samples from it. It is however possible to construct a Markov-chain that has $f_s(\mathbf{q})$ as its stationary distribution. The samples generated from such a Markov chain will not be independent. In fact, they are typically highly correlated. But if the Markov chain is ergodic (its temporal mean equals its ensemble mean) the Monte Carlo integration approximation will still converge, although it will converge much slower because of this dependency between the samples.

For a Markov-chain to have the stationary distribution $f_s(\mathbf{q})$ this distribution must be preserved by the transition distribution, $f(\mathbf{q}_t | \mathbf{q}_{t-1})$. This means that if $\mathbf{q}_{t-1}$ is distributed according to this stationary distribution, $f_{\mathbf{q}_{t-1}}(\mathbf{q}) = f_s(\mathbf{q})$, then so is $\mathbf{q}_t$, $f_{\mathbf{q}_t}(\mathbf{q}) = f_s(\mathbf{q})$. This puts some demands on $f(\mathbf{q}_t | \mathbf{q}_{t-1})$, because $f_{\mathbf{q}_t}(\mathbf{q}_t) = f(\mathbf{q}_t | \mathbf{q}_{t-1}) f_{\mathbf{q}_{t-1}}(\mathbf{q}_{t-1})$ means that

$$f_s(\mathbf{q}_t) = f(\mathbf{q}_t | \mathbf{q}_{t-1}) f_s(\mathbf{q}_{t-1}). \tag{1.66}$$

This is often assured by making the transition distribution satisfy the the stronger *global balance condition*,

$$f(\mathbf{q}_t | \mathbf{q}_{t-1}) f_s(\mathbf{q}_{t-1}) = f(\mathbf{q}_{t-1} | \mathbf{q}_t) f_s(\mathbf{q}_t), \tag{1.67}$$

which implies Equation 1.66.

A transition distribution fulfilling this can be constructed using the *Metropolis-Hastings* algorithm [24]. It uses a proposal distribution $\tilde{f}(\mathbf{q}_t | \mathbf{q}_{t-1})$, that is sampled once every time step to suggest a movement and then the movement is accepted with probability

$$\alpha(\mathbf{q}_t, \mathbf{q}_{t-1}) = \min\left(1, \frac{\tilde{f}(\mathbf{q}_{t-1} | \mathbf{q}_t) f_s(\mathbf{q}_t)}{\tilde{f}(\mathbf{q}_t | \mathbf{q}_{t-1}) f_s(\mathbf{q}_{t-1})}\right). \tag{1.68}$$

If the movement is not accepted at time t, then $\mathbf{q}_t = \mathbf{q}_{t-1}$. The proposal distribution $\tilde{f}(\mathbf{q}_t | \mathbf{q}_{t-1})$, can be almost anything as long as the resulting Markov-chain becomes irreducible (any state can be reached from any other state) and aperiodic (there is no state with a period greater than 1). A state has period k if all cycles returning to it are multiples of k time steps. The choice of $\tilde{f}(\mathbf{q}_t | \mathbf{q}_{t-1})$ affects the convergence time quite significantly though. Also note that for the fraction in (1.68) to be defined the following statement must hold for all $\mathbf{q}_t$ and $\mathbf{q}_{t-1}$,

$$\tilde{f}(\mathbf{q}_t | \mathbf{q}_{t-1}) \neq 0 \quad \Leftrightarrow \quad \tilde{f}(\mathbf{q}_{t-1} | \mathbf{q}_t) \neq 0. \tag{1.69}$$

More about Markov Chain Monte-Carlo can be found in [71].

Particle Filter

The Monte-Carlo idea of representing an arbitrary distribution as a set of samples from it can be extended to time series and can then handle non-linear dynamics as well as non-linear observations function. This has been done by Gordon *et al* [20] who called it the *bayesian bootstrap* filter and by Kitagawa [43] who called it *Monte Carlo filtering*. The technique have been made popular for tracking algorithms in the vision community by Isard and Blake [31] who called it *condensation*. Today it is more commonly known as the *particle filter*.

The distribution of interest $f\left(\mathbf{q}_t \left|\mathcal{O}_{0\cdots t}\right.\right)$ is represented with a set of m samples called particles, $\mathcal{S}_t = \left\{\mathbf{q}_t^{(1)}, \mathbf{q}_t^{(2)}, \cdots, \mathbf{q}_t^{(m)}\right\}$. The distribution of the first frame, $\mathcal{S}_0$, is assumed to be known, and the algorithm works in the same general manner as the Kalman filter. The distribution of the previous frame, $\mathcal{S}_{t-1}$ is propagated forward to the next frame by applying some state transfer function $\mathbf{h}\left(\cdot\right)$, Equation 1.31, to each of the particles in $\mathcal{S}_{t-1}$ using random samples, $\mathbf{w}_t^{(j)}$, for the noise distribution. This forms a prediction distribution,

$$\mathcal{S}_t^- = \left\{\mathbf{q}_t^{-(j)} = \mathbf{h}\left(\mathbf{q}_t^{(j)}, \mathbf{w}_t^{(j)}\right) | j = 1\cdots m\right\}, \tag{1.70}$$

representing the distribution $f\left(\mathbf{q}_t \left|\mathcal{O}_{0\cdots t-1}\right.\right)$. One weight, $\alpha_t^{(j)}$, for each of the elements of $\mathcal{S}_t^-$ is then generated from the observation probabilities,

$$\alpha_t^{(j)} = f\left(\mathbf{o}_t \left|\mathbf{q}_t^{-(j)}\right.\right). \tag{1.71}$$

Finally, $\mathcal{S}_t$ is generated by randomly choosing a single sample from $\mathcal{S}_t^-$ m times. The weights are used to define the probabilities of choosing each particle. That is, the particle $\mathbf{q}_t^{-(j)}$ is chosen with probability

$$\frac{\alpha_t^{(j)}}{\sum_{j=1}^{m} \alpha_t^{(j)}}. \tag{1.72}$$

The same sample might be chosen several times in which case $\mathcal{S}_t$ will contain several identical particles. To follow this approach strictly, the final step would be to approximate the expected value of $f\left(\mathbf{q}_t \left|\mathcal{O}_{0\cdots t}\right.\right)$, denoted $\hat{\mathbf{q}}_t$, as the mean of the elements in $\mathcal{S}_t$, e.g. $\frac{1}{m}\sum_{j=1}^{m}\mathbf{q}_t^{(j)}$. However it is more common to estimate the mean as a weighted mean over $\mathcal{S}_t^-$,

$$\hat{\mathbf{q}}_t = \sum_{j=1}^{m} \alpha_t^{(j)} \mathbf{q}_t^{-(j)}, \tag{1.73}$$

which utilises the generated samples better.

Importance Sampled Particle Filter

The notion of importance sampling also extends naturally to time series, and the improved convergence rate means that the number of samples, m, can be kept down. This is especially useful when the dimension of the state space is high in which case a lot of particles have to be used. Some importance function $\hat{f}\left(\mathbf{q}_t\right)$ is needed. It may be almost any distribution and can depend on the observations $\mathcal{O}_{0\cdots t}$ e.g. $\hat{f}\left(\mathbf{q}_t\right) = \hat{f}\left(\mathbf{q}_t \left|\mathcal{O}_{0\cdots t}\right.\right)$. In the particle filter, the prediction step generates a set of samples, $\mathcal{S}_t^-$, from $f\left(\mathbf{q}_t \left|\mathcal{O}_{1\cdots t-1}\right.\right)$. This is with importance sampling replaced with samples, $\mathbf{q}_t^{-(j)}$, from $\hat{f}(\mathbf{q}_t)$ instead. According to the importance sampling algorithm, this set has to be weighted. Those weights can be introduced as additional factors in the α weights already used by the particle filter, e.g.

$$\alpha_t^{(j)} = \frac{f\left(\mathbf{q}_t^{-(j)} \left| \mathcal{O}_{1\cdots t-1}\right.\right)}{\hat{f}\left(\mathbf{q}_t^{-(j)}\right)} f\left(\mathbf{o}_t \left| \mathbf{q}_t^{-(j)}\right.\right). \tag{1.74}$$

The original prediction likelihood, $f\left(\mathbf{q}_t^{-(j)} \left| \mathcal{O}_{1\cdots t-1}\right.\right)$, could be calculated from $\mathcal{S}_{t-1}$ as $\sum_{j=i}^{m} f\left(\mathbf{q}_t^{-(j)} \left| \mathbf{q}_{t-1}^{(i)}\right.\right)$ but as before with the expected value, better accuracy is achieved by estimating it from $\mathcal{S}_{t-1}^{-}$,

$$f\left(\mathbf{q}_t^{-(j)} \left| \mathcal{O}_{1\cdots t-1}\right.\right) = \sum_{j=i}^{m} \alpha_t^{(j)} f\left(\mathbf{q}_t^{-(j)} \left| \mathbf{q}_{t-1}^{-(i)}\right.\right). \tag{1.75}$$

Kernel density estimation

There exists quite a lot of algorithms to estimate the full continuous distribution $f(x)$ from a set of samples $\mathcal{X} = \left\{x^{(1)}, x^{(2)}, \cdots\right\}$. One of the simplest methods is to use a histogram. In that case the estimated distribution will depend on how the bins a_k are chosen. Typically a starting value a_0 and a bandwidth, d_x, are specified manually and then the centre points of the bins are defined as $a_k = a_0 + k d_x$. A histogram $h(\cdot)$ is then defined as

$$h(a_k) = \sum_{i=1}^{|\mathcal{X}|} \left\{ \begin{array}{ll} 1 & \text{if } a_k - \frac{d_x}{2} \leq x^{(i)} < a_k + \frac{d_x}{2} \\ 0 & \text{otherwise} \end{array} \right., \tag{1.76}$$

and the estimate can made by introducing the *box kernel* function

$$f_{\text{box}}(x \left| x_0, d_x\right.) = \left\{ \begin{array}{ll} \frac{1}{d_x} & \text{if } x_0 - \frac{d_x}{2} \leq x < x_0 + \frac{d_x}{2} \\ 0 & \text{otherwise} \end{array} \right., \tag{1.77}$$

and make the estimation from the normalised histogram

$$f(x) \approx \sum_{a_k} \frac{h(a_k)}{|\mathcal{X}|} f_{\text{box}}(x \left| a_k, d_x\right.) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} f_{\text{box}}\left(x \left| \left[x^{(i)}\right], d_x\right.\right), \tag{1.78}$$

where $\left[x^{(i)}\right]$ denotes rounding to the closest bin a_k. This estimation will depend on the starting point, a_0 and the bandwidth d_x. To get rid of the dependency on a_0, the kernel can be centred around each of the data points instead. This is known as the box *kernel density estimation*,

$$f(x) \approx \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} f_{\text{box}}\left(x \left| x^{(i)}, d_x\right.\right). \tag{1.79}$$

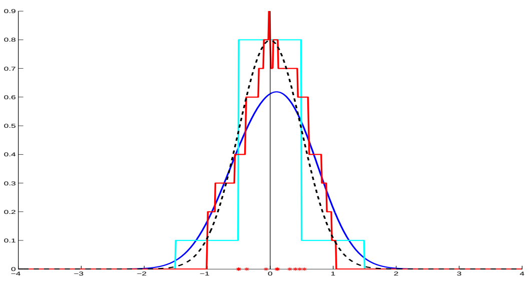


Figure 1.1: 10 samples (red stars) from $\mathcal{N}\left(x\left|0,\frac{1}{2}\right.\right)$ (black dashed) and the probability distribution of x estimated using histogram (cyan), box kernel density estimation (red) and Gaussian kernel density estimation (blue).

It still depends on the bandwidth which has to be specified. To get a smoother estimate, the box kernel can be replaced with for example a Gaussian kernel. In that case the bandwidth parameter will control the variance of the Gaussian kernel. Figure 1.1 compares the three different method on on synthetic data.

Another way to view the kernel density estimation is to consider the task of measuring some value x, such as for example the centre position of a car. If several measurements $x^{(i)}$ are made, each of them provide information on the true value x in form of $f_i\left(x\left|x^{(i)}\right.\right)$, which could be considered an approximation of the probability distribution $f\left(x\right)$. The contributions from the different measurements can then be combined by approximating $f\left(x\right)$ as the mean over the distributions $f_i\left(x\left|x^{(i)}\right.\right)$,

$$f\left(x\right) \approx \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} f_i\left(x\left|x^{(i)}\right.\right). \tag{1.80}$$

If measurement are made with for example Gaussian noise with some known variance, v, it is natural to chose $f_i\left(x\left|x^{(i)}\right.\right)$ to be Gaussian distributed with mean $x^{(i)}$ and variance v for all i. This will give the kernel density estimation, and it indicates how the bandwidth (variance of the kernel) should be chosen. But it is also possible to let $f_i\left(x\left|x^{(i)}\right.\right)$ be different distributions for different i if the measurements are performed in different ways with different precision.

This can be formalised by assuming that there exists outliers in the measurements and use a Bayesian approach. In that case the task is to estimate $f\left(x\left|\mathcal{X}\right.\right)$ where the distribution of the samples given a known x is assumed to be either of some known distribution, $f_i\left(x^{(i)}\left|x\right.\right)$, (for example $\mathcal{N}\left(x^{(i)}\left|x, v\right.\right)$) or outliers drawn from any distribution. The samples are also assumed to be conditional independent, conditioned on x, which means that the distributions $f_i\left(x^{(i)}\,|x\right)$ and $f_j\left(x^{(j)}\,|x\right)$ are independent if $i \neq j$. If $\mathcal{X}' \subset \mathcal{X}$ are the inliers, Bayes rule gives

$$f\left(x\,|\mathcal{X}'\right) = \frac{f\left(\mathcal{X}'\,|x\right) f\left(x\right)}{f\left(\mathcal{X}'\right)} = \frac{1}{c} \prod_{i=1}^{|\mathcal{X}'|} f\left(x^{(i)}\,|x\right), \tag{1.81}$$

where the prior, $f\left(x\right)$ has been assumed uniform and c is a constant making sure that $\int f\left(x\,|\mathcal{X}'\right) \mathrm{d}x = 1$. The distribution $f\left(x\,|\mathcal{X}\right)$, can be found by assuming some prior over the inlier subsets, and integrating out the assignments of samples into inliers and outliers,

$$f\left(x\,|\mathcal{X}\right) = \sum_{\mathcal{X}' \subset \mathcal{X}} f\left(x\,|\mathcal{X}'\right) f\left(\mathcal{X}'\,|\mathcal{X}\right). \tag{1.82}$$

If the prior $f\left(\mathcal{X}'\,|\mathcal{X}\right)$ is chosen as uniform over all possible subsets of $\mathcal{X}$, this sum will have $2^{|\mathcal{X}|}$ terms. Here $|\mathcal{X}|$ can be quite large if for example all feature points extracted from an image are considered. In many cases however the object of interest only constitutes a small part of the image which means that using a prior that puts higher likelihoods on subsets with a smaller number of inliers makes sense. Taking that to its extreme by assuming that exactly one of the measurements is an inlier and letting $f\left(\mathcal{X}'\right)$ be uniform over all single element subsets gives the kernel density estimation equation

$$f\left(x\,|\mathcal{X}\right) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} f_i\left(x\,\middle|x^{(i)}\right). \tag{1.83}$$

This prior assumptions holds even if there are more than one inlier since the outliers are assumed to come from any distribution, including the inlier distribution. It might however be possible to increase the accuracy of the estimator by relaxing this assumption. Assuming 1-2 inliers will however increase the computational time from linear to quadratic. But the execution time can be kept down by using a prior that only assign positive priors to sets $\mathcal{X}'$ with elements close to each other. Almost linear performance can be achieved by sorting the set of samples and breaking the recursion when the difference becomes too big. Figure 1.2 compares a few different prior assumptions.

Example: Condensation and iCondensation

Isard and Blake [31] uses a particle filter to track a single curve, typically the occluding border of some object. The state space, $\mathbb{S}$, consists of some set of B-spline curves. Such curves are parametrised by a set of control points, which means that each state $\mathbf{q} \in \mathbb{S}$ corresponds to a set of b control points, $(x_k, y_k) \in \mathbb{R}^2$,

$$(\mathbf{x}, \mathbf{y}) = (x_1, x_2, \cdots, x_b, y_1, y_2, \cdots, y_b) \tag{1.84}$$

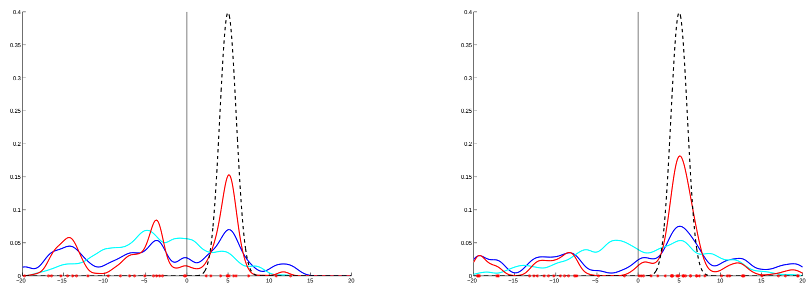


Figure 1.2: 5 samples from $\mathcal{N}\left(x\left|5,1\right.\right)$ (black dashed) mixed with 25(left)/50(right) uniformly distributed outliers, $\mathcal{X}$ (red stars), and estimates of $f\left(x\left|\mathcal{X}\right.\right)$ using Equation 1.82 with kernels $f_i\left(x\left|x^{(i)}\right.\right) = \mathcal{N}\left(x\left|x^{(i)},1\right.\right)$ and three different priors $f\left(\mathcal{X}'\left|\mathcal{X}\right.\right)$: (i) Exactly one inlier uniformly over all samples (blue). (ii) Exactly two inliers uniformly over all two element subsets (cyan). (iii) Exactly two inliers with the prior of $\mathcal{X}' = \{a,b\}$ proportional to $\mathcal{N}\left(a\left|\frac{a+b}{2},1\right.\right)\mathcal{N}\left(b\left|\frac{a+b}{2},1\right.\right)$ (red).

The B-spline curve, $B_{\mathbf{q}}\left(s\right)$, $0 \leq s \leq b$ is defined by interpolating between the control points using polynomials, where the polynomials are constructed in a way that makes their first and some higher order derivatives match on the junction points. This makes the curves smooth and without corners. Unfortunately, letting $\mathbb{S}$ consist of all such curves makes it too big and leads to unstable tracking. The space can be reduced by for example performing PCA on a set of representative curves and only keeping the a first principal components. This will give a mean shape $\bar{\mathbf{q}} = (\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and a matrix of variation modes $\mathbf{W}$, that gives the control points

$$(\mathbf{x}, \mathbf{y}) = \bar{\mathbf{q}} + \mathbf{q}\mathbf{W} \tag{1.85}$$

from a state vector of coefficients

$$\mathbf{q} = (c_1, c_2, \cdots c_a) \in \mathbb{R}^a = \mathbb{S}. \tag{1.86}$$

Another possibility is to to consider some given curve $\bar{\mathbf{q}} = (\bar{\mathbf{x}}, \bar{\mathbf{y}})$ under affine transformations. This is done by letting

$$\mathbf{W} = \begin{pmatrix} \mathbf{0}_b{}^{\mathrm{T}} & \mathbf{1}_b{}^{\mathrm{T}} \\ \mathbf{1}_b{}^{\mathrm{T}} & \mathbf{0}_b{}^{\mathrm{T}} \\ \bar{\mathbf{x}} & \mathbf{0}_b{}^{\mathrm{T}} \\ \mathbf{0}_b{}^{\mathrm{T}} & \bar{\mathbf{y}} \\ \mathbf{0}_b{}^{\mathrm{T}} & \bar{\mathbf{x}} \\ \bar{\mathbf{y}} & \mathbf{0}_b{}^{\mathrm{T}} \end{pmatrix}, \tag{1.87}$$

where $\mathbf{0}_b{}^{\mathrm{T}}$ and $\mathbf{1}_b{}^{\mathrm{T}}$ are b dimensional row vectors containing only zeroes or ones respectively.

The dynamic model used is linear and constructed by assuming that the accelerations of the coefficients, c_k, is Gaussian distributed with mean zero. This implies that the state $\mathbf{q}$ also contains the velocities. But it can also be achieved by letting $\mathbf{q}_t$ depend on both $\mathbf{q}_{t-1}$ and $\mathbf{q}_{t-2}$. With some coefficient matrix $\mathbf{H}$, the dynamic model becomes

$$\mathbf{q}_t = \mathbf{H} \begin{pmatrix} \mathbf{q}_{t-1} \\ \mathbf{q}_{t-2} \end{pmatrix} + \mathbf{w}_t, \tag{1.88}$$

where $\mathbf{w}_t$ as before is Gaussian distributed noise with mean 0 and covariation matrix $\mathbf{Q}$. The parameters $\mathbf{H}$, $\bar{\mathbf{q}}$ and $\mathbf{Q}$ can be learnt offline from training data as described in [5].

The observations are greyscale images, $\mathbf{o}_t$, and the observation likelihoods $f\left(\mathbf{o}_t \left| \mathbf{q}_t \right.\right)$ are generated using a one dimensional feature point detector that detect high contrast features, such as edges. Given a state $\mathbf{q}_t$, it translates into a curve $B_{\mathbf{q}_t}(s)$, $0 \le s \le b$. Along this curve d equally spaced points at $s = \frac{i}{d}$ for $i = 1, 2, \cdots, d$ is considered. At each such point, $B_{\mathbf{q}_t}\left(\frac{i}{d}\right)$, the feature point detector is used to find the feature point, $\mathbf{x}_i$, closest to $B_{\mathbf{q}_t}\left(\frac{i}{d}\right)$ along the curve normal. It is then assumed that $\mathbf{x}_i$ with probability α originating from the tracked curved and with probability $1-\alpha$ originates from clutter. If it originates from the tracked curve, it is assumed to be Gaussian distributed along the normal with mean $B_{\mathbf{q}_t}\left(\frac{i}{d}\right)$ and some variance σ. If it originates from clutter it is assumed uniformly distributed along the normal with spatial density λ. The feature points are all assumed independent, which gives

$$f\left(\mathbf{o}_t \left| \mathbf{q}_t \right.\right) = \prod_{i=1}^{d} (1-\alpha)\,\lambda + \frac{\alpha}{\sqrt{2\pi}\sigma} e^{-\frac{1}{\sigma^2}\left|B_{\mathbf{q}_t}\left(\frac{i}{d}\right)-\mathbf{x}_i\right|^2}. \tag{1.89}$$

Once the prediction function, Equation 1.88, and an observation likelihood, Equation 1.89 are given, the particle filter algorithm from Section 1.3.1 can be applied to estimate $f\left(\mathbf{q}_t \left| \mathcal{O}_{0\cdots t} \right.\right)$ and its mean, $\hat{\mathbf{q}}_t$. The output of the algorithm is the curves $B_{\hat{\mathbf{q}}_t}(s)$.

Isard and Blake extends [31] this algorithm to use importance sampling and calls the extended version *icondensation*. This is done in the framework of hand-tracking where the curve to be tracked is the outline of the hand. The importance function $\hat{f}(\mathbf{q}_t)$ is constructed from a skin colour blob detector, that detects regions in the image that is skin coloured. The number of detected regions is denoted n, and for each such region its mass centre, $\mathbf{b}_k$ is calculated.

The state variables $\mathbf{q} = (c_1, c_2, \cdots, c_a)$ are partitioned into two sets of variables corresponding to the translation, $\mathbf{q}^{\mathrm{T}}$, and deformation, $\mathbf{q}^{\mathrm{D}}$ respectively. If $\mathbf{W}$ in the form from Equation 1.87 is used, this means $\mathbf{q}^{\mathrm{T}} = (c_1, c_2)$ and $\mathbf{q}^{\mathrm{D}} = (c_3, c_4, c_5, c_6)$.

The relationship between the skin colour blob mass centre, $\mathbf{b}_k$, and the centroid of the tracked curve, $\mathbf{q}_t^{\mathrm{T}}$ is assumed to be Gaussian,

$$f\left(\mathbf{q}_t^{\mathrm{T}} \left| \mathbf{b}_k \right.\right) = \mathcal{N}\left(\mathbf{q}_t^{\mathrm{T}} - \mathbf{b}_k \left| \mu, \boldsymbol{\Sigma} \right.\right), \tag{1.90}$$

where the parameters μ and $\boldsymbol{\Sigma}$ is learnt by manually extracting curves from training images and comparing their centroid with the blob mass centres extracted by the blob detector. The importance function for the fist part of the state vector is defined as a mixture over all detected blobs,

$$\hat{f}\left(\mathbf{q}_t^{\mathrm{T}}\right) = \sum_{k=1}^{n} \frac{1}{n} \mathcal{N}\left(\mathbf{q}_t^{\mathrm{T}} - \mathbf{b}_k \left| \mu, \boldsymbol{\Sigma} \right.\right). \tag{1.91}$$

The distribution of the remaining coefficients, $\mathbf{q}_t^{\mathrm{D}}$, is copied from the original distribution $f\left(\mathbf{q}_t^{-} \left| \mathcal{O}_{1\cdots t-1} \right.\right)$,

$$\hat{f}\left(\mathbf{q}_t^{\mathrm{D}}\right) = \int_{\mathbb{R}^2} f\left(\mathbf{q}_t \left| \mathcal{O}_{1\cdots t-1} \right.\right) \mathrm{d}\mathbf{q}_t^{\mathrm{T}}, \tag{1.92}$$

and the full importance function is

$$\hat{f}\left(\mathbf{q}_t\right) = \hat{f}\left(\mathbf{q}_t^{\mathrm{T}}\right) \hat{f}\left(\mathbf{q}_t^{\mathrm{D}}\right). \tag{1.93}$$

This distribution can be simulated from by using the samples from $f\left(\mathbf{q}_t \left| \mathcal{O}_{1\cdots t-1} \right.\right)$ in $\mathcal{S}_t^{-}$ and then replacing $\mathbf{q}_t^{\mathrm{T}}$ in those samples with samples from $\hat{f}\left(\mathbf{q}_t^{\mathrm{T}}\right)$.

To estimate the weights, $\alpha_t^{(j)}$, the importance function has to be evaluated at the samples, $\mathbf{q}_t^{-(j)}$,

$$\hat{f}\left(\mathbf{q}_t^{-(j)}\right) = \hat{f}\left(\mathbf{q}_t^{\mathrm{T}-(j)}\right) \hat{f}\left(\mathbf{q}_t^{\mathrm{D}-(j)}\right), \tag{1.94}$$

The first factor is given by Equation 1.91. The second factor is somewhat more complicated. In order to make it possible to evaluate it in an efficient way, the distributions $f\left(\mathbf{q}_t^{\mathrm{T}} \left| \mathbf{q}_{t-1}^{\mathrm{T}} \right.\right)$ and $f\left(\mathbf{q}_t^{\mathrm{D}} \left| \mathbf{q}_{t-1}^{\mathrm{D}} \right.\right)$ are assumed to be independent. This assumption is true if the coefficient matrix of the dynamic model, $\mathbf{H}$, can be written in the block matrix form

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{-1}^{\mathrm{T}} & \mathbf{0} & \mathbf{H}_{-2}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{-1}^{\mathrm{D}} & \mathbf{0} & \mathbf{H}_{-2}^{\mathrm{D}} \end{pmatrix}, \tag{1.95}$$

and the noise covariation matrix, $\mathbf{Q}$ can be written in the block matrix form

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_{-1}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{-1}^{\mathrm{D}} \end{pmatrix}. \tag{1.96}$$

The prediction probability $f\left(\mathbf{q}_t \left| \mathbf{q}_{t-1} \right.\right)$ can then be split into

$$f\left(\mathbf{q}_t \left| \mathbf{q}_{t-1} \right.\right) = f\left(\mathbf{q}_t^{\mathrm{T}} \left| \mathbf{q}_{t-1}^{\mathrm{T}} \right.\right) f\left(\mathbf{q}_t^{\mathrm{D}} \left| \mathbf{q}_{t-1}^{\mathrm{D}} \right.\right). \tag{1.97}$$

This makes it possible to calculate the second factor $\hat{f}\left(\mathbf{q}^{\mathrm{D}}\right)$ using the samples in $\mathbb{S}_{t-1}^{-}$, e.g.

$$\hat{f}\left(\mathbf{q}_t^{\mathrm{D}-(j)}\right) = \sum_i \alpha_{t-1}^{(i)} f\left(\mathbf{q}_t^{\mathrm{D}-(j)} \middle| \mathbf{q}_{t-1}^{\mathrm{D}-(i)}\right). \tag{1.98}$$

Also, the distribution, $f\left(\mathbf{q}_t^{-} \left|\mathcal{O}_{1\cdots t-1}\right.\right)$, from which the samples originally was drawn, needs to be evaluated. This can also be achieved by using the the samples in $\mathbb{S}_{t-1}^{-}$, e.g.

$$f\left(\mathbf{q}_t^{-(j)} \left|\mathcal{O}_{1\cdots t-1}\right.\right) = \sum_i \alpha_{t-1}^{(i)} f\left(\mathbf{q}_t^{-(j)} \middle| \mathbf{q}_{t-1}^{-(i)}\right). \tag{1.99}$$

The weights, $\alpha_t^{(i)}$ are then given by Equation 1.74,

$$\alpha_t^{(j)} = \frac{1}{\hat{f}\left(\mathbf{q}_t^{\mathrm{T}-(j)}\right)} \frac{\sum_i \alpha_{t-1}^{(i)} f\left(\mathbf{q}_t^{-(j)} \middle| \mathbf{q}_{t-1}^{-(i)}\right)}{\sum_i \alpha_{t-1}^{(i)} f\left(\mathbf{q}_t^{\mathrm{D}-(j)} \middle| \mathbf{q}_{t-1}^{\mathrm{D}-(i)}\right)} f\left(\mathbf{o}_t \middle| \mathbf{q}_t^{-(j)}\right). \tag{1.100}$$

This expression is then approximated by assuming the two distributions $f\left(\mathbf{q}_t^{\mathrm{T}-(j)} \left|\mathcal{O}_{1\cdots t-1}\right.\right)$ and $f\left(\mathbf{q}_t^{\mathrm{D}-(j)} \left|\mathcal{O}_{1\cdots t-1}\right.\right)$ to be independent,

$$f\left(\mathbf{q}_t^{-(j)} \left|\mathcal{O}_{1\cdots t-1}\right.\right) \approx f\left(\mathbf{q}_t^{\mathrm{T}-(j)} \left|\mathcal{O}_{1\cdots t-1}\right.\right) f\left(\mathbf{q}_t^{\mathrm{D}-(j)} \left|\mathcal{O}_{1\cdots t-1}\right.\right). \tag{1.101}$$

This contradicts the idea allowing the particle filter to maintain several hypothesis coupling position and shape, but seems to give good results in practise. The factor $\sum_i \alpha_{t-1}^{(i)} f\left(\mathbf{q}_t^{\mathrm{D}-(j)} \middle| \mathbf{q}_{t-1}^{\mathrm{D}-(i)}\right)$ in Equation 1.100 will now cancel and give

$$\alpha_t^{(j)} \approx \frac{\sum_i \alpha_{t-1}^{(i)} f\left(\mathbf{q}_t^{\mathrm{T}-(j)} \middle| \mathbf{q}_{t-1}^{\mathrm{T}-(i)}\right)}{\hat{f}\left(\mathbf{q}_t^{\mathrm{T}-(j)}\right)} f\left(\mathbf{o}_t \middle| \mathbf{q}_t^{-(j)}\right). \tag{1.102}$$

The icondensation [32] paper also contains a different observation model based on directional derivatives taking along the curve normals instead of feature points. It also extends the dynamic model to allow the model to reinitiate by allowing it to jump from any state to some state distributed according to some prior state distribution that does not depend on the observations.

1.3.2 Multi target

A common way to extend the single target tracking algorithms to multi target is to split up the problem into four sub problem.

- Track initialisation that detects and initialised new tracks.
- Data association that associates the available measurements with the objects being tracked. Possible in a probabilistic manner like the PDAF describe above.
- Single target tracking of each object separately e.g. using one Kalman or particle filter per object.
- Track termination that removes lost tracks.

This way the tracking only have to consider likelihoods over single target state spaces, $\mathbb{S}$, and the data association has to consider probabilities over some space consisting of all possible assignments of detections to targets, which is a finite discrete space.

Joint Probabilistic Data Association Filter

Extending the PDAF algorithm from Section 1.3.1 to the joint probabilistic data association filter [90], or JPDAF, to handle the multi target case requires only the calculation of the β weights to be changed. The rest of the PDAF algorithms can be used as it is described above. This will solve the data association problem. Track initialisation and termination will have to be solved separately. To make this efficient the objects tracked are clustered into clusters containing objects with overlapping validation gates, and each such cluster is handled separately. If the validation gates of two objects do not overlap, they will not interact at all and there is no point in considering them jointly as that will give the exact same results as considering them one by one.

As before, there is n_t observation in frame t denoted $\mathbf{o}_{t,i}$ for $i = 1, 2, \cdots, n_t$, and the set of all of them is denoted $\mathcal{O}_{t,*}$. There is now also m_t objects being tracked, $\mathbf{q}_{t,j}$, for $j = 1, 2, \cdots, m_t$, and the predicted observation for each of them is $\mathbf{o}_{t,j}^-$. All possible assignments of targets to measurements are considered under the assumption that each target generates zero or one measurement within its validation gate. These different assignments are denoted $\mathcal{A}^k$ and enumerated $k = 1, 2, \cdots, h$, where h is the number of possible assignments. For each k, the function $a_t^k(j) = i$ assign object j to measurement i. For an object not associated to any measurement, $a_t^k(i) = 0$. From this function the number of false detections n_{false} can be calculated (n_{false} depends of t and k, but those indexes have been dropped). The JPDAF is derived by assuming that the number of false detections, n_{false}, is Poisson distributed with parameter λv. As before, v is the area of the entire image and α_2 is the probability that the detector will fail to detect an object. The probability of the events, $p\left(\mathcal{A}^k\right) = p\left(\mathcal{A}^k \left| \mathcal{O}_{t,*}, \mathbf{o}_{t,1}^-, \cdots, \mathbf{o}_{t,m_t}^-\right.\right)$, is (see [90] for details)

$$p\left(\mathcal{A}^k\right) = \frac{\lambda^{n_{\text{false}}} \alpha_2^{m_t - n_t + n_{\text{false}}} \left(1 - \alpha_2\right)^{n_t - n_{\text{false}}}}{c} \prod_{\{j | a_k(j) > 0\}} f\left(\mathbf{o}_{t,a_k(j)} \left| \mathbf{o}_{t,j}^-, \mathbf{S}_{t,j}\right.\right), \tag{1.103}$$

where c is a constant making sure $\sum_{k=1}^{h} p\left(\mathcal{A}^k\right) = 1$, and $f\left(\mathbf{o}_{t,a_k(j)} \left|\mathbf{o}_{t,j}^{-}, \mathbf{S}_{t,j}\right.\right)$ is given by Equation 1.59. The predicted observation, $\mathbf{o}_{t,j}^{-}$, of object j, and its variance, $\mathbf{S}_{t,j}$, are given by the Kalman filter. The probability that observation i belongs to object j, denoted $\beta_{t,i}^{j}$, can be found by summing over all events for which this condition is true,

$$\beta_{t,i}^{j} = \sum_{\{k | a_k(j)=i\}} p\left(\mathcal{A}^k\right), \tag{1.104}$$

and the probability that object j were not detected, given the observations is

$$\beta_{t,0}^{j} = 1 - \sum_{i=1}^{n_t} \beta_{t,i}^{j}. \tag{1.105}$$

Once the β weights have been derived jointly, the rest of the PDAF algorithm as described above can be applied to each of objects separately to update the distributions of each object for the next frame.

Multi Hypothesis Tracking

One problem with the above algorithms is that when objects come close to each other the data association will become very uncertain and detections from nearby objects will affect the state distribution quite a lot. This can be avoided by not only consider the different assignments of measurements to objects in the current frame, but considering the assignments jointly over all previous frames. That way a set of hypothesis has to be maintained, where each hypothesis consists of some assignment of measurements to objects over all previous frames.

It is then also possible to have different hypotheses represent a different number of objects present, and thus solve the entry and exit problems jointly with the data association problem. An algorithm for this was suggested by Reid [66] called Multi Hypothesis Tracking or MHT.

For each frame a set of h hypothesis, $\left\{\mathcal{H}_t^1, H_t^2, \cdots, H_t^h\right\}$, is maintained. Each hypothesis, k, assigns zero or one measurement to each object, j, for each frame. That information is condensed, recursively each frame, into a mean state, $\hat{\mathbf{q}}_{t,j}^{k}$, and a covariation matrix, $\mathbf{P}_{t,j}^{k}$, for each object using a Kalman filter, e.g.

$$\mathcal{H}_t^k = \left\{\hat{\mathbf{q}}_{t,1}^{k}, \mathbf{P}_{t,1}^{k}, \hat{\mathbf{q}}_{t,2}^{k}, \mathbf{P}_{t,2}^{k}, \cdots, \hat{\mathbf{q}}_{t,m_t^k}^{k}, \mathbf{P}_{t,m_t^k}^{k}\right\}, \tag{1.106}$$

were m_t^k is the number of objects in hypotheses k in frame t.

When the measurements in the current frame are processed, each hypothesis from the previous frame is expanded into several hypotheses with each of the possible assignments of objects to the measurements in the current frame, including the possibilities that some of the measurements are from new previously unobserved objects. Those measurements are assigned to an object id one larger than the maximum previous object id. This way the number of hypotheses will explode, but it is in some cases possible to keep the number of hypotheses manageable using the ideas described below. As in the previous section, let the function $i = a_t^k(j)$ represent the object, j, to measurement, i, associations of hypotheses k in the frame t. From it the following values can be calculated (here the t and k indexes have here been dropped for clarity)

- n_{old} - The number of objects in the previous frame that are detected in this frame.
- n_{new} - The number of new objects detected in this frame.
- n_{false} - The number of false detections (as before).
- $n_t = n_{\text{old}} + n_{\text{new}} + n_{\text{false}}$ - The total number of detections (as before).

Also, let $\psi_t(k)$ be the parent hypothesis from the previous frame that hypothesis, k, in the current frame is based on. That is, hypothesis $\mathcal{H}_t^k$ is constructed from $\mathcal{H}_{t-1}^{\psi_t(k)}$ from the previous frame and the associations $a_t^k(\cdot)$ in the current frame.

To evaluate the different hypothesis the probabilities of each of the hypothesis, given the observations, is used,

$$\beta_t^k = p\left(\mathcal{H}_t^k \middle| \mathcal{O}_{0\cdots t,*}\right) \tag{1.107}$$

This probability is found by reusing the Kalman filter assumption that measurements from tracked objects are Gaussian distributed, and assuming that

- False detections are uniformly distributed over the image with area v.
- New objects appear uniformly over the entire image.
- The probability of the detector failing to detect an object is α_2.
- The number of false detections, n_{false}, is Poisson distributed with intensity λv.
- The number of new objects, n_{new}, is Poisson distributed with intensity $\lambda_n v$.

This gives (see [66] for details)

$$\beta_t^k = \frac{1}{c}\left(1-\alpha_2\right)^{n_{\text{old}}} \alpha_2^{m_{t-1}^{\psi_t^k} - n_{\text{old}}} \lambda^{n_{\text{false}}} \lambda_n^{n_{\text{new}}} \prod_{\left\{j \middle| a_t^k(j)>0\right\}} f\left(\mathbf{o}_{t,a_t^k(j)} \middle| \mathbf{o}_{t,j}^{k-}, \mathbf{S}_{t,j}^k\right) \beta_{t-1}^{\psi_t^k}, \tag{1.108}$$

were c is a constant making sure that $\sum_{k=1}^h \beta_t^k = 1$. As described below, a validation gate will be used to keep down the number of hypothesis. This means that the truncated

Gaussian from Equation 1.59 can be used for $f\left(\mathbf{o}_{t,a_t^k(j)}\left|\mathbf{o}_{t,j}^{k-},\mathbf{S}_{t,j}^k\right.\right)$. The predicted observations, $\mathbf{o}_{t,j}^{k-}$, and their covariance matrices, $\mathbf{S}_{t,j}^k$, are given by the Kalman filters.

Finally to produce the output, the hypotheses, k, with maximum probability, β_t^k, will be chosen as the most likely one and the mean values from it are the output. This is no longer strictly a smoothing filter, since it is optimising over all possible measure/object assignments choosing the most likely only and then filtering under the assumption that this choice was correct.

Some care has to be taken to keep down the number of hypothesis. This is mainly done by removing unlikely hypotheses, combining similar hypotheses and clustering objects. But already when a new hypothesis is formed some sanity checks are performed and the hypotheses failing those are discarded directly, without being considered. The checks performed are

- Each object may not be associated with more than one measurement.
- Each measurement may not be associated with more than one object.
- Objects may only be associated with measurements within their validation gates.

If two objects are far enough from each other there will be no measurements that lies within the validation gate of both of them. In this case considering them one by one will give the exact same result as considering them jointly, and will require significantly less hypothesis. This means that clustering close objects and processing each cluster separately, can reduce the number of hypotheses quite significantly. The clustering will however have to change over time as different objects become close to each other.

As a frame is processed the clusters of the previous frame are inherited. Then each measurement is compared with each of the object states and if it is within the validation gate of an object, i.e. if it is close enough, it will be assigned to the cluster containing that object. If there are several objects close to each other this might result in a measurement being assigned to several clusters. All those clusters are combined into a new cluster with the number of hypotheses in the new cluster being the product of the number of hypothesis in the original clusters. The probabilities, β_t^k of the hypothesis in the new cluster is the product of the probabilities of the hypothesis in the old clusters. New clusters are created for measurements not within the validation gate of any object.

After each frame has been processes the set of hypothesis is reduced by discarding all hypotheses whose probability β_t^k is less than some constant. Also, all hypotheses that are very similar are combined into a single hypothesis. For two hypothesis to be similar, they should have the same number of objects present and the mean values and covariance matrices should be closer than some threshold,

After this reduction of hypothesis it might be possible to split clusters again. This is done by considering assignments that all hypothesis agree upon. If all hypothesis assigns the same measurement, i, to one object j, e.g $a_t^k(j) = i$ for all k and some fixed i and j, then the object j is excluded from the cluster and placed in a cluster of its own.

This is actually an approximation, because even if it is certain which measurement in the current frame belongs to the object, the different hypothesis might represent different assignments for measurements in previous frames, and thus the mean and covariances of the object in the different hypothesis will vary, and has to be combined in some way. If the measurement noise is low this will be a small approximation, but with higher measurement noise it might be a problem. Also, for the event that all hypothesis agree on an assignment to occur, the clutter density has to be low enough not to place any false detections at all within the validation gate of the object in this frame.

A final optimisation can be found by concluding that the Kalman filter does not have to be executed for every object in every hypotheses. Typically each object will have exactly the same assignments in several hypotheses, and then the Kalman filter only has to be executed once for each such group and the result can be linked to each of the hypothesis.

Probabilistic multi-hypothesis tracking

Even after those approximations the number of hypothesis in the MHT algorithm explodes exponentially. To prevent that Streit and Luginbuhl [77] have suggested an alternative called probabilistic multi-hypothesis tracking or PMHT. It operated in much the same way as the JPDAF described above in that it generates synthetic measurements based on how close the real measurement as are to the predicted measurements. It does not however have to enumerate every possible assignment of targets to measurements. Instead the EM-algorithm is used to calculate the weights.

The algorithm works in a sliding window fashion. A set of τ frames from $t = t_0$ to $t = t_0 + \tau - 1$ are considered at each step, and tracks are produced for those frames. Then only the result for the first frame, t_0, is stored and the window is shifted one frame forward, $t_0 \leftarrow t_0 + 1$, and new tracks are estimated.

The idea is to model the assignment of measurements to objects as unknown random variables, $a_t^i \in \{1, 2, \cdots, m_t\}$, where $a_t^i = j$ means that measurement i is assigned to target j in frame t. Note that all measurments are assumed to originate from objects, i.e. no clutter. Note also that here measurements are assigned to targets in contrast to the JPDAF case where targets were assigned to measurements. This means that more than one measurement can be assigned to a single target. In cases where the sensor provides point detections this is undesirable as in that case it is often assumed that a single target produces zero or one measurements as is done in the JPDAF case. Here such restrictions can't be enforced, but they can be encouraged by the choice of the prior of a_t^i, see below. If the input however is the results of a sliding window based object detector it might actually be desirable to assume that a single target produces several detections (measurements) as that is typically the case for such detectors. In that case some other type of prior has to be chosen that prevent two objects from occupying the same physical space, which introduces more intricate dependencies between the objects.

The EM-algorithm is an iterative algorithm that assumes an initial guess of the state, $\tilde{\mathbf{q}}_{t,j}$, of all objects, $1 \leq j \leq m_t$, in all frames, $t_0 \leq t < t_0 + \tau$, is available. This guess will be refined by making a new estimate of the states, $\hat{\mathbf{q}}_{t,j}$. This new estimate is then used as the initial guess in the next iteration, $\tilde{\mathbf{q}}_{t,j} \leftarrow \hat{\mathbf{q}}_{t,j}$, and the process is iterated until convergence. The process can be initiated by using the predictions made by the Kalman filter as the first initial guess.

The first step is to estimate the probability that observation i belongs to the object j,

$$\beta_{t,i}^{j} = p\left(a_t^i = j \left| \mathcal{O}_{t,*} \right.\right) = \frac{f\left(\mathbf{o}_{t,i} \left| \tilde{\mathbf{q}}_{t,j} \right.\right) p\left(a_t^i = j\right)}{\sum_{j=1}^{m_t} f\left(\mathbf{o}_{t,i} \left| \tilde{\mathbf{q}}_{t,j} \right.\right) p\left(a_t^i = j\right)}, \tag{1.109}$$

where $p\left(a_t^i\right)$ is the prior of a_t^i. It is supposed to include the probability of the object not being detected at all, and it is defined as a function of the weights of the previous iteration, $\tilde{\beta}_{t,i}^{j}$,

$$\omega_{t,i}^{j} = p\left(a_t^i = j\right) = \frac{1}{n_t} \sum_{i=1}^{n_t} \tilde{\beta}_{t,i}^{j}. \tag{1.110}$$

In the simplest case, a linear observation model, with observation matrix $\mathbf{O}$, and Gaussian observation noise, with covariance matrix $\mathbf{R}$ can be used. In that case $\mathbf{O}$ and $\mathbf{R}$ will not vary over time or from object to object. The weights can be calculated as

$$\beta_{t,i}^{j} = \frac{\mathcal{N}\left(\mathbf{o}_{t,i} \left| \mathbf{O}\tilde{\mathbf{q}}_{t,j}, \mathbf{R} \right.\right) \omega_{t,i}^{j}}{\sum_{j=1}^{m_t} \mathcal{N}\left(\mathbf{o}_{t,i} \left| \mathbf{O}\tilde{\mathbf{q}}_{t,j}, \mathbf{R} \right.\right) \omega_{t,i}^{j}}. \tag{1.111}$$

Form these weights, one synthesised measurement for each object is produced as a weighted mean,

$$\hat{\mathbf{o}}_{t,j} = \frac{\sum_{i=1}^{n_t} \beta_{t,i}^{j} \mathbf{o}_{t,i}}{\sum_{i=1}^{n_t} \beta_{t,i}^{j}}. \tag{1.112}$$

The covariance matrix of that synthesised measurement becomes

$$\hat{\mathbf{R}}_{t,j} = \frac{\mathbf{R}}{\sum_{i=1}^{n_t} \beta_{t,i}^{j}}, \tag{1.113}$$

which now depends on both time and object. Note that $\beta_{t,i}^{j}$ is normalised over the objects and in the denominator here it is summed over the measurements. That sum might very well be greater than one, which would represent the case where several measurements have been assigned to one target. In that case the variance of the synthesised measurement becomes smaller than the variance of the original measurements. This makes sens as the synthesised measurement in this case is a weighted mean over several samples from the same distribution. If on the other hand the sum in the denominator is less than one, the variance of the synthesised measurement will become larger than that of the original measurements. This represents the case where only one measurement originates from the target, but the system don't know which one it is and thus becomes more uncertain.

Once the synthesised measurements have been calculated a new estimate of the states, $\mathbf{q}_{t,j}$, of all objects, $1 \leq j \leq m_t$, in all frames, $t_0 \leq t < t_0 + \tau$, can be made using Kalman filters. To allow all measurements, both past and future, to influence the state estimates, the Kalman filter is first execute forward in time and then backwards in time. This new estimate is then used as the initial guess for the next iteration.

In the presented form, the PMHT algorithm assumes that all measurements originates from objects, e.g. no clutter. To allow it to handle false detections, Gauvrit *et al* [18] have suggested to add an additional flat object, $j = 0$, representing the clutter. Observations from this object will be uniformly distributed over the observed area v, with some intensity λ,

$$f\left(\mathbf{o}_{t,i}\left|\tilde{\mathbf{q}}_{t,0}\right.\right) = \frac{1}{\lambda v}. \tag{1.114}$$

To introduce the assumption that a single target can only generate a single measurement, Ruan and Streit [88] have suggested to let the priors, $\omega^j_{t,i}$ only depend on the number of objects tracked and the number of observations made. They can in that case be calculated before the iterations of the EM-algorithm is started and then kept constant. In the paper it is claimed that it is possible to derive expressions for such estimations by assuming that an object will not be detected with some probability, α_2, and that the number of false detections are Poisson distributed with mean λv. An explicit formula is only given for the single target case, $m_t = 1$, which is shown to be

$$\omega^1_{t,i} = \frac{1-\alpha_2}{n_t\left(1-\alpha_2\right)\alpha_2\lambda v} \qquad \text{for all } i \tag{1.115}$$

Multi target state space

Extending the single target state space $\mathbb{S}$ to multi target is not straightforward. The natural way would be to extend $\mathbb{S}$ into $\mathbb{S}^n$ and let each state represent the joint state of all objects. But this requires that the number of objects, n, is known and don't vary over time. To allow the number of objects to vary, a state space constructed like

$$\mathbb{S}^\infty = \cup_{n=0}^{\infty}\mathbb{S}^n \tag{1.116}$$

is needed. But this is a very strange space. It is no longer a vector space, e.g it is not possible to form linear combinations of its elements, which makes it hard to define expected values. It is however measurable, which means that it is possible to define the integral of functions $f : \mathbb{S}^\infty \rightarrow \mathbb{R}$ by letting the integral be the sum over the integrals restricted to the separate subspaces $\mathbb{S}^k$. The space $\mathbb{S}^0$ consists of a single element representing the event that no objects are present and the measure of this has to be set to something. Here it will be set to 1.

This makes it possible to define probability distribution functions on $\mathbb{S}^\infty$ and talk about the probability of events in this space as integrals of some probability distribution function. It is however no longer possible to compare likelihoods as the different states have different dimensions. In for example $\mathbb{R}^n$ it makes sens to compare likelihoods because if $\mathbf{x} \in \mathbb{R}^n$ is a stochastic variable with continious distribution $f(\mathbf{x})$, the probability of the event that $\mathbf{x}$ lies within some small distance δ of $\mathbf{x}_0$ will be

$$\int_{|\mathbf{x}-\mathbf{x}_0|<\delta} f(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx f(\mathbf{x}_0) \int_{|\mathbf{x}-\mathbf{x}_0|<\delta} \mathrm{d}\mathbf{x} = f(\mathbf{x}_0)\, A_\delta, \tag{1.117}$$

where the approximation is good if δ is small enough compared to the smoothness of $f(\cdot)$. A_δ is a constant (the volume of an n-dimensional ball with radius δ) that depends on δ and n, but not $\mathbf{x}_0$. This means that if $f(\mathbf{x}_0) > f(\mathbf{x}_1)$, then the probability of x being within δ of x_0 is larger than the probability of x being within δ of x_1 regardless of which δ is chosen, as long as it is small enough. This is not the case in $\mathbb{S}^\infty$, since the dimension n will now depend of $\mathbf{x}_0$ and that means A_δ is no longer constant, but will also depend on $\mathbf{x}_0$. This means that a likelihood from some subspace $\mathbb{S}^n$ of $\mathbb{S}^\infty$ can be made both larger and smaller than a likelihood from some subspace $\mathbb{S}^m$ with a different dimension, $m \neq n$, by only changing the the overall scale used in both coordinate systems. This means that the property of one likelihood being larger than another will depend on for example whether the states are expressed in meters or millimetres. Two likelihoods from the same subspace $\mathbb{S}^n$ will still be comparable. In the same way likelihood ratios of different dimensions are undefined and can be given arbitrary values by rescaling the coordinate system.

One way do avoid this problem is to discretise the state space, $\mathbb{S}^\infty$, e.g specifying δ. That way A_δ can be calculated and it will be possible to work with probabilities instead of likelihoods. This will make everything well defined, but results will still depend on the step size of the discretisation, δ. That dependency becomes more visible as δ is an explicit parameter instead of being hidden as the chosen scale of the coordinate system. A continuous state space is however used in many cases.

To get a feeling of such spaces, consider the simple example illustrated in Figure 1.3. The world consist of a continuous line from 0 to 3 on which $0 - 2$ objects move. The state of an object q is where along this line its centre is located. That is, $\mathbb{S}^1 = \{x \in \mathbb{R}\,|0 \leq x \leq 3\}$ and $\mathbb{S}^\infty = \mathbb{S}^2 \cup \mathbb{S}^1 \cup \mathbb{S}^0$. This world is observed by a 2-pixel camera centred in the world displaying a foreground/background segmentation. Both pixels and objects are one unit wide and the pixels, $o_t = (o_{t,1}, o_{t,2})$, are expected to be 1 if more than half of them is covered by an object otherwise 0, but are observed with noise. That is $f(o_{t,k}) = f(o_{t,k}\,|\hat{o}_{t,k})$ describes the observation noise, where $\hat{o}_{t,k}$ is the expected value of the pixel k. This means that the observation space is a discrete space consisting of four elements, 00, 01, 10 and 11. The expected value of first pixel, $\hat{o}_{t,1}$ is 1 if there is an object whose state q satisfies $0.5 \leq q < 1.5$ otherwise 0. Likewise, the expected value of the second pixel, $\hat{o}_{t,2}$, is 1 if there is an object whose state q satisfies $1.5 \leq q < 2.5$ other wise 0. Half-open intervals are used here to prevent a single object from being able to cover both pixels if placed in the exact centre. There are no constraints

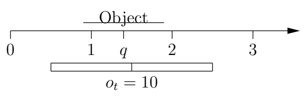


Figure 1.3: Example of a 1D world with one object present viewed by a 2-pixel camera.

00	0 objects present or 1 object present with $q < 0.5$ or $q \geq 2.5$ or 2 objects present with $q_1 < 0.5$ or $q_1 \geq 2.5$ and $q_2 < 0.5$ or $q_2 \geq 2.5$
01	1 object present with $0.5 \leq q < 1.5$ or 2 objects present with $0.5 \leq q_1 < 1.5$ and $0.5 \leq q_2 < 1.5$ or 2 objects present with $0.5 \leq q_1 < 1.5$ and $q_2 < 0.5$ or $q_2 \geq 2.5$ or 2 objects present with $q_1 < 0.5$ or $q_1 \geq 2.5$ and $0.5 \leq q_2 < 1.5$
10	1 object present with $1.5 \leq q < 2.5$ or 2 objects present with $1.5 \leq q_1 < 2.5$ and $1.5 \leq q_2 < 2.5$ 2 objects present with $1.5 \leq q_1 < 2.5$ and $q_2 < 0.5$ or $q_2 \geq 2.5$ or 2 objects present with $q_1 < 0.5$ or $q_1 \geq 2.5$ and $1.5 \leq q_2 < 2.5$
11	2 objects present with $0.5 \leq q_1 < 1.5$ and $1.5 \leq q_2 < 2.5$ or 2 objects present with $1.5 \leq q_1 < 2.5$ and $0.5 \leq q_2 < 1.5$.

Table 1.1: Relationship between the expected observations symbol $\mathbf{o}_t$ (left column) and the event in the state space (right column).

preventing two objects from occupying the same space at the same time.

The four different observation symbols will split the state space $\mathbb{S}^\infty$ into four different events listed in Table 1.1. Figure 1.4 illustrates the state space $\mathbb{S}^\infty$ as an union of a plane, a line and a point with the four different events show in different colours.

When working with this kind of model, the image $\mathbf{o}_t$ is measured, and the likelihood of this measurement is given by the observation noise function

$$f\left(\mathbf{o}_t \left|\hat{\mathbf{o}}_t\right.\right) = \begin{cases} p_{00} & \text{if } \hat{\mathbf{o}}_t = 00 \\ p_{01} & \text{if } \hat{\mathbf{o}}_t = 01 \\ p_{10} & \text{if } \hat{\mathbf{o}}_t = 10 \\ p_{11} & \text{if } \hat{\mathbf{o}}_t = 11 \end{cases}, \tag{1.118}$$

e.g the values of p_{00}, p_{01}, p_{10} and p_{11} are given by the measurement. Each point in the state space corresponds to a specific expected observation $\hat{o}_t$, which means that from the measurement, $f\left(o_t \left|q\right.\right)$ can be derived. It will be a discontinuous function looking like the illustration in Figure 1.4, where the values of the different regions will be p_{00}, p_{01}, p_{10} and p_{11}. Using Bayes rule on this gives

$$f\left(\mathbf{q} \left|\mathbf{o}_t\right.\right) = \frac{f\left(\mathbf{o}_t \left|\mathbf{q}\right.\right) f\left(\mathbf{q}\right)}{f\left(\mathbf{o}_t\right)} = \frac{f\left(\mathbf{o}_t \left|\mathbf{q}\right.\right) f\left(\mathbf{q}\right)}{\int f\left(\mathbf{o}_t \left|q\right.\right) f\left(\mathbf{q}\right) \mathrm{d}q}. \tag{1.119}$$

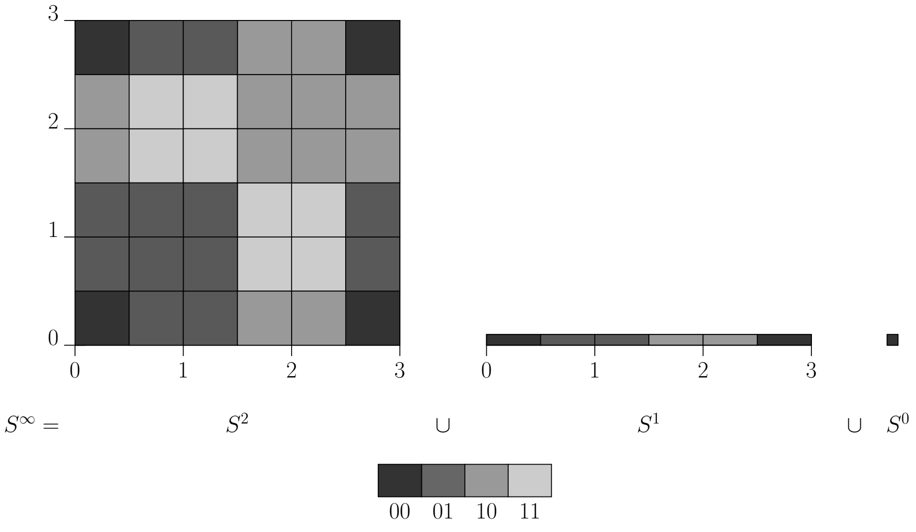


Figure 1.4: Illustration of the multi object state space as a union of a plane a line and a point, with the different events corresponding to the different observation symbols marked in different colours.

With a uniform prior, $f(\mathbf{q})$ factors out and cancels, and the integral can be calculated. For this example, the area of the squares in $\mathbb{S}^2$ in Figure 1.4 is 0.25, the length of the segments in $\mathbb{S}^1$ is 0.5 and the measure of $\mathbb{S}^0$ is 1. This gives

$$\int f(o_t|q)\,\mathrm{d}q = p_{00}(0.25*4+0.5*2+1)+p_{01}(0.25*12+0.5*2+0)+ \\ +p_{00}(0.25*12+0.5*2+0)+p_{00}(0.25*8+0.5*2+0)\,. \quad (1.120)$$

In particle the prior $f(\mathbf{q})$ can be used to zero out areas of the state space where for example object occupy the same physical space and to incorporate the information learnt from previous frames.

It has been claimed that with this kind of observations it is possible to use $f(\mathbf{o}_t|\mathbf{q})\,\mathbf{f}(\mathbf{q})$ as likelihoods in $\mathbb{S}^\infty$ and that this will make the likelihoods comparable. This is for example done by Isard and MacCormick [33] and Zhao and Nevatia [92]. However it does not remove the problem of the varying dimension, but if only the state at a single point in time is considered, comparing such likelihoods is the same thing as discretising the space into the regions shown Figure 1.4 and comparing the probabilities of each such subset. This is a somewhat strange discretising, but it makes sens in that this is the sets the measurements can distinguish between. Likelihood ratios will no longer depend on the scale of the coordinate system but instead they will depend on the type of camera and its placement, orientation and calibration. However, if a dynamic model is introduced, connecting the states over time, likelihood ratios conditioned on past observations as well as the current observation will again become undefined (dependent on the coordinate system scale). This means that likelihoods of state sequences can only be compared with likelihoods of state sequences when the entire sequence of state dimensions are identical.

Another way to extend $\mathbb{S}^1$ to several objects is to add a single state S_0 representing the the situation of the object not being present. Then it is possible to let the space

$$\hat{\mathbb{S}}^n = \left(\mathbb{S}^1 \cup S_0\right)^n \tag{1.121}$$

track a varying number of present objects. Consider as before the case of $0 - 2$ object,

$$\hat{\mathbb{S}}^2 = \left(\mathbb{S}^1 \cup S_0\right) \times \left(\mathbb{S}^1 \cup S_0\right) = \mathbb{S}^2 \cup \left(\mathbb{S}^1 \times S_0\right) \cup \left(S_0 \times \mathbb{S}^1\right) \cup \left(S_0 \times S_0\right). \tag{1.122}$$

This is a plane, two lines and a point. The only difference from before is that now there are two lines instead of one. The difference is that in this space the first and the second object are different, e.g. the previous state "one object present at position x" is now replaced with two different states "first object present at position x" and "second object present at position x".

Markov Chain Monte-Carlo

Some care is needed when applying the Markov chain Monte-Carlo simulation from Section 1.3.1 to this kind of multi target state space. It is a fairly complex state space and it might be difficult to cover the entire state space with only one kind of proposal distribution. Fortunately it is possible to use several different types of moves each specified using a separate proposal distribution, $\tilde{f}_k\left(\mathbf{q}_t \left|\mathbf{q}_{t-1}\right.\right)$, $k = 1, 2, \cdots$, and in each step choose randomly which of the moves to use. If each of the move types can be shown to fullfill the global balance condition, Equation 1.67, the full combined Markov chain will do so too [80]. Typically there will be move types for each of the $\mathbb{S}^n$ subspaces moving the objects present around within this subspace. There will also be move types jumping between adjacent subspaces, $\mathbb{S}^n$ and $\mathbb{S}^{n+1}$ by adding and removing a single object. The moves staying within one subspace can be constructed exactly as before using the Metropolis-Hastings algorithm. To handle the moves moving between the subspace more care has to be taken as the distributions involved becomes singular. Green *et al* [21, 67] has extended the Metropolis-Hastings algorithm to these cases. They derive the theory for very general measurable spaces. A somewhat simpler derivation that does not rely as heavily on measure theory is provided by [85]. Below is yet another approach to the same theory. It is intended to give an intuitive understanding of the theory rather than being a strict treatment.

To represent singular distributions the *Dirac delta function*, $\delta_t\left(x\right)$, will be used. It is defined by the property,

$$\int_{\mathcal{A}} g\left(x\right) \delta_t\left(x\right) \mathrm{d}x = \begin{cases} g\left(t\right) & \text{if } t \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}, \tag{1.123}$$

for any continuous function $g(\cdot)$. By allowing the proposal distribution, $\tilde{f}(\mathbf{q}_t|\mathbf{q}_{t-1})$, to contain such function all cases typically of interests in many applications can be handled by treating $\tilde{f}(\mathbf{q}_t|\mathbf{q}_{t-1})$, as a distribution of $\mathbf{q}_t$ over $\mathbb{S}^\infty$. In that case the ratio in the acceptance probability (1.68) will always be a ratio of likelihoods of the same dimension. That's the case because if $\mathbf{q}_{t-1} \in \mathbb{S}^n$ and $\mathbf{q}_t \in \mathbb{S}^m$ then $\tilde{f}(\mathbf{q}_t|\mathbf{q}_{t-1})$ will be a likelihood from $\mathbb{S}^m$ and $f_s(\mathbf{q}_{t-1})$ will be a likelihood from $\mathbb{S}^n$. The denominator is the product of the two, which means that it is a likelihood from $\mathbb{S}^m \times \mathbb{S}^n = \mathbb{S}^{m+n}$. In the same way, $\tilde{f}(\mathbf{q}_{t-1}|\mathbf{q}_t)$ is a likelihood from $\mathbb{S}^n$ and $f_s(\mathbf{q}_t)$ is a likelihood from $\mathbb{S}^m$ which makes the nominator a likelihood from $\mathbb{S}^n \times \mathbb{S}^m = \mathbb{S}^{m+n}$. However, for this fraction to be defined it is no longer enough that the denominator is not 0. All $\delta_t(x)$ function must cancel out in the formal expression

$$\frac{\tilde{f}(\mathbf{q}_{t-1}|\mathbf{q}_t)}{\tilde{f}(\mathbf{q}_t|\mathbf{q}_{t-1})}. \tag{1.124}$$

This is often referred to as *dimension matching*. This formulation makes it clear that it is not only the dimension, i.e. the number of delta functions, that has to match, but also the positions of those delta functions. This means that a move that allows new objects to enter also has to allow objects to exit and the exit sub-move has to be the inverse move of the entry sub-move. Checking that the delta functions cancel out in the fraction above gives a way of verifying that this is the case and that the sub-moves thus corresponds in the way they have to.

As an example of this consider the space $\mathbb{R}^2 \cup \mathbb{R}$, and the task of generating samples from

$$f_s(\mathbf{q}) = \begin{cases} f_1(q) & \text{if } \mathbf{q} = (q) \in \mathbb{R} \\ f_{21}(q_1) f_{22}(q_2) & \text{if } \mathbf{q} = (q_1, q_2) \in \mathbb{R}^2 \end{cases}, \tag{1.125}$$

where $f_1(\cdot)$, $f_{21}(\cdot)$ and $f_{22}(\cdot)$ are one dimensional distributions over $\mathbb{R}$. Consider a few moves from $\mathbf{x} = \mathbf{q}_{t-1}$ to $\mathbf{y} = \mathbf{q}_t$. One move could be to add a new object by moving from $\mathbf{x} = (x)$ to $\mathbf{y} = (y_1, y_2)$ by letting $y_1 = x$ and choose the position of the new object, y_2, from a $\mathcal{N}(\cdot|0,1)$ distribution. The corresponding exit move will then be to remove the object at y_2 and let $x = y_1$. The proposal distribution, becomes

$$\tilde{f}(\mathbf{y}|\mathbf{x}) = \begin{cases} \delta_x(y_1)\mathcal{N}(y_2|0,1) & \text{if } \mathbf{x} \in \mathbb{R} \text{ and } \mathbf{y} \in \mathbb{R}^2 \\ \delta_{x_1}(y) & \text{if } \mathbf{x} \in \mathbb{R}^2 \text{ and } \mathbf{y} \in \mathbb{R} \\ 0 & \text{otherwise} \end{cases}, \tag{1.126}$$

and the acceptance probability becomes the minimum of 1 and

$$
\begin{aligned}
\frac{\tilde f(\mathbf{x}|\mathbf{y})f_s(\mathbf{y})}{\tilde f(\mathbf{y}|\mathbf{x})f_s(\mathbf{x})} &= \begin{cases} \frac{\delta_{y_1}(x)f_{21}(y_1)f_{22}(y_2)}{\delta_x(y_1)\mathcal{N}(y_2|0,1)f_1(x)} & \text{if } \mathbf{x}\in\mathbb{R} \text{ and } \mathbf{y}\in\mathbb{R}^2 \\ \frac{\delta_y(x_1)\mathcal{N}(x_2|0,1)f_1(y)}{\delta_{x_1}(y)f_{21}(x_1)f_{22}(x_2)} & \text{if } \mathbf{x}\in\mathbb{R}^2 \text{ and } \mathbf{y}\in\mathbb{R} \end{cases} . \\
&= \begin{cases} \frac{f_{21}(y_1)f_{22}(y_2)}{\mathcal{N}(y_2|0,1)f_1(x)} & \text{if } \mathbf{x}\in\mathbb{R} \text{ and } \mathbf{y}\in\mathbb{R}^2 \\ \frac{\mathcal{N}(x_2|0,1)f_1(y)}{f_{21}(x_1)f_{22}(x_2)} & \text{if } \mathbf{x}\in\mathbb{R}^2 \text{ and } \mathbf{y}\in\mathbb{R} \end{cases}
\end{aligned} \tag{1.127}
$$

Note that this fraction is not defined for moves within the same subspace. This is not a problem since the probability of such a move (using this move type) is 0 and can thus be defined arbitrarily.

Another type of move could be to combine two objects that are close to each other into one object by letting $y = \frac{x_1+x_2}{2}$. The corresponding splitting move will then have to be constructed from a single sample u from some one dimensional distribution, say $\mathcal{N}(\cdot\,|0,1)$, by letting $x_1 = y - u$ and $x_2 = y + u$. The proposal distribution, becomes

$$
\tilde f(\mathbf{y}|\mathbf{x}) = \begin{cases} \delta_x\left(\frac{y_1+y_2}{2}\right)\mathcal{N}\left(\frac{y_2-y_1}{2}|0,1\right) & \text{if } \mathbf{x}\in\mathbb{R} \text{ and } \mathbf{y}\in\mathbb{R}^2 \\ \delta_{\frac{x_1+x_2}{2}}(y) & \text{if } \mathbf{x}\in\mathbb{R}^2 \text{ and } \mathbf{y}\in\mathbb{R} \\ 0 & \text{otherwise} \end{cases} , \tag{1.128}
$$

and the acceptance probability becomes the minimum of 1 and

$$
\begin{aligned}
\frac{\tilde f(\mathbf{x}|\mathbf{y})f_s(\mathbf{y})}{\tilde f(\mathbf{y}|\mathbf{x})f_s(\mathbf{x})} &= \begin{cases} \frac{\delta_{\frac{y_1+y_2}{2}}(x)f_{21}(y_1)f_{22}(y_2)}{\delta_x\left(\frac{y_1+y_2}{2}\right)\mathcal{N}\left(\frac{y_2-y_1}{2}|0,1\right)f_1(x)} & \text{if } \mathbf{x}\in\mathbb{R} \text{ and } \mathbf{y}\in\mathbb{R}^2 \\ \frac{\delta_y\left(\frac{x_1+x_2}{2}\right)\mathcal{N}\left(\frac{x_2-x_1}{2}|0,1\right)f_1(y)}{\delta_{\frac{x_1+x_2}{2}}(y)f_{21}(x_1)f_{22}(x_2)} & \text{if } \mathbf{x}\in\mathbb{R}^2 \text{ and } \mathbf{y}\in\mathbb{R} \end{cases} . \\
&= \begin{cases} \frac{f_{21}(y_1)f_{22}(y_2)}{\mathcal{N}\left(\frac{y_2-y_1}{2}|0,1\right)f_1(x)} & \text{if } \mathbf{x}\in\mathbb{R} \text{ and } \mathbf{y}\in\mathbb{R}^2 \\ \frac{\mathcal{N}\left(\frac{x_2-x_1}{2}|0,1\right)f_1(y)}{f_{21}(x_1)f_{22}(x_2)} & \text{if } \mathbf{x}\in\mathbb{R}^2 \text{ and } \mathbf{y}\in\mathbb{R} \end{cases}
\end{aligned} \tag{1.129}
$$

Particle Filter

Isard and MacCormick has suggested [33] to apply a particle filter on $\mathbb{S}^\infty$ to do multi target tracking of pedestrians. They call the resulting algorithm *BraMBLe*.

The state of a single object is defined to be its centre position on the ground plane, its velocity in the ground plane, a set of shape parameters that specifies a generalised cylinder and a unique id-number. The state $\mathbf{q}_t$, representing the entire scene is defined to be a set of such single object states.

The dynamic model used states that

- object arrivals are Poisson distributed,
- object departures are Poisson distributed,
- the objects translations are damped constant velocity with additive Gaussian noise,
- the shape parameters obeys a 1st order auto-regressive process and
- unique id-numbers never changes and are assigned to each entry.

This forms a prediction function, $\mathbf{h}(\cdot)$, that given a state from the previous frame, $\mathbf{q}_{t-1}$, and a noise vector $\mathbf{w}_t$ forms a prediction of the current state $\mathbf{q}_t^- = \mathbf{h}(\mathbf{q}_{t-1}, \mathbf{w}_t)$.

The observations $\mathbf{o}_t$ are the images from a camera viewing the scene. For a given predicted state $\mathbf{q}_t^-$, the expected scene is rendered using the generalised cylinders of all objects present. From this a binary mask is produced that for each block of 5×5 pixels indicates whether that block show the background or one of the moving objects, the foreground. A set of filters are applied to the image and the distributions of the filter-responses are modelled as mixtures-of-Gaussians. One distribution for background responses and one for foreground responses are learnt offline. The filters are small as compared to the 5×5 pixel blocks, which means that the responses for the different blocks are fairly independent. By assuming them to be independent, an observation likelihood, $f\left(o_t \left| q_t^- \right.\right)$, is formed by taking the product over all blocks.

This allows a set of m particles, $\left\{ \mathbf{q}_t^{(j)} \middle| j = 1 \cdots m \right\}$, and corresponding weights $\alpha_t^{(j)}$, representing the $f\left(\mathbf{q}_t \left| \mathcal{O}_{0 \cdots t} \right.\right)$ distribution, to be calculated as described in Section 1.3.1. The final step of the particle filter is to calculate the mean of those particles to form a single output state. This will not work here as $\mathbb{S}^\infty$ is not a vector space and thus it is not possible to form linear combinations. Instead each object id, Φ, is considered separately, and the probability of this object being visible is estimated as

$$\Pi_\Phi = \sum_{\left\{ j \middle| q_t^{(j)} \text{has an object with id } \Phi \right\}} \alpha_t^{(j)}. \tag{1.130}$$

If this probability is larger than some threshold, the object is considered visible and its position is calculated as the weighted mean over its position in all particles in which it is present. The weight used for its position in particle j is $\frac{\alpha_t^{(j)}}{\Pi_\Phi}$.

Note that the likelihoods are never compared. Instead the probability of being in each subspace is calculated by integrating out everything else. In this case there is actually several subspaces of the same dimension as the different objects have different id-numbers.

There will be one subspace for each possible set of object id-numbers. The subspace with highest probability is found approximately by estimating the probability of each object being visible separately. This avoids an optimisation over all possible subsets, which would be required if no approximation were made. This is done separately for each frame, which means that there can be inconsistent jumps between the subsets if a less likely mode becomes the dominant mode when new observations are made.

1.4 Traffic Surveillance

In this section a few examples especially targeted at traffic surveillance will be presented. Manny of them of will in some way make use of the more general methods presented in the previous section. Unless otherwise mentioned, the setup will consist of a single, static, calibrated camera viewing an intersection or road segment, and all road users are assumed to move on a known ground plane. The task is to generate tracks describing the motion of the road users using the video produced by the camera.

1.4.1 3D Wireframe Models

Koller, Danilidis and Nagel presents [46] a parametrised wireframe 3D model of a vehicle with 12 length parameters that allows the model to represent different types of vehicles, such as sedan, hatchback, station waggon, minivan and pickup. The values of the parameters are assumed to be know a priori. Also, the vehicles are assumed to move on a known ground plane, which means that the position (of the centre) of the object can be specified with a two dimensional vector, (x, y) and the orientation is a single angle θ, which also defines the traveling direction. The state of a object will consist of those values together with a translational velocity, $\dot{r}$, and an angular velocity $\dot{\theta}$, e.g

$$\mathbf{q}_t = \left(x_t, y_t, \theta_t, \dot{r}_t, \dot{\theta}\right) \in S. \tag{1.131}$$

To use this model in a Kalman fashion, a dynamic model, $\mathbf{q}_t = \mathbf{h}(\mathbf{q}_{t-1}, \mathbf{w}_t)$, is needed. It is constructed by assuming zero acceleration, which gives a deterministic update, and then adding Gaussian noise, $\mathbf{w}_t = (w_{t,1}, \cdots, w_{t,5})$, with mean zero and some covariance matrix $\mathbf{Q}$. The prediction equations used is found by integrating the differential equations

$$\left\{ \begin{array}{lcl} \frac{\mathrm{d}x}{\mathrm{d}t} & = & \dot{r}\cos\theta \\ \frac{\mathrm{d}y}{\mathrm{d}t} & = & \dot{r}\sin\theta \\ \frac{\mathrm{d}\theta}{\mathrm{d}t} & = & \dot{\theta} \\ \frac{\mathrm{d}\dot{r}}{\mathrm{d}t} & = & 0 \\ \frac{\mathrm{d}\dot{\theta}}{\mathrm{d}t} & = & 0 \end{array} \right. , \tag{1.132}$$

from one time step to the next. If the distance between the time steps is normalised to 1, this gives the prediction function $\mathbf{h}(\cdot)$ as

$$\mathbf{h}(\cdot):\begin{cases} x_t &= x_{t-1} + \dot{r}\frac{\sin\left(\theta_{t-1}+\dot{\theta}_{t-1}\right)-\sin(\theta_{t-1})}{\dot{\theta}_{t-1}} + w_{t,1} \\ y_t &= x_{t-1} - \dot{r}\frac{\cos\left(\theta_{t-1}+\dot{\theta}_{t-1}\right)-\cos(\theta_{t-1})}{\dot{\theta}_{t-1}} + w_{t,2} \\ \theta_t &= \theta_{t-1} + \dot{\theta}_{t-1} + w_{t,3} \\ \dot{r}_t &= \dot{r}_{t-1} + w_{t,4} \\ \dot{\theta}_t &= \dot{\theta}_{t-1} + w_{t,5} \end{cases}, \tag{1.133}$$

which is not defined if $\dot{\theta}_{t-1} = 0$. That problem is solved by defining the fractions in the expression to be 1 if $\dot{\theta}_{t-1}$ is small. That would correspond to a pure translational motion.

The observations $\mathbf{o}_t$ are formed by taking the 3D wireframe model and placing it in the world coordinate system on the ground plane centred at (x_t, y_t), rotated θ_t radians and projecting it into the camera image. The wireframe model consits of a set of 3D line segments that when projected into the image forms a set of line segments in the image. The line segments that are not visible (e.g. those on the back side of the vehicle) are removed using a hidden line removal algorithm. Also if the sun direction is known, a vehicle shadow can be estimated by projecting the model onto the ground plane and then projecting the shadow into the camera image. This gives a few additional line segments. This makes the observation, $\mathbf{o}_t$, a set of m line segments, the *model segments*,

$$\mathbf{o}_t = \{\mathbf{o}_{t,1}, \cdots, \mathbf{o}_{t,m}\}, \tag{1.134}$$

where each line segment $\mathbf{o}_{t,i} = (c_x, c_y, \phi, l)$ is represent by its centre position, (c_x, c_y), angle ϕ and length l. These parameters are related to the endpoints (x_1, y_1) and (x_2, y_2) of the line segment according to

$$\begin{array}{rcl} c_x &=& \frac{x_1+x_2}{2} \\ c_y &=& \frac{y_1+y_2}{2} \\ \phi &=& \arctan\left(\frac{y_2-y_1}{x_2+x_1}\right) \\ l &=& \sqrt{\left(x_2-x_1\right)^2 + \left(y_2-y_1\right)^2} \end{array}. \tag{1.135}$$

The prediction achieved by a detector extracting such segments from an image is typically significantly worse along the line direction than perpendicular to it. By assuming the variance of the positions of the endpoints to be $\sigma_{\|}^2$ along the line and $\sigma_{\perp}^2$ perpendicular to it and that the detections are Gaussian distributed, a covariance matrix for $\mathbf{o}_{t,i}$ can be found, according to

$$\tilde{\mathbf{R}}_i = \begin{pmatrix} \frac{\sigma_{\|}^2\cos^2\phi+\sigma_{\perp}^2\sin^2\phi}{2} & \frac{\left(\sigma_{\|}^2-\sigma_{\perp}^2\right)\sin^2\phi\cos^2\phi}{2} & 0 & 0 \\ \frac{\left(\sigma_{\|}^2-\sigma_{\perp}^2\right)\sin^2\phi\cos^2\phi}{2} & \frac{\sigma_{\perp}^2\cos^2\phi+\sigma_{\|}^2\sin^2\phi}{2} & 0 & 0 \\ 0 & 0 & \frac{2\sigma_{\perp}^2}{l^2} & 0 \\ 0 & 0 & 0 & 2\sigma_{\|}^2 \end{pmatrix}. \tag{1.136}$$

The covariance matrix of $\mathbf{o}_t$ will then be a block diagonal matrix,

$$\mathbf{R}_t = \begin{pmatrix} \tilde{\mathbf{R}}_1 & \mathbf{0}_{4\times 4} & \mathbf{0}_{4\times 4} & \cdots & \mathbf{0}_{4\times 4} \\ \mathbf{0}_{4\times 4} & \tilde{\mathbf{R}}_2 & \mathbf{0}_{4\times 4} & \cdots & \mathbf{0}_{4\times 4} \\ & & \cdots & & \\ \mathbf{0}_{4\times 4} & \cdots & \mathbf{0}_{4\times 4} & \mathbf{0}_{4\times 4} & \tilde{\mathbf{R}}_m \end{pmatrix}. \tag{1.137}$$

This gives the observations function $\mathbf{o}_t = \mathbf{o}(\mathbf{q}_t, \mathbf{v}_t)$, which consists of a nonlinear function that generates a set of line segments from the state $\mathbf{q}_t$ and a noise process, $\mathbf{v}_t$, which is Gaussian distributed and added to endpoints of those line segments.

To generate observations of this form the input image is analysed by a straight line detector that will approximate an entire image as a set of n straight line segments. These segments are called the *data segments*, and there will typically be significantly more data segments than model segments. Applying the extended Kalman filter to this model becomes a bit tricky since the correspondences between the data segments and the model segments are not known, but it can be done in an iterative manner that also updates the assignments. The state distribution is approximated as Gaussian,

$$f\left(\mathbf{q}_t \left| \mathcal{O}_{0\cdots t} \right.\right) = \mathcal{N}\left(\mathbf{q}_t \left| \hat{\mathbf{q}}_t, \hat{\mathbf{P}}_t \right.\right) \tag{1.138}$$

and assumed known for the previous frame, $t-1$. It is propagated forward in time

$$f\left(\mathbf{q}_t \left| \mathcal{O}_{0\cdots t-1} \right.\right) = \mathcal{N}\left(\mathbf{q}_t \left| \hat{\mathbf{q}}_t^{\,-}, \hat{\mathbf{P}}_t \right.\right) \tag{1.139}$$

with

$$\hat{\mathbf{q}}_t^{\,-} = \mathbf{h}(\hat{\mathbf{q}}_{t-1}, \mathbf{0}), \tag{1.140}$$

$$\mathbf{P}_t^- = \mathbf{H}_t \mathbf{P}_{t-1} \mathbf{H}_t^{\mathrm{T}} + \mathbf{Q}, \tag{1.141}$$

where $\mathbf{H}_t$ is the jacobian of $\mathbf{h}(\cdot)$ defined in Equation 1.133 evaluated at $\hat{\mathbf{q}}_{t-1}$. From this predicted state it is possible to form an expected set of line segments $\mathbf{o}_t^- = \mathbf{o}(\mathbf{q}_t, \mathbf{0})$ which is approximated to be Gaussian distributed with covariance matrix $\tilde{\mathbf{S}}_t = \mathbf{O}_t \mathbf{P}_t^- \mathbf{O}_t^{\mathrm{T}}$, where $\mathbf{O}_t$ is the jacobian of the observation function $\mathbf{o}(\cdot)$ defined above evaluated at $\hat{\mathbf{q}}_t^{\,-}$, which typically is estimated numerically. The predicted observation $\mathbf{o}_t^-$ consists of a set of line segments, $\mathbf{o}_{t,j}^-$. Covariance matrices, $\tilde{\mathbf{S}}_{t,j}$ corresponding to each of them can be extracted from $\tilde{\mathbf{S}}_t$ by extracting the 4×4 matrix blocks along the diagonal.

The next step is to calculate the innovation $\mathbf{o}_t - \mathbf{o}_t^-$. Which means that the correspondence between the measured data segments $\mathbf{o}_t$ and the predicted model segments $\mathbf{o}_t^-$ has to be established. Each model segment j is assigned to the data segment i that minimises the Mahalanobis distance

$$\left(\mathbf{o}_{t,j}^- - \mathbf{o}_{t,i}\right)^{\mathrm{T}} \left(\tilde{\mathbf{S}}_{t,j} + \tilde{\mathbf{R}}_{t,i}\right)^{-1} \left(\mathbf{o}_{t,j}^- - \mathbf{o}_{t,i}\right). \tag{1.142}$$

Model segments where this distance is larger than some threshold are discarded. After the associations are fixed, it is straightforward to apply the rest of the extended Kalman filter as it is described above. The updating step and the segment assignment step is placed in a loop and iterated to test a few different assignments. Once a previously tested assignment is reached again, or a certain number of iterations have been performed, the loop can be terminated, and the results from the assignments with minimal residual is chosen as the result of the updating step. As residual, the average innovation weighted with weights that takes the length of the segment into account, is used.

Note that $\tilde{\mathbf{S}}_t$ is the variance of the actual position of the line segments. The segments are then measured with measurements noise and the covariance matrix of the measurement is $\mathbf{S}_t = \tilde{\mathbf{S}}_t + \mathbf{R}_t$, which is the covariance matrix used by the extended Kalman filter. Here $\mathbf{R}_t$ refers to the model segments and not the data segments as before.

Initialisation of the model is performed by a motion segmentation step that tracks image features. Features moving in a coherent way is clustered together into vehicles and projected onto the ground plane. The centre of the projected features are then assumed to be the centre of the vehicle on the ground plane and and the average displacement direction gives the orientation. By making this kind of detections in two or more adjacent frames, estimates of the velocities are made from discrete time derivatives. This assumes that all road users are separated well enough not to be clustered into the same cluster, e.g. no occlusions, and that the shape parameters of the car model is known.

A more elaborate initialisation algorithm is provided by Tan, Sullivan and Baker [79]. It considers all assignments of a single model segment to a single data segment. Such an assignment induces some restrictions on the possible values the state variables x, y and θ for the state to be consistent with the assignment. By using a generalised Hough transform the state consistent with the largest number of assignments can be chosen.

Consider first a single model/data segment assignment. If there were no measurement noise such an assignment would imply two possible values for θ differing by π radians. If measurement noise is also considered those two possibilities will be increased into two intervals. In the paper [79] a probability distribution $f_\theta\left(\theta\left|i,j\right.\right)$ of θ is derived given model segment j is assigned to the data segment i. The shape of that function will depend on the accuracy of the line detector used which typically increases with longer lines (where the length is measured in the image) and it will depend on the 3D angle of the line. A line perpendicular to the ground plane will not give any information at all, which means $f_\theta\left(\theta\left|i,j\right.\right)$ becomes uniform on the entire interval. Lines almost perpendicular will thus make $f_\theta\left(\theta\left|i,j\right.\right)$ close to uniform, while lines almost parallel to the plane will make $f_\theta\left(\theta\left|i,j\right.\right)$ close to two sharp peaks. The model/data segment assignments are then integrated out and the kernel density estimation from Equation 1.83 is used to combine the measurements of the different segments into a single distribution of θ. With n model segments and m data segments that gives a probability distribution of θ given the observed image,

$$f(\theta) = \frac{1}{nm} \sum_{i=1}^{m} \sum_{j=1}^{n} f_\theta \left(\theta \left| i, j \right. \right). \tag{1.143}$$

This function is approximated with a histogram that is Gaussian smoothed, and strong peaks in the the histogram are extracted as candidate values for θ. Typically there will be four strong peaks at $\pi/2$ intervals, since there are two dominate directions for all lines of a vehicle.

For each candidate θ, all data and model segments not contributing to its peak are discarded. Each remaining segments will restrict the position, (x, y), of the vehicle to slide along some line segment called a *confusion line segment*. The contribution of the different segments can be combined by considering an accumulator image $A(\cdot)$ where each pixel $A(x, y)$ is the number of confusion line segments passing through the position (x, y). By smoothing this image by a Gaussian convolution and detecting peaks, a set of candidate positions (x, y) can be extracted for each candidate θ.

This results in a set of candidate triplets (x, y, θ). They are compared using some feature score and the one with the most likely feature score is chosen and the rest discarded. To calculate this feature score each visible line segment l is considered separately and a $w \times h$ subimage, I, is extracted form the input frame. The subimage is centred around the data segment with the segment parallel to its x-axis and the endpoints of the segment on the its left and right borders. The feature score, s_l is then defined as

$$s_l = \max_{1 \leq y < h} \left(m_{y+1} - m_y \right) - \min_{1 \leq y < h} \left(m_{y+1} - m_y \right), \tag{1.144}$$

where

$$m_y = \frac{1}{w} \sum_{x=1}^{w} I(x, y). \tag{1.145}$$

The distribution, $f(s_l)$, of s_l is approximated by a normalised histogram offline from some training examples. With some independence assumption, the log likelihood of each candidate triplet becomes

$$\sum_{l} \log \left(f \left(s_l \right) \right), \tag{1.146}$$

and the candidate triplet with highest likelihood is chosen.

Some discussion is given on how to handle the case when there is more than one object present. After the object with highest score is detected, the line segments corresponding to it can be removed and the algorithm repeated under some geometrical constraints preventing objects form occupying the same 3D space to find the next one. It is also suggested to handle the case of zero objects by thresholding the feature score likelihood. This would give some ad hoc way of estimating the number of objects present, but there is nothing in the theory to back this up. In the more complicated experiments presented, the image was manually segmented into regions containing exactly one car each.

1.4.2 Bag Of Words Detectors

The above ideas can be generalised to use several different kinds of features extracted from images and also to allow variations in the model to be able to track non-rigid objects such as pedestrians. The bag of words detector is a general purpose type of detector with such capabilities that also allows the models to be learnt from training data. The idea is to break down an image into a set of visual words from some predefined bag of words, and use training data to learn which words corresponds to which positions on what types of objects.

Leibe *et al* presented [52] one such model that also allows reasoning about the number of objects in the scene using an minimum description length (MDL) approach. It is a general purpose detector, but results are presented on detecting road users such as pedestrians, cars, motorbikes and cows. The model does not consider full 3D objects, but considers objects to be two dimensional but with varying scale. Detecting for example cars can still be done by using separate detectors for cars viewed from different angles.

Codebook

A codebook is generated from a set of training images by applying a feature point detector and extracting a local feature descriptor vector around those detections. Several such detectors and descriptors exists, a good choice is Lowes's SIFT [54]. It will in addition to the position in the image also detect the scale of the feature point, which means that the same features can be detected even if the image is rescaled, and the scale of the feature will specify how much the image has been rescaled. This makes the detector independent of object size. The feature vectors are then clustered into clusters containing similar features. Again several clustering methods are available and the suggestion in the paper by Leibe *et al* is to use agglomerative clustering. It has the advantage over for example k-means clustering in that the number of clusters don't have to be specified manually, but is determined by the clustering algorithm.

Agglomerative clustering is an iterative process that operates on a set of disjoint clusters $\mathcal{X}_i$, where each cluster consists of a set of feature vectors $\mathbf{x}_{i,j}$, i.e. $\mathcal{X}_i = (\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \cdots)$. It can be initiated by letting each cluster consists of a single feature vector. A similarity measure, $\mu(\mathcal{X}, \mathcal{Y})$, between two clusters, $\mathcal{X}$ and $\mathcal{Y}$ has to be chosen. This can for example be the group average over some similarity measure, $\mu(\mathbf{x}, \mathbf{y})$ between two feature vectors $\mathbf{x}$ and $\mathbf{y}$, giving

$$\mu(\mathcal{X}, \mathcal{Y}) = \frac{1}{|\mathcal{X}|\,|\mathcal{Y}|} \sum_{\mathbf{x}\in\mathcal{X}} \sum_{\mathbf{y}\in\mathcal{Y}} \mu(\mathbf{x}, \mathbf{y}) . \tag{1.147}$$

Here $\mu(\mathbf{x}, \mathbf{y})$ could be the cross correlation coefficient or the negated euclidean distance. The algorithm iteratively merges the two clusters with largest similarity as long as the similarity measure stay above some specified threshold, α.

Existing implementations of this algorithm for general similarity measures are typically quite slow and requires a lot memory, which makes it impossible to use them for clustering large sets of feature vectors. However if the similarity measure fullfills *Burynooghe's reducibility property* an efficient algorithm exists (see [52] for references) that gives run time performance and memory requirements similar to k-means. This property demands that when two clusters, $\mathcal{X}$ and $\mathcal{Y}$ are merged, the similarity between those two clusters and any third cluster, $\mathcal{Z}$ may only decrease. More formaly, the implication

$$\left\{ \begin{array}{lcl} \mu(\mathcal{X},\mathcal{Y}) & \geq & \mu(\mathcal{X},\mathcal{Z}) \\ \mu(\mathcal{X},\mathcal{Y}) & \geq & \mu(\mathcal{Y},\mathcal{Z}) \end{array} \right. \Rightarrow \left\{ \begin{array}{lcl} \mu(\mathcal{X},\mathcal{Z}) & \geq & \mu(\mathcal{X}\cup\mathcal{Y},\mathcal{Z}) \\ \mu(\mathcal{Y},\mathcal{Z}) & \geq & \mu(\mathcal{X}\cup\mathcal{Y},\mathcal{Z}) \end{array} \right. \tag{1.148}$$

has to be true for all clusters $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$. This property holds for the group average similarity measure regardless of which similarity measure between two features are chosen.

After the clustering processes, each cluster, $\mathcal{X}_i$ will be considered a visual word and the representation of each visual word will be the mean value,

$$\mathbf{m}_i = \frac{1}{|\mathcal{X}_i|} \sum_{\mathbf{x}\in\mathcal{X}_i} \mathbf{x}. \tag{1.149}$$

This representation only makes sens for some similarity measures, $\mu(\mathcal{X},\mathcal{Y})$, such as for example the euclidian distance. A codebook $\mathcal{C} = (\mathbf{m}_1, \mathbf{m}_2, \cdots)$ consisting of a set of visual words is formed in this way. An image can be decomposed into visual words by using the feature point detector, and for each detected feature point, extracting a feature vector $\mathbf{x}$ and identifying all visual words with a similarity larger than the threshold α that was used to terminate the clustering, i.e. all visual words $\mathbf{m}_i$ for which $\mu(\mathbf{x}, \mathbf{m}_i) \geq \alpha$.

Implicit Shape Model

An *implicit shape model*, or ISM, consists of a codebook, $\mathcal{C} = (\mathbf{m}_1, \mathbf{m}_2, \cdots)$, and for each visual word $\mathbf{m}_i$ in this codebook a probability distribution $f_\mathbf{w}(\mathbf{w}\,|\mathbf{m}_i)$, with $\mathbf{w} = (w_x, w_y, w_s)$, over the spacial location $(w_x, w_y) \in \mathbb{R}^2$ of where on the object the visual word $\mathbf{m}_i$ is located and at what scale $w_s \in \mathbb{R}$. The coordinates (w_x, w_y) are specified in an object coordinate system with origo in the centre of the object and the object at some reference scale.

The distribution $f_\mathbf{w}(\mathbf{w}\,|\mathbf{m}_i)$ is represented by a set of samples, $\mathcal{W}_i = \left(\mathbf{w}_i^{(1)}, \mathbf{w}_i^{(2)}, \cdots\right)$. Those samples are constructed from training images containing only a single object at the reference scale positioned in origo. From those images all feature points, $\mathbf{l}_k$, with corresponding feature vectors $\mathbf{x}_k$, are extracted. Then $\mathbf{w}_i^{(j)} = \mathbf{l}_k$ is stored as a sample for each visual word, $\mathbf{m}_i$, that matches, i.e for all i where $\mu(\mathbf{x}_k, \mathbf{m}_i) \geq \alpha$. The index j is used to enumerate the samples assigned to each visual word, i, separately while the index k enumerates all samples found in the training images. Note that there is not a one to one correspondence between k and (i, j) as one sample from the training images can be assigned to several visual words. The distribution $f_{\mathbf{w}}\left(\mathbf{w}\left|\mathbf{m}_i\right.\right)$ is estimated using kernel density estimation (Equation 1.83) with a uniform three dimensional ellipsoidal kernel,

$$f_{\text{ellipse}}\left(\mathbf{x}\left|\mathbf{x}_0, r_0\right.\right) = \frac{1}{3\pi r^3}\begin{cases} 1 & \text{if } \left|\mathbf{x}-\mathbf{x}_0\right|^2 \leq r_0^2 \\ 0 & \text{otherwise} \end{cases}, \tag{1.150}$$

with a radius r_0 of 5% of the reference object size. Since the samples have already been assigned to visual words it is here enough to deal with the index i of the visual word. That is,

$$f_{\mathbf{w}}\left(\mathbf{w}\left|\mathbf{m}_i\right.\right) = f\left(\mathbf{w}\left|i\right.\right) = \frac{1}{|\mathcal{W}_i|}\sum_{j=1}^{|\mathcal{W}_i|} f_{\text{ellipse}}\left(\mathbf{w}\left|\mathbf{w}_i^{(j)}, r_0\right.\right). \tag{1.151}$$

Consider a new test image containing an object that has been translated to position (c_x, c_y) and rescaled a factor c_s and let $\mathbf{c} = (c_x, c_y, c_s)$. The feature points in the object coordinate system, $\mathbf{w}$, is transformed into feature points in the image, $\mathbf{l} = (l_x, l_y, l_s)$, by

$$\begin{cases} l_x &= c_x + w_x c_s \\ l_y &= c_y + w_y c_s \\ l_s &= w_s c_s \end{cases} \Leftrightarrow \begin{cases} w_x &= \frac{l_x - c_x}{c_s} \\ w_y &= \frac{l_y - c_y}{c_s} \\ w_s &= \frac{l_s}{c_s} \end{cases}. \tag{1.152}$$

These functions are denoted

$$\mathbf{l} = \mathbf{g}_{\text{l}}\left(\mathbf{w}, \mathbf{c}\right) \qquad \Leftrightarrow \qquad \mathbf{w} = \mathbf{g}_{\text{w}}\left(\mathbf{l}, \mathbf{c}\right).$$

This means that the distribution of the feature points in the image becomes

$$f\left(\mathbf{l}\left|\mathbf{m}_i, \mathbf{c}\right.\right) = \left|\frac{d\mathbf{g}_{\text{w}}}{d\mathbf{l}}\right| f_w\left(\mathbf{g}_{\text{w}}\left(\mathbf{l}, \mathbf{c}\right)\left|\mathbf{m}_i\right.\right) = \frac{1}{c_s^3} f_w\left(\mathbf{g}_{\text{w}}\left(\mathbf{l}, \mathbf{c}\right)\left|\mathbf{m}_i\right.\right). \tag{1.153}$$

This is a rescaling of the probability distribution $f_w(\cdot)$ based on the scale of the object. It can be implemented by using the following property of the kernel,

$$\frac{1}{c_s^3} f_{\text{ellipse}}\left(\mathbf{g}_{\text{w}}\left(\mathbf{l}, \mathbf{c}\right)\left|\mathbf{w}_i^{(j)}, r\right.\right) = f_{\text{ellipse}}\left(\mathbf{l}\left|\mathbf{g}_{\text{l}}\left(\mathbf{w}_i^{(j)}, \mathbf{c}\right), c_s r\right.\right) \tag{1.154}$$

e.g. letting the radius, r of the kernel follow the scale of the object c_s. It has the effect of increasing the smoothing with the size of the objects, which is a desired behaviour,

$$f\left(\mathbf{l}\left|\mathbf{m}_i, \mathbf{c}\right.\right) = \frac{1}{|\mathcal{W}_i|}\sum_{j=1}^{|\mathcal{W}_i|} f_{\text{ellipse}}\left(\mathbf{l}\left|\mathbf{g}_{\text{l}}\left(\mathbf{w}_i^{(j)}, \mathbf{c}\right), c_s r\right.\right). \tag{1.155}$$

By assuming a uniform prior on the object centre, $\mathbf{c}$, its posterior distribution becomes

$$f\left(\mathbf{c}\left|\mathbf{l}, \mathbf{m}_i\right.\right) = \frac{1}{a} f\left(\mathbf{l}\left|\mathbf{m}_i, \mathbf{c}\right.\right), \tag{1.156}$$

where a is a constant making sure $\int f(\mathbf{c}\,|\mathbf{l}, \mathbf{m}_i)\,\mathrm{d}\mathbf{c} = 1$.

To locate objects in an input image, I, a set of n feature points $\mathbf{l}_j$ are detected and their feature vectors, $\mathbf{x}_j$ are extracted. Each of them is compared to the visual words in the codebook and a distribution, $f(\mathbf{m}_i\,|\mathbf{x}_j)$, over the visual words given the feature vector is estimated. It is assumed to be uniform over the visual words matching the feature, e.g.

$$f(\mathbf{m}_i\,|\mathbf{x}_j) = \frac{1}{c}\begin{cases} 1 & \text{if } \mu(\mathbf{x}_j, \mathbf{m}_i) \geq \alpha \\ 0 & \text{otherwise} \end{cases}, \tag{1.157}$$

where c is a normalising constant making sure that $\int f(\mathbf{m}_i\,|x_j) = 1\mathrm{d}\mathbf{m}_i$. The distribution of the object centre, given a single feature point, can be found by integrating out the visual word

$$f(\mathbf{c}\,|\mathbf{l}_j, \mathbf{x}_j) = \sum_{i=1}^{|\mathcal{C}|} f(\mathbf{c}\,|\mathbf{l}_j, \mathbf{m}_i)\, f(\mathbf{m}_i\,|\mathbf{x}_j). \tag{1.158}$$

All the detected feature points can be combined using the kernel density estimation from Equation 1.83,

$$f(\mathbf{c}\,|I) = \frac{1}{n}\sum_{j=1}^{n} f(\mathbf{c}\,|\mathbf{l}_j, \mathbf{x}_j). \tag{1.159}$$

Putting it all together gives

$$f(\mathbf{c}\,|I) = \frac{1}{an}\sum_{j=1}^{n}\sum_{i=1}^{|\mathcal{C}|}\frac{1}{|\mathcal{W}_i|}\sum_{k=1}^{|\mathcal{W}_i|} f_{\text{ellipse}}\left(\mathbf{l}\,\middle|\,\mathbf{g}_{\mathbf{l}}\left(\mathbf{w}_i^{(k)}, \mathbf{c}\right), c_s r\right) f(\mathbf{m}_i\,|\mathbf{x}_j) \tag{1.160}$$

Local maximas of this distribution is found by generating a histogram of the distribution and finding local maximas in this. Those local maximas indicate positions where it is likely for an object to be located and are considered detection hypothesis. The position of these detections are then refined using mean shift [11] which results in a set of hypothesis, $\mathcal{H} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots\}$.

Pixel wise Segmentation

To allow the MDL based reasoning mentioned above about how many objects there are present in an image, a centre position and scale of each hypothesis is not enough. Especially not in cases where the objects occlude each other. In those cases a more precise pixel wise segmentation is used. To make that possible the training data has to be extended with pixel wise segmentations of the objects. When that is done, each sample $\mathbf{w}_i^{(k)}$ in the training database can be augmented with a binary segmentation mask $B_i^{(k)}$ centred at the detected feature. This mask will indicate how the segmentation should look around a given visual word at a given position relative to the object centre.

Assume an object is located at $\mathbf{c}$, and let B be a segmentation of an input image I such that each pixel, $B(\mathbf{a})$, is one if it belongs to an object located at $\mathbf{c}$ and otherwise zero. The distribution $f(\mathbf{l}\,|\mathbf{c},\mathbf{m}_i)$ is given by Equation 1.155, and the probability of $B(\mathbf{a})=1$ can be expressed as

$$p\left(B\left(\mathbf{a}\right)=1|\mathbf{c}\right)=$$
$$\frac{1}{b}\sum_j\sum_i\frac{1}{|\mathcal{W}_i|}\sum_{k=1}^{|\mathcal{W}_i|}f_{\text{ellipse}}\left(\mathbf{l}_j\left|\mathbf{g}_{\text{l}}\left(\mathbf{w}_i^{(k)},\mathbf{c}\right),c_s r\right.\right)f\left(\mathbf{m}_i\,|\mathbf{x}_j\right)B_i^{(k)}(\mathbf{a}-\mathbf{g}_{\text{w}}\left(\mathbf{l}_j,\mathbf{c}\right)),\tag{1.161}$$

where b is a normalising constant. Likewise,

$$p\left(B\left(\mathbf{a}\right)=0|\mathbf{c}\right)=$$
$$\frac{1}{b}\sum_j\sum_i\frac{1}{|\mathcal{W}_i|}\sum_{k=1}^{|\mathcal{W}_i|}f_{\text{ellipse}}\left(\mathbf{l}_j\left|\mathbf{g}_{\text{l}}\left(\mathbf{w}_i^{(k)},\mathbf{c}\right),c_s r\right.\right)f(\mathbf{m}_i\,|\mathbf{x}_j)\left(1-B_i^{(k)}(\mathbf{a}-\mathbf{g}_{\text{w}}(\mathbf{l}_j,\mathbf{c}))\right).\tag{1.162}$$

Using these two equations a probabilistic pixel-wise segmentation mask B can be found for each object hypothesis $\mathbf{c}$.

Hypothesis Verification using MDL

Each object in the input image typically gives rise to several conflicting (i.e. overlapping) hypothesis in $\mathcal{H}$. Thus, the subset best describing the image should be chosen as the output of the detector. This is done here using a *minimum description length*, or MDL, formulation. The concept is borrowed from information theory and the idea is that the simplest explanation of the image probably is the correct one.

In the current setting this is formalised by considering all possible subsets of the hypotheses, $\mathcal{H}'\subset\mathcal{H}$. Each such subset can be used to give an approximate description of the parts of the input image that the detected objects cover. To give a complete description, the pixels in the parts not covered by the detections as well as the errors made by the approximative description has to be described. Let $n(\mathcal{H}')$ be the number of pixels covered by the objects in $\mathcal{H}'$ and let n_{all} be the total number of pixels in the image. Furthermore, let $\xi(\mathcal{H}')$ be the error introduced by describing the $n(\mathcal{H}')$ pixels covered by the detections in the image using the detections in $\mathcal{H}'$. Finally, let $m(\mathcal{H}')$ be the number of parameters needed to describe the detections in $\mathcal{H}'$. The description length, $l(\cdot)$, is defined as

$$l(\mathcal{H}')=k_1\left(n_{\text{all}}-n\left(\mathcal{H}'\right)\right)+k_2\xi\left(\mathcal{H}'\right)+k_3 m\left(\mathcal{H}'\right),\tag{1.163}$$

where the constant parameters k_1, k_2 and k_3 is the number of bits required to encode each part of the information.

Each pixel $\mathbf{a}$ in the input image is considered explained by and assigned to the detection giving it the highest probability,

$$p\left(B\left(\mathbf{a}\right) = 1\left|\mathcal{H}'\right.\right) = \max_{\mathbf{c}\in\mathcal{H}'} p\left(B\left(\mathbf{a}\right) = 1|\mathbf{c}\right), \tag{1.164}$$

if this probability is larger than $p\left(B\left(\mathbf{a}\right) = 0\left|\hat{\mathbf{c}}_{\mathbf{a}}\right.\right)$. Here $\hat{\mathbf{c}}_{\mathbf{a}}$ is the hypothesis generating the maximum at pixel $\mathbf{a}$, i.e. $\hat{\mathbf{c}}_{\mathbf{a}} = \operatorname{argmax}_{\mathbf{c}\in\mathcal{H}'} p\left(B\left(\mathbf{a}\right) = 1|\mathbf{c}\right)$. If that is not the case the pixel is assigned to the unexplained parts of the image. The set of explained pixels is denoted $\mathcal{X}$, which gives $n\left(\mathcal{H}'\right) = |\mathcal{X}|$ and the error made is defined as

$$\xi\left(\mathcal{H}'\right) = \sum_{\mathbf{a}\in\mathcal{X}} \left(1 - p\left(B\left(\mathbf{a}\right) = 1\left|\mathcal{H}'\right.\right)\right). \tag{1.165}$$

It is a sum over all explained pixels of the probabilities that each assignment is wrong. The number of parameters needed, $m\left(\mathcal{H}'\right)$, is defined as the expected area of the detected object.

The final result of the detector is found by optimising over all possible subsets $\mathcal{H}' \subset \mathcal{H}$ and locating the subset

$$\mathcal{O} = \min_{\mathcal{H}'\subset\mathcal{H}} l\left(\mathcal{H}'\right). \tag{1.166}$$

For this to be solvable in reasonable time, only pairwise interaction between objects are considered. This means that hypothesis with three or more objects overlapping the same area in the image will be assigned a larger description length compared to the exact objective function, Equation 1.163. This is claimed to be a desired feature since it can be interpreted as a prior distribution putting a lower prior on hypotheses with a lot of objects clustered close together.

The optimisation becomes a submodular second order pseudo boolean problem, which can be solved exactly using for example graph cuts[48]. It is however claimed [53] that a greedy approximation gives good enough results here.

3D object detection

If the camera calibration is known and the objects are assumed to be flat rectangles with known aspect ratio standing on a known ground plane, the two dimensional hypothesis, $(x', y', s') = \mathbf{c} \in \mathcal{H}$ can be upgraded to three dimensional, $\mathbf{d} = (x, y, h)$. Here (x', y') is the centre position of the object in the image, s is the scale of object bounding box in the image, (x, y) is the objects centre position on the ground plane in the world coordinate system, e.g. in meters, and h is its height in the world coordinate system.

This is used by Leibe *et al* [50] to fuse detections from different cameras by clustering the three dimensional hypothesis generated from different views. They also consider car detection by using 5 different car detectors trained for different orientations of the car. Detections in the same image from different car detectors are also fused in the same way to allow overlapping detections of different orientations to be fused into a single hypotheses with a more precise orientation. In that way a set of three dimensional hypothesis $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots\}$ is formed and the MDL hypothesis verification step is modified to work with these three dimensional hypothesis and instead optimise over all subsets $\mathcal{D}' \subset \mathcal{D}$.

The camera calibration can be described by the distribution $f(\mathbf{c}\,|\mathbf{d})$, which introduces some model uncertainty in the calibration. The per pixel probabilities can then be expressed in terms of the three dimensional detection by marginalising over the two dimensional detections

$$p(B(\mathbf{a}) = i\,|\mathbf{d}) = \frac{1}{b}\sum_{\mathbf{c}} p(B(\mathbf{a}) = i\,|\mathbf{c})\, f(\mathbf{c}\,|\mathbf{d}), \tag{1.167}$$

for $i = 0, 1$ and a normalisation constant b. The assumption that pixels always belongs to the strongest possible hypothesis can here be replaced with the assumption that the pixel belongs to the object closest to the camera. Each pixel $\mathbf{a}$ is considered explained if there exists one or several hypothesis $\mathbf{d} \in \mathcal{D}'$ such that $p(B(\mathbf{a}) = 1\,|\mathbf{d}) > p(B(\mathbf{a}) = 0\,|\mathbf{d})$, otherwise it is considered unexplained. Let $\hat{\mathbf{d}}_{\mathbf{a}}$ be the hypothesis closest to the camera among all such hypothesis. The per pixel probabilities can then be extended to sets of hypothesis by letting

$$p(B(\mathbf{a}) = 1\,|\mathcal{D}') = p\left(B(\mathbf{a}) = 1|\hat{\mathbf{d}}_{\mathbf{a}}\right). \tag{1.168}$$

As before, the set of explained pixels is denoted $\mathcal{X}$.

The three dimensional hypothesis also allows for a natural prior distribution, $f(\mathbf{d}) = f(h)\, f(x, y)$, to be introduced. For tracking pedestrians, Leibe *et al* [51] suggests to use the height prior (for h in meters), $f(h) = \mathcal{N}\left(h\,|1.7, 0.2^2\right)$, and to use a position prior, $f(x, y)$, that is uniform over the area where the detector is expected to perform well. This becomes a strong connection between the scale and position of the objects in the image which improves detection results. To introduce this prior, the error is redefined into

$$\xi(\mathcal{D}') = \sum_{\mathbf{a}\in\mathcal{X}} (1 - p(B(\mathbf{a}) = 1\,|\mathcal{D}'))\, f\left(\hat{\mathbf{d}}_{\mathbf{a}}\right). \tag{1.169}$$

The remaining parts of the description length, $l(\mathcal{O}')$ (Equation 1.163), remains the same as before and the set of three dimensional detections is found by maximising over all subsets $\mathcal{D}'$,

$$\mathcal{O} = \min_{\mathcal{D}'\subset\mathcal{D}} l(\mathcal{D}'). \tag{1.170}$$

Tracking

The set of detections from the previous section can be passed to any of the detection based trackers presented above. But by formulation the tracking problem as the same kind of MDL hypothesis selection optimisation it is possible to solve it jointly with the detection problem. This will be described in the next Section. Leibe *et al* [50] have made this kind of formulation of the tracking problem. In that case, the three dimensional detections for each frame,

$$\mathcal{O}_t = \left\{\mathbf{o}_{t,1}, \mathbf{o}_{t,2}, \cdots \mathbf{o}_{t,|\mathcal{O}_t|}\right\}, \tag{1.171}$$

is generated as above in a separate step and then the tracking uses those detections as input. For some objects, such as cars, several detectors are used trained on cars with different orientations, which means that also the orientation of the detected object, θ, can be estimated. The detections are also augmented with a colour histogram, $a(\cdot)$, formed over the supporting pixels, wighted with the per pixel probabilities, Equation 1.167. Each detection is thus represented by $\mathbf{o}_{t,i} = (x_{t,i}, y_{t,i}, h_{t,i}, \theta_{t,i}, a_{t,i}(\cdot))$,

The tracker tries to estimate a state sequence for each object in the scene. The state of a single object at time t is given by $\mathbf{q}_t = (x, y, \theta, v, w, a(\cdot))$, where (x, y) is the object centre position on the ground plane, θ is its orientation, v its velocity, w its angular velocity and $a(\cdot)$ a $8 \times 8 \times 8$ colour histogram. Two different dynamic models are introduced, one for pedestrians and one for cars, where the car model includes the restriction that the car cannot turn if it is not in motion. Both models are constructed by a set of differential equations. In both cases $\frac{\mathrm{d}x}{\mathrm{d}t} = v\cos\theta$ and $\frac{\mathrm{d}y}{\mathrm{d}t} = v\sin\theta$ are used. The pedestrian model also uses $\frac{\mathrm{d}\theta}{\mathrm{d}t} = w$, while the car model uses $\frac{\mathrm{d}\theta}{\mathrm{d}t} = wv$. Using these equation, predictions about the current state, $\mathbf{q}_t^- = \left(x_t^-, y_t^-, \theta_t^-, v_t^-, w_t^-, a_t^-(\cdot)\right)$, can be made from the state in the previous frame, $\mathbf{q}_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1}, v_{t-1}, w_{t-1})$, using either

$$\left\{\begin{array}{l} x_t^- = x_{t-1} + v_{t-1}\cos\theta_{t-1} \\ y_t^- = y_{t-1} + v_{t-1}\sin\theta_{t-1} \\ \theta_t^- = \theta_{t-1} + w_{t-1} \\ v_t^- = v_{t-1} \\ w_t^- = w_{t-1} \\ a_t^- = a_{t-1} \end{array}\right. \quad \text{or} \quad \left\{\begin{array}{l} x_t^- = x_{t-1} + v_{t-1}\cos\theta_{t-1} \\ y_t^- = y_{t-1} + v_{t-1}\sin\theta_{t-1} \\ \theta_t^- = \theta_{t-1} + w_{t-1}v_{t-1} \\ v_t^- = v_{t-1} \\ w_t^- = w_{t-1} \\ a_t^- = a_{t-1} \end{array}\right. . \tag{1.172}$$

The observations made are the detections which gives position, orientation and colour histogram. The observation probability distribution of the position part of the state is defined using a rotation matrix

$$\mathbf{R}_\theta = \left(\begin{array}{cc} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{array}\right), \tag{1.173}$$

and different variations along the travelling direction, σ^2_{mov}, and perpendicular to it, σ^2_{trn}. It is assumed to be

$$f\left((x_t, y_t)\,\middle|\mathbf{q}_t^-\right) = \mathcal{N}\left((x_t, y_t)\,\middle|\left(x_t^-, y_t^-\right), \mathbf{R}_{\theta_t^-}{}^{\mathrm{T}}\left(\begin{array}{cc} \sigma^2_{\text{mov}} & 0 \\ 0 & \sigma^2_{\text{trn}} \end{array}\right)\mathbf{R}_{\theta_t^-}\right). \tag{1.174}$$

The orientation part is also assumed Gaussian distributed with some variance σ_{str}^2,

$$f\left(\theta_t \left| \theta_t^- \right.\right) = \mathcal{N}\left(\theta_t \left| \theta_t^-, \sigma_{\text{str}}^2 \right.\right). \tag{1.175}$$

The colour histograms are compared using their *Bhattacharyya coefficient* [38, 13],

$$f\left(a_t(\cdot) \left| a_t^-(\cdot) \right.\right) = \sum_i \sqrt{a_t(i)\, a_t^-(i)}, \tag{1.176}$$

which is closely related to the average probability of classification error. The height of the detection provides very little extra information as it is closely coupled to the position, and is not used. Combining Equation 1.174, 1.175 and 1.176 gives the likelihood of an observation, $\mathbf{o}_t$, assuming independence,

$$f\left(\mathbf{o}_t \left| \mathbf{q}_t^- \right.\right) = f\left((x_t, y_t) \left| \mathbf{q}_t^- \right.\right) f\left(\theta_t \left| \theta_t^- \right.\right) f\left(a_t(\cdot) \left| a_t^-(\cdot) \right.\right). \tag{1.177}$$

The current state of the tracker is formed as a weighed sum over the prediction and all detections. To do that the non observed parts of the state has to be estimated for each detection, $\mathbf{o}_{t,i}$, by for example

$$\begin{cases} v_{t,i} = \sqrt{\left(x_{t,i} - x_{t-1}\right)^2 + \left(y_{t,i} - y_{t-1}\right)^2} \\ w_{t,i} = \theta_{t,i} - \theta_{t-1} \end{cases}. \tag{1.178}$$

The detections are weighted with their likelihoods and the prediction with some constant, $e^{-\lambda}$,

$$\mathbf{q}_t = \frac{e^{-\lambda}\mathbf{q}_t^- + \sum_i f\left(\mathbf{o}_{t,i} \left| \mathbf{q}_{t-1} \right.\right)\left(x_{t,i}, y_{t,i}, \theta_{t,i}, v_{t,i}, w_{t,i}, a_{t,i}(\cdot)\right)}{e^{-\lambda} + \sum_i f\left(\mathbf{o}_{t,i} \left| \mathbf{q}_{t-1} \right.\right)}. \tag{1.179}$$

The tracker is initiated at every detection in every frame and for each of them a hypothetical track is generated by running the tracker both forward and backwards in time. This generates a set of track hypothesis, $\mathcal{T}$, and a subset of them is chosen by optimising over all subsets $\mathcal{T}' \subset \mathcal{T}$ in a similar manner as above. Here the tracks will compete for the detections as the detections competed for the pixels above. Each track hypothesis, $\mathbf{r}_i \in \mathcal{T}$, is a sequence of states, $\mathbf{r}_i = \{\mathbf{q}_{i,t}\}$ for $t = 0, 1, \cdots$. For each track, i, there will be a set of corresponding three dimensional detections, $\mathbf{o}_{t,i}$, for some t, typically not every possible t, i.e. the observation is missing for some time points. Each observation will be assigned to the track in $\mathcal{T}'$ that maximises its likelihood, $f\left(\mathbf{o}_{t,i} \left| \mathbf{q}_t \right.\right)$. In each image, each pixel is assigned to a single detection as above. Let $\mathcal{X}_{t,i}$ be the set of pixels assigned to the detection $\mathbf{o}_{t,i}$. The description length error is here defined to be

$$\xi\left(\mathcal{T}'\right) = \sum_i \sum_t e^{-\lambda(\tau - t)} \sum_{\mathbf{a} \in \mathcal{X}_{t,i}} \left(1 - p\left(B(\mathbf{a}) = 1 \left| \mathbf{o}_{t,i} \right.\right)\right) f\left(\mathbf{o}_{t,i}\right) f\left(\mathbf{o}_{t,i} \left| \mathbf{q}_{t,i} \right.\right). \tag{1.180}$$

A temporal weighting have been used to put more weight of the last frame, τ. For offline tracking or in surveillance situation where results can be provided with a few seconds delay it would have been more natural to put the same weight on all frames. With this new error term, the MDL hypothesis selections procedure can be applied and the subset $\mathcal{T}'$ with minimum description length can be found.

Joint Tracking and Detection

Leibe *et al* [51] has combined this MDL based hypothesis selection for tracking and detection into a single hypothesis selection optimisation where tracks will compete for detections and detections will compete for pixels. To do this over an entire video sequence might be too ambitious a task. Instead tracks are optimised over all frames while detections only are optimised over the current frame. As the system then moves on to the next frame the detections for past frames are fixed. This means that only detections matching the current optimal sequence, which might change when future frames are received, or matches that are strong enough to by themselves force a new track to be initiated, are kept as detections.

1.4.3 Background Foreground Trackers

It is also possible to reason about the number of objects present using a Bayesian formulation. In that case the multi target state space presented in Section 1.3.2 is typically used. This is done by for example Song and Nevatia [92], who models pedestrians as four three dimensional ellipsoids corresponding to the head, torso, left leg and right leg. The model is assumed to be standing on a known ground plane and it is projected into a calibrated camera. The pixels it covers in the image is expected to be detected as foreground by some background/foreground segmentation algorithm and all other pixels are expected to be background.

Three articulations of the model are considered, legs together, right leg forward and left leg forward. The orientation is discretised into a few orientations. This forms a discrete, finite, set of articulations and orientations which is enumerated with labels, $l \in \mathcal{L}$. The state of an object also contains its position on the ground plane, (x, y), and two scaling factors, (s_h, s_f), varying the height and the fatness of the model. Each single object $\mathbf{q}_i$ is thus represented by $\mathbf{q}_i = (l, x, y, s_h, s_f) \in \mathbb{S}$. The full state, $\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots) \in \mathcal{S}^\infty$, is a configuration of objects.

As observations a binary background foreground segmentation, B, is used. For pixels, $\mathbf{a}$, where foreground is detected $B(\mathbf{a}) = 1$ and for pixels where background is detected $B(\mathbf{a}) = 0$. The observation probability distribution is formed by projecting the ellipses representing the state $\mathbf{q}$ into the image, forming an expected background foreground segmentation image $\hat{B}$. All pixels are considered independent and their distributions specified by two constants p_{00} and p_{11}, i.e. for all pixels $\mathbf{a}$,

$$\begin{aligned} f\left(B(\mathbf{a})=0\left|\hat{B}(\mathbf{a})=0\right.\right) &= p_{00} \\ f\left(B(\mathbf{a})=1\left|\hat{B}(\mathbf{a})=0\right.\right) &= 1-p_{00} \\ f\left(B(\mathbf{a})=0\left|\hat{B}(\mathbf{a})=1\right.\right) &= 1-p_{11} \\ f\left(B(\mathbf{a})=1\left|\hat{B}(\mathbf{a})=1\right.\right) &= p_{11} \end{aligned}, \tag{1.181}$$

and for the entire observation,

$$f\left(B\left|\hat{B}\right.\right)=\prod_{\mathbf{a}} f\left(B(\mathbf{a})\left|\hat{B}(\mathbf{a})\right.\right). \tag{1.182}$$

By counting the two types of errors

$$n_{10}=\sum_{\mathbf{a}} B(\mathbf{a})\left(1-\hat{B}(\mathbf{a})\right) \tag{1.183}$$

and

$$n_{01}=\sum_{\mathbf{a}}\left(1-B\left(\mathbf{a}\right)\right)\hat{B}(\mathbf{a}), \tag{1.184}$$

this can be written

$$f\left(B\left|\hat{B}\right.\right)=\alpha e^{-\lambda_{10}n_{10}-\lambda_{01}n_{01}}, \tag{1.185}$$

where λ_{10} and λ_{01} are constants depending on p_{00} and p_{11} and α contains all terms independent of $\hat{B}$ and $\mathbf{q}$ (but dependent on B).

An maximum a posterior, MAP, estimate of the state $\mathbf{q}$ in the current image is found by maximising $f\left(\mathbf{q}\left|B\right.\right)$ over all possible $\mathbf{q}$. The same maximum is found by maximising $f\left(B\left|\hat{B}\right.\right)f\left(\mathbf{q}\right)$, where $f\left(\mathbf{q}\right)$ is a state prior. Here only a single frame is considered, which means that comparing observation likelihoods is equivalent with discretising the state space into regions as illustrated by Figure 1.4 and explained in Section 1.3.2. However, the prior used removes this property. It is chosen as a product over all objects present,

$$f\left(\mathbf{q}\right)=\prod_{i=1}^{|\mathbf{q}|} f\left(\mathbf{q}_i\right), \tag{1.186}$$

which means it will implicitly put a lower prior on any single state with more objects present since $f\left(\mathbf{q}_i\right)<1$. However the number of states with a given number of objects grows with the number of objects with means that the prior assumed gives an uniform prior over the number of objects. The single object prior is chosen to be

$$f\left(\mathbf{q}_i\right)=e^{-\lambda_1 v(\mathbf{q}_i)}\left(1-e^{-\lambda_2 v(\mathbf{q}_i)}\right)\frac{1}{v_{\text{tot}}}\mathcal{N}\left(s_h\left|1.5,1.9\right.\right)\mathcal{N}\left(s_f\left|0.8,1.2\right.\right)f(l), \tag{1.187}$$

where $v(\mathbf{q}_i)$ is the area covered by projecting $\mathbf{q}_i$ into the image and v_{tot} is the total area of the image. The prior over the articulation/orientation labels $f(l)$ is chosen to make the prior of the walking models lower than the standing models in order to penalise the higher complexity of the walking models. These priors will depend on the scale of the coordinate system and comparing the prior of two states with a different number of objects can be given any outcome by changing the coordinate system scale.

The MAP estimate is search for by generating a set of samples $\mathcal{Q} = \left\{\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \cdots\right\}$ by doing Markov chain Monte-Carlo sampling from $f\left(B \middle| \hat{B}\right) f\left(\mathbf{q}\right)$, and then choosing the sample with maximum likelihood as the MAP estimate $\mathbf{q}^*$,

$$\mathbf{q}^* = \operatorname{argmax}_{\mathbf{q}\in\mathcal{Q}} f\left(B \middle| \hat{B}\right) f\left(\mathbf{q}\right) \tag{1.188}$$

The MCMC algorithm is introduced in Section 1.3.1 and extended to this kind of multi-object spaces in Section 1.3.2. It is based on a proposal distribution $\tilde{f}\left(\mathbf{q}^{(j)} \middle| \mathbf{q}^{(j-1)}\right)$, which has to be specified. It is here defined by an algorithm generating $\mathbf{q}^{(j)}$ by randomly choosing one of five modification schemes and applying it to $\mathbf{q}^{(j-1)}$. The five possible modifications are

- Add a random detection, perturbed with Gaussian noise, from a random pedestrian detector as a new object. Here any set of pedestrian detectors could be plugged in. If the detector don't give all state variables in $\mathbf{q}_i$, the remaining is sampled from their prior.

- Remove a random object.

- Randomly change the label l of a random object.

- Let $\mathbf{q}^{(j)} = \mathbf{q}^{(j-1)} + k \left.\frac{d\log f(\mathbf{q}|B)}{d\mathbf{q}}\right|_{\mathbf{q}=\mathbf{q}^{(j-1)}} + \mathbf{w}$, where k is some fixed parameter and $\mathbf{w}$ is Gaussian noise.

- Replace one object by performing the second modification followed by the first modification.

It is straightforward to generalise this to handle other types of objects. Song and Nevatia [72] present a version for detecting cars modelled as boxes with known dimensions. There tracking is also introduced by considering the set $\mathcal{Q}$ above as a set of hypothesis for each frame. By indexing the state vectors with the frame number t, e.g $\mathbf{q}_t = (\mathbf{q}_{t,1}, \mathbf{q}_{t,2}, \cdots)$ and $\mathcal{Q}_t = \left\{\mathbf{q}_t^{(1)}, \mathbf{q}_t^{(2)}, \cdots\right\}$, a dynamic model can be introduced as the probability distribution $f\left(\mathbf{q}_t \middle| \mathbf{q}_{t-1}\right)$. It is here constructed by matching the objects in $\mathbf{q}_t$ to the objects found in $\mathbf{q}_{t-1}$ based on their overlapping area. If $\mathbf{q}_t$ and $\mathbf{q}_{t-1}$

are reorder to place the l matching objects in the same order at the beginning of the vectors,

$$f\left(\mathbf{q}_t \left| \mathbf{q}_{t-1} \right.\right) = \prod_{i=1}^{l} f\left(\mathbf{q}_{t,i} \left| \mathbf{q}_{t-1,i} \right.\right) \prod_{i=l+1}^{|\mathbf{q}_t|} f_{\text{new}}\left(q_{t,i}\right) \prod_{i=l+1}^{|\mathbf{q}_{t-1}|} f_{\text{end}}\left(q_{t-1,i}\right), \tag{1.189}$$

where $f\left(\mathbf{q}_{t,i} \left| \mathbf{q}_{t-1,i} \right.\right)$ is some single object motion model and $f_{\text{new}}\left(q_{t,i}\right)$ and $f_{\text{end}}\left(q_{t-1,i}\right)$ are the entry and exit probabilities that here might depend on the position to make it more likely for cars to appear or disappear at the borders of the intersection.

This forms an hidden Markov model and the maximum likelihood state sequence is found by using Viterbi optimisation, see Chapter 4. The Viterbi optimisation does give the global optimum, but in this case it is not used to search the entire state space, but only a subset of hypothesis, which means that only a local optimum is found. In Chapter 4 modifications to the Viterbi algorithm are presented that allows it to be used to search the entire state space.

This algorithm allows temporal consistency arguments to enter the reasoning about how many objects are present, but it assumes that the MCMC sampling actually finds the states on maximum likelihood state sequence based on a single frame only. This means that the algorithms can remove false positives, but will fail if a car is not detected at all in a single frame.

1.4.4 Globally Optimised

Cost flow network

Zang and Nevatia [91] has suggested to solve the problem of missing detections by allowing detections to be linked over small time gaps. The problem is here formulated as a search for the optimal assignment of measurements to objects. The space of all possible assignments is typically a much smaller space than the multi object state space, since only the areas of the state space were there are detections are considered. This makes it possible to optimise over the entire space if the maximum number of consecutive frames an object might be missed by the detector can be assumed lower than some low threshold. It is then possible to solve for a global optimum using cost flow networks.

The observations in this setup is a set of detections from a single frame object detector, $\mathbf{o}_i \in \mathcal{O}$, where each observation $\mathbf{o}_i = (x, y, s, a, t)$, consists of a position, (x, y), scale, s, appearance, a and frame index t. The task is to partition this set into disjoint trajectories, $\mathcal{T}_k = \{\mathbf{o}_{k,1}, \mathbf{o}_{k,2}, \cdots\}$, and false alarms. Such a partitioning is represented by the set of trajectories, $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots\}$. A maximum a posterior estimate, $\mathcal{T}^* = \operatorname{argmax}_{\mathcal{T}} f\left(\mathcal{T} \left| \mathcal{O} \right.\right)$, is found by optimising

$$\prod_{\mathbf{o} \in \mathcal{O}} f\left(\mathbf{o} \left| \mathcal{T} \right.\right) \prod_{\mathcal{T}_k \in \mathcal{T}} f\left(\mathcal{T}_k\right) \tag{1.190}$$

over $\mathcal{T}$ under the constraints

$$\mathcal{T}_k \cap \mathcal{T}_l = \emptyset \text{ for all } k \neq l. \tag{1.191}$$

This optimisation can be done in polynomial time using cost flow networks (see [91] for details). The first part of the probability distribution is derived from the probability of a false detection, β,

$$f\left(\mathbf{o}\left|\mathcal{T}\right.\right) = \begin{cases} 1-\beta & \text{if } \mathbf{o} = \mathbf{o}_{k,j} \text{ for some } k \text{ and } j \\ \beta & \text{otherwise} \end{cases}. \tag{1.192}$$

The probability of a track

$$f\left(\mathcal{T}_k\right) = f_{\text{new}}\left(\mathbf{o}_{k,1}\right) \prod_{l=2}^{|\mathcal{T}_k|} f\left(\mathbf{o}_{k,l}\left|\mathbf{o}_{k,l-1}\right.\right) f_{\text{end}}\left(\mathbf{o}_{k,|\mathcal{T}_k|}\right), \tag{1.193}$$

where $f_{\text{new}}(\cdot)$ and $f_{\text{end}}(\cdot)$ are assumed constant and equal and somehow estimated during the optimisation. The link probabilities

$$f\left(\mathbf{o}_{k,l}\left|\mathbf{o}_{k,l-1}\right.\right) = f\left(\mathbf{o}_j\left|\mathbf{o}_i\right.\right) = \\ f\left(s_j\left|x_i, y_i, s_i, t_j - t_i\right.\right) f\left(x_j, y_j\left|x_i, y_i, t_j - t_i\right.\right) f\left(a_j\left|a_i\right.\right) f\left(t_j - t_i\right), \tag{1.194}$$

where $f\left(s_j\left|x_i, y_i, s_i, t_j - t_i\right.\right)$ and $f\left(x_j, y_j\left|x_j, y_j, t_j - t_i\right.\right)$ are assumed Gaussian and $f\left(t_j - t_i\right)$ is constructed from the missing rate of the detector and is truncated to 0 for large $t_j - t_i$. The appearance a is a Colour histogram and the probability distribution $f\left(a_j\left|a_i\right.\right)$ is formed by comparing the two Colour histograms.

The global optimal MAP estimate, $\mathcal{T}^*$, is calculated, which results in object tracks even if the objects are not detected for a few consecutive frames and there are clutter detections. Long term occlusion between objects does not work however. This is solved in a iterative (no longer globally optimal) manner where in a first step the non occluded objects are tracked using the above algorithm. Then those tracks are fixed and synthesised detections are added to $\mathcal{O}$ representing detections that might have been occluded by the now fixed objects. The algorithm is the repeated to locate the occluded tracks. The entire process is repeated until no more tracks are located or some specified number of iterations have been reached.

Iterative single object HMM

Berclaz *et al* [4] has suggested to use a single object tracker to extract the tracks of multiply objects one by one. This single object tracker is constructed by discretising the ground plane into a set of positions forming a uniform two dimensional grid. The state space is formed from the finite set, $\mathbb{S}^1 = (S_1, S_2, \cdots)$, of gird points and a special state S_0

representing the state of the object being located outside the region of interest, e.g the state space used is $\mathbb{S} = \hat{\mathbb{S}}^1 = \mathbb{S}^1 \cup S_0$. The objects tracked are pedestrians and they are assumed to be flat rectangles standing at the grid points facing the camera. The observations formed from the input images, I_t, consists of background foreground segmentations and colour histograms form over the area in the image an objects projects into. The initial state $\mathbf{q}_0$, and the expected colour histogram of that object is assumed known a priori. In the case that $\mathbf{q}_0 = S_0$, the colour histogram is made flat to allow an object with unknown colour histogram to enter the scene. The probability distribution $f(\mathbf{q}_t | I_t)$ is derived based on the colour histogram and the background segmentation, and an MAP estimate of of the state sequence $\mathbf{q}_1, \mathbf{q}_2, \cdots$ is found using Viterbi optimisation.

This single object tracker is then used to form a multi object tracker by applying it to 4 second batches of the input video. After it is executed once, a single track is found. Then this track is fixed and the algorithm repeated under the restriction that the new track may not overlap the fix tracks. When no more tracks can be found, the 10 first frames of the 4 second batch is discarded and the results from them stored as output. Also colour histograms are estimated for the tracks found to be used in the next iteration. Then 10 new frames are appended to the batch and the entire process repeated.

Even though the single object tracker gives a global optimum for a single object in each 4 second batch, the resulting algorithm don't have to give the global optimum for the multi object problem for longer sequences. The resulting state sequences are not even guaranteed to be consistent, even though the large overlap of the batches makes this likely. Also, the colour histograms of the objects are assumed known at the start of each batch and not part of the state space, which means no optimisation is done over different colour histograms.

Single frame

Schoenemann and Cremers [68] has suggested a globally optimal tracker that is constructed from a shape based object detector, detection the occluding contour line of an object. The detector assumes that exactly one object is present and finds its globally optimal position given a single frame. No dynamic model is used. This means that the optimisation is done over a single state for a single frame and not over state sequences.

1.5 Discussion

Detecting simple events, such as a pedestrian entering a shop, a vehicle turning left at an intersection, or a person loitering, can be done by using a robust tracker. By placing restrictions on the tracker, such as "vehicles may only appear at the borders of the image and not in the middle of the intersection", the event detection will be a simple matter of checking the endpoints of the tracks produced, and, in case of loitering, the time spend in the scene. In this thesis we present a novel tracker that uses online Viterbi optimisation to find the set of tracks with maximum likelihood among all tracks conforming to constraints such as those mentioned above.

The main purpose is that the tracking algorithm should produce the overall picture right. That is, decide where and when objects enter and exit the scene. To do that robustly the objects have to be tracked. There is however no need for an exact estimate of the object position or speed, as the goal is only to get the overall picture right. The typical application is to count objects and separate between for example left turning and right turning vehicles. Once the overall picture is in place more exact positions can be calculated using one of the single target tracking algorithms presented above.

Using a continuous state space, $\mathbb{S}^{\infty}$, is theoretically somewhat complicated. The solution used in this thesis is to discretise it, which is not a problem in this case as the goal is the overall picture and not exact positions. That allows the dynamics of all objects, including entries and exits, to be modelled by a single hidden Markov model.

A major problem with the approach is that the state space is huge. Typically at least some 10000 states is needed for a single object, and tracking n objects simultaneously will reuire 10000^n states. Two novel versions of the Viterbi optimisation is presented that do not need to calculate the likelihood of every state in every frame. The first operates online, and in some cases in real time, but will with some probability, only provide approximative solutions. This probability can be estimated offline from unlabelled training sequences using the second one that provides the global optimum.

Relation to classical approaches

A classical solution to generate tracks from object detections is Kalman filtering [39], but since it is a linear model that assumes Gaussian probabilities it often fails in heavy clutter. In those cases particle filtering [30, 20] are often preferred as it allows multiple hypothesis to be maintained, and thus postpones the resolving of problematic situation until more information has been obtained.

When several objects are considered, one model for each tracked object is often used. However, the data association problem [70] of deciding which detections should be used to update which models has to be solved. This is done by the MHT [65], for the Kalman filtering framework, where all possible associations are considered. Less likely hypotheses are pruned as the number of possible associations increases exponentially with time. This exponential increase is avoided in the JPDAF [90] by presuming the associations at each time step to be independent from associations of the previous time step. The complexity can be lowered even further by also assuming independence among the different associations at one time step, which is done by the PMHT [76]. Here the data association problem does not have to be solved explicitly in every frame. Instead the probability of each measurement belonging to each model is estimated.

The problems of detecting when objects enter or exit the scene has to be solved separately in the cases above. When using a single model for all objects, as proposed in this thesis, neither the data association problem, nor the entry/exit problems has to be solved explicitly in every frame. Instead we optimise over all possible sequences of solutions over time. In [12] the PMHT is extended with the notion of track visibility to solve the problem of track initiation. However, their system is still based on the creation of candidate tracks that may not be promoted to real tracks, but they will still influence other real tracks.

Relation to other HMM and particle filter approaches

For single target tracking the particle filter have been a successful solution. Extending this to handle a varying number of objects is not straightforward as the state now contains a discrete component, the number of objects currently present. One common way is to use one particle filter for each object, but that again means that the problems of track initialisation, track termination and data association has to be solved separately, which is not the case when those events are modelled within the state space, $\mathbb{S}^{\infty}$. A single particle filter can then be applied to this space or it can be discretised and an HMM used to model the dynamics.

This has previously been suggested by [26] where a state space is exhaustively searched for an optimum in each frame. However the authors assume a known positions in the previous frame. In another approach [10] the discretising grid is repositioned in each frame, centred at the current best estimate with its mesh size and directions given as the eigenvalues and eigenvectors of the error covariance. More recently, [9] shows, in the single object case, that a particle filter is capable of matching the performance of an HMM tracker [8] at a fraction of the computational cost. However in [59] it is shown that by placing some restrictions on the HMMs the computational complexity can be reduced form $O(n^2)$ to $O(n)$, and that HMMs with $100,000$ states can be used for real time tracking, which is more than enough for single target tracking. In both these approaches fairly densely discretised state spaces are used. We show in this work that state spaces discretised more sparsely can be used. Especially in applications where the main interest is to solve the global problem of deciding which object goes where.

Most real-time HMM-based trackers [59], [42] and [22] do not use the standard Viterbi dynamic programming algorithm [64], which finds the global maximum likelihood state sequence. The main problem of using this algorithm is that it requires the entire set of future observations to be available. Instead they estimate the state posterior distribution given the past observations only. The particle filter also results in this kind of posterior state distribution, which means that both the particle filter and this kind of HMM trackers suffer from the problem of trying to estimate the single state of the system from this distribution. Later processing stages or data-displays usually requires a single state and not a distribution.

Common ways to do this is to estimate the mean or the maximum (MAP) of this posterior distribution, but this have a few problems:

1. A mean of a multi modal distribution is some value between the modes. The max-

imum might be a mode that represents a possibility that is later rejected. In both cases this can result in inconsistent tracks. In this thesis we instead use optimisation that considers future observation and thereby chooses the correct mode.

2. In the multi object case the varying dimensionality of the states makes the mean value difficult to define. In [33] it is suggested to threshold the likelihood of each object in the configurations being present. Then the mean state of each object for which this likelihood is above some threshold is calculated separately.

3. Restrictions placed in the dynamic model, such as illegal state transactions, are not enforced, and the resulting state sequence might contain illegal state transactions. For the particle filter also restrictions in the prior state distribution might be violated. In [33] for example, the prior probability of two objects overlapping in the 3D scene is set to zero, as this is impossible. However the output mean state value may still be a state where two objects overlap, as the two objects may originate from different sets of particles.

The problem is that only single states are considered. In this thesis state sequences are considered, which means that impossible state transactions will never appear in the results. Neither will impossible states, as the optimisation produces a single state sequence as the optimum, and there never is any need to calculate the mean over different states.

Relation to other non causal approaches

Classically tracking algorithms have been constructed to be strictly casual and only considers past frames when making a decision of the current state of the tracked objects. In many surveillance applications this is an unnecessary restriction though. When doing for example people counting or loitering detection there is typically no problem in allowing a delay of several seconds between an event happening in the scene and the detection made by the system. In those cases it is possible to let the decisions not only depend of past frames but also on future frames.

Recent work on tracking have started to move from causal solutions into the non causal solutions. Zang and Nevatia [91] considers tracking as an offline problem where all data is available when the algorithm starts. Berclaz *et al* [4] uses a 4 second long sliding window which is optimised in an offline fashion and then only the 10 first frames are stored as output. Leibe *et al* [50] formulates their tracker in an online casual way but reevaluates its view of what happened in the past as new frames are received. This means that the final result for those past reevaluated frames do depend on future frames.

The work presented in this theses shows how to use classical hidden Markov models with this online but non casual approach. It is done in a way that guarantees consistency in the produced tracks, i.e. no jumping between different hypothesis, which is not the case with the sliding window approach. Also it can assess whether a global optimum were actually found or not.

1.6 Contributions

The contribution of this thesis is

1. A theoretical derivation of the probability distribution function of the correlation coefficient between two patches where the two patches are either uncorrelated or only differ by some translation, rescaling and additive Gaussian noise
2. A suggestion of using recursive quantile estimators instead of learning factors for estimating background models.
3. Two background/foreground segmentation algorithms efficient enough to be used in real time embedded in cameras that can handle the varying lighting condition and static backgrounds of outdoor traffic surveillance.
4. An online version of the Viterbi algorithm (dynamic programming) that will output the resulting globally optimal maximum likelihood state sequence of an hidden Markov model even before the last observation symbol have been received and can thus be used for infinite state sequences.
5. A modification to the Viterbi algorithm (dynamic programming) that will make it produce the globally optimal maximum likelihood state sequence of an hidden Markov model without evaluating every possible state in every frame enabling the use of very large, possible infinite, state spaces.
6. An online real time tracking algorithm that can track a varying number of objects of different types with sensor fusion form several cameras observing the same scene.

Chapter 2

Matching Image Patches using Correlation

2.1 Introduction

The correlation between two signals (cross correlation) is a standard tool for assessing the degree to which two signals are similar. It is a basic approach to match two image patches, for feature detection [14] as well as a component of more advanced techniques [7]. The technique has several advantages. Firstly, the cross correlation is fairly easy to compute. Fourier methods can be used to compute the cross correlation fast, when used for matching a patch in a general position in an image. Secondly, the cross correlation is independent of translations and scaling in the intensity domain. Thus it is fairly independent of lighting variations.

Numerous authors use cross-correlation for matching, [7, 78]. The technique has been shown to give good results in many situations where the patches has enough structure. However, there has been little attention to probabilistic models of the correlation coefficient. The contribution of this work is to make it usable even when there is no structure, e.g. a blank wall. The correlation between two almost uniform patches is close to 0, which will make the entire wall foreground if a thresholded cross-correlation were used for foreground/background-segmentation, as illustrated in Figure 2.1. The theory in this paper makes it possible to asses how good the correlation value is in determining if two patches are the same based on the amount of structure in the patch, c.f. Figure 2.1 (last column). In Chapter 3 the use of cross correlation as a feature for background foreground segmentation will be investigated.

In [40] image patches are matched by first transforming the signal to a binary signal and then forming a correlation coefficient called increment sign correlation. They also calculate the distribution function of the increment sign correlation assuming a Gaussian noise model in image intensity. Much information is, however, lost in the binarisation and the remaining theory is only applicable for binary signals.

In this paper we derive the distribution function of the cross-correlation coefficient in two different cases: (i) the cross-correlation coefficient between two random independent

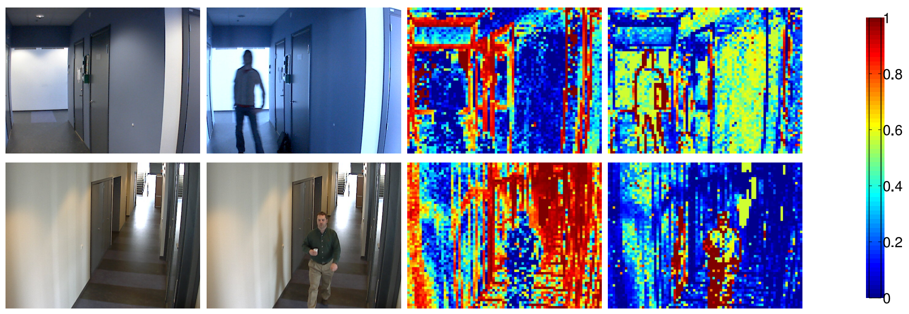


Figure 2.1: A background image (first column), the current frame (second column), the normalised cross correlation, c, between each 8×8 block (third column) and the foreground probability (last column). Note that the correlation, c is low both for the foreground object and the uniformly coloured wall, while the foreground probability is close to 0.5 (e.g unknown) for the uniformly coloured regions, close to 0 for most background with structure and close to 1 for foreground with structure. See also `fgbg_ncc.avi` at http://www.maths.lth.se/˜ardo/thesis/.

patches and (ii) between two patches that differ only by scale, translation and additive Gaussian noise. Using these two distributions the patch matching problem is formulated as a binary classification problem and the probability of two patches matching is derived using Bayes' formula.

In Section 3.1.3 of Chapter 3, the theory is applied to the problem of background foreground segmentation, which can be made more robust to changes in lighting using patches instead of individual pixels. Furthermore the quality of the segmentation can be assessed automatically and precisely. This is necessary as the background might contain patches with very little structure. Those patches are detected as more uncertain than patches with more structure.

2.2 Correlation Coefficient

In this chapter the cross correlation between small patches, typically 4×4 or 8×8 pixels, will be studied. It does not depend on the two dimensional structure of the patch, which allows each patch, $\mathbf{a}$ to be represented as a one dimensional vector, $\mathbf{a} = (a_1, a_2, \cdots a_d)$, where a_k is the grey level of pixel k, and d is the total number of pixels in the patch. The ordering of the pixels is not important as long as the same order is always used. The following notation for the mean, $\bar{\mathbf{a}} = \frac{1}{d}\sum a_k$, the displacement, $\hat{a}_k = a_k - \bar{\mathbf{a}}$ and the length (amount of structure or variance), $|\hat{\mathbf{a}}|^2 = \sum \hat{a}_k^2$ will be used. The correlation coefficient, c, between two patches, $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$c = \frac{\sum \hat{a}_k \hat{b}_k}{|\hat{\mathbf{a}}| \left|\hat{\mathbf{b}}\right|} = \frac{\hat{\mathbf{a}}}{|\hat{\mathbf{a}}|} \cdot \frac{\hat{\mathbf{b}}}{\left|\hat{\mathbf{b}}\right|} , \tag{2.1}$$

where $\cdot$ denotes scalar multiplication. Note that $c = \cos\alpha$, with α the angle between the two vectors $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$.

The patch matching problem can be formulated as a binary Bayesian classification problem, with one feature, c. In other words, given a known patch and one or several candidate patches it is possible to calculate the probability, for each of the candidate patches, that they are noisy, rescaled and translated versions of the known patch using Bayes' formula. To do that the distribution function, $f_{\text{bg}}(c)$, of correlating the known patch with a noisy rescaled and translated version of itself is compared with the distributing function, $f_{\text{fg}}(c)$, of correlating the known patch with a random uncorrelated patch. The foreground distribution, $f_{\text{fg}}(c)$, will only depend on the dimension d, while the background distribution, $f_{\text{bg}}(c)$, will also depend on the amount of structure, i.e. the length $|\hat{\mathbf{a}}|$, in the known patch and the noise level.

2.2.1 Foreground Distribution

To derive the foreground distribution, consider a known patch $\mathbf{p}$ and a random patch $\mathbf{r}$ with independent Gaussian distributed pixels with identical mean value. This is a rather crude approximation of the distribution of patches that assumes that two unrelated patches are uncorrelated. As is shown by the experiments below, this is not the situation in the general case. The approximation is however not as severe in typical traffic surveillance scenes where the background mostly consists of pavement. Also, looking at this simpler case first will make it easier to understand the derivation of the background distribution, which is more complicated but follows the same general idea and the background distribution does very accurately model the general case as is also shown in the experiments section.

The correlation coefficient, c, can be calculated from $\mathbf{p}$ and $\mathbf{r}$ in four steps.

1. Remove the mean of $\mathbf{p}$ and $\mathbf{r}$.

2. Rotate coordinate system to place $\mathbf{p}$ on the first coordinate axis.

3. Scale and $\mathbf{r}$ to unit length.

4. Let $c = r_1$.

Removing the mean is an orthogonal projection on the plane $(1, 1, 1, \cdots)$. The resulting vector $\hat{\mathbf{r}}$ is still normally distributed on a $d - 1$ dimensional subspace, now with mean zero. Scaling $\mathbf{r}$ to unit length means integrating the distribution function along rays from the origin. In the case of a rotational symmetric distribution this results in an uniform distribution on the $d-1$ dimensional unit hyper-sphere, which is unchanged by rotating the coordinate system. The same result is found in the case of a Gaussian distribution with a general non singular covariance matrix. Before an explicit expression of f_{fg} and a strict proof is given in Lemma 2, the following lemma will be proved. It represents step 1-2 above. Plots of the distribution function is found in Figure 2.2 (left).

Lemma 1. *Let* $\mathbf{r}$ *be a d-dimensional Gaussian distributed random vector with mean* $\mathbf{u}$*, and covariance matrix* $\mathbf{\Sigma}$*, i.e.* $f(\mathbf{r}) = \mathcal{N}(\mathbf{r}\,|\mathbf{u}, \mathbf{\Sigma})$. *Let* $\mathbf{p}$ *be a fixed given d-dimensional vector. Let* $c = \cos\alpha$ *be the correlation coefficient between* $\mathbf{r}$ *and* $\mathbf{p}$. *Then there exists a* $d-1$ *dimensional random vector* $\hat{\mathbf{r}}'$*, such that the angle between* $\hat{\mathbf{r}}'$ *and the first coordinate axis is* α*, and*

$$f(\hat{\mathbf{r}}') = \mathcal{N}(\hat{\mathbf{r}}'\,|\mathbf{u}', \mathbf{\Sigma}')\ , \tag{2.2}$$

with $\mathbf{\Sigma}'$ *and* $\mathbf{u}'$ *defined in the proof below. Furthermore,*

$$\begin{array}{ccc} \mathbf{u} = m\mathbf{1}_d & \Rightarrow & \mathbf{u}' = \mathbf{0}_{d-1} \\ \mathbf{u} = a\mathbf{p} + b\mathbf{1}_d & \Rightarrow & \mathbf{u}' = (\mathbf{a}\,|\mathbf{p}|\,, 0, 0, \cdots, 0) \\ \mathbf{\Sigma} = \sigma^2\mathbf{I}_d & \Rightarrow & \Sigma' = \sigma^2\mathbf{I}_{d-1} \\ \mathbf{\Sigma}\ \textit{non singular} & \Rightarrow & \mathbf{\Sigma}'\ \textit{non singular} \end{array}\ , \tag{2.3}$$

where $a, b, m \in \mathbb{R}$.

Proof. The displacement of $\mathbf{r}$ can be written as the linear transformation $\hat{\mathbf{r}} = \mathbf{A}\mathbf{r}$, with $\mathbf{A} = \mathbf{I}_d - \frac{1}{d}\mathbf{1}_{d\times d}$, where $\mathbf{1}_{d\times d}$ is a $d\times d$ matrix with all elements set to 1, e.g.

$$\mathbf{A} = \begin{pmatrix} 1-\frac{1}{d} & -\frac{1}{d} & -\frac{1}{d} & \cdots & -\frac{1}{d} \\ -\frac{1}{d} & 1-\frac{1}{d} & -\frac{1}{d} & \cdots & -\frac{1}{d} \\ & & \cdots & & \\ -\frac{1}{d} & -\frac{1}{d} & -\frac{1}{d} & \cdots & 1-\frac{1}{d} \end{pmatrix}\ . \tag{2.4}$$

This means that $f(\hat{\mathbf{r}}) = \mathcal{N}\left(\hat{\mathbf{r}}\,\middle|\mathbf{A}\mathbf{u}, \mathbf{A}\mathbf{\Sigma}\mathbf{A}^{\mathrm{T}}\right)$. This distribution is singular as $\hat{\mathbf{r}}$ always lies on the $d-1$ dimensional hyper-plane $\sum \hat{r}_k = 0$. By rotating the coordinate system using the matrix

$$\mathbf{B} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & \cdots & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{-2}{\sqrt{6}} & 0 & 0 & \cdots & 0 \\ \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{1}{\sqrt{12}} & \frac{-3}{\sqrt{12}} & 0 & \cdots & 0 \\ \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{1}{\sqrt{20}} & \frac{-4}{\sqrt{20}} & \cdots & 0 \\ & & \cdots & & & & \\ \frac{1}{\sqrt{d^2-d}} & \frac{1}{\sqrt{d^2-d}} & \frac{1}{\sqrt{d^2-d}} & \frac{1}{\sqrt{d^2-d}} & \cdots & \frac{1}{\sqrt{d^2-d}} & \frac{-d+1}{\sqrt{d^2-d}} \\ \frac{1}{\sqrt{d}} & \frac{1}{\sqrt{d}} & \frac{1}{\sqrt{d}} & \frac{1}{\sqrt{d}} & \cdots & \frac{1}{\sqrt{d}} & \frac{1}{\sqrt{d}} \end{pmatrix} \tag{2.5}$$

the last coordinate, will always be 0. The length of all row-vectors in $\mathbf{B}$ is one and the scalar product between any two row vectors is zeros. This means that $\mathbf{B}$ is an orthonormal matrix and therefor a rotation (and possible mirroring) matrix. The distribution of the rotated vector becomes

$$f\left(\mathbf{B}\hat{\mathbf{r}}\right) = \mathcal{N}\left(\mathbf{B}\hat{\mathbf{r}}\left|\mathbf{B}\mathbf{A}\mathbf{u}, \mathbf{B}\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\right.\right). \tag{2.6}$$

By using $\mathbf{A} = \mathbf{I}_d - \frac{1}{d}\mathbf{1}_{d\times d}$ and splitting $\mathbf{B}$ in the block matrix form

$$\mathbf{B} = \left(\begin{array}{c} \mathbf{B}' \\ \frac{1}{\sqrt{d}}\mathbf{1}_d{}^{\mathrm{T}} \end{array}\right), \tag{2.7}$$

the matrix $\mathbf{BA}$ can be expanded into

$$\mathbf{BA} = \left(\begin{array}{c} \mathbf{B}' - \mathbf{B}'\frac{1}{d}\mathbf{1}_{d\times d} \\ \frac{1}{\sqrt{d}}\mathbf{1}_d{}^{\mathrm{T}} - \frac{1}{\sqrt{d}}\mathbf{1}_d{}^{\mathrm{T}}\frac{1}{d}\mathbf{1}_{d\times d} \end{array}\right). \tag{2.8}$$

Each row of $\mathbf{B}'$ sums to 0, which means that $\mathbf{B}'\mathbf{1}_{d\times d} = \mathbf{0}_{d\times d}$. By also using that $\mathbf{1}_d{}^{\mathrm{T}}\mathbf{1}_{d\times d} = d\mathbf{1}_d{}^{\mathrm{T}}$, this simplifies to

$$\mathbf{BA} = \left(\begin{array}{c} \mathbf{B}' \\ \mathbf{0}_d{}^{\mathrm{T}} \end{array}\right). \tag{2.9}$$

This allows the variance of $f\left(\mathbf{B}\hat{\mathbf{r}}\right)$ in Equation 2.6 to be simplified into

$$\mathbf{B}\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}} = \left(\begin{array}{cc} \mathbf{B}'\boldsymbol{\Sigma}\mathbf{B}'^{\mathrm{T}} & \mathbf{0}_{d-1} \\ \mathbf{0}_{d-1}{}^{\mathrm{T}} & 0 \end{array}\right). \tag{2.10}$$

The expressions of Equation 2.9 shows that the last coordinate of $\mathbf{BAp} = \mathbf{B}\hat{\mathbf{p}}$ and $\mathbf{BAr} = \mathbf{B}\hat{\mathbf{r}}$ will be 0. To make every coordinate but the first one 0 in $\hat{\mathbf{p}}$ it is possible to find a rotational matrix, $\mathbf{C}$, such that $\mathbf{CB}\hat{\mathbf{p}} = \left(\left|\hat{\mathbf{p}}\right|, 0, 0, \cdots, 0\right)$. This rotational matrix is independent of $\mathbf{r}$ and will be or the form

$$\mathbf{C} = \left(\begin{array}{cc} \mathbf{C}' & \mathbf{0}_{d-1} \\ \mathbf{0}_{d-1} & 1 \end{array}\right), \tag{2.11}$$

since the last coordinate was already 0. This means that the last coordinate of $\mathbf{CB}\hat{\mathbf{r}}$ will remain 0.

Let $\hat{\mathbf{r}}'$ and $\hat{\mathbf{p}}'$ be the $d-1$ first coordinates of $\mathbf{CB}\hat{\mathbf{r}}$ and $\mathbf{CB}\hat{\mathbf{p}}$,

$$\mathbf{CB}\hat{\mathbf{p}} = \left(\begin{array}{c} \mathbf{C}'\mathbf{B}'\hat{\mathbf{p}} \\ 0 \end{array}\right) = \left(\begin{array}{c} \hat{\mathbf{p}}' \\ 0 \end{array}\right), \tag{2.12}$$

$$\mathbf{CB}\hat{\mathbf{r}} = \left(\begin{array}{c} \mathbf{C}'\mathbf{B}'\hat{\mathbf{r}}' \\ 0 \end{array}\right) = \left(\begin{array}{c} \hat{\mathbf{r}}' \\ 0 \end{array}\right). \tag{2.13}$$

Then the angle, α, between $\hat{\mathbf{r}}'$ and $\hat{\mathbf{p}}'$ will be the same as the angle between $\hat{\mathbf{r}}$ and $\hat{\mathbf{p}}$ as the omitted coordinate is 0 for both $\hat{\mathbf{r}}$ and $\hat{\mathbf{p}}$ and the angle is invariant to coordinate system rotation. As noted above the correlation coefficient between $\mathbf{r}$ and $\mathbf{p}$ is $\cos\alpha$. This gives $\hat{\mathbf{p}}' = (|\hat{\mathbf{p}}|, 0, 0, \cdots, 0)$ and

$$f(\hat{\mathbf{r}}') = \mathcal{N}(\hat{\mathbf{r}}' | \mathbf{u}', \mathbf{\Sigma}'), \tag{2.14}$$

with,

$$\mathbf{u}' = \mathbf{C}'\mathbf{B}'\mathbf{u} \tag{2.15}$$

and

$$\mathbf{\Sigma}' = \mathbf{C}'\mathbf{B}'\mathbf{\Sigma}\mathbf{B}'^{\mathrm{T}}\mathbf{C}'^{\mathrm{T}} \tag{2.16}$$

The vector $\hat{\mathbf{p}}'$ is located on the first coordinate axis and thus the angle between $\hat{\mathbf{r}}'$ and $\hat{\mathbf{p}}'$ is the angle between $\hat{\mathbf{r}}'$ and the first coordinate axis. This concludes the proof of the first statement regarding the existence or $\hat{\mathbf{r}}'$ and that it is Gaussian distributed. Regrading the later statements, consider them one by one:

- If $\mathbf{u} = m\mathbf{1}_d$ the conclusion that each row of $\mathbf{B}'$ sums to 0 also means that $\mathbf{u}' = \mathbf{C}'\mathbf{B}'\mathbf{u} = \mathbf{0}_d$.

- If $\mathbf{u} = a\mathbf{p} + b\mathbf{1}_d$, then

$$\mathbf{u}' = \mathbf{C}'\mathbf{B}'(a\mathbf{p} + b\mathbf{1}_d) = a\mathbf{C}'\mathbf{B}'\mathbf{p} = a\hat{\mathbf{p}}' = (a|\hat{\mathbf{p}}|, 0, 0, \cdots, 0). \tag{2.17}$$

- If $\mathbf{\Sigma} = \sigma^2\mathbf{I}_d$, then σ^2 will factor out and $\mathbf{B}'\mathbf{B}'^{\mathrm{T}} = \mathbf{I}_{d-1}$ since $\mathbf{B}'$ is orthonormal as noted above and $\mathbf{C}'\mathbf{C}'^{\mathrm{T}} = \mathbf{I}_{d-1}$ since $\mathbf{C}'$ is a rotation matrix. In this case $\mathbf{\Sigma}' = \sigma^2\mathbf{I}_{d-1}$.

- Finally, if $\mathbf{\Sigma}$ is non singular so is $\mathbf{\Sigma}' = \mathbf{C}'\mathbf{B}'\mathbf{\Sigma}\mathbf{B}'^{\mathrm{T}}\mathbf{C}'^{\mathrm{T}}$

□

Using Lemma 1 it is now possible to prove the following lemma.

Lemma 2. *The distribution function of the correlation coefficient, c, between a given fixed patch $\mathbf{p}$ and a random, d dimensional, patch, $\mathbf{r}$, with Gaussian distributed pixels with identical mean and non singular covariance matrix, is for $d > 3$,*

$$f_{fg}(c) = \frac{\Gamma\left(\frac{d}{2}\right)}{\sqrt{\pi}\Gamma\left(\frac{d}{2} - \frac{1}{2}\right)} \left(1 - c^2\right)^{\frac{d-3}{2}} . \tag{2.18}$$

Proof. Let $\mathbf{r}$ be a d-dimensional Gaussian distributed random vector with mean $\mathbf{u} = (m, m, m, \cdots, m)$ and covariance matrix $\mathbf{\Sigma}$. Let $\mathbf{p}$ be a d-dimensional given fixed vector. According to Lemma 1, there exits a vector $\hat{\mathbf{r}}'$ with distribution

$$f_{\hat{\mathbf{r}}'}(\hat{\mathbf{r}}') = \mathcal{N}(\hat{\mathbf{r}}' | \mathbf{0}_{d-1}, \mathbf{\Sigma}') \ , \tag{2.19}$$

and the correlation coefficient between $\mathbf{p}$ and $\mathbf{r}$ is $\cos\alpha$ for α, the angle between $\hat{\mathbf{r}}'$ and the first coordinate axis. According to Lemma 1, the matrix $\mathbf{\Sigma}'$ is non-singular and can thus be factored into $\mathbf{\Sigma} = \mathbf{S}\mathbf{S}^{\mathrm{T}}$ using Cholesky factorisation with $\mathbf{S}$ invertible. This means

$$f_{\hat{\mathbf{r}}'}(\hat{\mathbf{r}}') = \mathcal{N}\left(\mathbf{S}^{-1}\hat{\mathbf{r}}' | \mathbf{0}_{d-1}, \mathbf{I}\right) \ . \tag{2.20}$$

By normalising $\hat{\mathbf{r}}'$ to unit length $\cos\alpha$ will be equal to the first coordinate, x_1, of $\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}$. The distribution function of $\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}$ is found by integrating the distribution of $\hat{\mathbf{r}}'$,

$$f_{\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}}(\mathbf{x}) = \int_{\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|} = \mathbf{x}} f_{\hat{\mathbf{r}}'}(\hat{\mathbf{r}}') \, . \tag{2.21}$$

The length of $\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}$ is 1, which means that for $|\mathbf{x}| \neq 1$ the integration set will be the empty set and $f_{\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}}(\mathbf{x}) = 0$. To evaluate this integral for any other fix $\mathbf{x}$, choose a rotation matrix $\mathbf{R}$ such that $\mathbf{R}\mathbf{x} = (1, 0, 0, \cdots)$ and rotate the coordinate system by making the variable substitution $\mathbf{s} = \mathbf{R}\hat{\mathbf{r}}'$. The integration set will then become,

$$\frac{\mathbf{s}}{|\mathbf{s}|} = \mathbf{R}\mathbf{x} = (1, 0, 0, \cdots) \, , \tag{2.22}$$

which is the first coordinate axis. By letting $\mathbf{s} = (t, 0, 0, \cdots)$, the integral becomes

$$f_{\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}}(\mathbf{x}) = \int_{t=0}^{\infty} f_{\hat{\mathbf{r}}'}\left(\mathbf{R}^{\mathrm{T}}(t, 0, \cdots)\right) \mathrm{d}t = \int_{t=0}^{\infty} \mathcal{N}\left(\mathbf{S}^{-1}\mathbf{R}^{\mathrm{T}}(t, 0, \cdots) | \mathbf{0}_{d-1}, \mathbf{I}\right) \mathrm{d}t. \tag{2.23}$$

By introducing $\mathbf{a} = \mathbf{S}^{-1}\mathbf{R}^{\mathrm{T}}(1, 0, \cdots)$, and making another variable substitution, $s = |\mathbf{a}|\, t$ with $\mathrm{d}s = |\mathbf{a}|\, \mathrm{d}t = \left|\mathbf{S}^{-1}\mathbf{R}^{\mathrm{T}}\right| \mathrm{d}t = \left|\mathbf{S}^{-1}\right| \mathrm{d}t$ since $\left|\mathbf{R}^{\mathrm{T}}\right| = 1$, this becomes,

$$f_{\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}}(\mathbf{x}) = \int_{s=0}^{\infty} \mathcal{N}\left(\frac{\mathbf{a}}{|\mathbf{a}|} s \,|\mathbf{0}_{d-1}, \mathbf{I}\right) \frac{1}{|\mathbf{S}^{-1}|} \mathrm{d}s. \tag{2.24}$$

The length of $\frac{\mathbf{a}}{|\mathbf{a}|}$ is 1 and $\mathcal{N}(\cdot | \mathbf{0}_{d-1}, \mathbf{I})$ is spherically symmetric, which means that integrating it along the direction defined by $\mathbf{a}$ will generate some constant independent of $\mathbf{a}$. This means that the entire integral will be some constant, u, and the full probability distribution becomes,

$$f_{\frac{\hat{\mathbf{r}}'}{|\hat{\mathbf{r}}'|}}(\mathbf{x}) = \begin{cases} u & \text{if } \sum_{k=1}^{d-1} \mathbf{x}_k^2 = 1, \\ 0 & \text{otherwise} \end{cases} \tag{2.25}$$

for some constant u. Setting $x_1 = c$ and integrating out all other variables gives the distribution of the correlation coefficient. The integrand is zero outside the unit hypersphere $\sum_{k=1}^{d-1} \mathbf{x}_k^2 = c^2 + \sum_{k=2}^{d-1} \mathbf{x}_k^2 = 1$, which means the integral can be restricted to this set,

$$f_c(c) = \int_{\sum_{k=2}^{d-1} x_k^2 = 1-c^2} u \mathrm{d}S_{d-2} \ , \tag{2.26}$$

where $\mathrm{d}S_{d-2}$ is the $d-2$ dimensional spherical area element. This results in that the integration becomes restricted to the surface of the $d-2$ dimensional unit hyper-sphere. The integrand is constant, which means that the integral evaluates into this constant times the surface area of the $d-2$ dimensional hyper-sphere with radius $\sqrt{1-c^2}$, which makes $f_{\mathrm{fg}}(c)$ proportional to $\sqrt{1-c^2}^{d-3}$. The normalising constant is found by making sure that $f_c(\cdot)$ integrates to one, i.e.

$$\int_{-1}^{1} \sqrt{1-c^2}^{d-3} \mathrm{d}c = \frac{\sqrt{\pi}\Gamma\left(\frac{d}{2}-\frac{1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \ , \tag{2.27}$$

which concludes the proof. □

The lemma does not use the assumption of the patch being Gaussian distributed, only that the integral of the probability distribution function along any ray from origo to infinity is constant. It would probably be possible to the extend the lemma to a larger class of distributions, including at least all spherical symmetric distributions.

2.2.2 Background Distribution

By following the same general ideas as in the previous section, the distribution of the correlation coefficient between a given patch and a noisy sample of the same patch can be derived. The distribution is, however, no longer symmetrical around origo.

Lemma 3. *The distribution function of the correlation coefficient, c, between a given fixed patch* $\mathbf{p}$ *of dimension* d *and* $\mathbf{r} = a\mathbf{p} + b\mathbf{1}_d + \mathbf{w}$, *where* $f(\mathbf{w}) = \mathcal{N}\left(\mathbf{w} \left| \mathbf{0}_d, \sigma^2 I_d\right.\right)$ *and* $a, b \in \mathbb{R}$, *is for* $d > 3$,

$$f_{bg}(c|\hat{\sigma}) = \frac{\sqrt{1 - c^2}^{d-4}}{\sqrt{\pi}} e^{\frac{c^2-1}{2\hat{\sigma}^2}} \cdot$$
$$\cdot \sum_{k=0}^{d-2} \binom{d-2}{k} \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{d-2}{2}\right)} \left(\frac{c}{\sqrt{2}\hat{\sigma}}\right)^{d-2-k} \begin{cases} 1 + \frac{c}{|c|} - \frac{c\Gamma\left(\frac{k+1}{2}, \frac{c^2}{2\hat{\sigma}^2}\right)}{|c|\Gamma\left(\frac{k+1}{2}\right)} & k \text{ even} \\ \frac{\Gamma\left(\frac{k+1}{2}, \frac{c^2}{2\hat{\sigma}^2}\right)}{\Gamma\left(\frac{k+1}{2}\right)} & k \text{ odd} \end{cases} , \tag{2.28}$$

where $\hat{\sigma} = \frac{\sigma}{a|\hat{\mathbf{p}}|}$. *Here* $l = \frac{1}{\hat{\sigma}}$ *can be thought of as a a signal to noise ratio of the observed patch.*

Proof. Let $\mathbf{p}$ be a d-dimensional given fixed vector. Let $\mathbf{r}$ be a d-dimensional Gaussian distributed random vector with mean $a\mathbf{p} + b\mathbf{1}_d$ and covariance matrix $\Sigma = \sigma^2\mathbf{I}_d$. According to Lemma 1,

$$f_{\hat{\mathbf{r}}'}\left(\hat{\mathbf{r}}'\right) = \mathcal{N}\left(\hat{\mathbf{r}}'\left|\left(a\left|\hat{\mathbf{p}}\right|, 0, 0, \cdots, 0\right), \sigma^2\mathbf{I}_{d-1}\right.\right), \tag{2.29}$$

and the correlation coefficient between $\mathbf{p}$ and $\mathbf{r}$ is $\cos\alpha$ for α, the angle between $\hat{\mathbf{r}}'$ and the first coordinate axis. This angle will not change by rescaling $\hat{\mathbf{r}}'$, and the notation becomes smoother if $\frac{\hat{\mathbf{r}}'}{a|\hat{\mathbf{p}}|}$ is used instead. The distribution of it is

$$f_{\frac{\hat{\mathbf{r}}'}{a|\hat{\mathbf{p}}|}}\left(\mathbf{x}\right) = \mathcal{N}\left(\mathbf{x}\left|\left(1, 0, 0, \cdots, 0\right), \hat{\sigma}^2\mathbf{I}_{d-1}\right.\right), \tag{2.30}$$

with $\hat{\sigma} = \frac{\sigma}{a|\hat{\mathbf{p}}|}$. The distribution of $c = \cos\alpha = \frac{x_1}{|\mathbf{x}|}$ is found by integrating this distribution,

$$f_c\left(c\right) = \int_{\frac{x_1}{|\mathbf{x}|}=c} f_{\frac{\hat{\mathbf{r}}'}{a|\hat{\mathbf{p}}|}}\left(\mathbf{x}\right)\mathrm{d}\mathbf{x} = \left(\frac{1}{\sqrt{2\pi}\hat{\sigma}}\right)^{d-1}\int_{\frac{x_1}{|\mathbf{x}|}=c} e^{-\frac{|(1,0,0,\cdots,0)-\mathbf{x}|^2}{2\hat{\sigma}^2}}\mathrm{d}\mathbf{x}. \tag{2.31}$$

The squared distance from the mean can be simplified by

$$\left|\left(1, 0, 0, \cdots, 0\right) - \mathbf{x}\right|^2 = \left(1 - x_1\right)^2 + \sum_{k=2}^{d-1} x_k^2 = 1 - 2x_1 + \left|\mathbf{x}\right|^2. \tag{2.32}$$

To evaluate the integral, introduce the hyper-spherical coordinates [84] (in $d-1$ dimensions)

$$\begin{aligned} x_1 &= t\cos\left(\phi_1\right), \\ x_2 &= t\sin\left(\phi_1\right)\cos\left(\phi_2\right), \\ x_3 &= t\sin\left(\phi_1\right)\sin\left(\phi_2\right)\cos\left(\phi_3\right), \\ &\cdots \\ x_{d-1} &= t\sin\left(\phi_1\right)\cdots\sin\left(\phi_{d-2}\right), \\ \mathrm{d}x &= t^{d-2}\sin^{d-3}\left(\phi_1\right)\sin^{d-4}\left(\phi_2\right)\cdots\sin\left(\phi_{d-3}\right)\mathrm{d}t\mathrm{d}\phi_1\mathrm{d}\phi_2\cdots\mathrm{d}\phi_{d-2}, \end{aligned} \tag{2.33}$$

which gives

$$\begin{aligned} f_c\left(c\right) = &\left(\frac{1}{\sqrt{2\pi}\hat{\sigma}}\right)^{d-1}\cdot \\ &\cdot\int_{\cos\phi_1=c} e^{-\frac{1-2t\cos(\phi_1)+t^2}{2\hat{\sigma}^2}} t^{d-2}\sin^{d-3}\left(\phi_2\right)\cdots\sin\left(\phi_{d-3}\right)\mathrm{d}r\mathrm{d}\phi_1\cdots\mathrm{d}\phi_{d-2}. \end{aligned} \tag{2.34}$$

By factoring the integrand and writing it as an iterative integral, all influence of $\phi_2 \cdots \phi_{d-2}$ can be factored out and evaluated separately,

$$v = \int_{\phi_2=0}^{\pi} \sin^{d-4}(\phi_2)\, \mathrm{d}\phi_2 \cdots \int_{\phi_{d-3}=0}^{\pi} \sin(\phi_{d-3})\, \mathrm{d}\phi_{d-3} \int_{\phi_{d-2}=0}^{2\pi} \mathrm{d}\phi_{d-2}\ . \tag{2.35}$$

This is the surface area of a $d-3$ dimensional unit hyper-sphere,

$$v = \frac{2\pi^{\frac{d-2}{2}}}{\Gamma(\frac{d-2}{2})}. \tag{2.36}$$

The variable ϕ_1 is constant, which means that all that remains is to integrate over t, and this integral can be expressed in terms of c instead of ϕ_1 by using the variable substitution $c = \cos\phi_1$,

$$\int_{\cos\phi_1=c} e^{-\frac{1-2t\cos(\phi_1)+t^2}{2\hat{\sigma}^2}} t^{d-2} \sin^{d-3}(\phi_1)\, \mathrm{d}t = \\ = \int_{t=0}^{\infty} e^{-\frac{1-2tc+t^2}{2\hat{\sigma}^2}} t^{d-2} \sqrt{1-c^2}^{\,d-3} \mathrm{d}t. \tag{2.37}$$

By collecting all constant terms into k, the integral becomes

$$f_c(c) = k \int_{t=0}^{\infty} e^{-\frac{1-2tc+t^2}{2\hat{\sigma}^2}} t^{d-2} \mathrm{d}t, \tag{2.38}$$

with the constant term

$$k = \left(\frac{1}{\sqrt{2\pi}\hat{\sigma}}\right)^{d-1} \frac{2\pi^{\frac{d-2}{2}}}{\Gamma(\frac{d-2}{2})} \sqrt{1-c^2}^{\,d-3} = \frac{\sqrt{1-c^2}^{\,d-4}}{\Gamma\left(\frac{d-2}{2}\right) \sqrt{2}^{\,d-3} \sqrt{\pi}\hat{\sigma}^{d-1}}. \tag{2.39}$$

A closed form solution to the integral is found by rewriting $1-2tc+t^2 = (t-c)^2 - c^2 + 1$ and using the variable substitution $\hat{t} = \frac{t-c}{\sqrt{2}\hat{\sigma}}$, which results in

$$\int_{t=0}^{\infty} e^{-\frac{1-2tc+t^2}{2\hat{\sigma}^2}} t^{d-2} \mathrm{d}t = \int_{t=-\frac{c}{\sqrt{2}\hat{\sigma}}}^{\infty} e^{\frac{c^2-1}{2\hat{\sigma}^2}} e^{-r^2} \left(\sqrt{2}\hat{\sigma}t + c\right)^{d-2} \sqrt{2}\hat{\sigma}\mathrm{d}t. \tag{2.40}$$

By factoring out some constants, this integral can be written

$$e^{\frac{c^2-1}{2\hat{\sigma}^2}} \left(\sqrt{2}\hat{\sigma}\right)^{d-1} \int_{t=-\frac{c}{\sqrt{2}\hat{\sigma}}}^{\infty} e^{-r^2} \left(t + \frac{c}{\sqrt{2}\hat{\sigma}}\right)^{d-2} \mathrm{d}t. \tag{2.41}$$

After introducing the constant $a = \frac{c}{\sqrt{2}\hat{\sigma}}$, the binomial theorem and a change of the order of summation and integration gives

$$\int_{-a}^{\infty} e^{-t^2} (t+a)^{d-2} \,\mathrm{d}r \quad = \quad \sum_{k=0}^{d-2} \binom{d-2}{k} a^{d-2-k} \int_{-a}^{\infty} e^{-t^2} t^k \mathrm{d}r \qquad , \tag{2.42}$$

This can be simplified using partial integration recursively, by factoring the integrand, $e^{-t^2} t^k = t e^{-t^2} t^{k-1}$, and then taking the primitive of the factor te^{-t^2}. Let

$$\begin{aligned} p_k = \int_{-a}^{\infty} e^{-t^2} t^k \mathrm{d}t = \\ = \left[\frac{e^{-t^2}}{-2} t^{k-1} \right]_{-a}^{\infty} - \int_{-a}^{\infty} \frac{e^{-t^2}}{-2} (k-1)\, t^{k-2} \mathrm{d}t = \\ = \frac{e^{-a^2} (-a)^{k-1}}{2} + \frac{k-1}{2} p_{k-2} \ . \end{aligned} \tag{2.43}$$

This forms two recursive equations, one for even indexes, denoted p_k^{e}, and one for odd indexes, denoted p_k^{o}. They can be written in the form,

$$\begin{aligned} p_l^{\mathrm{o}} = p_{2l+1} = l p_{l-1}^{\mathrm{e}} + \frac{e^{-a^2} (-a)^{2l}}{2} \\ p_l^{\mathrm{e}} = p_{2l} = \frac{2l-1}{2} p_{l-1}^{\mathrm{e}} + \frac{e^{-a^2} (-a)^{2l-1}}{2} \end{aligned} \quad , \tag{2.44}$$

with starting values

$$\begin{aligned} p_0^{\mathrm{o}} = p_1 = \frac{e^{-a^2}}{2} \\ p_0^{\mathrm{e}} = p_2 = \frac{\sqrt{\pi}}{2} (\mathrm{erf}\,(a) + 1) \end{aligned} \quad . \tag{2.45}$$

These equations is of the standard form $x_l = a_l x_{l-1} + b_l$, which has the general solution [73],

$$x_l = c \prod_{i=1}^{l} a_i + \sum_{j=1}^{l} \left(\prod_{i=j+1}^{l} a_i \right) b_j, \tag{2.46}$$

where c is some constant determined by the starting value. The products needed to solve the two equations can be expressed using the gamma function $\Gamma(\cdot)$,

$$\prod_{i=j+1}^{l} i = \frac{\Gamma(l+1)}{\Gamma(j+1)}, \tag{2.47}$$

$$\prod_{i=j+1}^{l} \frac{2i-1}{2} = \frac{\Gamma\left(l+\frac{1}{2}\right)}{\Gamma\left(j+\frac{1}{2}\right)}. \tag{2.48}$$

Applying those formulas to the two recursive equations gives

$$p_l^{\mathrm{o}} = \frac{e^{-a^2}}{2}\Gamma(l+1) + \sum_{j=1}^{l} \frac{\Gamma(l+1)}{\Gamma(j+1)} \frac{e^{-a^2} a^{2j}}{2}, \tag{2.49}$$

and,

$$p_l^{\mathrm{e}} = \frac{\sqrt{\pi}}{2}\left(\operatorname{erf}(a)+1\right)\frac{\Gamma\left(l+\frac{1}{2}\right)}{\Gamma\left(\frac{1}{2}\right)} - \sum_{j=1}^{l} \frac{\Gamma\left(l+\frac{1}{2}\right)}{\Gamma\left(j+\frac{1}{2}\right)} \frac{e^{-a^2} a^{2j-1}}{2}. \tag{2.50}$$

Changing the indexes back to k gives the original sequence,

$$p_k = \begin{cases} \frac{e^{-a^2}}{2}\sum_{j=0}^{\frac{k-1}{2}} \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma(j+1)} a^{2j} & k \text{ odd} \\ \frac{e^{-a^2}}{2}\sum_{j=1}^{\frac{k}{2}} \frac{-\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(j+\frac{1}{2}\right)} a^{2j-1} + \frac{\Gamma\left(\frac{k+1}{2}\right)}{2}\left(\operatorname{erf}(a)+1\right) & k \text{ even} \end{cases}. \tag{2.51}$$

The sum from equation 2.42 will then become a polynomial in a and $\operatorname{erf}(a)$

$$\sum_{k=0}^{d-2} \binom{d-2}{k} a^{d-2-k} p_k = \sum_{i,j} c_{i,j} a^i \operatorname{erf}^j(a), \tag{2.52}$$

with some coefficients $c_{i,j}$. For d larger than about 10, these coefficients become very large and this form is thus numerically very unstable. An alternative solution is to use the standard sum

$$\sum_{0}^{n} \frac{b^i}{\Gamma(i+1)} = \frac{e^b \Gamma(1+n,b)}{\Gamma(1+n)}, \tag{2.53}$$

to write p_k as

$$p_k = \begin{cases} \frac{1}{2}\Gamma\left(\frac{k+1}{1}, a^2\right) & k \text{ odd} \\ \frac{1}{2}\Gamma\left(\frac{k+1}{2}\right) + \frac{a}{2|a|}\left(\Gamma\left(\frac{k+1}{2}\right) - \Gamma\left(\frac{k+1}{2}, a^2\right)\right) & k \text{ even} \end{cases} \tag{2.54}$$

The proof is concluded by putting it all together, resulting in $f_c(c) =$

$$\frac{\sqrt{1-c^2}^{d-4}}{\sqrt{\pi}} e^{\frac{c^2-1}{2\hat{\sigma}^2}} \sum_{k=0}^{d-2} \binom{d-2}{k} \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{d-2}{2}\right)} a^{d-2-k} \begin{cases} 1 + \frac{a}{|a|} - \frac{a\Gamma\left(\frac{k+1}{2}, a^2\right)}{|a|\Gamma\left(\frac{k+1}{2}\right)} & k \text{ even} \\ \frac{\Gamma\left(\frac{k+1}{2}, a^2\right)}{\Gamma\left(\frac{k+1}{2}\right)} & k \text{ odd} \end{cases}. \tag{2.55}$$

For $c < 0$, this simplifies to

$$\frac{\sqrt{1-c^2}^{d-4}}{\sqrt{\pi}} e^{\frac{c^2-1}{2\hat{\sigma}^2}} \sum_{k=0}^{d-2} \binom{d-2}{k} \frac{\Gamma\left(1/2\,k+1/2, a^2\right)}{\Gamma\left(\frac{d-2}{2}\right)} a^{d-2-k}, \tag{2.56}$$

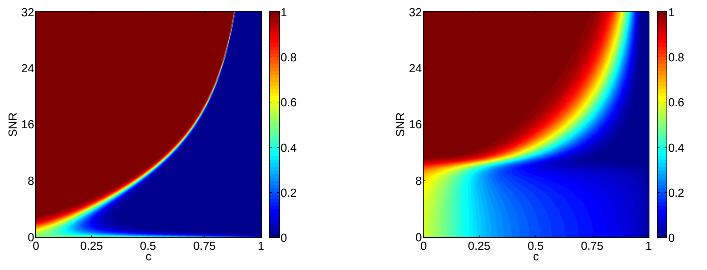


Figure 2.2: *Left:* $p_{\text{fg}|c,l}$ plotted as a function of the correlation c and and the true SNR l. *Right:*$p_{\text{fg}|c,\tilde{l}}$ plotted as a function of correlation c and measured SNR $\tilde{l}$.

and for $c \geq 0$, it simplifies to

$$\frac{\sqrt{1-c^2}^{d-4}}{\sqrt{\pi}} e^{\frac{c^2-1}{2\hat{\sigma}^2}} \left(l + \sum_{k=0}^{d-2} \binom{d-2}{k} \frac{\Gamma\left(1/2\,k+1/2, a^2\right)}{\Gamma\left(\frac{d-2}{2}\right)} a^{d-2-k} \left(-1\right)^{k+1} \right), \tag{2.57}$$

with

$$l = \sum_{k=0,\text{k even}}^{d-2} \binom{d-2}{k} \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{d-2}{2}\right)} a^{d-2-k} 2\,, \tag{2.58}$$

which concludes the proof. $\square$

2.2.3 Bayes' Formula

Given a known background patch it is possible to estimate if the current input patch of a video sequence is this patch or something else. If it is the same patch, the correlation coefficient should be distributed according to f_{bg}, otherwise according to f_{fg}. With $l = \frac{1}{\hat{\sigma}}$ the signal to noise ratio, Bayes' formula and the assumption of a uniform prior, gives the probability

$$p\left(\text{foreground}\,|c,l\,\right) = p_{\text{fg}|c,l} = \frac{f_{\text{fg}}\left(c\,|\,\frac{1}{l}\,\right)}{f_{\text{fg}}\left(c\,|\,\frac{1}{l}\,\right) + f_{\text{bg}}\left(c\right)}. \tag{2.59}$$

This probability is plotted in Figure 2.2 (left) as a function of c and l.

2.2.4 SNR Measurement

To use Equation 2.59 this the signal to noise ratio $l = \frac{a|\hat{p}|}{\sigma}$ has to be estimated for each observed patch $\mathbf{r} = a\mathbf{p} + b\mathbf{1}_d + \mathbf{w}$. As can be seen in Figure 2.2 (left), $p_{\text{fg}|c,l}$ is a very steep function, which means that l has to be measured very precisely. To do that a very accurate noise model has to be used, which is not always available. In this work it is instead assumed that the noise level, σ, is not a known constant but a stochastic variable. It is common in Bayesian literature to assume a Gamma distribution as the prior of the precision of a normal distribution. It is mostly motivated by the fact that it makes the calculations easy. That's the case here too, and a gamma distribution is assumed. If $\tilde{\sigma}$ is the estimated noise level and k some fix parameter specifying how uncertain the measurement is, then the distribution of σ is assumed to be

$$f\left(\frac{1}{\sigma}\middle|\frac{1}{\tilde{\sigma}}\right) = f_\Gamma\left(\frac{1}{\sigma}\middle|k, \frac{1}{k\tilde{\sigma}}\right), \tag{2.60}$$

where $f_\Gamma(\cdot)$ is the gamma distribution function

$$f_\Gamma(x|k,\theta) = x^{k-1}\frac{e^{-x/\theta}}{\theta^k\Gamma(k)}, \tag{2.61}$$

whose expected value is $k\theta = \frac{1}{\tilde{\sigma}}$. The parameter θ is a scale parameter, which means that rescaling $1/\sigma$ with the length of the observed patch $|\hat{\mathbf{r}}|$ (a fixed number) results in

$$f\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|\frac{|\hat{\mathbf{r}}|}{\tilde{\sigma}}\right) = f_\Gamma\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|k, \frac{|\hat{\mathbf{r}}|}{k\tilde{\sigma}}\right). \tag{2.62}$$

This will average out the $p_{\text{fg}|c,\hat{\sigma}}$ function making it much smoother.

Also the length $a|\hat{\mathbf{p}}|$ of the true patch observed, without noise, is needed. Unfortunately the length of the observed noisy patch, $|\hat{\mathbf{r}}|$, is a biased measurement of this, especially for low signal to noise ratios. The distribution of $|\hat{\mathbf{r}}|$ can be derived by looking at

$$\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\right)^2 = \sum_1^d \frac{\hat{r}_k^2}{\sigma^2}. \tag{2.63}$$

Here $\hat{r}_k$ is normal distributed with variance σ^2. This means that $\frac{|\hat{\mathbf{r}}|^2}{\sigma^2}$ is non-central Chi^2 distributed with $\lambda = \frac{\left(a|\hat{P}|\right)^2}{\sigma^2}$ [37],

$$f_{\text{ncChi}^2}(x|\lambda,d) = \frac{1}{2}\left(\frac{x}{\lambda}\right)^{\frac{d-2}{4}} e^{-\frac{\lambda+x}{2}} I_{\frac{d-2}{2}}\left(\sqrt{\lambda x}\right), \tag{2.64}$$

where $I_v(z)$ is a modified Bessel function of the first kind. The signal to noise ratio, $\frac{|\hat{\mathbf{r}}|}{\sigma}$ is the square root of this expression, which makes it non-central chi distributed. That distribution can be derived from f_{ncChi^2} with

$$f_{\text{ncChi}}(x|\lambda,d) = 2x f_{\text{ncChi}^2}\left(x^2\middle|\lambda^2,d\right) = \frac{x^{d/2}}{\lambda^{\frac{d-2}{2}}} e^{-\frac{x^2+\lambda^2}{2}} I_{\frac{d-2}{2}}(\lambda x). \tag{2.65}$$

Note that the λ parameters of f_{ncChi} and f_{ncChi^2} differs by a square to form more natural parameters, e.g. $\lambda = \frac{a|\hat{P}|}{\sigma}$ for f_{ncChi}. This gives

$$f\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|\frac{a|\hat{\mathbf{p}}|}{\sigma}\right) = f_{\text{ncChi}}\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|\frac{a|\hat{\mathbf{p}}|}{\sigma}, d\right), \tag{2.66}$$

which, using Bayes formula can be converted into

$$f\left(\frac{a|\hat{P}|}{\sigma}\middle|\frac{|\hat{\mathbf{r}}|}{\sigma}\right) = \frac{f\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|\frac{a|\hat{P}|}{\sigma}\right) f\left(\frac{a|\hat{P}|}{\sigma}\right)}{\int_0^\infty f\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|\frac{a|\hat{P}|}{\sigma}\right) f\left(\frac{a|\hat{P}|}{\sigma}\right) \mathrm{d}\frac{a|\hat{P}|}{\sigma}} . \tag{2.67}$$

Here $f\left(\frac{a|\hat{P}|}{\sigma}\right)$ is an unknown prior, but by looking at histograms of SNR:s the exponential distribution were deemed a plausible prior,

$$f\left(\frac{a|\hat{P}|}{\sigma}\right) = f_{\text{exp}}\left(\frac{a|\hat{P}|}{\sigma}\middle|\lambda_e\right). \tag{2.68}$$

Here $f_{\text{exp}}(x|\lambda_e) = \lambda_e e^{-\lambda_e x}$ for $x \geq 0$ and $f_{\text{exp}}(x|\lambda_e) = 0$ for $x < 0$. The parameter λ_e were set to 1.

Using the distributions in (2.62) and (2.67), the distribution of the true SNR, $\frac{a|\hat{P}|}{\sigma}$, given the measured SNR, $\frac{|\hat{\mathbf{r}}|}{\tilde{\sigma}}$, can be found by integrating out $\frac{|\hat{\mathbf{r}}|}{\sigma}$, which gives

$$f\left(\frac{a|\hat{P}|}{\sigma}\middle|\frac{|\hat{\mathbf{r}}|}{\tilde{\sigma}}\right) = \int_0^\infty f\left(\frac{a|\hat{P}|}{\sigma}\middle|\frac{|\hat{\mathbf{r}}|}{\sigma}\right) f\left(\frac{|\hat{\mathbf{r}}|}{\sigma}\middle|\frac{|\hat{\mathbf{r}}|}{\tilde{\sigma}}\right) \mathrm{d}\frac{|\hat{\mathbf{r}}|}{\sigma} . \tag{2.69}$$

This distribution can be used in the same way together with the probability $p_{\text{fg}|c,l}$ from Equation 2.59 to find the the probability of foreground given the measured SNR, $\tilde{l} = \frac{|\hat{\mathbf{r}}|}{\tilde{\sigma}}$, by integrating out the true SNR,

$$p_{\text{fg}|c,\tilde{l}} = \int_0^\infty p_{\text{fg}|c,\frac{a|\hat{P}|}{\sigma}} f\left(\frac{a|\hat{P}|}{\sigma}\middle|\frac{|\hat{\mathbf{r}}|}{\tilde{\sigma}}\right) \mathrm{d}\frac{a|\hat{P}|}{\sigma} . \tag{2.70}$$

These integrals can be evaluated numerically by discretising the signal to noise ratio l. Figure 2.2 (right) were generated by restricting l to 512 uniformly spaced values between 0 and 64. The top half, $l > 32$ is discarded to make sure the remaining plot not suffers from any border effects, e.g. depends on function values for $l > 64$.

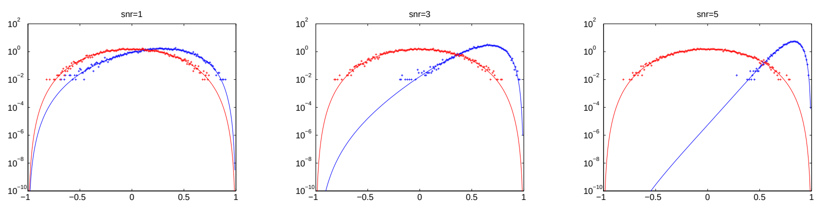


Figure 2.3: Logarithmic plots of simulated distribution functions (crosses) for different signal to noise ratio, l, together with the theoretical functions (solid lines). f_{fg} is red and f_{bg} is blue. $d = 16$

2.3 Experiments

2.3.1 Simulated patches

For $d = 16$ the integral in (2.31) can be evaluated symbolically using maple, which allows it to be plotted for different values of the signal to noise ratio, l. In Figure 2.3 such plots are compared to simulated results generated from a random patch P. Here c is calculated between a fixed patch P and a perturbed version $P + N$ for 10000 random N. The result is binned into 0.01 wide bins.

2.3.2 Real Data - Foreground

The foreground probability distribution is in Figure 2.4 compared to histograms generated by randomly choosing two patches and calculating the cross correlation between them. This is done for two different cases. The green histogram is generated by choosing patches from the image shown to the right in the same figure. The blue is generated by choosing patches from 71 different images each from a different surveillance scene. The figure shows that there is a significantly higher probability for unrelated patches to be correlated than the assumption that they are uncorrelated would suggest. For a surveillance case where most of the background is pavement though, that approximation might not be so severe.

2.3.3 Real Data - Constant lighting

Using a Sony XCD-X710 firewire camera, 4000 frames of a static indoor scene with constant lighting, were captured. The background image was estimated as the mean over all frames, and two patches with different amount of structure were chosen, and the correlation coefficient between the estimated mean and each of the frames were calculated, and a histogram estimating this distribution is plotted in Figure 2.5 (left, middle) together

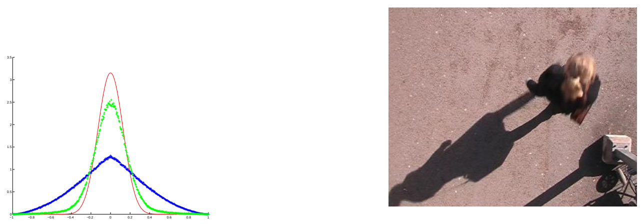


Figure 2.4: The left plot shows f_{fg} (red) and histogram estimations of it from the scene shown to the right (green) and from 71 different surveillance scenes (blue)

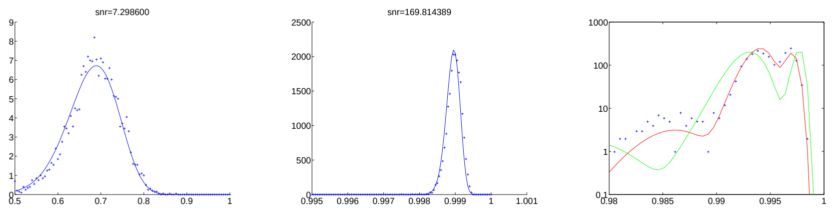


Figure 2.5: *Left, Middle*: Plots of f_{bg} estimated from real video data (crosses), with constant lighting, together with the theoretical function (solid line) for two different signal to noise ratio, $\hat{\mu}$. $d = 64$. *Right*: Plots of f_{bg} estimated from real video data (crosses), with varying lighting, together with the theoretical function (solid line). The green line assumes a constant noise-level over the entire sequence, while the red line assumes the noise-level to be an affine function of the intensity level.

with the theoretical function f_{bg} from (2.28). The two functions seems to agree fairly well.

2.3.4 Real Data - Varying lighting

The experiment from the previous section were repeated, but now the lighting conditions were varied by turning on and of the light in the room as well as pulling the window curtains back and forth. The signal to noise ratio were estimated in each frame. A patch with fair amount of structure was chosen to avoid the biased measurements of low signal to noise ratios. The result is plotted in Figure 2.5 (right) together with the theoretical function f_{bg} averaged over the different signal to noise ratios. The green line assumes a constant noise-level over the entire sequence, while the red line assumes the noise-level to be an affine function of the intensity level. The latter gives a much better fit, which shows that a very precise noise-model is needed to utilise the full potential of this model, and that the smoothing from Section 2.2.4 is really needed if such a precise noise-model is not available.

2.4 Conclusions

The distribution function of the cross-correlation coefficient is derived in two different cases: (i) the cross-correlation coefficient between two random independent patches and (ii) between two patches that differ only by scale, translation and additive Gaussian noise. Those functions are compared with histograms generated from simulations as well as with histograms generated from real data. In both cases the histograms and the distribution functions concur very well.

For real world scenarios where an exact noise model is not available, uncertainty of the signal to noise estimate have been introduced into the model. This gives a much smoother model that is robust to poor noise estimations.

The foreground distribution does not model the foreground very accurately as it assumes unrelated patches to be uncorrelated which empirical tests shows is not the case. It is however possible to estimate it as a histogram as is done in the experimental section and use that instead of the theoretical function.

On the other hand, the background distribution model the background very well.

Chapter 3
Background/Foreground Segmentation

In this chapter two methods for estimating the background in an image sequence taken by a stationary video camera is presented. It is shown that it is possible to extract the background from moving objects in real time and in a very reliable way, also in outdoor scenes where the lighting conditions is changing rapidly due to passing clouds. This is done by introducing a set of intensity independent feature values. Two approximations of the probability distribution functions of the features are suggested. One based on histograms and one based on quantiles. In both cases it is possible to update the estimated parameters very efficiently.

The objective is to extract the foreground and consequently also the background from a sequence of images. Problems facing us include

- keeping execution time short,
- slowly varying lighting conditions,
- rapidly varying lighting conditions, and
- what should be considered background.

The general idea is to extract features from the image sequence corresponding either to single pixels or blocks of pixels and then look at the distribution of those feature values over time. One distribution of the feature value when the corresponding pixels show the background and one when they show the foreground are estimated. The probability that a set of pixels in a given input frame shows the foreground can then be calculated using Bayes formula, which makes it possible to generate a probabilistic foreground/background segmentation image. In Section 3.3 it is described how to generate a binary segmentation from such probabilities. Alternatively, the probabilities can be passed directly to a tracking algorithm that can utilise a probabilistic foreground/background segmentation, such as the one described in Chapter 4.

In contrast to many other background/foreground segmentation algorithms the one proposed here is computationally efficient enough to be used embedded together with some tracking algorithm inside a modern network camera and still perform well with static backgrounds with locally varying illumination. This we believe is needed to build for example large scale automated surveillance system for urban environments where the background typically consist of static pavement with a locally varying illumination due to passing clouds or other shadows.

Many recent solutions aims for continuously varying backgrounds and thus typically requires significantly more processing power. The proposed algorithm is for example about 100 times faster than [58] (243 fps on a 2.4GHz P4).

3.1 Features

3.1.1 General filter based features

In order to compute a feature at each pixel a convolution,

$$R(x, y) = I * H = \iint_{\mathbb{R}^2} I_t(x - a, y - b)H(a, b)\mathrm{d}a\mathrm{d}b, \tag{3.1}$$

can be used where H is a spatial filter mask. This gives a filter response at every point $(x, y) \in \mathbb{R}^2$ and the statistical properties of these can be used to classify background and foreground. The well known Stauffer–Grimson [75] estimator is obtained by letting $H = \delta_{0,0}$ be the Dirac measure at the origin in which case $I * \delta_{0,0} = I$, i.e. base the estimator on the raw pixel data.

It is a well know problem that many background estimators are sensitive to rapid changes in lighting. Such rapid changes are often present and can occur for example when a cloud suddenly occludes the sun, when moving objects cast shadows, or for fast changes in indoor lighting.

In order to deal with this problem it would be preferable to use features that are independent to lighting changes. Changing the lighting will result in a rescaling of the intensity values in the image. The lighting is typically not constant over the entire image, but except for on the borders on sharp shadows it is varying smoothly. For this reasons, assume that there exists some constant c such that when the background is shown,

$$I_{t+1}(x + \Delta x, y + \Delta y) = cI_t(x + \Delta x, y + \Delta y), \quad (\Delta x, \Delta y) \in \Omega, \tag{3.2}$$

where Ω is some neighbourhood of $(0, 0)$. The condition (3.2) is called that the image sequence is *locally proportional*. Local proportionality is usually fulfilled, at least approximately, for most points (x, y) if Ω is sufficiently small. It is however not fulfilled for example on the boundary of a moving shadow, but it is fulfilled on both sides of the boundary. To take advantage of the locally proportional assumption, introduce two filters $\Phi(x, y)$ and $\Psi(x, y)$ such that Φ and Ψ are non-zero only on Ω. Recall that, for a function $f : \mathbb{R}^2 \to \mathbb{R}$, the notation

$$\operatorname{supp} f = \{(x, y) \,|\, f(x, y) \neq 0\}. \tag{3.3}$$

It follows that $\operatorname{supp}\Phi \subseteq \Omega$ and $\operatorname{supp}\Psi \subseteq \Omega$.

By using

$$G_t = \frac{I_t * \Psi}{I_t * \Phi} \tag{3.4}$$

as features, it follows from the locally proportional assumption that

$$G_{t+1} = \frac{I_{t+1} * \Psi}{I_{t+1} * \Phi} = \frac{cI_t * \Psi}{cI_t * \Phi} = \frac{I_t * \Psi}{I_t * \Phi} = G_t. \tag{3.5}$$

This means that for points (x, y) that fulfil the local proportionality the features $G_t(x, y)$ are independent of changes in lighting.

3.1.2 Haar based features

The convolution for computing (3.4) can be done using FFT with a computational cost of $\mathcal{O}(wh \log(wh))$ for $w \times h$ images. However, if the filters Φ and Ψ, are simple functions like for example Haar wavelets then the well known *integral image* [83] can be used to speed up the computation. Let

$$J(x, y) = \int_{-\infty}^{x} \int_{-\infty}^{y} I(a, b)\mathrm{d}a\mathrm{d}b \tag{3.6}$$

be the integral image of a grey scale image I. Then J can be computed with a computational cost of about $4wh$ additions for an $w \times h$ image I. For notational convenience, just the filter Φ is treated below and assume that

$$\Phi(x, y) = \begin{cases} 1, & |x| \leq c, |y| \leq c \\ 0, & \text{otherwise} \end{cases}. \tag{3.7}$$

In order to fulfil $\operatorname{supp}\Phi \subseteq \Omega$, the constant $c > 0$ should be chosen sufficiently small. It follows that

$$(I * \Phi)(x, y) = J(x-c, y-c) + J(x+c, y+c) - J(x-c, y+c) - J(x+c, y-c) \tag{3.8}$$

requiring only four additions for each pixel (x, y). A general Haar wavelet in $\mathbb{R}^2$ is a linear combination of at most 4 functions like (3.7) resulting in maximum of 16 additions for each pixel (x, y).

The background and foreground probability distribution functions of those feature responses can be arbitrarily complex. In Section 3.2.1 it is shown how to estimate such distributions functions in a non parametric way using histograms.

3.1.3 Cross correlation

The images delivered by an off-the-shelf camera is typically preprocessed by the camera in some way. This means that a lighting change might no longer be a simple rescaling of the image. When using such data empirical studies show that in addition to independence of intensity scaling, independence of intensity translation will improve results. The classical cross correlation coefficient (2.1) is independent to both scaling and translation in the intensity domain.

By dividing the input image I into n small (typically 8×8) patches, $\mathbf{r}_j$, $j = 1, 2, \cdots, n$, it is possible to use the cross correlation between those patches and some background patches can be used as features. In that case the theory from Chapter 2 gives both the foreground and background distributions functions. They depend only on the signal to noise ratio, which has to be measured.

By letting the background patch $\mathbf{p}_j$ be the temporal mean of $\frac{\hat{\mathbf{r}}_j}{|\hat{\mathbf{r}}_j|}$ as defined in Cahpter 2 it will be possible to estimate the background patch even if the lighting conditions vary. The patch $\mathbf{p}_j$ is in Chapter 2 assumed to be known exactly. This is never the case when something is estimated from real data, but the variance of the estimate will decrease with increased learning factor. This means that the by choosing the learning factor large enough, the approximations induced by assuming $\mathbf{r}_j$ to be known exactly can be made arbitrarily good. Also, the noise-level, $\tilde{\sigma}$, has to be estimated. That is done by estimating the variance of $\hat{\mathbf{r}}_j - |\hat{\mathbf{r}}_j|\,\mathbf{p}_j$. After those estimates are done, Equation 2.70 gives the probability of foreground for a given input patch, $p_{\text{fg}|c,\tilde{l}}$.

The presented approach is very well suited for processing *motion-JPEG* compressed video. That is a video compression standard that stores each frame of a video as a JPEG image. The JPEG compression algorithm divides the image into 8x8 blocks and performs a discrete cosine transform (DCT) och each block. All calculation presented above can be performed in this DCT domain, which means that the algorithm can operate on motion-JPEG compressed videos without uncompressing them fully. Processing a compressed image will be more efficient on low end systems due to better utilisation of the cache memories as the JPEG-image is already organised in 8x8 blocks. In the DCT-domain the first coefficient is the mean value, which means that the operation of removing the mean simply means skipping the first coefficient. After that, Equation 2.1 for calculating the correlation coefficient from zero mean vectors is the same in the DCT-domain as in the intensity-domain. This kind of implementation becomes very fast. A 320×240 videos is processed at 243 fps on a 2.40GHz P4, 640×480 at 70 fps and 1280×1024 at 17 fps.

3.1.4 General intensity independent features

In this section a general result for intensity independent features will be shown. Let $\mathbb{R}_+$ be the set of positive real numbers. Then an intensity independent feature $\mathbf{f} : \mathbb{R}_+^n \rightarrow \mathbb{R}^m$ is characterised by the property $\mathbf{f}(\mathbf{x}) = \mathbf{f}(c\mathbf{x})$ for any real $c > 0$ and $\mathbf{x} \in \mathbb{R}_+^n$. It is easily seen that

$$f(\mathbf{x}) = \sum_{j=1}^{n} m_j \log(x_j) \tag{3.9}$$

is intensity independent if $\sum_{j=1}^{n} m_j = 0$. The following theorem shows that the reverse is also true, i.e. all intensity independent features can be written as sums of the form (3.9). For convenience, introduce the notation $\log(\mathbf{x})$ to denote the vector $(\log(x_1), \ldots, \log(x_n))$ and similarly for $\exp(\mathbf{x})$.

Theorem 3.1.1. *A feature* $\mathbf{f} : \mathbb{R}_+^n \rightarrow \mathbb{R}^m$*, where* $n > 1$*, is intensity independent if and only if it can be written as*

$$\mathbf{f}(x) = \mathbf{g}(\mathbf{M}\log(\mathbf{x})), \qquad \mathbf{x} \in \mathbb{R}_+^n, \tag{3.10}$$

for some $\mathbf{g} : \mathbb{R}^{n-1} \rightarrow \mathbb{R}^m$ *and* $\mathbf{M}$ *is an* $(n-1) \times n$ *matrix with row sums equal to* 0.

Proof. Let $\mathbf{x} \in \mathbb{R}_+^n$. Then, $\mathbf{f}(\mathbf{x}) = \mathbf{f} \circ \exp \circ \log(\mathbf{x})$. Let $\mathbf{P}$ be a $n \times n$ non singular matrix such that the first row of $\mathbf{P}$ contains only ones and the other rows are orthogonal to this. The the row sums are $= 0$ except for the first row. It follows that $f(\mathbf{x}) = f(\exp(\mathbf{P}^{-1}\mathbf{P}\log(x)))$. Set $\mathbf{h}(\mathbf{y}) = \mathbf{f}(\exp(\mathbf{P}^{-1}\mathbf{y}))$. Set

$$\mathbf{P} = \begin{pmatrix} \mathbf{1}_n{}^{\mathrm{T}} \\ \mathbf{M} \end{pmatrix}, \tag{3.11}$$

where $\mathbf{1}_n{}^{\mathrm{T}}$ is a $1 \times n$ matrix with only ones and $\mathbf{M}$ an $(n-1) \times n$ matrix. It then follows that $\mathbf{1}_n{}^{\mathrm{T}} \log(c\mathbf{x}) = n\log(c)\mathbf{1}_n{}^{\mathrm{T}} \log(\mathbf{x})$ and $\mathbf{M}\log(c\mathbf{x}) = \mathbf{M}\log(\mathbf{x})$. The intensity independence gives that $\mathbf{h}(\mathbf{P}\log(c\mathbf{x})) = \mathbf{h}(\mathbf{P}\log(\mathbf{x}))$ for all $\mathbf{x} \in \mathbb{R}_+^n$ and $c > 0$, implying that $\mathbf{h}(a_0, \mathbf{b}) = \mathbf{h}(a, \mathbf{b})$, where $a_0, a \in \mathbb{R}$ and $\mathbf{b} = \mathbf{M}\log(\mathbf{x}) \in \mathbb{R}^{n-1}$. The theorem follows by setting $\mathbf{g}(\mathbf{b}) = \mathbf{h}(a_0, \mathbf{b})$. $\square$

3.2 Background Models

To deal with noise and points (x, y) in background that do not fulfil the locally proportional condition the idea of continuously updating probability distribution functions will be used. The idea is to estimate the background and the foreground distribution of the feature values and then use Bayes rule to find the probability of the current frame showing the foreground.

In the general case this is to ambitious a task though, and one will have to suffice with estimating the distribution of the feature values observed. This is a mixture between the foreground and the background distribution, but if it is assumed that the background is shown most of the time this mixture can be considered an approximation of the background distribution. The foreground distribution can be assumed to be for example uniform.

With this method it is well known that one can deal with slowly varying changes and also in some cases (semi)-periodic fast changes such as for example the branches of a tree swaying in the wind.

In the case of the correlation coefficient the background model will be constructed from the features generated by removing the mean and normalising the the length of each block. The cross correlation can then be calculated between the new input frame and the mean of this distribution.

3.2.1 Recursive Histogram Estimation

Instead of using a parametrised probability distribution functions whose parameters are updated, the distributions are here represented with histograms. Comparing with the parametrised probability functions such as multi modal Gaussian distributions the histogram makes no assumptions on the function. It is furthermore very straightforward to use being both easy to update and very fast in execution. Let (x_k, y_k), $k = 1, \ldots, m$, be a set of fixed points in the images. The probability distribution of the values

$$G_t(x_k, y_k), \qquad k = 1, \ldots, m, \tag{3.12}$$

will be estimated and the estimate is updated dynamically keeping it accurate for all times $t \geq 0$. Let $f_{k,t}(\cdot)$ be this probability distribution function, which is dependent on which pixel k and what time t.

If the probability distribution function $f_{k,t}(\cdot)$ is assumed to vary slowly with t it can be estimated from the histogram, $q_{t,k}(\cdot)$, obtained by computing $G_t(x_k, y_k)$ for some values of t while keeping k constant. In order to compute the histogram, first choose some bins $a_1, \ldots a_n$ and round off the value of $G_t(x_k, y_k)$ to the nearest bin. The nearest bin to $G_t(x_k, y_k)$ is denoted by $\overline{G}_t(x_k, y_k)$. By using a normalised histogram,

$$p_{t,k}(a_j) = \frac{q_{t,k}(a_j)}{\sum_{j=1}^{n} q_{t,k}(a_j)}, \tag{3.13}$$

the approximated probability distribution function can be written

$$f_{t,k}(x) = \begin{cases} p_{t,k}(a_j) \frac{2}{a_{j+1} - a_{j-1}} & \text{if } \frac{a_{j-1}+a_j}{2} \leq x < \frac{a_j + a_{j+1}}{2}, \\ 0 & \text{otherwise,} \end{cases} \tag{3.14}$$

where a_j are the bins. Rescaling $q_{t,k}(\cdot)$ will not change the values of this probability distribution function. This means that it is sufficient to estimate $q_{t,k}(\cdot)$ up to scale. That is, if c_t are some unknown constants, it will be enough to estimate

$$\hat{q}_{t,k}(\cdot) = c_t q_{t,k}(\cdot). \tag{3.15}$$

Equation 3.13 will still hold if $q_{t,k}(\cdot)$ is replaced by $\hat{q}_{t,k}(\cdot)$ since c_t will factor out and cancel. In order to update the probability function with a new measurement $\overline{G}_t(x_k, y_k)$, introduce

$$\gamma_{t,k}(x) = \begin{cases} 1 & \text{if } \overline{G}_t(x_k, y_k) = x, \\ 0 & \text{otherwise,} \end{cases} \tag{3.16}$$

and update the normalised histogram using a learning factor α,

$$p_{t,k}(x) = (1-\alpha)p_{t-1,k}(x) + \alpha\gamma_{t,k}(x). \tag{3.17}$$

Replacing $p_{t-1,k}(x)$ with $\frac{\hat{q}_{t-1,k}(a_j)}{\sum_{j=1}^{n}\hat{q}_{t-1,k}(a_j)}$ and rearranging the equation gives

$$\frac{\sum_{j=1}^{n}\hat{q}_{t-1,k}(a_j)}{1-\alpha}p_{t,k}(x) = \hat{q}_{t-1,k}(a_j) + \frac{\alpha\sum_{j=1}^{n}\hat{q}_{t-1,k}(a_j)}{1-\alpha}\gamma_{t,k}(x). \tag{3.18}$$

Collecting all constant terms on the left hand side and combining them into the c_t constant will make it possible to use the update equation

$$\hat{q}_{t,k}(x) = \hat{q}_{t-1,k}(a_j) + \frac{\alpha\sum_{j=1}^{n}\hat{q}_{t-1,k}(a_j)}{1-\alpha}\gamma_{t,k}(x). \tag{3.19}$$

The factor,

$$\frac{\alpha\sum_j \hat{q}_{t-1,k}(a_j)}{(1-\alpha)}, \tag{3.20}$$

becomes independent of the pixel k if α is kept constant and if the starting value is

$$q_{0,k}(a_j) = \begin{cases} 1 & \text{if } \overline{G}_0(x_k, y_k) = a_j, \\ 0 & \text{otherwise,} \end{cases} \tag{3.21}$$

for all k. It follows that the update is done very fast, requiring only one addition per histogram. Note also that $\sum_j \hat{q}_{t,k}(a_j)$ only depends on t and not k. Thus, computing the probability $p_{t,k}(a_j)$ requires only division by a number that is the same for all positions (x_k, y_k).

Discredited feature values

A problem is that the features $G_t(x_k, y_k)$ may take arbitrarily large values and this will cause problem when choosing the bins for a histogram. To simplify that process, assume that it is required that the feature values are integers in the interval $[0, n-1]$. This requirement can be fullfilled by finding an affine transformation $h : \mathbb{R} \to \mathbb{R}$ such that $h(G_t(x_k, y_k)) \in [0, n-1]$ as often as possible. Let $m_{t,k}$ and $\sigma_{t,k}$ be the mean and standard deviation of the values of $G_t(x_k, y_k)$, respectively, and update these values according to

$$m_{t,k} = (1-\alpha)m_{t-1,k} + \alpha g_t(x_k, y_k) \tag{3.22}$$

and

$$\sigma_t = (1-\alpha)\sigma_{t-1,k} + (1-\alpha)|G_t(x_k, y_k) - m_{t,k}|, \tag{3.23}$$

where α is the learning factor. The feature $G_t(x_k, y_k)$ can the be transformed by the affine transformation

$$\frac{G_t(x_k, y_k) - m_{t,k}}{\sigma_{t,k}} \frac{n}{4} + \frac{n}{2} \tag{3.24}$$

which has mean equal to $n/2$ and standard deviation $n/4$. To be sure that the values never get out side the interval $[0, n-1]$, introduce the transformed features

$$H_t(x_k, y_k) = \left[\min\left(\max\left(\frac{G_t(x_k, y_k) - m_{t,k}}{\sigma_{t,k}} \frac{n}{4} + \frac{n}{2}, 0\right), n-1\right)\right], \tag{3.25}$$

where $[x]$ denotes rounding to nearest integer $\leq x$.

This gives a very general method to estimate the background distribution, but it requires quite a lot of memory. This can be a problem in cases when the memory bandwidth is limited, which is typically the case for the smaller platforms embedded in network cameras. This can result in a low frame rate even if there is very few mathematical operations performed.

3.2.2 Recursive Quantile Estimation

Another approach that requires less memory is to use a learning factor as is described in Chapter 1. One problem with that thou is that the mean will not only be taken over background values, but over foreground as well. The version in Section 1.2.3 that only updated the mean if the current pixel value is within 2.5 standard deviations of the mean mitigates this problem, but for that to work there has to be a decent estimate to begin with, and there is no guarantee that this solution will converge, as is shown by a counterexample in the simulations below.

Another solution is to use the median instead of the mean and to estimate the variance from the 25/75% quantile. Möller *et al* shows [60] how to estimates quantiles recursively, using a control sequence $c_t = max(c_0/t, c_{\text{min}})$. The median $B_{0.50,t}(\mathbf{x})$, 25% quantile $B_{0.25,t}(\mathbf{x})$ and 75% quantile $B_{0.75,t}(\mathbf{x})$ of each pixel $I_t(\mathbf{x})$ in a image sequence is found with

$$B_{\gamma,t} = \begin{cases} B_{\gamma,t-1} + \gamma c_t & \text{if } B_{\gamma,t-1} < I_t \\ B_{\gamma,t-1} - (1-\gamma)\, c_t & \text{if } B_{\gamma,t-1} > I_t \\ B_{\gamma,t-1} & \text{if } B_{\gamma,t-1} = I_t \end{cases} . \tag{3.26}$$

From these quantiles the variance, V_t, can be estimated using

$$\sqrt{V_t} = \frac{B_{0.75,t} - B_{0.25,t}}{\mathcal{N}_{\text{cdf}}^{-1}(0.75) - \mathcal{N}_{\text{cdf}}^{-1}(0.25)}, \tag{3.27}$$

where $\mathcal{N}_{\text{cdf}}^{-1}(x)$ is the inverse of the normal cumulative distribution function

$$\mathcal{N}_{\text{cdf}}(x) = \int_{-\infty}^{x} \mathcal{N}(t\,|0,1)\,\mathrm{d}t, \tag{3.28}$$

$$\mathcal{N}_{\text{cdf}}^{-1}(0.75) - \mathcal{N}_{\text{cdf}}^{-1}(0.25) \approx 1.349. \tag{3.29}$$

Simulations

Figure 3.1 (left column) shows several plots of the simulated intensity of a single pixel in grey. In the top left plot the pixel always shows the background which is measured with additive Gaussian noise. The blue thick line shows the estimated background model using a learning factor to estimate the mean and variance and the two dashed blue lines shows an offset of two standard deviation from this mean. The red lines shows the corresponding values but based on the 25%, 50% and 75% quantile estimations instead. Both estimates agree equally well with the ground truth after they have converged when there is no foreground. The learning factor and the step size have been chosen to make the convergence time of the two estimates approximately equal.

In the second plot of the left column the pixel is assumed to show the foreground 1% of the time. The foreground is modelled as uniformly distributed between 0 and 255. The quantile based estimator still gives the same result while the mean based overestimates the variance. In the bottom two plots, the amount of foreground is increased even further and now the learning factor based estimator over estimates the variance even further and also overestimates the mean value when it is lower than 127 and underestimate it when it is larger than 127. The quantile based estimator still gives reasonable results. The mean of the last estimates archived right before the background intensity was changed and at the end of the sequence are shown in Table 3.1. Also, normalised histograms of the data together with plots of the probability distribution functions estimated are shown in Figure 3.2.

When using the mixtures of Gaussians based algorithms by Stauffer and Grimson presented in Section 1.2.3 the difference between using a learning factor as they do and the quantile estimates are not as striking. Figure 3.1 (right column) shows how the algorithm from Section 1.2.3 performs on the same input data. It contains a plot of the mean and variance of the component with largest weight W_t. That would correspond to a unimodal background. The algorithm performs very well in most cases. The variance is somewhat underestimated, but it should be possible to compensate for that by figuring out how much the estimate is biased. What's troubling though is that when there is a lot of foreground present this algorithm might lock on to something completely wrong and then stick to that as has happened in the lower right plot. The result is also tabulated in Table 3.1.

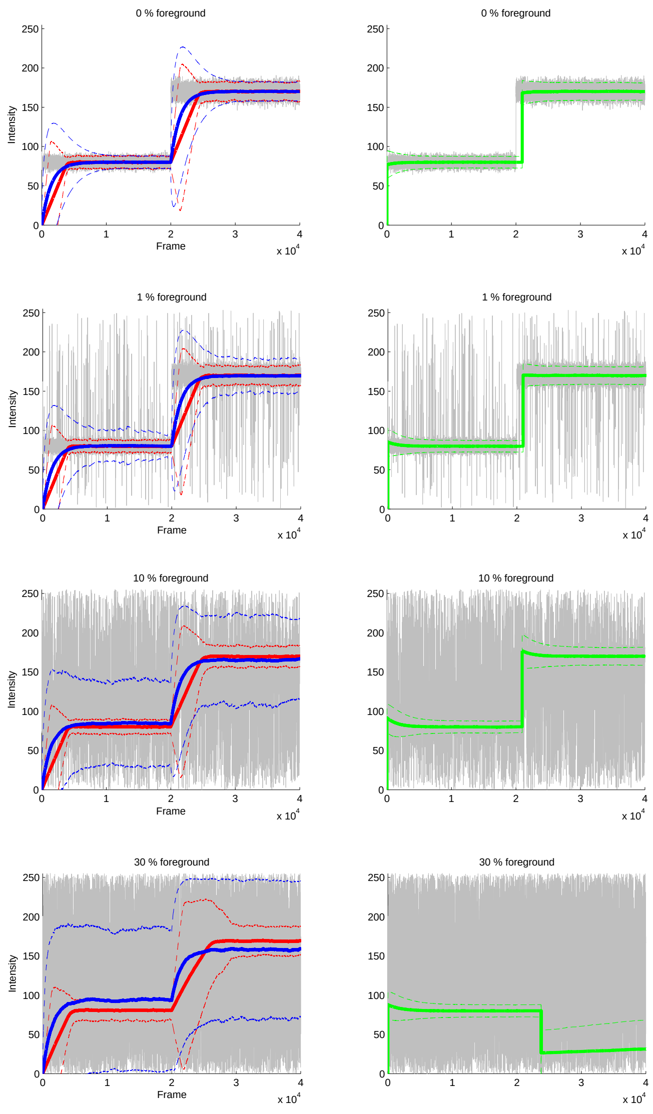


Figure 3.1: Simulated intensity of a single gaussian distributed background pixel (grey) mixed with different amount of uniformly distributed foreground. *Left column:* It's mean (thick blue) and standard deviation (dashed blue) estimates using a learning factor $\alpha = 0.9993$. It's mean (thick red) and standard deviation (dashed red) estimates using a recursive quantile estimator with $c_t = 0.02$. *Right column:* Results from using the Stauffer and Grimson algorithm on the same data (green).

Amount of	Mean Value				Standard Deviation			
foreground	GT	LF	RQ	SG	GT	LF	RQ	SG
0%	80	80.01	79.99	80.02	4	4.03	3.99	3.71
1%	80	80.34	79.88	79.91	4	9.48	3.98	3.69
10%	80	84.75	80.24	80.00	4	27.30	4.55	3.81
30%	80	93.86	80.69	79.97	4	45.04	6.54	3.93
0%	170	170.08	170.10	170.10	6	6.08	6.00	5.60
1%	170	169.47	169.85	169.89	6	10.75	6.02	5.61
10%	170	165.14	169.56	169.98	6	28.00	6.74	5.64
30%	170	157.91	168.89	29.85	6	44.00	9.43	17.41

Table 3.1: Results of estimating the mean and variance of a Gaussian distributed background distribution mixed with difference amounts of uniformly distributed foreground. Three different methods: learning factor (LF), recursive quantile (RQ) and Stauffer/Grimson (SG) are compared with the ground truth (GT).

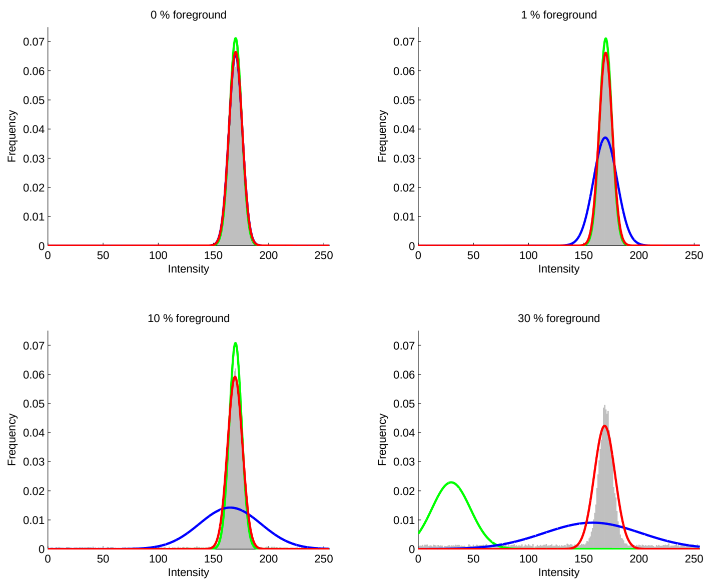


Figure 3.2: Normalised histogram over the the second half of the simulated pixel values from Figure 3.1 (grey) and the background estimations made by the three different algorithms: learning factor (blue), recursive quantile (red) and Stauffer/Grimson (green).

3.3 Post processing

This far the analysis have been performed on each pixel independently. To gain robustness, the relationship between the pixels can be used. If some a priori information about the size of the objects of interest is available this can be used to produce a likelihood of an object being present by taking the product over an area of this size. This is done in Chapter 4 where such products are formed over both space and time and they are then optimised over different sequences of configurations of objects.

Another approach that does not require as much a priori information is to use a Markov random field model to utilise the fact that neighbouring pixels often both are foreground or both background and that the transition from background to foreground often happens where there is an edge in the image. A binary background foreground segmentation can then be found using for example dynamic graph cuts [44], which gives the maximum likelihood segmentation and are guaranteed to give the global optimum.

Both these techniques takes the product over pixels, which assumes these likelihoods to be independent. If likelihoods are calculated for each pixel by centring a block at each pixel those blocks will overlap quite significantly and thus be very dependent. To avoid that only non overlapping blocks can be used which gives a background foreground segmentation with lower resolution than the original image.

3.4 Colour Images

For colour images the same machinery as above can be applied by for example treating each colour channel separately. This requires that the colour channels are independent which is normally not the case for RGB. However, it does hold approximately for other types of colour codings, such as YCbCr. It is also possible to use several features, defined for example by a set of filters Φ_j and Ψ_j, $j = 0, 1, \dots$. This improves the accuracy at the cost of more memory requirements and computational load. Here again independence has to be assumed if the model should be manageable.

3.5 Experiments

A system was implemented in C running on image sequences having 352×288 in resolution. For each pixel (x_k, y_k) we

1. compute the Haar features $g_t(x_k, y_k)$
2. update $m_{t,k}$ and $\sigma_{t,k}$
3. compute the affinely transformed features $h_t(x_k, y_k)$
4. update the corresponding histograms.

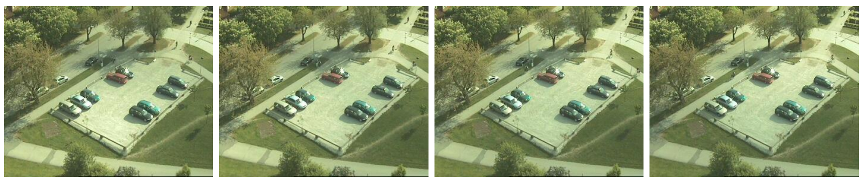


Figure 3.3: Four frames from the cloudy input sequence used to test the proposed algorithm.

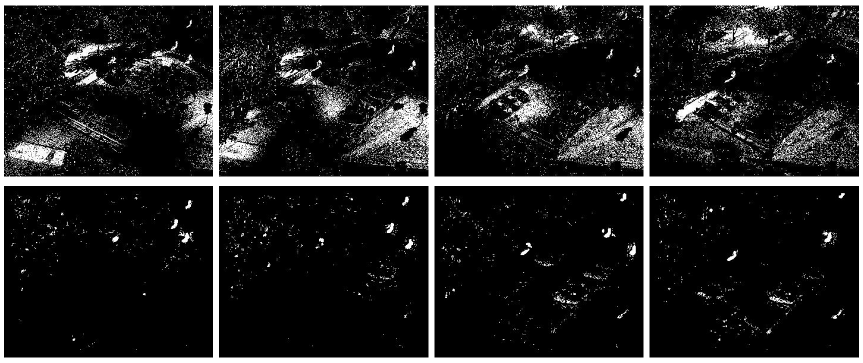


Figure 3.4: The first row shows the result from the method in [75] and to the second row the result from the proposed method when applied to the frames in Figure 3.3.

This runs at about 20 frames per second on a 2.4 GHz P4 processor. A binary image, estimating the foreground, was obtained by setting a threshold for the probabilities. This threshold was the same for each pixel (x_k, y_k).

3.5.1 Parking lot

By using a 16 minutes (20 fps) video sequence monitoring a parking lot, the proposed algorithm was compared to the one suggested in [75]. The sequence was recorded on a cloudy and windy day. Four frames, just as a cloud shadow passes over the ground, are shown in Figure 3.3.

The corresponding binary output images from the two algorithms are shown in Figure 3.4. Figure 3.4 clearly shows that the cloud shadows, as displayed by the method [75], is almost entirely gone with the proposed method. Furthermore, the bicycles are still shown as foreground.

The input sequence is 352x288 pixels and the proposed algorithm processes 21 frames per second on a 2.4 GHz P4. This is significantly faster than our implementation of the algorithm in [75] that processes four frames per second on the same computer and input

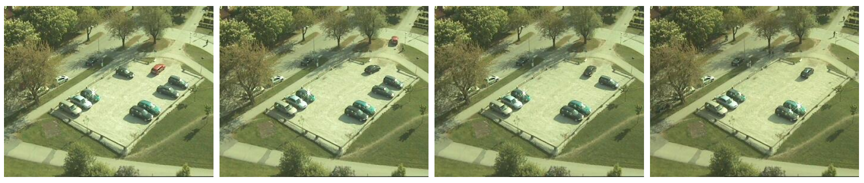


Figure 3.5: All events detected from the parking lot monitoring algorithm based on the proposed algorithm.

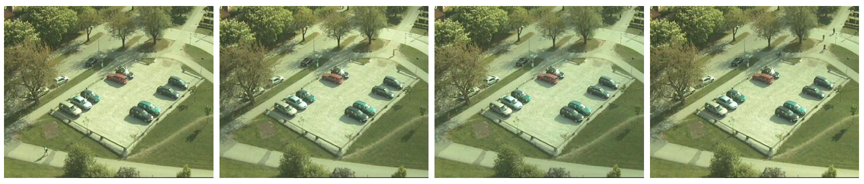


Figure 3.6: First four events detected from the parking lot monitoring algorithm based on [75].

sequence.

In order to obtain quantitative comparison between the two algorithms a simple application was implemented on top of them. It is a first stage in a parking lot monitoring system that counts the number of parked cars in real time. The application detects the event that a car is entering or exiting the parking. This is done by counting the number of foreground pixels, N, in a predefined rectangle covering the entrance of the parking lot. If $N > \sigma_1$, where σ_1 is some threshold, an event is detected and the image is saved. Then when $N \leq \sigma_2$, for some $\sigma_2 < \sigma_1$ a new event is triggered. The saved images are then inspected and compared with the ground truth.

The cloudy sequence mentioned above contains four events, all consisting of a car exiting the parking lot. And the proposed algorithm found all four of them and no false positives. The four images saved are shown in Figure 3.5. Executing the same event detection, based on [75], on the same input sequence resulted in 18 events detected. The four cars exiting are still detected and the addition 14 false positives are cloud shadows moving past the parking lot entrance. The first four detections are shown in Figure 3.6.

As a final test the proposed event detector were tested on a 16 hour image sequence acquired from 16:30 in the afternoon until 08:30 the next morning. The sequence contains 32 events, both cars exiting and entering, and quite a lot of lighting variations as it contains both a sunset and a sunrise. The proposed system detected 34 events, including the 32 correct ones. The first additional false detection consisted of five pedestrians simultaneously entering the parking lot, and the second of a car driving past the entrance from one place in the parking lot to another. In both cases the proposed back-

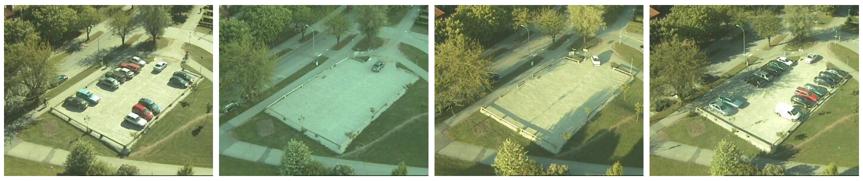


Figure 3.7: Four detected events chosen from the events detected by the parking lot monitoring algorithm based on the proposed algorithm when tested on a 16 hour sequence.

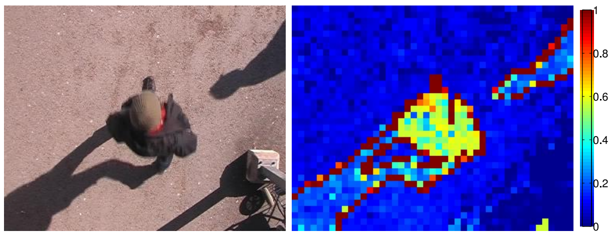


Figure 3.8: Input frame, and foreground probability, $P_{\mathrm{fg}|c,\tilde{l}}$. See also Figure 2.1 and `fgbg_ncc.avi` at http://www.maths.lth.se/˜ardo/thesis/.

ground/foreground segmentation algorithm generated the expected data, and the misstates were made by the simplistic event detection algorithm. Four detected events chosen from the entire sequence are shown in Figure 3.7.

3.5.2 Cross Correlation

Another implementation using the cross correlation features were made to test the probabilistic segmentation approach. Figure 3.8 shows some results. Most of the image area occupied by the pedestrian is detected as foreground. A large part of the jacket is very uncertain though, as it is uniformly coloured and partly underexposed. The interior of the shadow is detected as background with slightly less probability than the rest of the ground as the SNR is lower. The border of the shadow is detected as foreground because here the patches overlap the border and thus the assumption about the light being constant within the patch no longer holds.

The implementation has also been tested on the training video sequences of Axis's Open Evaluation of Motion Detection Algorithms. It consists of 10 different quite challenging sequences from different scenes acquired with different types of cameras and resolutions, with varying weather condition and illumination both indoor and outdoor. The exact same parameters were used in all cases. Results are shown in Figure 2.1

Sequence	False Positive (%)	False Negative (%)	Total (%)
TimeOfDay	2.47	0.95	3.42
ForegroundAperture	7.26	0.03	7.28
Bootstrap	0.88	7.39	8.27
Camouflage	5.60	2.11	7.71
LightSwitch	46.48	0.00	46.48
MovedObject	0.00	0.00	0.00
WavingTrees	3.13	9.07	12.20

Table 3.2: Results from applying the proposed algorithm to the dataset from [81]. For each of the 7 sequences the percentages of misclassified pixels are presented.

and Figure 3.9. Figure 3.9 also shows the results from a binary Markov random field (MRF) segmentation of the probability image, see Section 1.2.5. There is also a video, `fgbg_ncc.avi`[1], that shows a few seconds form some of those videos and the results. The results are mostly correct. In the second sequence of Figure 2.1 there is one shadow detected as foreground because the wall it falls on is lit by some complex far from constant lighting. Also part of the shadows in sequence seven of Figure 3.9 shows up as foreground, partly due to an overexposed specular reflection in the floor. Finally, there is some overexposed ridge that shows up as foreground in the last sequence.

The proposed algorithm were also tested on the dataset from [81] available on-line[2], which is also used by [61]. This dataset consists of 7 sequences with resolution 160x120. For each sequence one frame has been manually segmented into foreground and background. The result from the proposed algorithm followed by a binary MRF segmentation was compared to those ground truth frames and results are presented in Table 3.2 and Figure 3.10. The LightSwitch sequence fails because the background model is in this cased trained on dark underexposed frames that does not contain the same structures as the light frames. If the LightSwitch sequence is excluded this gives on average 6.48% misclassified pixels, which is better than the results presented in [81, 61], 7.82% and 7.33% misclassified pixels respectively. The proposed algorithm is also significantly faster. Those 160x120 sequences are processed at 690 fps on a 2.4GHz P4.

[1]http://www.maths.lth.se/˜ardo/thesis/

[2]http://research.microsoft.com/users/jckrumm/WallFlower/TestImages.htm

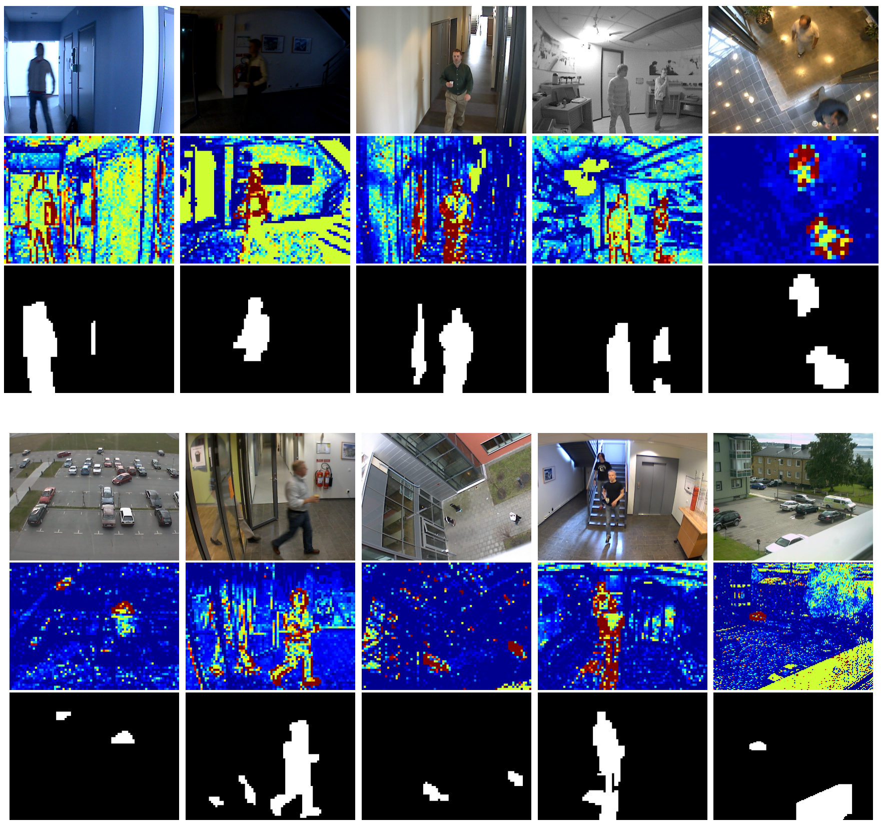


Figure 3.9: Some results of applying the proposed foreground/background segmentation to the training videos of Axis's Open Evaluation of Motion Detection Algorithms. For each of the 10 sequences the figure shows a single frame, $P_{\text{fg}|c,\tilde{l}}$ for that frame, and the result of segmenting the frame using a MRF. The colour coding of $P_{\text{fg}|c,\tilde{l}}$ is the same as given in Figure 2.2 (right). Videos from a few of these sequence are shown in `fgbg_ncc.avi` at http://www.maths.lth.se/˜ardo/thesis/.

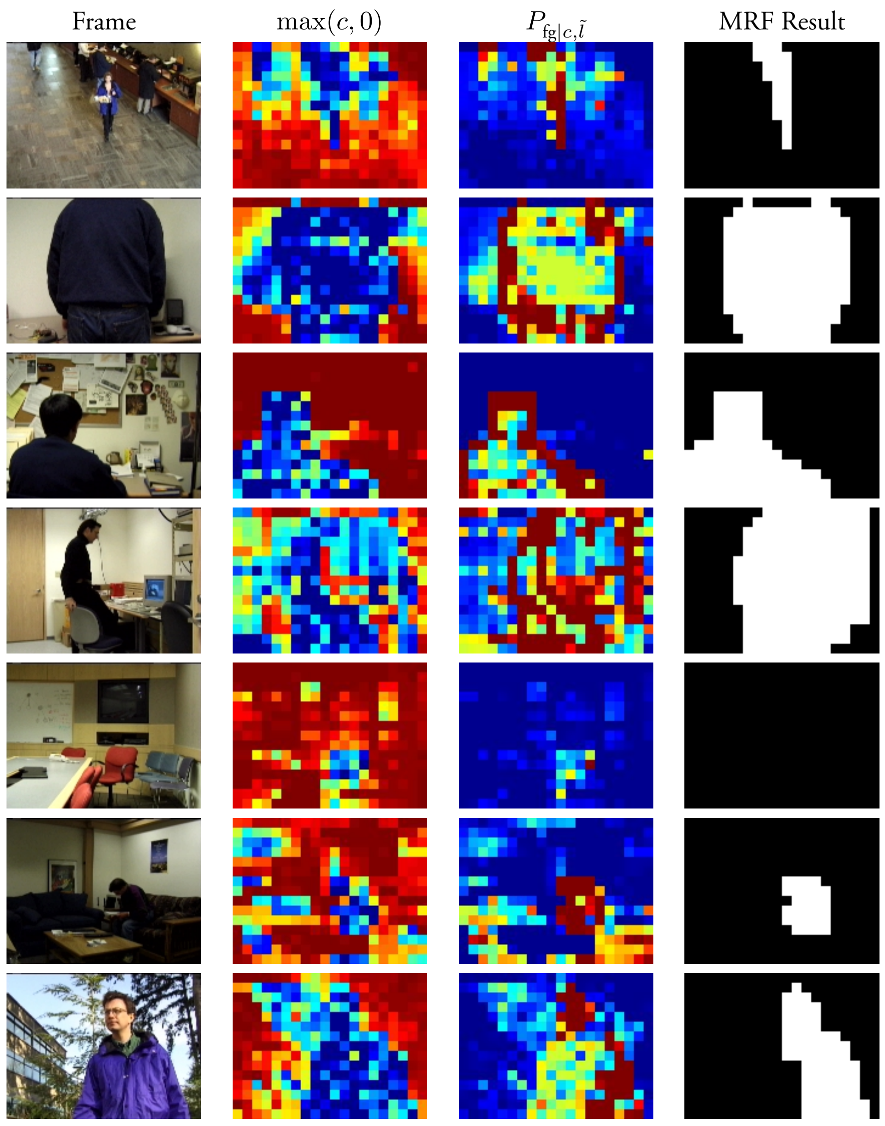


Figure 3.10: Some results of applying the proposed foreground/background segmentation to the test images for wallflower paper [81]. For each of the sequences the figure shows a single frame, the normalised cross correlation with this frame and a estimated background c, $P_{\text{fg}|c,\tilde{l}}$ for that frame, and the result of segmenting the frame using a MRF. The colour coding of $P_{\text{fg}|c,\tilde{l}}$ and c is the same as given in Figure 2.2. Note that this colour coding only covers the range $0..1$, so c is truncated below 0.

Chapter 4

Hidden Markov Models

In this Chapter a real time system is presented that tracks a varying number of moving objects. The entire state space is modelled by a Hidden Markov Model (HMM) [64], where each state represents a configuration of objects in the scene and the state transactions represent object movements as well as the events of objects entering or leaving the scene. The solution is found by optimising the observation likelihood over different state sequences. Results are generated with several frames delay in order to incorporate information from both past and future frames in the optimisation. This delay varies depending on the input data.

The optimisation is performed over entire sequences and not only single frames. The first frame is included in the sequence, which means that no initialisation is needed. Also, the joint state of all objects is modelled by a single HMM, which means that no data association is needed. An offline version of the optimisation algorithm is also presented that is guaranteed to find the global optimum. It can be applied to a set of unlabelled recordings form a specific installation containing the typical behaviour observed in that scene. The result of that analysis can be used to automatically tune a parameter of the online algorithm to make it produced the global optimum with any given probability, strictly less than one (for this installation). Alternatively it can be used to calculate the probability of finding the global optimum given some CPU or memory usage limitations.

The optimisation algorithm is constructed from two novel modification to the Viterbi dynamic programming algorithm [64]. The first allows it to be used on infinite time sequences and still produce the global optimum. The problem with the original Viterbi algorithm is that it assumes that all observations are available before any results can be produced. The modification presented here allows results to be computed before all observations are received, and still it generates the same globally optimal state sequence as is done when all observations are available. However, there is a delay of several frames between obtaining an observation and the production of the optimum state for that frame. The second modification makes it possible to use dynamic programming for very large state spaces by only calculating the likelihoods of a subsets of all the states and an upper bound on the rest. If enough likelihoods were calculated it is possible to still find the global optimum. Whether this is the case or not can be automatically decided by the algorithm.

A very simple object model is used. It states that all objects are boxes of some given size and the observations made are the result of a background foreground segmentation. The boxes are projected into the camera images and the probability of all pixels within the projected boxes being foreground and all pixel outside the boxes being background is calculated. By combining data from several cameras and using the efficiency of HMM to optimise over different state sequences, it is shown that this simple model can actually achieve reliable results.

4.1 Hidden Markov models

A good introduction to the hidden Markov model can be found in [64]. It is defined as a discrete time stochastic process with a set of n states, $\mathbb{S} = \{S_0, \ldots, S_n\}$ and a constant transitional probability distribution $a_{i,j} = p(q_{t+1} = S_j | q_t = S_i)$, where $\mathcal{Q}_{0\cdots\tau} = (q_0, \ldots, q_\tau)$ is a state sequence for the time $t = 0, 1, \ldots, \tau$. The initial state distribution is denoted $\pi = (\pi_0, \ldots, \pi_n)$, where $\pi_i = p(q_0 = S_i)$. The state of the process cannot be directly observed, instead some sequence of observations, $\mathcal{O}_{0\cdots\tau} = (\mathbf{o}_0, \ldots, \mathbf{o}_\tau)$ are measured, and the observation probability distribution, $b_j(\mathbf{o}_t) = b_{j,t} = p(o_t | q_t = S_j)$, depends on the current state. The Markov assumption gives that

$$p(q_{t+1} \mid q_t, q_{t-1}, \ldots, q_0) = p(q_{t+1} \mid q_t), \tag{4.1}$$

and the probability of the observations satisfies

$$p(\mathbf{o}_t \mid q_t, q_{t-1}, \ldots, q_0) = p(\mathbf{o}_t \mid q_t). \tag{4.2}$$

4.1.1 Viterbi optimisation

From a hidden Markov model $\lambda = (a_{i,j}, b_j, \pi)$ and an observation sequence, $\mathcal{O}_{0\cdots\tau}$, the most likely state sequence,

$$\mathcal{Q}^*_{0\cdots\tau} = \operatorname{argmax}_{\mathcal{Q}_{0\cdots\tau}} p(\mathcal{Q}_{0\cdots\tau} | \lambda, \mathcal{O}_{0\cdots\tau}) \tag{4.3}$$

to produce $\mathcal{O}_{o\cdots\tau}$ can be determined using the classical Viterbi optimisation [64] or *dynamic programming* as described below. Note that

$$p(\mathcal{Q}_{0\cdots\tau} | \lambda, \mathcal{O}_{0\cdots\tau}) = \frac{p(\mathcal{Q}_{0\cdots\tau}, \mathcal{O}_{0\cdots\tau} | \lambda)}{p(\mathcal{O}_{0\cdots\tau} | \lambda)} \tag{4.4}$$

and that $p(\mathcal{O}_{0\cdots\tau} | \lambda)$ does not depend on the state sequence $\mathcal{Q}_{0\cdots\tau}$. This means that an equivalent formulation would be

$$\mathcal{Q}^*_{0\cdots\tau} = \operatorname{argmax}_{\mathcal{Q}_{0\cdots\tau}} p(Q_{0\cdots\tau}, O_{0\cdots\tau} | \lambda), \tag{4.5}$$

This expression can be optimised by defining

$$\delta_t(i) = \max_{q_0,\dots,q_{t-1}} p(q_0, \dots, q_{t-1}, q_t = S_i, \mathbf{o}_0, \dots, \mathbf{o}_t). \tag{4.6}$$

Note that the Markov assumption implies that

$$p(q_0, \dots, q_t, \mathbf{o}_0, \dots, \mathbf{o}_t) = p(\mathbf{o}_t \mid q_t) p(q_t \mid q_{t-1}) p(q_0, \dots, q_{t-1}, \mathbf{o}_0, \dots, \mathbf{o}_{t-1}). \tag{4.7}$$

For $t = 0$, $\delta_0(i)$ becomes $p(q_0 = S_i, \mathbf{o}_0)$, which can be calculated as $\delta_0(i) = \pi_i b_{i,0}$, and for $t > 0$ it follows that $\delta_t(i) = \max_j(\delta_{t-1}(j) a_{j,i}) b_{i,t}$. By also keeping track of $\psi_t(i) = \operatorname{argmax}_j(\delta_{t-1}(j) a_{j,i})$ the optimal state sequence can be found by backtracking from $q_\tau^* = \operatorname{argmax}_i \delta_\tau(i)$, and letting $q_t^* = \psi_{t+1}(q_{t+1}^*)$ for $t < \tau$.

4.1.2 Infinite time sequences

To handle the situations where $\tau \to \infty$ consider any given time $t_1 < \tau$. The observation symbols $\mathbf{o}_t$, for $0 \leq t \leq t_1$, have been measured, and $\delta_t(i)$ as well as $\psi_t(i)$ can be calculated. Setting $q_{t_1}^* = \operatorname{argmax}_i \delta_{t_1}(i)$ is no longer guaranteed to be optimal as future measurements might generate a different global optimum. This means that the optimal state for $t = t_1$ is unknown. Consider instead some set of states, $\mathcal{X}_t$, at time t such that the global optimum $q_t^* \in \mathcal{X}_t$. For time t_1 this is fulfilled by letting $\mathcal{X}_{t_1} = \mathbb{S}$, the entire state space. For $\mathcal{X}_t$, $t < t_1$, shrinking sets of states can be found by letting $\mathcal{X}_t$ be the image of $\mathcal{X}_{t+1}$ under ψ_{t+1}, that is

$$\mathcal{X}_t = \{S_i | i = \psi_{t+1}(j) \text{ for some } S_j \in \mathcal{X}_{t+1}\}. \tag{4.8}$$

If the dependencies of the model is sufficiently localised in time, then for some time $t_2 < t_1$, there will be exactly one state $q_{t_2}^*$ in $\mathcal{X}_{t_2}$, which have to be the global optimum for time t_2 since $\mathcal{X}_t$ is constructed to be guaranteed to contain the global optimum. The optimal state q_t^* for all $t \leq t_2$ can be obtained by backtracking from $q_{t_2}^*$. No future observations made can alter the optimal state sequence for $t \leq t_2$. This algorithm will be referred to as *online Viterbi optimisation*.

4.1.3 Infinite state spaces

The problem with using the Viterbi optimisation for large state spaces is that $\delta_t(i)$ has to be calculated and stored for all states i at each time t, regardless of how insignificantly small their probability might be. By instead only storing the m largest $\delta_t(i)$ and an upper bound, $\delta_{\max}(t)$ on the rest, significantly less work is needed. If m is large enough, the entire globally optimal state-sequence might be found by backtracking among the stored states. The algorithm presented below can decide if this is the case or not for a given sequence and a given m. If the global optimum is not found, m can be increased and the algorithm executed again, or an approximative solution can be found among the stored states.

Typically the algorithm is executed off-line for a set of example sequences to find the smallest m that will make the algorithm find the global optimum in each of them. A value of m can then be chosen that will make the algorithm find the global optimum with any given probability (strictly less than one). Then when running the online version, this value of m is fixed. This means that the online algorithm when applied to a test sequence of the same kind as the example sequences will produce the global optimum with this probability. When running online this will correspond to the probability of finding the global optimum for any subsequence of the same length as the example sequences. This can be interpreted as a form of expected error rate. Note that no ground truth is needed for the example sequences, which means that this can be used in a fully automated calibration processes where the only manual labour needed would be to mount the camera and give the calibration program access to it to make recordings.

The details are shown in Algorithm 1. The main idea is to maintain an upper bound $\tilde{\delta}_t(i) \geq \delta_t(i)$ where in most cases equality holds. It will be proven that if the algorithm returns that a global optimum was found, then $\tilde{\delta}_t(i) = \delta_t(i)$ along the optimal state sequence. Thereby a state sequence with higher likelihood than an upper bound on all other sequences have been found. This means that it has an higher likelihood than any other state sequence and is thus a global optimum.

This upper bound will be represented by $\hat{\delta}_t(\cdot)$ and $\delta_{\text{max}}(t)$, where $\hat{\delta}_t(\cdot)$ is formed from $\tilde{\delta}_t(\cdot)$ by reordering the indexes to make it a decreasing function. Only the m most likely states will be stored, i.e. the values $\hat{\delta}_t(i)$ for $1 \leq i \leq m$, and $\delta_{\text{max}}(t)$ will be stored as an upper bound on all other states, $i > m$.

For each time step, during forward propagation (lines 4-21), the algorithm investigates the states reachable from the m stored states. The reachable function is defined as $R(\hat{\mathcal{S}}) = \left\{ i | a_{j,i} > 0 \text{ for some } j \in \hat{\mathcal{S}} \right\}$, which is the set of states reachable from a set of states, $\hat{\mathcal{S}}$, in one time step. For each of them, $\psi_t(i)$ and $\delta_t(i)$ should be calculated using a maximum over $\delta_{t-1}(\cdot)$. If this maximum is one of the m stored states an exact value of $\hat{\delta}_t(\cdot)$ is calculated. Otherwise only an upper bound is calculated using $\delta_{\text{max}}(t)$. If that is the case $\hat{\psi}_t(i)$ is set to -1. If it is possible to backtrack (line 22-29) without ever reaching a $\hat{\psi}_t(i) = -1$ a global optimum is found. Also, the constant $a_{\text{max}} = \max_{i,j} a_{i,j}$ is used, and the permutation, $h_t(\cdot)$ that performs the reordering of $\tilde{\delta}_t(\cdot)$ has to be stored, i.e. $\tilde{\delta}_t(h_t(j)) = \hat{\delta}_t(j)$. Below in the algorithm line 1-4 initiates, line 5-24 is the forward propagation and line 25-35 is the backtracking.

To prove that this algorithm is correct, Proposition 1 below shows that in the general case $\hat{\delta}_t(i)$ in Algorithm 1 is an upper bound on $\delta_t(h_t(i))$ from the Viterbi algorithm. Then, in Proposition 2, it is shown that when the algorithm returns that a global optimum were found, $\hat{\delta}_t(i)$ is actually equal to $\delta_t(h_t(i))$ at least for (i, t) on the optimal state sequence. In the typical case it is true for most (i, t) though.

Algorithm 1 InfiniteViterbi

```
1: $\tilde{\delta}_0(i) = \pi_i b_{i,0}$
2: $(\hat{\delta}_0, h_0) = \mathrm{sort}(\tilde{\delta}_0)$
3: $\delta_{\text{max}}(0) = \max_{i>m}(\hat{\delta}(i))$
4: Discard $\hat{\delta}_0(i)$ for $i > m$
5: **for** $t = 1$ to $\tau$ **do**
6:     $\mathcal{S} = R\left(\{h_{t-1}(i) | i = 1, \ldots, m\}\right)$
7:     **for all** $i \in \mathcal{S}$ **do**
8:         $(\delta_{\text{stored}}, j_{\text{max}}) = \max_{1 \leq j \leq m}(\hat{\delta}_{t-1}(j) a_{h_{t-1}(j),i})$
9:         $\delta_{\text{discarded}} = \delta_{\text{max}}(t-1) a_{\text{max}}$
10:        **if** $\delta_{\text{stored}} > \delta_{\text{discarded}}$ **then**
11:            $\tilde{\delta}_t(i) = \delta_{\text{stored}} b_{i,t}$
12:            $\hat{\psi}_t(i) = j_{\text{max}}$
13:        **else**
14:            $\tilde{\delta}_t(i) = \delta_{\text{discarded}} b_{i,t}$
15:            $\hat{\psi}_t(i) = -1$
16:        **end if**
17:    **end for**
18:    $(\hat{\delta}_t, h_t) = \mathrm{sort}(\tilde{\delta}_t)$
19:    Find some $b_{\text{max}} \geq b_{i,t}$ for all $i \notin \mathcal{S}$
20:    $\delta_{\text{maxcalc}} = \max_{i>m | h_t(i) \in \mathcal{S}} \hat{\delta}_t(i)$
21:    $\delta_{\text{maxdisc}} = \delta_{\text{max}}(t-1) a_{\text{max}} b_{\text{max}}$
22:    $\delta_{\text{max}}(t) = \max(\delta_{\text{maxcalc}}, \delta_{\text{maxdisc}})$
23:    Discard $\hat{\delta}_t(i)$ for $i > m$
24: **end for**
25: $\hat{q}_\tau = \mathrm{argmax}_{i \leq m}(\hat{\delta}_\tau(i))$
26: **if** $\hat{\delta}_\tau(\hat{q}_\tau) \leq \delta_{\text{max}}(\tau)$ **then**
27:    **return** Optimum not found, retry with larger $m$
28: **end if**
29: **for** $t = \tau - 1$ to $0$ **do**
30:    $\hat{q}_t = \hat{\psi}_{t+1}(\hat{q}_{t+1})$
31:    $\tilde{q}_t = h_t(\hat{q}_t)$
32:    **if** $\hat{q}_t == -1$ **then**
33:        **return** Optimum not found, retry with larger $m$
34:    **end if**
35: **end for**
36: **return** Global optimum found!
```

Proposition 1: $\hat{\delta}_t(i)$ and δ_{max} calculated by Algorithm 1 together forms an upper bound on $\delta_t(i)$ calculated by the classic Viterbi algorithm described in Section 4.1.1 such that

$$\delta_t(i) \leq \begin{cases} \hat{\delta}_t\left(h_t^{-1}(i)\right) = \tilde{\delta}_t(i) & \text{if } h_t^{-1}(i) \leq m, \\ \hat{\delta}_{\text{max}}(t) & \text{otherwise.} \end{cases} \tag{4.9}$$

Proof. $\hat{\delta}_t$ is a sorted version of $\tilde{\delta}_t(i)$, generated on line 18 of Algorithm 1, and $h_t(i)$ is the permutation used to perform the sorting, $\tilde{\delta}_t\left(h_t\left(i\right)\right) = \hat{\delta}_t\left(i\right)$. A permutation is a bijective functions, which means it inverse $h_t^{-1}\left(\cdot\right)$ always exists. For $t = 0$, the proposition holds (with equality for $h_t^{-1}(i) \leq m$) because of the initialisation performed on line 1-4. To prove it for $t > 0$, induction can be used. By assuming the proposition true for $t-1$, δ_{stored}, calculated in line 8, gives an upper bound on all transactions from the m previously stored states. That is, for all j s.t. $h_{t-1}^{-1}(j) \leq m$,

$$\delta_{\text{stored}} = \max_{\left\{j | h_{t-1}^{-1}(j) \leq m\right\}} \tilde{\delta}_{t-1}(j) a_{j,i} \geq \tilde{\delta}_{t-1}(j) a_{j,i} \geq \delta_{t-1}(j) a_{j,i}. \tag{4.10}$$

On line 9, $\delta_{\text{discarded}}$ gives an upper bound on the rest. That is, for all j s.t. $h_{t-1}^{-1}(j) > m$,

$$\delta_{\text{discarded}} = \delta_{\text{max}}(t-1) a_{\text{max}} \geq \delta_{t-1}(j) a_{\text{max}} \geq \delta_{t-1}(j) a_{j,i}. \tag{4.11}$$

These two bounds can be combined into into one bound for all previous states j. That is for all j,

$$\delta_{t-1}(j) a_{j,i} \leq \begin{cases} \delta_{\text{stored}} & \text{if } h_{t-1}^{-1}(j) \leq m \\ \delta_{\text{discarded}} & \text{otherwise} \end{cases} . \tag{4.12}$$

By taking the maxium over j on both sides, an upper bound on the maximum over all previous states, $\max_j(\delta_{t-1}(j) a_{j,i})$, can be formed. This maximum is used to calculate $\delta_t(i)$ from $\delta_{t-1}\left(\cdot\right)$ in the Viterbi algorithm, which gives the bound

$$\tilde{\delta}_t(i) = \max(\delta_{\text{discarded}}, \delta_{\text{stored}}) b_{i,t} \geq \delta_{t-1}(j) a_{j,i} b_{i,t}. \tag{4.13}$$

This holds for all j, which means it holds for the maximum over all j as well, and that gives the bound on $\delta_t(i)$,

$$\tilde{\delta}_t(i) \geq \max_j(\delta_{t-1}(j) a_{j,i}) b_{i,t} = \delta_t(i). \tag{4.14}$$

The bound used in the algorithm, $\hat{\delta}\left(i\right)$ is formed by sorting, $\tilde{\delta}\left(i\right)$, and discarding the probabilities of all but the m largest values, which proves the first line of the proposition. To prove the second a similar situation arises for δ_{maxcalc} and δ_{maxdisc} calculated in line 20 and 21 of Algorithm 1. Typically, there will for each frame be quite a lot of states whose probability is calculated and then discarded. The maximum over those states is denoted δ_{maxcalc}. That is for all $i \in \mathcal{S}$ s.t. $h_t^{-1}(i) > m$,

$$\delta_{\text{maxcalc}} = \max_{i \in \mathcal{S} | h_t^{-1}(i) > m} \tilde{\delta}_t(i) \geq \delta_t(i) \geq \delta_{t-1}(j) a_{j,i} b_i. \tag{4.15}$$

There will also typically be even more states whose probability is not calculated at all, and an upper bound for them is denoted δ_{maxdisc}. That is for all $i \notin \mathcal{S}$

$$\delta_{\text{maxdisc}} = \delta_{\text{max}}(t-1)a_{\text{max}}b_{\text{max}} \geq \delta_{t-1}(j)a_{\text{max}}b_{\text{max}} \geq \delta_{t-1}(j)a_{j,i}b_{i,t}. \tag{4.16}$$

Note that for all those states, $i \notin \mathcal{S}$, it holds that $h_t^{-1}(i) > m$ since $\hat{\delta}_t(i)$ is constructed from the m states in $\mathcal{S}$ with largest probability. Combining δ_{maxcalc} and δ_{maxdisc} gives for all i s.t. $h_t^{-1}(i) > m$,

$$\delta_{\text{max}} = \max(\delta_{\text{maxcalc}}, \delta_{\text{maxdisc}}) \geq \max_j(\delta_{t-1}(j)a_{j,i})b_{i,t} = \delta_t(i), \tag{4.17}$$

which concludes the proof. □

Proposition 2: If Algorithm 1 finds a solution (i.e. line 27 or line 33 is never reached) it is the global optimal state sequence.

Proof. If the algorithm finds a solution all $\tilde{\psi}_t(\tilde{q}_t) \neq -1$. In that case $\tilde{\delta}_t(\tilde{q}_t) = \delta_t(\tilde{q}_t)$ (shown below) and thus the exact probability of the state sequence $\tilde{q}_t$. But according to Proposition 1 it is also an upper bound on all $\delta_t(i)$, which includes the likelihood of the global optimal state sequence, $\delta_\mathcal{T}(q_\mathcal{T})$. This means that a state sequence, $\hat{q}_t$, is found with a likelihood larger than or equal to the likelihood of all other state sequences, a global optimum.

To conclude the proof it has to be shown that $\tilde{\delta}_t(\tilde{q}_t) = \delta_t(\tilde{q}_t)$. For $t = 0$ it is clear. Use induction and assume it is true for $t-1$. Since $\tilde{\psi}_t(\tilde{q}_t) \neq -1$, $\tilde{\delta}_t(\tilde{q}_t)$ is calculated on line 11, which means

$$\tilde{\delta}_t(\tilde{q}_t) = \tilde{\delta}_{t-1}(\tilde{q}_{t-1})a_{\tilde{q}_{t-1},\tilde{q}_t}b_{\tilde{q}_t} = \delta_{t-1}(\tilde{q}_{t-1})a_{\tilde{q}_{t-1},\tilde{q}_t}b_{\tilde{q}_t} = \delta_t(\tilde{q}_t). \tag{4.18}$$

□

4.2 Using HMM for tracking

4.2.1 Single object tracking

An HMM such as described above can be used for tracking objects in a video sequence produced by a stationary camera. Initially we assume that the world only contains one mobile object and that this object sometimes is visible in the video sequence and sometimes located outside the scene. The assumption of only one object will later be removed.

The state space of the HMM, denoted $\mathbb{S}^1$, is constructed from a finite set of grid points $\mathbf{x}_i \in \mathbb{R}^2$, $i = 1, \ldots, n$ typically spread in a homogeneous grid over the image. The state S_i represents that the mass centre of the object is at position $\mathbf{x}_i$. A special state S_0, representing the state when the object is not visible, is also needed. The $a_{i,j}$ constants, representing probabilities of the object appearing, disappearing or moving from one position to another, can be measured from training data, or some default values can be assumed. Experiments have shown that their exact values are not so important as long as they provide some reasonable restrictions on how fast objects can move. It is also important where the entry or exit probabilities are greater than 0, as this is where objects may appear or disappear.

The observation symbols, $O_t(\cdot)$, of this model will be background/foreground segmentation images from a camera observing the scene. The shape of the image of an object located in state S_i is defined as the set of pixels, $\mathcal{C}_{S_i}$, in the image that the object covers. This shape can be learnt from training data offline, or it can be approximated by assuming some rough estimate of the object shape. For 2D tracking the objects can for example be assumed to be circular with some radius r, in which case $\mathcal{C}_{S_i} = \{\mathbf{x} \mid |\mathbf{x} - \mathbf{x}_i| < r\}$. For 3D tracking, the objects can be assumed to be box-shaped with some given dimensions standing on the ground plane. The set $\mathcal{C}_{S_i}$ can be found by using a calibrated camera and projecting this box onto the image plane. When the HMM is in state S_i, the pixels in C_{S_i} are expected to be foreground pixels and all other pixels are expected to be background pixels, as there is only one object in the world.

Figure 4.1 illustrates this model. For a given state S_i, a box of some given dimensions is placed on the ground plane centred on $\mathbf{x}_i$. Using a calibrated camera this model is then projected into the image and the expected background/foreground segmentation image, $\hat{O}_t(\mathbf{x}) \in \{0, 1\}$, is generated. This will however differ from the true noise-free background/foreground segmentation image, $\hat{O}'_t(\mathbf{x}) \in \{0, 1\}$, due to for example

- The shape of the object is not modelled perfectly, but approximated as a crude box
- The object may not be solid everywhere, but have transparent parts (such as the windows of a car)
- The state space is discrete and thus the object position not exact.
- There might be small uninteresting objects (dry leaves, rodents, ...) running around in the background that should be ignored
- The noise model of the background/foreground segmentation is not be perfect.
- ...

By using a set of background/foreground segmentation images that has a know state q_t, e.g. has been manually classified, good dimensions of the box can be learnt by using maximum likelihood estimates. Also, the probabilities

$$p_{\text{fg}} = p(\hat{O}'_t = 1 | \hat{O}_t = 1) \tag{4.19}$$

and

$$p_{\text{bg}} = p(\hat{O}'_t = 0 | \hat{O}_t = 0) \tag{4.20}$$

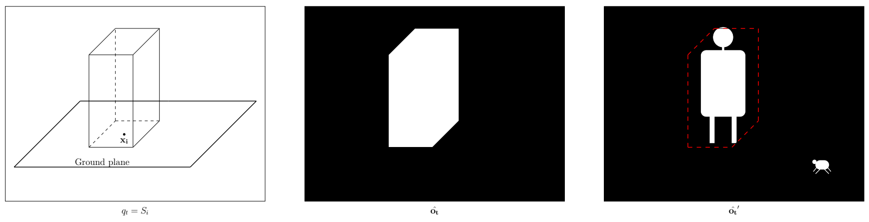


Figure 4.1: Left: illustration of the state S_i consisting of a single object located at position x_i. Middle: the expected observation $\hat{\mathbf{O}}_t$ given the current state $\mathbf{q}_t = S_i$. Right: the true noise-free observation, $\hat{\mathbf{O}}_t'$.

can be estimated. Typically these are well above 1/2, and it is here assumed that they are constant over time and does not depend on the pixel position $\mathbf{x}$, but there is no problem in allowing p_{fg} and p_{bg} vary over t or $\mathbf{x}$.

The observation probabilities, $b_{i,t} = p(O_t|q_t = S_i)$, are given by the background foreground segmentation used. In the algorithm from Chapter 2 Equation 2.59 gives the probability of foreground, $p_{\text{fg}|c,\tilde{l}}(c)$, where c is the correlation coefficient between an image patch in the current frame and the background model. To keep the notation simple, the observations will be defined to be these probabilities, $O_t(\mathbf{x}) = p_{\text{fg}|c,\tilde{l}}(c_{\mathbf{x}})$, where $c_{\mathbf{x}}$ is the correlation for block corresponding to the background/foreground pixel $\mathbf{x}$. It is also possible to use a binary background/foreground segmentation algorithm for example [1] or [75] in which case $O_t(\mathbf{x})$ will be the output of the algorithm, e.g. either 0 or 1. For each pixel $\mathbf{x}$ this gives gives

$$\begin{aligned} p(O_t|\hat{O}_t') &= \begin{cases} O_t & \text{if } \hat{O}_t' = 1 \\ 1 - O_t & \text{if } \hat{O}_t' = 0 \end{cases} \\ &= O_t\hat{O}_t' + (1 - O_t)\left(1 - \hat{O}_t'\right) \end{aligned} . \tag{4.21}$$

The true background/foreground segmentation, $\hat{O}_t'$, is not know, but the state q_t is given, which means the expected foreground/background segmentation $\hat{O}_t$ is know, and the constants p_{fg} and p_{bg} gives the distribution $p(\hat{O}_t'|\hat{O}_t)$. This means

$$\begin{aligned} p(O_t|\hat{O}_t) = \sum_{\hat{O}_t' \in \{0,1\}} p(O_t|\hat{O}_t')p(\hat{O}_t'|\hat{O}_t) = \\ = \begin{cases} O_t p_{\text{fg}} + (1 - O_t)\left(1 - p_{\text{fg}}\right) & \text{if } \hat{O}_t = 1 \\ O_t\left(1 - p_{\text{bg}}\right) + (1 - O_t)\, p_{\text{bg}} & \text{if } \hat{O}_t = 0 \end{cases} . \end{aligned} \tag{4.22}$$

Note that a pixel in the background/foreground segmentation image produced in the algorithm from Chapter 2 corresponds to a 8×8 block in the original input images. This means that the blocks represented by adjacent pixels do not overlap and thus becomes fairly independent. This is more or less true for other background/foreground segmentations as well. If the pixels are assumed to be independent, the observation probability, $b_{i,t} = p(O_t(\cdot)|q_t = S_i)$, can be found as the product over all pixels,

$$b_{i,t} = \prod_{\mathbf{x} \in \mathcal{C}_{S_i}} \left(O_t(\mathbf{x}) p_{\text{fg}} + (1 - O_t(\mathbf{x}))\left(1 - p_{\text{fg}}\right)\right) \cdot \\ \cdot \prod_{\mathbf{x} \notin \mathcal{C}_{S_i}} \left(O_t(\mathbf{x})\left(1 - p_{\text{bg}}\right) + (1 - O_t(\mathbf{x})) p_{\text{bg}}\right), \quad (4.23)$$

and thereby all parts of the HMM are defined.

4.2.2 Multi object HMMs

Generalising the one object model in the previous section into two or several objects is straightforward. For the two object case the states become $S_{i,j} \in \mathbb{S}^2 = \mathbb{S} \times \mathbb{S}$ and the shapes, $\mathcal{C}_{S_{i,j}} = \mathcal{C}_{S_i} \cup \mathcal{C}_{S_j}$. The transitional probabilities become $a_{i_1,j_1,i_2,j_2} = a_{i_1,i_2} a_{j_1,j_2}$. Figure 4.2 illustrates such a model.

One problem that arises here is that the observation likelihood of two objects perfectly overlapping at $\mathbf{x}_k$ is identical to the observation likelihood of one object at $\mathbf{x}_k$. This is typically solved by introducing a prior that states that the probability of object occupying the same physical 3D space is zero.

Solving this model using the Viterbi algorithm above gives the tracks of all objects in the scene, and since there is only one observation in every frame, the background/foreground segmented image, no data association is needed. Also, the model states contain the entry an the exit events, so this solution also gives the optimal entry and exit points.

4.2.3 Extending to several object types

If all objects in a scene are not of approximately the same physical dimensions, several object types can be introduced. In an intersection it might be of interest to track pedestrians, cars and buses and distinguish between the different types. To do this some discrete set of types $\mathcal{T}$ is introduced that will contains the different types, e.g. for the example above there will be three elements in $\mathcal{T}$ indicating pedestrian, car or bus respectively. The single object state space, $\mathbb{S}^1$, is then replaced with $\mathbb{S}^1 \times \mathcal{T}$ and the rest of the theory follows as it is presented above. The dimensions of the boxes representing the objects can now be allowed to depend on which type of object it is.

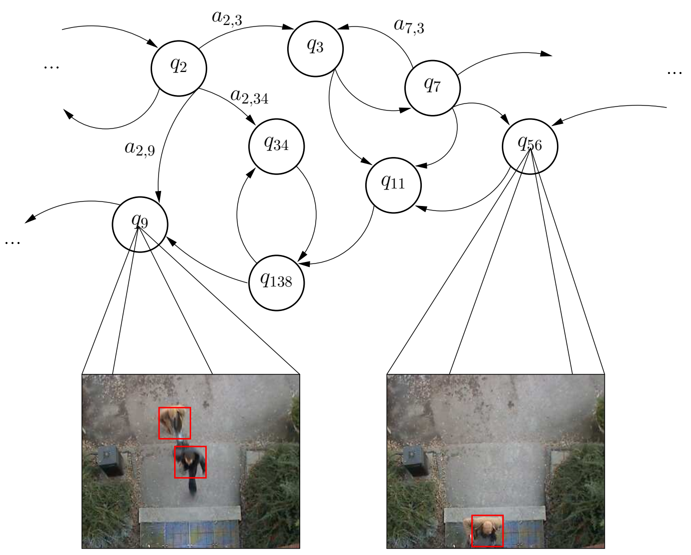


Figure 4.2: Illustration of the HMM used to track multiple objects. Each state represents a configurations of objects in the scene. The transactions represents object movements as as well as the events of objects entering or leaving the scene.

4.2.4 Calculating b_{max}

Algorithm 1 contains a single model specific step, the calculation of b_{max}. It is an upper bound on the observation probability of all states not explicitly tested, i.e. all states not in $\mathcal{S}$. Even though this is typically performed offline, the state-space is too large to make it possible to evaluate the likelihoods of all states in a single frame.

Note that rescaling all observation probabilities of a single frame will not change the position of the maximum. Also, by using the logarithm of the probabilities it is possible to replace the products with sums and use integral images [83] to speed up the calculations. If $b_{0,t}$ is the observation probability of the state representing an empty scene, i.e. $\mathcal{C}_{S_0} = \emptyset$, it is convenient to rescale all observation probabilities with $\frac{1}{b_{0,t}}$. By using the transformed features

$$L = \log \frac{O_t p_{\text{fg}} + (1 - O_t) \left(1 - p_{\text{fg}}\right)}{O_t \left(1 - p_{\text{bg}}\right) + (1 - O_t)\, p_{\text{bg}}}, \tag{4.24}$$

the logarithm of the rescaled observation probabilities can be calculated as sums over the foreground pixels only,

$$\tilde{b}_{i,t} = \log \frac{b_{i,t}}{b_{0,t}} = \sum_{\mathbf{x} \in \mathcal{C}_{S_i}} L(\mathbf{x}). \tag{4.25}$$

Optimising using $\tilde{b}_{0,t}$ instead of $b_{0,t}$ will give the same optimal state sequence as the probability of all state sequences have been rescaled with the same positive amount, $\prod_t b_{0,t}$, and the logarithm does not change the position of the maximum.

Consider the brute force algorithm for finding b_{max} that tests all possible states by recursively adding more and more objects. It is initiated by letting $b_{\text{max}} = -\infty$ and then calling Algorithm 2 with the parameter $i = 0$, i.e. TryAllStateBF(0), which will make the recursion start at the state $\mathbf{q}_0$ representing no objects present.

Algorithm 2 TryAllStateBF(i)

```
1: $b = \sum_{\mathbf{x} \in \mathcal{C}_{S_i}} L(\mathbf{x})$
2: if $i \notin \mathcal{S}$ then
3:     if $b > b_{\text{max}}$ then
4:         $b_{\text{max}} = b$
5:     end if
6: end if
7: for all $j \in \mathbb{S}^1$ do
8:     Find $k$ s.t. $\mathbf{q}_k$ is the state formed from $\mathbf{q}_i$ by adding an object at grid point $\mathbf{x}_j$
9:     call TryAllStateBF($k$)
10: end for
```

To form a more efficient algorithm, introduce a new image L_+ that consists of the positive parts of L, i.e.

$$L_+ = \max(L, 0). \tag{4.26}$$

The sum over this image, $\sum_{\mathbf{x}} L_+(\mathbf{x})$, forms a loose upper bound representing the case that every pixel is expected to take its most likely value. To tighten this bound, notice that as the recursion progresses down a branch more and more objects will be added and thus $\mathcal{C}_{S_i}$ will increase. This means that the sum over the pixels not in this set, $\sum_{\mathbf{x} \notin \mathcal{C}_{S_i}} L_+(\mathbf{x})$, forms an upper bound on how much the observation probability can increasing by continuing down the current branch. If this is not enough to beat the currently best state found, there is no point in continuing investigating the current branch. This is used in in Algorithm 3 to eliminate entire branches from the recursion.

4.2.5 Multiple cameras

Extending this to multiple overlapping or non-overlapping cameras is relatively straightforward. By calibrating the set of cameras and identifying a common coordinate system

Algorithm 3 TryAllState(i)

```
1: $b = \sum_{\mathbf{x} \in \mathcal{C}_{S_i}} L(\mathbf{x})$
2: if $i \notin \mathcal{S}$ then
3:     if $b > b_{\text{max}}$ then
4:         $b_{\text{max}} = b$
5:     end if
6: end if
7: if $b + \sum_{\mathbf{x} \notin \mathcal{C}_{S_i}} L_+(\mathbf{x}) \leq b_{\text{max}}$ then
8:     return
9: end if
10: for all $j \in \mathbb{S}^1$ do
11:     Find $k$ s.t. $\mathbf{q}_k$ is the state formed from $\mathbf{q}_i$ by adding an object at grid point $\mathbf{x}_j$
12:     call TryAllState($k$)
13: end for
```

for the ground plane, the objects centres, $\mathbf{x}_k$, can be modelled as moving in this common coordinate system. Thereby a single HMM modelling the events on this ground plane can be used. The observations for the model are the images from all the cameras. Using the calibration of the cameras, each centre point can be projected into the shape of the object in each of camera images, $\mathcal{C}^c_{\mathbf{x}_k}$, where $c = 1, 2, \ldots$, represents the different cameras. The set $\mathcal{C}^c_{\mathbf{x}_k}$ might be the empty set if an object at position $\mathbf{x}_k$ is not visible in camera c. By indexing Equation 4.23 on the camera c, with O^c_t the background/foreground image produced from camera c,

$$b^c_{i,t} = \prod_{\mathbf{x} \in \mathcal{C}^c_{S_i}} \left(O^c_t(\mathbf{x}) p_{\text{fg}} + (1 - O^c_t(\mathbf{x}))(1 - p_{\text{fg}}) \right) \cdot \\ \cdot \prod_{\mathbf{x} \notin \mathcal{C}^c_{S_i}} \left(O^c_t(\mathbf{x})(1 - p_{\text{bg}}) + (1 - O^c_t(\mathbf{x})) p_{\text{bg}} \right), \quad (4.27)$$

the total observation probability becomes

$$b_{i,t} = \prod_c b^c_{i,t}. \quad (4.28)$$

4.2.6 Region of Interest

In many cases it is neither interesting nor computational possible to consider all objects visible in the camera field of view. Typically the state space is restricted to only consider objects within a specific region of interest. But this leads to problems along the border of this region as objects moving outside this region, but close enough to occlude it in the

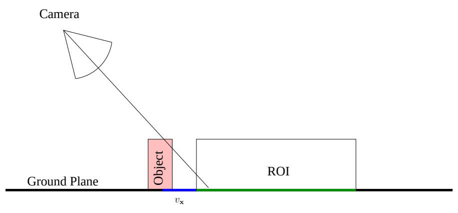


Figure 4.3: This setup shows a camera viewing a ground plane at a tilted angle. A region of interest is specified in 3 dimensions with its height equal to the height of the objects. The thick blue line indicates the region in which objects outside the region of interest will occlude the region of interest.

camera view, will affect the background/foreground segmentation within. Regardless of how the region of interest is chosen (including the entire image), there may always be objects outside this region having some part of them projecting into the region of interest in the image.

This can be resolved by modelling the entire world as a single ground plane with an infinite number of equally shaped objects moving along this plane. The positions of the objects outside a given region of interest are considered uninteresting and are integrated out. This means that S_i will represent the state “one object within the region of interest at position i, and any number of objects at any positions outside”.

To form an observation-model for this kind of states some prior distribution of how many objects are located within a certain area has to be assumed. It could for example be assumed that on average there will be λ objects per m^2. In that case, the number of objects present within an area, v, is Poisson distributed with mean λv,

$$p_o\left(n, v\right) = p\left(n \text{ objects present in } v\right) = \frac{\left(\lambda v\right)^n e^{-\lambda v}}{n!}. \tag{4.29}$$

Using the camera calibration, the geometry of the setup such as shown in Figure 4.3 can be calculated. The region of interest is a three dimensional box, with the same height as the objects. For every pixel $\mathbf{x}$ the area $v_{\mathbf{x}}$ of the region outside the region of interest where objects occlude the pixel $\mathbf{x}$ can be calculated. The probability of the background being visible at that pixel, given that no object within the region of interest occludes it is $p_o\left(0, v_{\mathbf{x}}\right) = e^{-\lambda v_{\mathbf{x}}}$. The probability of the background at that pixel being occluded given that an object within the region of interest occludes it is of course 1. This can be introduced into the observation probability in (4.23), by replacing p_{bg} with

$$p'_{\text{bg}}\left(\mathbf{x}\right) = e^{-\lambda v_{\mathbf{x}}} p_{\text{bg}} + \left(1 - e^{-\lambda v_{\mathbf{x}}}\right)\left(1 - p_{\text{fg}}\right), \tag{4.30}$$

which is no longer constant, but varies over the pixels, $\mathbf{x}$.

The prior model assumed should then also be used in the state probability by introducing a factor, $p_o(n, v)$ to Equation 4.28, where v is the area on the ground plane covered by the region of interest, and n is the number of objects present within. With this modification (4.28) becomes

$$b_{i,t} = \frac{(\lambda v)^n e^{-\lambda v}}{n!} \prod_c b_{i,t}^c. \tag{4.31}$$

All the calculations can be performed offline and stored in an calibration image that for each pixel, x, stores the value of $p'_{bg}(x)$. The extra work needed online is then reduced to a pixel look up in this image.

4.2.7 Static Obstacles

In many real life situations it is hard to find suitable places for mounting the cameras. There might be static obstacles, such as houses, occluding parts of the scene or there might be obstacles in the scene, such as traffic lights or lampposts, that occlude objects in some parts of the scene, but might be occluded them selves by object in some other parts of the scene. To handle these kind of situations such static obstacles can be located, either manually or by some automatic learning based of the tracks produced. Then a static map, $p'_{\text{fg}}(\mathbf{x})$, replacing p_{fg} can be introduced in the same way as $p'_{\text{bg}}(\mathbf{x})$ was introduced in the previous Section. This map will be

$$p'_{\text{fg}}(\mathbf{x}) = \begin{cases} 0.5 & \text{if pixel } x \text{ shows static obstacle} \\ p_{\text{fg}} & \text{otherwise} \end{cases}. \tag{4.32}$$

This will result in that the pixels showing the static obstacles will be ignored while they are expected to show an object as in this case it is not know whether the object is in front of or behind the obstacle. On the other hand when the pixels showing the static obstacles are expected to show the background, there will be a penalty for them showing foreground as this indicates that there actually is an object in front of the obstacle.

This will make the tracker useful in situations where several cameras is needed to view the entire scene because of obstacles occluding different parts of the scene form every camera angle.

If the position of the obstacles on the ground plane were also to be calibrated it would be possible to, for a given state, decide whether the obstacle is in front of or behind the tracked objects and thereby decide whether to expect foreground or background at those pixels. This is achieved by removing the obstacle pixels from the $\mathcal{C}_{S_i}$ sets corresponding to objects located behind the obstacles.

4.2.8 Multi object HMMs using multiple spatially overlapping HMMs

Applying the presented HMM tracker to large scenes is however problematic. The number of states increases exponentially with the number of objects and with the size of the

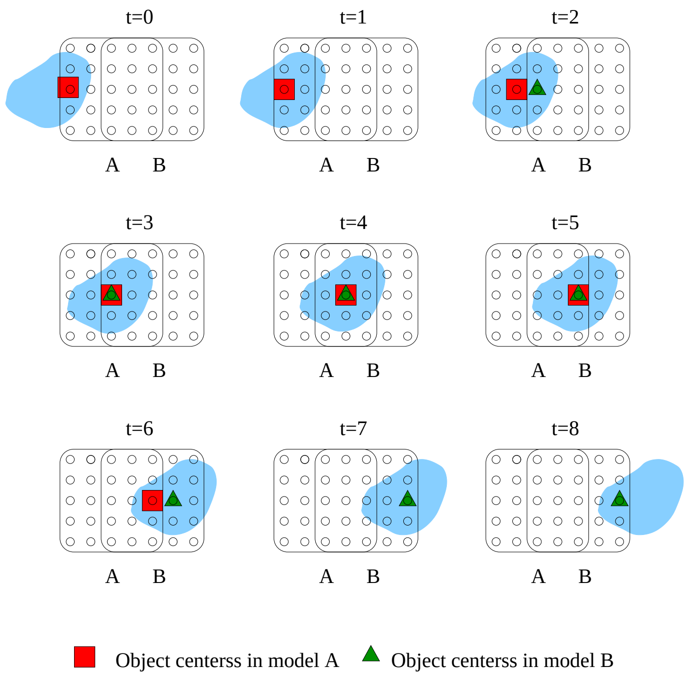


Figure 4.4: An example of two models, A and B, at 9 time intervals, with 5×5 states each overlapping by 3×5 states. The blue blob is an object passing by, and the red square shows the state of model A while the green triangle shows the state of model B.

scene. In practise an exact solution is only computationally feasible in real time for a small number of objects within a small region of interest.

To track objects travelling over larger regions requires an extremely large state space. It might be hard to find an upper bound $b_{max}(t)$ small enough, which means that the number of stored states, m, will be large too. In this case some approximations might be necessary in order to obtain real time performance. The assumption that distant objects do not affect the position or movements of each other more than marginally can be exploited by using several spatially smaller HMM models each only covering a small part of the region of interest. This also means that the number of objects simultaneously located within one such model would decrease, and the state space will be reduced significantly. Each of the models is optimised separately, and the results are combined as shown below.

In Figure 4.4 there is an example of two models, A and B, showing their states as an object passes by. First, only consider the state of model A, the red square. At time $t = 0$, the object is still outside both of the models. However model A detects it in one of its border-states because this is the state of model A that explains the situation best. Then model A follows the centre of the object correctly through $t = 1, \dots, 5$ and in $t = 6$ the same problem as for $t = 0$ arises again: The object has left model A, but is still detected in a border state.

The state of model B, the green triangle, shows the same characteristics as the state of model A. This problem of objects being erroneously detected at the border-states is reduced by using the region of interest introduced in Section 4.2.6, but might still occur. It can be solved by defining the meaning of the border-states to be "object located outside model" and ignore all such detections.

By having 2 or more states overlapping between the models, all grid points will be an internal state, i.e. not on the border, of some model. This means that when the two models are optimised one by one, model A will produce one track starting at $t = 2$ and ending with $t = 4$, while model B produces another track starting at $t = 4$ and ending at $t = 6$. At $t = 4$ both tracks will be in the same state which is used to concatenate the two tracks into a single track. This will be possible as long as there are three or more states overlapping. Typically more than three states will be used in order to get several overlapping states in the resulting tracks. This is a kind of data association problem, but the overlapping states makes it trivial.

If ambiguities arise when combining tracks, this means that the models have failed, typically because they are too small or too widely separated. The models will have to be large enough to resolve any object interaction that might appear, and they will have to overlap enough to ensure that any such interaction will occur well within some model. Otherwise the border effects might prevent the correct solution to be found.

To summarise, instead of solving one big model covering the entire region of interest with many objects, the model is broken down into several small models covering a smaller part of the full region of interest. Thus only a few objects have to be considered. Each of the small models is optimised separately and all detections in border-states are ignored. The resulting tracks that are overlapping, or almost overlapping, are concatenated into longer tracks that might traverse the entire region of interest.

This is of course an approximation and it is no longer possible to decided whether a single larger model covering the entire region of interest would produce the same results.

4.2.9 Results after constant delay

For real time application it might be more important to restrict the maximum delay, t_{max}, between receiving a frame and outputting the result for that frame, than to guarantee that a global maximum is found. That would also make the processing time of each frame and the memory needed less varying. This can be achieved by summing up the total likelihood converging in each node while backtracking,

$$\Delta_t\left(i\right) = \sum_{\psi_{t+1}(j)=i} \Delta_{t+1}\left(j\right). \tag{4.33}$$

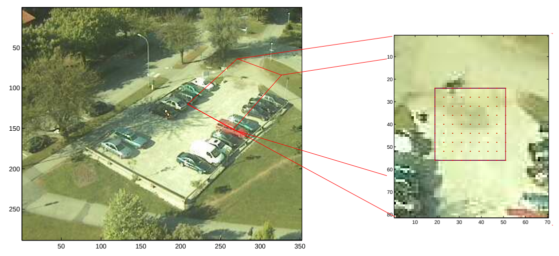


Figure 4.5: Parking lot monitoring by counting the number of cars entering or exiting. To the right an enlargement of the part of the frame where the processing is performed and the object centre points used in the discretised state space are marked.

For the last received frame t_1, set $\Delta_{t_1}(i) = \delta_{t_1}(i)$. As the algorithm is presented above, the backtracking should continue until $\Delta_{t_2}(i) = 0$ for all i but one. But if instead the backtracking is terminated while $\Delta_{t_2}(i) > 0$ for several i, the resulting state for t_2 could be chosen as $q_{t_2} = \operatorname{argmax}_i \Delta_{t_2}(i)$.

By doing so a decision has been made for t_2. To ensure that the resulting state sequence is a legal one, this decision has to be propagated forward. With some luck this was the correct decision to generate the global maximum, and the algorithm will be able to detect if this was the case by letting $\psi_{t_2}(i) = -1$ for all $\{i|\Delta_{t_2}(i) > 0, i \neq q_{t_2}\}$. For approximate solutions it is enough to set $\delta_t(i) = 0$ for all i and t that backtracks to any other state than q_{t_2}.

The additions introduced by evaluating (4.33) replace conditionals in the original backtracking algorithm. So, if conditionals and additions have the same cost, this part will have the same cost as the original backtracking. The extra computational effort needed is to propagate the decisions made forward. It works in much the same way as the backtracking and have the same complexity. For a given backtracking length this means that the constant delay improvement doubles the amount of work needed as compared to the standard backtracking. But the idea is to make the length lower, which also means that memory is saved. For the overall system the time spent backtracking is negligible, as almost all time is spent in the forward propagation of the Viterbi optimisation, Equation 4.6.

4.3 Experiments

4.3.1 Parking lot

As an initial test, a narrow entrance to a parking lot was monitored. This demonstrates that single car tracking with the proposed model works very well. Such a parking lot were monitored by an Axis 2120 camera, and by placing one single 9×9 model with the state

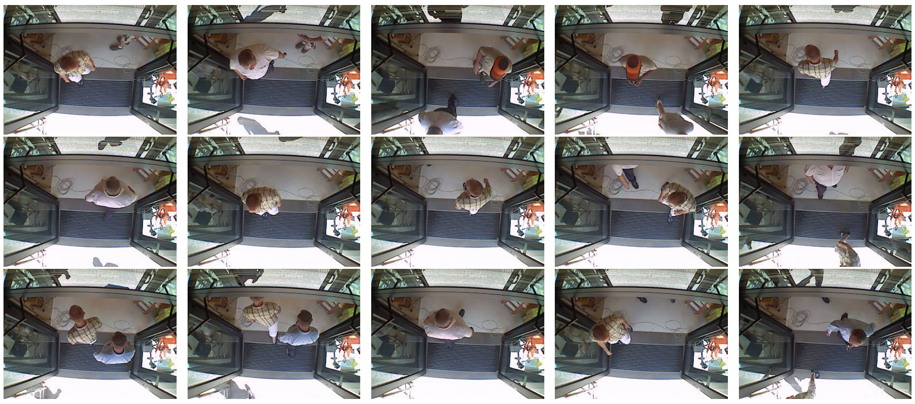


Figure 4.6: 15 of the 17 detections made in a test sequence with 17 passes.

space $\mathbb{S}^1$ at the entrance of a parking lot, see Figure 4.5, the entry and exit of cars were counted. The shape of the cars were chosen as a two dimensional rectangle. This space is small enough to evaluate every state in every frame, i.e. chose $m = \left|\mathbb{S}^1\right|$. Thereby no approximations are made in the optimisation and the resulting state sequence is the globally optimal maximum likelihood state sequence.

The test sequence used is 7 hour long, acquired from 16:30 in the afternoon until darkness falls. The sequence contains 17 events, both cars exiting and entering. All of them is correctly detected and one false detection is made consisting of 5 pedestrians entering the parking lot simultaneously. Since only a small cutout (70×80 pixels) of the image has to be processed, the implementation processes 502 fps on a 2.4GHz P4 including a simple background/foreground segmentation.

4.3.2 Footfall

By placing a camera above an entrance looking straight down it is possible to count the number of pedestrians entering and leaving. This is often called footfall counting. By tracking each person for a short distance the system decides if it is a person entering or exiting. This is done here by placing a single $\mathbb{S}^\infty$ model in the centre of the image that is 1m high and wide enough to cover the entire image.

To choose the m parameter, 45 recordings where made with varying length around 1 min, in different environments with different cameras at different heights. The algorithm where executed offline with a large value of m, and during the backtracking the position of the states on the maximum likelihood sequence in the sorted vector $\hat{\delta}(\cdot)$ were recorded. The maximum over those positions is the minimum m required for the algorithm to produce the global optimum for that particular sequence. Table 4.4 show the minimum

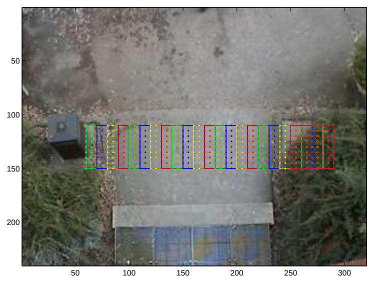


Figure 4.7: 20 models overlapping by seven states. Each new model is shown in a separate colour.

m found for the test sequences in this way by running the offline algorithm.

The mean over all 45 sequences is $\mu = 184$. If the minimum m required to find a global optimum is assumed exponential distributed, the online algorithm with $m = \mathcal{E}_{\text{cdf}}^{-1}\left(0.95\,|\mu\right) \approx 551$ should find the global optimum with probability 0.95. Here $\mathcal{E}_{\text{cdf}}^{-1}\left(x\,|\mu\right)$ is the invers of the exponential cumulative distribution function

$$\mathcal{E}_{\text{cdf}}\left(x\,|\mu\right) = \begin{cases} 1 - e^{-\mu x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{4.34}$$

The online tracker were executed on a short sequence containing 17 passes with $m = 551$. It was processed at 11.9 frames per second, on a 2.4 GHz P4, and all 17 passes were correctly detected and no false detection were made. For each detected passage a single frame were saved and 15 of those are shown in Figure 4.6.

The performance of using multiple spatially overlapping HMMs were tested by mounting a Axis 210 camera above an entrance looking straight down, as shown in Figure 4.8. 20 independent 9×9, S^1 models were placed along a row in the images, with an overlap of 7 states, see Figure 4.7. The overlapping tracks produced by each model were combined, and all tracks laying entirely on the border of its model were removed as described above. The remaining tracks that crosses the entire model from top to bottom were counted. Either as a person entering or as a person leaving depending on the direction of the track.

The test sequence consists of 14 minutes video where 249 persons passes the line. 7 are missed and 2 are counted twice. This gives an error rate of 3.6%. Compared to the parking lot test a larger cutout was processed (320×101 pixels) and 20 models instead of 1 were used. This resulted in a frame rate of 49 fps on the same 2.4GHz P4 as in Section 4.3.1. A few frames from the result are shown in Figure 4.8.

A few more footfall tests are presented in Chapter 5.

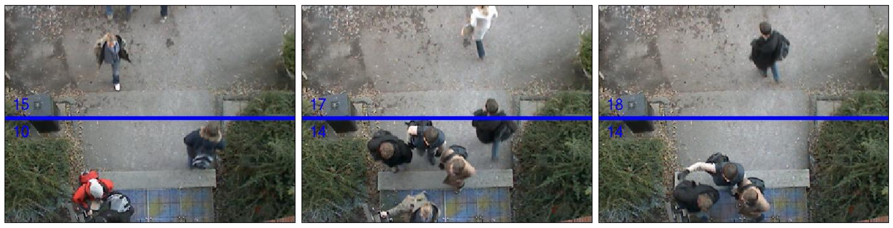


Figure 4.8: People entering and exiting an entrance. The numbers above and below the line gives the number of people crossing the line going out and in respectively. The system detects 4 people entering and 3 people exiting during the time shown in the figure, which is correct. See `lunchcut.avi` at http://www.maths.lth.se/˜ardo/thesis/for the video.

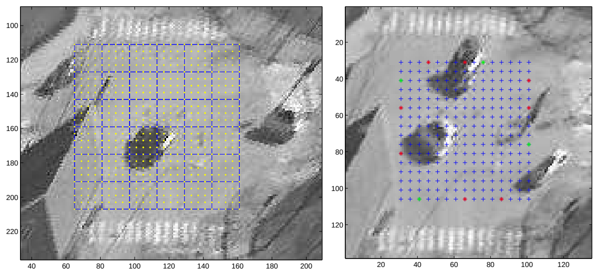


Figure 4.9: Left: A grid of 5×5 small tracking models (blue squares) with 9×9 (yellow dots) each. Right: Grid points in a single model. The red starts is the possible starting points and the green stars the possible exit points.

4.3.3 Traffic

For automatic analysis of traffic surveillance the first step is often to extract the trajectories of the vehicles in the scene. The multi object state space model was tested on this problem by analysing a 7 minutes surveillance video, acquired by an Axis 2120 camera, where 58 cars and a few larger vehicles passed through the centre of an intersection. A grid of 5×5 $\mathbb{S}^2$ models with 9×9 grid points each were placed to cover the intersection, i.e. each model were assumed to contain 0, 1 or 2 cars at each time ($\mathbb{S}^2$). This space is small enough to let the parameter $m = \left|\mathbb{S}^2\right|$. The video was rectified to make all cars of similar size. (roughly 25×25 pixels), and the shape of the objects were considered two dimensional rectangles of this size. The total resolution used was 224×324 pixels. The setup is shown in Figure 4.9 (left) where each blue square represents one model and the yellow dots the different states. Each of the models was optimised separately by using the exact Viterbi algorithm for infinite sequences. In the test 57 of the 58 cars were detected, but for some

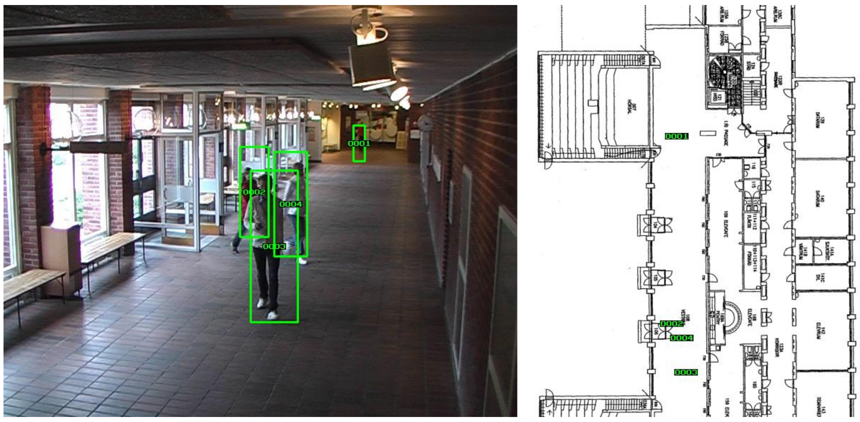


Figure 4.10: Result from corridor tracking example. 4 pedestrians are present and correctly detected. Each is marked in the blueprint with their identity number at their centre position as well as in the image with a box representing areas that should be foreground. See also `korr.avi` at http://www.maths.lth.se/˜ardo/thesis/.

of them only partial tracks were produced. One car was missed due to an interaction of three cars within the same model and two extra tracks were produced due to groups of bicycles. The larger vehicles were detected as one or two cars or not at all. In this case the system was only able to process 0.38 fps, which means that real time performance were not reached, but since the 25 models are evaluated separately it would be no problem spreading the processing on several CPUs or GPUs to reach the desired frame rate.

The first 5 minutes of the above sequence were also passed to a matlab implementation of the infinite state space method described in Section 4.1.3, with a single $\mathbb{S}^n$ model (which can handle 0-n objects being visible) covering the entire intersection, see Figure 4.9 (right). The number of hypothesis stored for each frame, m, were set to 20. This part contained in total 40 vehicles, including some larger than the normal car and one bus. The part where the multiple models above missed one car were included. The full tracks of all the 40 vehicles where correctly extracted. In addition to those correct tracks 5 more were discovered. Two were due to groups of bicycles, and 3 were due to larger vehicles, including the bus. This test was made in matlab at 0.78 fps (0.75 if $b_{\max}(t)$ also were calculated to evaluate whether the global maximum were reached).

4.3.4 Occlusion

To test how well the system could handle occlusions, another test was performed using a camera overviewing a corridor with significant perspective effects and occlusions, see Figure 4.10. The camera was calibrated and the world coordinate system registered to a

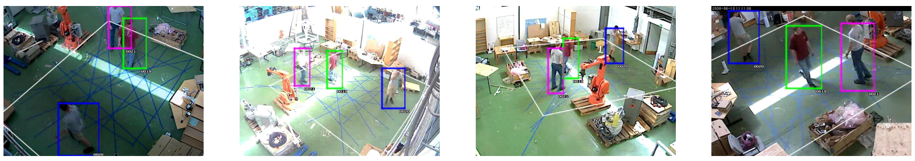


Figure 4.11: Example frames from the 4 camera views of the robot lab sequence. The objects detected by running the tracking within the white rectangle are marked. From left to right, camera 0 to 3. See also `robotlab.avi` at http://www.maths.lth.se/˜ardo/thesis/.

blueprint of the corridor. Pedestrians are modelled as $44 \times 44 \times 180$ cm boxes, and the set of possible object centre point $\mathbf{x}_k$, were generated as an regular grid in the blueprint. For each of the centre points the region of the image a pedestrian located at that point would cover could be calculated from the calibration. Entry and exit points were placed around the doors and along the bottom half of the image.

The sequence is 1:45 min and contains 9 events of people walking through the scene in different ways. All of them were correctly detected. But because of noise from the opening and closing of the doors and reflections a few short erroneously tracks were generated between the entry and exit points belonging to the same door, but they were all easy to filter out afterwards by removing all short tracks starting and ending at the same door. No other errors were made.

4.3.5 Multiple cameras

To test how multiple cameras can be utilised, four cameras were mounted in our lab looking down onto the same area. The four camera views are shown in Figure 4.11. In a first test a 30s sequence were used and a singe $\mathbb{S}^\infty$ model was optimised. The pedestrians were modelled as $40 \times 40 \times 160$ cm boxes and several tests were performed using different sets of cameras. Also, the size of the region of interest within which objects were tracked were varied. The minimum m required for the different setups are shown in Table 4.1.

Looking at the first 4 rows of that table it can be concluded that solving the tracking problem using only camera 2 is significantly harder that using only one of the other cameras. This makes sense because camera 2 is the one behind the robot (see Figure 4.11) and a significant part of the region of interest used here is occluded by the robot. By combining data from several cameras the problem becomes easier as more information is available. Note for example that combining camera 0 and 2 in the 1.5×1.5 m case (middle column) the minimum m required is lowered from 15 and 257 respectively down to 1.

The full 3 minute sequence were then processed. It contains 40 pedestrians passing the 6×6 meter area of interest in different ways. A single passage were missed and no

	Minimum m		
Camera set	1×1 m	1.5×1.5 m	2×2 m
0	1	15	
1	2	4	
2	38	257	
3	11	6	
0,1	1	2	
0,2	1	1	
0,3	1	1	
2,1	10	16	35
2,3	2	2	6
3,1	1	3	
0,2,3	1	1	3
1,0,3	1	3	17
1,2,0	1		
1,2,3	1	2	8
0,1,2,3	1	2	

Table 4.1: Minimum m required to find the global optimum using different combination of the four cameras and with the tracking performed on 1×1, 1.5×1.5 and 2×2 m region of interests.

false positives were made. The results are presented in `robotlab.avi`[1].

4.3.6 Loitering detection

Tests were performed on the PETS 2007 dataset[2]. It consists of 8 different scenarios, each recorded with four cameras from four different angles, Figure 4.12. In addition to this there is a background sequence that can be used for calibrating the system. The first three scenarios, S0, S1 and S2 were analysed to automatically detect loitering. Loitering is defined as a person who enters the field of view of camera 3, and remains within the scene for more than 60 seconds. The dataset comes with calibration parameters for the cameras, that were calculated from markers on the floor using the Tsai camera model [82]. The camera parameters were estimated using the freely available Tsai Camera Calibration Software[3] by Reg Willson.

The ground plane was discretised into a uniform 74x42 grid reaching from $(x, y) = (-4, -2)$ to $(x, y) = (4, 2.5)$ in world coordinates (meters) as defined by the camera calibration's provided. This roughly corresponds to the filed of view of camera 3. The

[1]http://www.maths.lth.se/˜ardo/thesis/

[2]http://pets2007.net/

[3]http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html

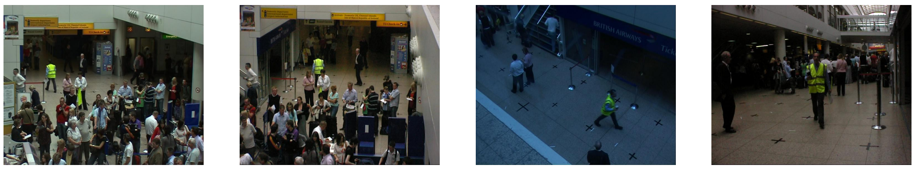


Figure 4.12: Example frames from the PETS 2007 dataset. From left to right, camera 1 to 4. See also http://pets2007.net/.

objects were modelled as 0.44x0.44x1.8 m boxes. Those boxes were projected into each of the four camera images, and bounding boxes parallel to the image axes where fitted. This allows the observation probability of each state to be calculated very fast using integral images.

Objects detected on the very border of the grid were ignored since such detections often correspond to object movements outside the grid. The number of frames an object stayed within the interior of the grid was counted, and if this exceeded 1500 (60 s), the loiter alarm was raised.

Result

The tracker's run time was measured on a 2.40GHz P4 for different values on m and different number of cameras. The background/foreground segmentation was precalculated, so its run time is not considered. For $m = 1$ the algorithm becomes equivalent to a greedy algorithm changing to the most likely state in each frame. The result is shown in Table 4.2.

The performance of the system was measured by evaluating how successful the loitering detection is in different setups. Each setup uses a different set of cameras. The correct result is to find one loiter in each of S1 and S2 and zero in S0. The results are shown in Table 4.3 and it is mostly correct. The errors made are that in three of the setups there were an extra loiter detected, which actually is the same loiter detected twice due to an identity mix up between two objects.

The dataset used comes with ground truth data. This data were compared to tracks generated by the suggested algorithm. For each frame with ground truth, the tracking results were searched for objects within 1m of the ground truth objects. If such objects could be found for all ground truth objects, that frame was considered OK. The percentage of good frames from S1 (ground truth for S0 and S2 were not available at the time) is presented In the "Tracker OK Frames" column of Table 4.3.

To also verify that the identity of the objects are not mixed up a mapping between the id-numbers in the ground truth and the id-number generated by the algorithm has to be established. This is done by choosing the mapping that give the best results. By

m	Cameras	Run time/frame (ms)
10	1	1
10	2	2
10	3	2
10	4	4
100	1	15
100	2	21
100	3	30
100	4	42
1000	1	200
1000	2	281
1000	3	429
1000	4	526

Table 4.2: Mean frame processing time for different values of the m parameter described in Section 4.1.3 using different number of cameras.

using this mapping pairs of ground truth and generated tracks are formed and compared by counting the number of frames the difference between the objects position in the two tracks is less than 1m. The percentage of such frames is presented in the "Tracker OK Ids" column.

The two "Tracker OK" columns should be interpreted as follows. If "Frames" is close to 100%, but "Ids" is lower, the objects were tracked, but the identities mixed up. If "Frames" is low some object were missed altogether. A few frames from the resulting tracks are shown in Figure 4.16 and the full video is available as `pets.avi`[4].

4.3.7 Multiple object types

In a traffic scene there is several types of road users of different types. In this section three different types will be considered: pedestrians (modelled as $0.5 \times 0.5 \times 1.8$ m boxes), cars (modelled as $4 \times 2 \times 1.3$ m boxes) and buses (modelled as $10 \times 3 \times 4$ m boxes). The state space of the previous sections have been extended to also include the orientation of the objects and the type. A 55s recording of an intersection with 4 cameras viewing the intersection from different locations was made, see Figure 4.13. A static obstacle mask were manually created as shown in Figure 4.14. A single $\mathbb{S}^{\infty}$ model was used that covered an area of 63×41 m on the ground plane. The state of a single object were extended to not only consist of its position but also of its orientation and type. This means that the state space is significantly larger than in the previous tests. To lower the run-time a point tracker was used to track points moving through the intersection and then the state

[4]http://www.maths.lth.se/~ardo/thesis/

M	Cameras used	Loiter detections S0	S1	S2	Tracker OK (%) Frames	Ids
100	3	0	1	1	84	73
100	1 3	0	1	1	76	62
100	2 3	0	1	2	85	80
100	3 4	0	1	1	84	73
100	1 2 3	0	1	2	88	84
100	1 3 4	0	1	1	76	62
100	2 3 4	0	1	2	85	81
100	1 2 3 4	0	1	1	88	84

Table 4.3: Tracking result using different sets of cameras. The number of loitering detections found in each of the three sequences are shown. S0 contains no loiters, while S1 and S2 contains one each. The "Tracker OK Frames" column shows the percentage of frames the the algorithm has detected objects less tan 1m from the objects in the ground truth. "Tracker OK Ids" shows the percentages of object frames where the id-number is not mixed up between objects.

space was limited to positions along those point tracks and orientations defined by the motion of those points. The sequence contains 8 cars, 2 pedestrians and 2 buses and it was processed in matlab, which required 4 days for a 2.40 GHz P4. All the objects were detected correctly. Figure 4.15 shows a single frame from the tracking results and videos are available[5] as `4cam_intersection.avi` and `4cam_intersection_rect.avi`.

[5] http://www.maths.lth.se/˜ardo/thesis/

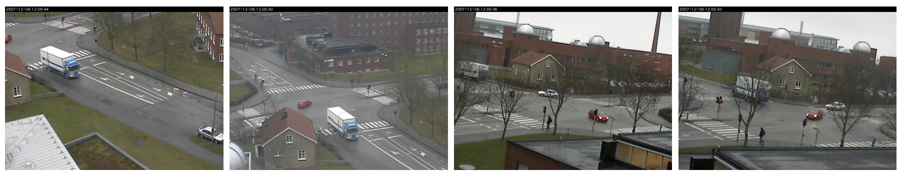


Figure 4.13: Input data for the multi object traffic test. Four cameras were used all looking at the same intersection from different angles. See also `4cam_intersection.avi` at http://www.maths.lth.se/˜ardo/thesis/.

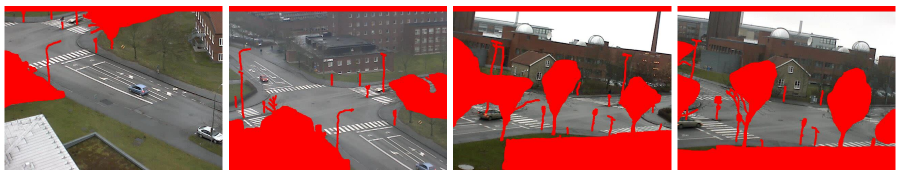


Figure 4.14: Static obstacle masks for the multi object traffic test.

4.4 Conclusions

In this chapter we have proposed a multi HMM model to model multiple targets in one single model with the advantage of solving all the problems of a multi target tracker by a Viterbi optimisation. This includes track initiation and termination as well as the model updating and data association problems. Furthermore, two extensions to the standard Viterbi optimisation are proposed that allows the method to be used in real-time applications with infinite time sequences and infinite state spaces. The first one allows the optimisation to be performed online but with a varying delay and a global optimum is still generated. The second one works with upper bounds instead of explicitly calculating the probability at very state at every time. The later extension only gives approximative solutions in the general case, but can also determine if an exact solution were found. The produced tracks are still guaranteed to follow any restrictions placed on the model, such as “objects may not overlap” or “objects many only (dis)appear along the borders”, which is typically not the case for system based on smoothing (e.g particle filter) instead of optimisation.

Several test have been performs that indicates that the system works very well and in real time situations if the region of interest within which objects are tracked can be kept small and the number of objects kept down. However, it scales badly as the size of this region or the number of objects is increased. The task on monitoring a narrow parking lot entrance can be done at 502 fps while the task of tracking all cars in an entire intersection only reaches 0.003 fps.

Some ideas on how to split up large areas, such as the intersection, into several smaller parts tracked independently is presented. This approach aims to utilise the fact that distant objects do not influence each other much, but could be considered independent. How this can be exploited further and possible build hierarchical models where the higher levels do not have to deal with the exact position of the objects will be the subject of interesting future research.

The focus of this work has been placed on the optimisation algorithms, the observation model used were fairly simple. Much more sophisticated pedestrian detectors exists and it should be straight forward to plug in such a detector instead of or in combination with the background foreground segmentation.

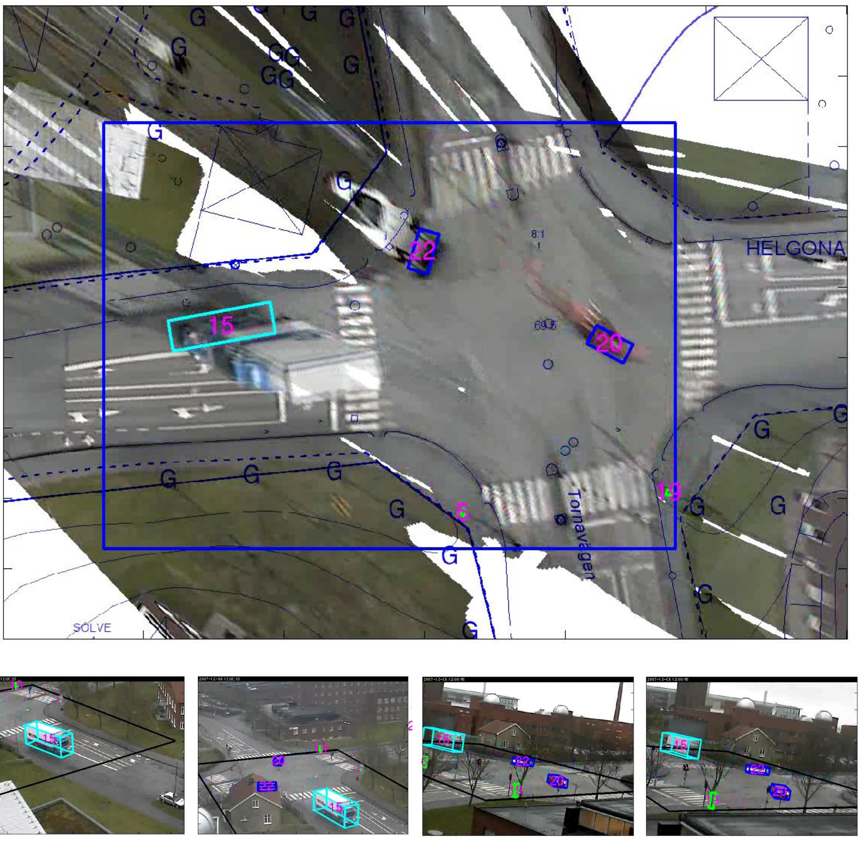


Figure 4.15: Results from the multi object traffic test. At the bottom row, the images from the four cameras shows the detected objects being marked. The object type is indicated with the colour of the marking and an identity number has been assigned to each object. The region of interest have been marked in black. At the top all four images projected onto the ground plane and superimposed onto a blueprint of the intersection. See also `4cam_intersection.avi` and `4cam_intersection_rect.avi` at http://www.maths.lth.se/˜ardo/thesis/.

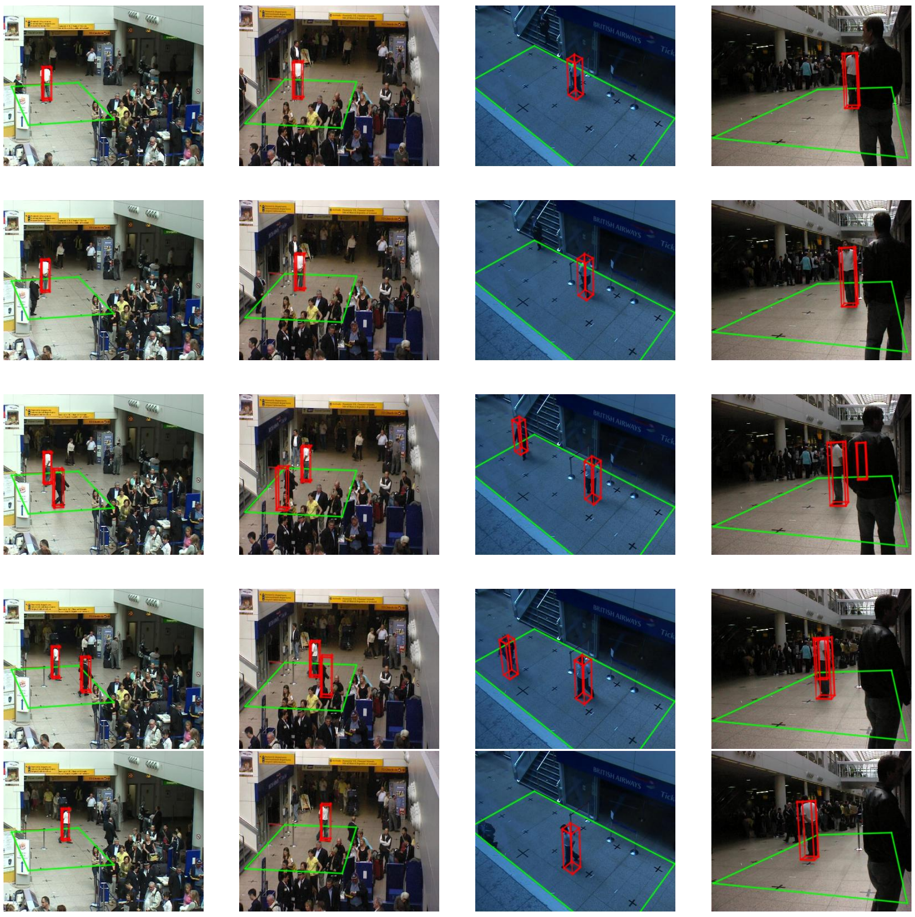


Figure 4.16: Resulting output of the tracker from frames 700-900 of scenario S1. The red boxes indicates the detected and tracked objects in the four cameras. The green rectangle indicates the area within which objects are tracked. See also `pets.avi` at http://www.maths.lth.se/˜ardo/thesis/.

Camera	Lens (mm)	Height (m)	Frames	Minimum m
Axis 207W	4.0	4.91	1199	14
Axis 207W	4.0	2.55	530	16
Axis 207W	6.0	2.55	590	63
Axis 207W	8.0	2.55	560	136
Axis 207W	2.8	2.00	1199	481
Axis 207W	2.8	2.20	70	1
Axis 207W	2.8	2.41	1199	4
Axis 207W	2.8	3.04	1050	49
Axis 207W	2.8	3.04	1199	117
Axis 207W	2.8	3.67	1199	78
Axis 207W	4.0	3.11	1100	192
Axis 207W	6.0	3.11	870	16
Axis 207W	8.0	3.11	840	230
Axis 207W	16.0	3.72	740	13
Axis 207W	4.0	3.72	1060	20
Axis 207W	6.0	3.72	900	757
Axis 207W	8.0	3.72	840	474
Axis 207W	12.0	4.23	880	256
Axis 207W	16.0	4.23	400	5
Axis 207W	4.0	4.23	770	486
Axis 207W	6.0	4.23	970	79
Axis 207W	8.0	4.23	1030	110
Axis 207W	12.0	4.60	660	44
Axis 207W	16.0	4.60	940	21
Axis 207W	4.0	4.60	840	832
Axis 207W	6.0	4.60	800	22
Axis 207W	8.0	4.60	730	3
Axis 209	3.0	2.62	1199	42
Axis 209	3.0	3.17	850	27
Axis 209	3.0	3.79	1199	746
Axis 209	3.0	4.29	1199	28
Axis 209	3.0	4.29	1199	852
Axis 207W	4.0	2.80	1199	1
Axis 207W	4.0	2.80	1199	2
Axis 207W	4.0	2.80	1199	1
Axis 207W	4.0	2.80	1199	6
Axis 209	3.0	3.50	1060	7
Axis 212	2.7	3.11	1199	253
Axis 212	2.7	2.90	1199	533
Axis 212	2.7	4.40	690	996
Axis 212	2.7	2.90	1199	31
Axis 212	2.7	3.80	1199	65
Axis 212	2.7	4.30	1199	49
Axis 212	2.7	5.22	1199	57
Axis 212	2.7	2.63	1199	87

Table 4.4: Minimum m required to find the global optimum for each of the 45 test sequences. For each test the camera type used, the focal length of the lens, the height above the floor of the camera and the length in frames of the sequence is presented.

Chapter 5

Applications

The algorithms presented in this thesis have been used in real world large scale traffic studies and some of these studies will be presented here. The footfall counter presented has also been developed into a commercial product and as of today sold in more than 30 countries world wide.

5.1 TrueView People Counter

Cognimatics AB[1] is a spinnoff company from the math department of Lund university that has used the footfall counter presented above to produce a commercial product called TrueView People Counter. The algorithm is ported to three different platforms and sold as three different products. There is a PC version that is sold as a windows program that can handle entrances up to at least 10m. There is a mini-PC version that can run on a 800 MHz mini-PC and handle two cameras simultaneously, each counting an entrance of up to 4m. Finally there is an embedded version that runs on a 150MHz arm processor embedded in the Axis 207, 209 or 212 cameras that can handle up to 4m wide entrances. The embedded version is the most successful product since it is easier to use as the counting is done within the camera and there is no need to connect the video stream to a separate PC doing the counting. These products have been tested quite extensively and a few of the tests are presented below.

The real time properties of the PC based counter were tested by connecting it directly to an Axis 210 camera mounted above a 10m wide entrance to a shopping centre and the results were compared to a high end commercial counter, already present at the entrance. The system was up and running for a week, with a mean daily error of 4%. The first 5 days at 25 fps (camera maximum), and during the last 2 days the frame rate was limited to 10 fps in order to test how high frame rate was required. The results are shown in Table 5.1 with a total of 4800 passes and a mean error-rate of 4%.

The 800MHz mini-PC implementation was used on 20 cameras spread out within a large grocery store. The cameras were placed in a way that made all entrances to two separate areas covered with counters. That way the total number of pedestrians entering and

[1]http://www.cognimatics.com

Date	In	Out	Mean	Sensor	Error
Fr 7/7	837	853	845	824	2.4%
Sa 8/7	696	747	721	683	5.5%
Su 9/7	563	559	561	586	4.2%
Mo 10/7	578	617	597	575	3.8%
Tu 11/7	690	701	697	680	2.5%
We 12/7	758	761	759	744	2.0%
Th 13/7	831	850	841	767	8.8%

Table 5.1: Results from running our footfall counter for a week at a 10m wide entrance to a shopping centre. In and Out are the number of people entering or leaving the building and Mean is the mean value of those two. Mean is compared to a high end counting system presented as Sensor and the Error is shown in the last column.

leaving those two areas could be calculated by summing up the results from the counters. The test was up and running for 7 months and during this time a total number of 4.9 million passes were counted. The difference between the number of persons entering and the number of persons leaving the two encircled areas was 0.19% and 0.02% respectively.

The 150Mhz embedded version has been tested in a shop in northern Spain during a 7 hour period with a total of 622 passes and an counting error of 0.3%, and in a shopping centre in Madrid at 4 different locations with a total of 621 passes and a counting error of 2.1%. The ground truth in this case was found by manually counting the number of passes by looking at video recordings made at the same time as the counter was running live.

By also measuring the time it takes for each object to move from one end of the tracking area to the other, an estimate of the speed of the object can be obtained. Using this estimate the counter can classify each passing object based on its travelling speed. This have been used in a pilot project together with the company Infracontrol[2].

A counter was mounted above a walkway where both bicycles and pedestrians are travelling. The counter classifies each passing object as either pedestrian or bicycle based on its speed and separate counts are kept for the two different types of objects. Infracontrol has performed some preliminary tests of the system by comparing the counting results with manual counts. Several tests were performed at different times of day on both sunny and cloudy days and both at day and night. The scene is at night illuminated by a street lamp. In total the tests covers 3 hours time and the result shows a precision of 96.43%. More detailed results are presented in Table 5.2.

[2]http://www.infracontrol.com

	Pedestrians		Bicycles		Total
	In	Out	In	Out	
Count	68	299	56	25	448
Errors	0	14	0	2	16
Error rate	0.00%	4.68%	0.00%	8.00%	3.57%
Precision	100.00%	95.32%	100.00%	92.00%	96.43%

Table 5.2: Results from several tests of a pedestrian/bicycle counter. The automatic detections presented on the row marked Count have been compared with manual observations and the number of errors made was counted. In total the tests covers 3 hours time.

5.2 Two-way bicycles on One-way streets

The city of Stockholm has traditionally had a rather small share of bicycle trips compared to many other Swedish cities. However, there are initiatives trying to promote cycling. One possibility is to generally allow biking against one-way traffic. This would extend the available network for bicycle trips and lead to shorter travel routes and times. The backside would be that it might also lead to new dangerous situations and conflicts between the cyclists travelling against one-way traffic and other road users.

The project aims at investigating the total safety effect of allowing cycling against one-way traffic, not only estimating the risk at a specific street. Therefore the design of the study also includes studying changes in the total bicycle flow and in the route choice, i.e. from which streets bicycle traffic transfers to the one-way streets. To be able to both establish the risk at specific sites and route changes the sites were in some cases chosen so as to be able to count cyclists at several alternative routes.

This study was ordered by the city of Stockholm and the main work was performed at the Department of Technology and Society.

Initially, 32 places were selected as potentially interesting for observations. However, finding a good place for camera installations turned out to be a complicated task. The cameras were normally attached to railings on balconies of apartments or offices, but in some places there were no buildings with balconies located near enough. At other sites there were potential camera positions but no electrical power available. Some owners of the buildings did not want to co-operate, or it was impossible to get in contact with them. Finally, only 22 places were filmed of which 18 were further analysed. Three of the excluded sites did not have any one-way streets entering or exiting the intersection (only being selected for counting bicyclists), and the fourth was excluded since the camera turned out to be too far away from the intersection to allow for proper analysis.

Eight camera units were used for the study, and they were moved between sites just before or after the weekend, resulting in three to four weekdays of recordings at each site. The recordings were made as 320×240 motion jpeg files. Figure 5.1 shows a single frame

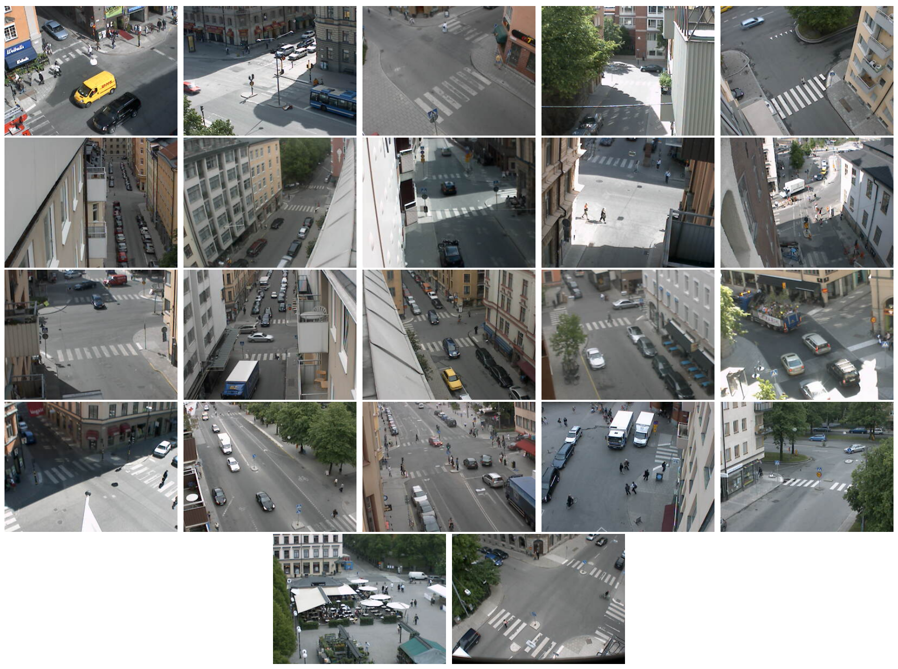


Figure 5.1: A single frame from each of the 22 intersections video recorded.

from each of the 22 intersections. A subset of the dataset [3] is available online including the manual counts made. More can be made available if there is sufficient interest.

The video material was processed and the objects moving in the "wrong" direction detected. To ensure the quality and validate the work of the video analysis system much work was still done manually. This included: a) calculation of the vehicle, pedestrian and cyclist flows for short periods at each site; b) visual control and sorting of the system detections, detection among them the situations which potentially might be conflicts.

Analysis of the recorded video films employs several techniques, that vary in degree of automation, complexity and computationall intensity. Generally, the more advanced technique, the more sensitive it is to errors, quality of input data and calibration procedures and the more validation it requires.

5.2.1 Systematic error in speed and position estimation

An image observed in a camera view is a 2-dimensional representation of the 3-dimensional reality, therefore it is not possible to calculate the distances between the objects in real-

[3]http://www.lth.se/index.php?id=15823

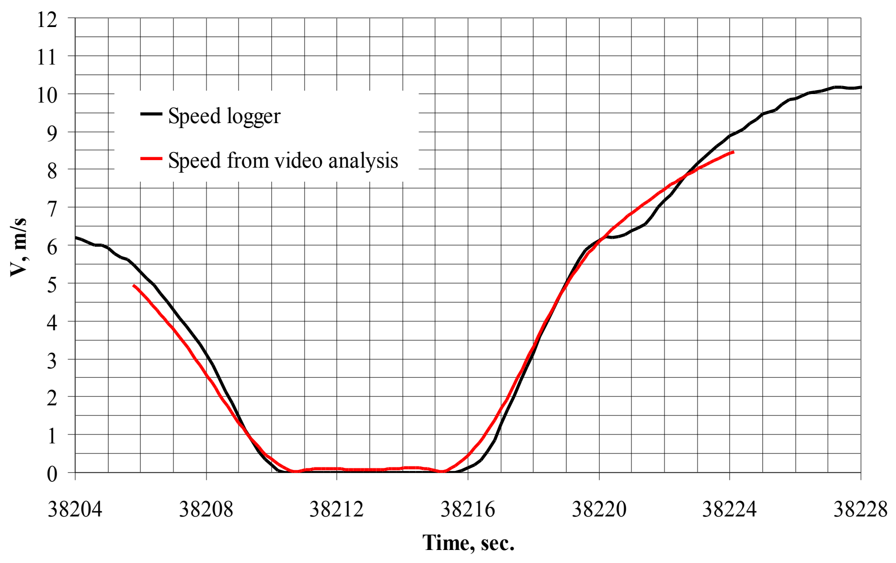


Figure 5.2: Speed estimated from video data vs. speed log in a car.

ity using the measurements in the image only. However, having some prior knowledge about the reality makes it possible to estimate 3D structure. For example, if the image is transformed as if taken from straight above the intersection (i.e. rectified), distances in the ground plane can be calculated using simple scaling.

The rectified image provides accurate distances between the points in a certain plane in the real world, usually the road plane. However, the distances between the object's parts which are elevated above the road plane would be distorted, and the higher the elevation the higher the error is. Making the approximation that the objects are flat and lie in the road plane introduces a systematic error in position estimation as, seen from aside, an object appear to take more place on the road than it actually does. This error depends on the object's height, orientation, distance from the camera, and the angle at which it is seen. Thus theoretically the error is not constant as the object passes through the camera view. In practise, the error varies between 1-2 meters for smaller road users (cars, cyclists and pedestrians), but for high vehicles, such as buses or lorries, it is much higher.

Despite the error in position, accurate estimates of speed can be made extracting a patch in the centre of the detected vehicles and matching those patches using sub pixel cross correlation [2]. If the position error size does not change much during the object passage, the speed (i.e. the change of the position) is nearly free from the error and thus

Site	Bicyclists	Pedestrians	Cars	Other	Sum
2	147	894	11	12	1063
4	100	44	19	7	170
5	110	54	9	14	187
6	63	938	26	126	1153
7	8	13	4	1	26
9	42	367	4	159	572
11	35	104	29	63	230
12	13	312	5	5	334
14	31	140	16	48	235
15	35	426	12	26	497
16	55	667	7	17	745
23	208	347	35	54	645
27	52	163	61	61	337
29	13	11	4	14	42
33	55	50	9	6	120
34	28	491	11	18	548
36	30	15	20	10	74
37	12	1	3	4	19
Total	1037	5037	285	645	6997
Percentage	15%	72%	4%	9%	100%

Table 5.3: The results of manual sorting of the detections at each site (average per day).

the speed estimations would have higher accuracy even without special corrections. Figure 6 illustrates some results from a test where a car with an installed speed logger was filmed and the logged speed was compared with estimations from the video data.

5.2.2 Results

The recording at 18 sites for 3-4 days resulted in 2.5 Tb of data and 900 hours of daytime video material. After the first stage of video analysis, the detection of objects moving in the "wrong" direction, this amount was decreased to approximately 27000 short video clips, with a total length of approximately 115 hours. It was decided that this material would be examined by two observers who classified the detections in 4 categories: bicyclists, pedestrians, cars and other (wrong detection or odd situations). This work took approximately one month of full-time work for the observers. The results are presented in Table 5.3. Since the observational periods were not the same at each site, the numbers are given as an average per day.

To estimate the accuracy of the automatic detection, manual cyclist counts were also

Site	Time	Manual count	Automatically detected			
		Bicyclists	Bicyclists	Pedestrians	Cars	Other
2	07.00-07.30	2	3	7	2	0
4	07.00-07.30	3	3	0	0	0
4	15.00-15.30	4	3	0	1	1
5	07.00-07.30	2	2	1	0	0
5	15.00-15.30	2	4	3	1	0
6	07.00-07.30	1	1	15	3	0
6	15.00-15.30	3	1	29	0	0
7	07.00-07.30	2	0	0	1	0
7	15.00-15.30	1	0	0	0	0
9	07.00-07.30	2	1	1	0	0
9	15.00-15.30	7	4	10	0	0
11	07.00-07.30	1	1	0	1	0
12	07.00-07.30	0	0	2	0	0
14	07.00-07.30	0	0	2	1	0
15	07.00-07.30	1	1	7	1	1
16	07.00-07.30	3	1	5	0	1
16	15.00-15.30	4	0	22	0	0
23	07.30-08.00	7	6	12	0	1
23	15.00-15.30	7	12	18	3	2
27	07.00-07.30	1	1	4	1	0
27	15.00-15.30	0	0	7	0	1
29	07.00-07.30	0	0	0	0	0
29	15.00-15.30	0	0	0	0	0
33	07.00-07.30	8	5	0	1	0
33	15.00-15.30	4	4	3	0	0
34	07.00-07.30	1	1	9	0	0
34	15.00-15.30	5	3	24	0	1
36	07.00-07.30	1	1	0	1	1
36	15.00-15.30	3	2	0	0	1
37	07.00-07.30	3	0	0	0	0
Sum		78	60	181	17	10

Table 5.4: Manually observed vs. automatically detected "wrong-way" cyclists. The automatic detections have been manually classified into bicyclists, pedestrians, cars and other. This means that the bicyclists column under automatically detected is the true positives, the sum of the other three columns is the false positives and the manual count bicyclists column should be close to the ground truth.

Period	"Wrong-way" detector			Track-based detection		
	Cyclists	Pedestrians	Conflicts	Cyclists	Pedestrians	Conflicts
1	8	1	2	9	12	2
2	4	0	1	3	10	0
3	2	3	2	3	5	2
4	3	0	1	2	3	0
Total	17	4	6	17	30	4

Table 5.5: Detection of "wrong-way" cyclists and conflicts by 2 techniques.

done at each site for one or two 0.5-hour periods (from the video films). The comparison between manual counts and automatic detection is given in Table 5.4.

Another task performed by the observers was to detect situations that involve "wrong-way" cyclists and that could potentially be classified as a conflict as defined by [29]. In total, only 43 such situations were found, none of them was classified as a serious conflict according to the definition of the Swedish Traffic Conflict Technique [29].

The information available after the video clips had been manually sorted was sufficient for the purpose of the study, therefore the extraction of road users' tracks was not done on a large scale, but only for a test purpose. Site 33 was chosen for this test as it had relatively high number of potential conflicts (6) concentrated during four 0.5-hour periods (i.e. totally 2 hours of video). The tracks were extracted for all the road users during this period. Even though there was a possibility to analyse only short sequences detected in the first stage, it was interesting to compare the performance of these two techniques in detection of "wrong-way" cyclists, too. As it was known that there are no serious conflicts to be found, the conflict criteria were set quite loose:

One of the road user in an encounter had to move in the "wrong" direction with time to collision < 2 sec. or time advantage < 1 sec. Table 5.5 shows the results of this test.

It turns out that the studied site was in a shade of a large tree during the most part of the day. This resulted in many false track detections located on the shade border. As the leaves were moving in the wind, the shades were detected as separate objects. These tracks were, however, very easy to sort out as they were abnormally long in time while the travel length did not exceed 1-2 meters.

5.2.3 Discussion

One of the main benefits of the automated video analysis is that it condenses the video material to where the events of interest are rare. The amount of raw video data collected in this study was hardly feasible to process employing only human observers. However, the amount of manual work done was still significant, therefore there is still a need for further automation of the process, e.g. use of the track detection and analysis technique on a large scale.

The detection rate of the "wrong-way" cyclists is quite high, most of the manually counted "wrong-way" cyclists were also detected by the system (60 out of 78 in Table 5.4, i.e 77%). However, configuration of the system for a high detection rate results in many false detections. Only 15% of all the detections were cyclist while the main part (72%) were pedestrians walking on the street. This problem can be partly resolved if a more advanced filter is introduced to distinguish between cyclists and pedestrians. The filter should include the threshold values for both the size and the average speed of a moving object.

Generally, the automated detection has lower detection rate than what a human observer has. However, in some cases (sites 2, 5 and 23 in Table 5.4) the video detector found cyclists that were missed by the observers. All three sites are very lively with pedestrians and cyclists mixed, crossing or moving on the street in all possible directions. Such environment is very distractive for a human observer, while the automated detector is not much affected as long as the space between road users is large enough to detect them as separate objects.

5.2.4 Conclusions

The developed automated video analysis system has a great potential for use in behavioural studies, especially when the studied events are rare. However, at the moment, the amount of false detections is still somewhat high and more advanced filtering algorithms are needed.

It is rarely possible to find a very good location to install a camera. Therefore more efforts should be put on developing techniques to compensate for a poor view by using data from several cameras.

There is still a need to improve the accuracy of position and speed estimation, which are important parameters for calculation of safety-related indicators. Since video analysis provides continuous description of road users' trajectories and speed profiles, it provides data for more comprehensive analysis of the behaviour and interactions and can be used, among other things, for validation and enhancement of the conflict techniques.

5.3 Bicycles in roundabout

There are two main ways of handling bicycles when building roundabouts. Either the bicycles are allowed to travel in the same lanes as the motor vehicles, or separate bicycle lanes are constructed. In both situations the bicycles have to interact with the motor vehicles. In the integrated case, when a single lane is shared, they compete for space on that lane and in the separated case the bicycle lanes have to cross the entry and exit lanes of the motor vehicles.

In an ongoing study, performed at the Department of Technology and Society at Lund University, this interaction is studied. Questions of interest includes:

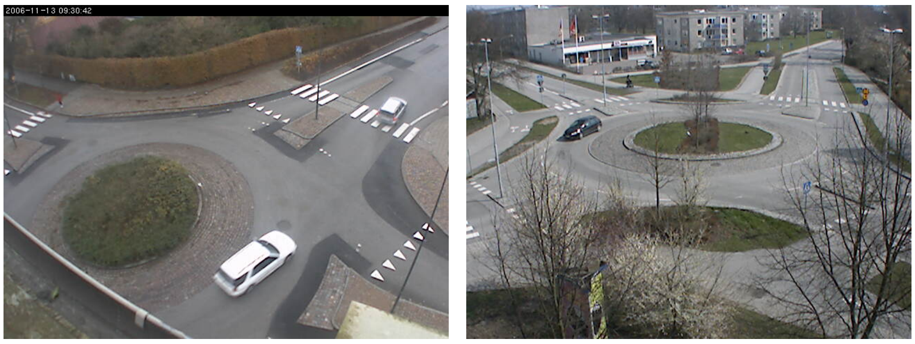


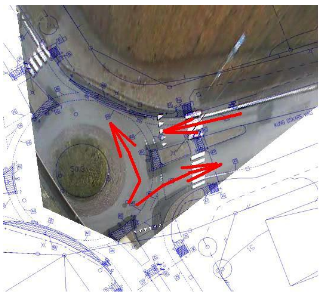
Figure 5.3: A single frame from each of the 2 roundabouts video recorded.

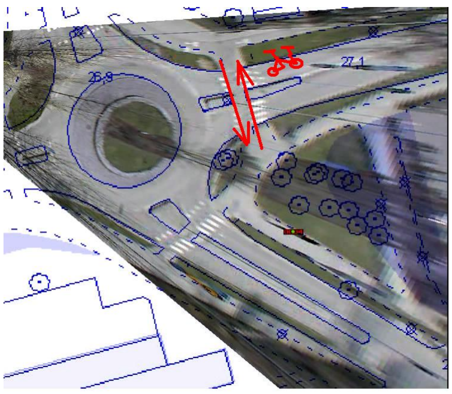
- In what situations will one or several of the road users have to break and how is it decided who will break?
- Is there any commutation going on?
- Are traffic regulations violated?
- Does the behaviours depend on the traffic volume?
- Does the bicyclist use the bicycle lanes or are they using the motor lanes?
- Do the bicycles use the walkway when there are no bicycle lanes?

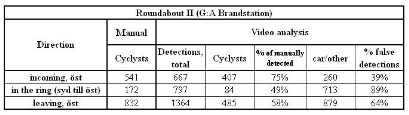
The main focus of the study is the safety of the bicyclists. It is measured by counting the number of dangerous events, such as evasive manoeuvres, that occurs in the roundabout.

Part of the study was performed by using video surveillance of two different roundabouts. The same camera equipment as described above in Section 5.2 was used to make recordings. This time using a resolution of 640×480 and the sites were filmed continuously for 7 and 9 days respectively. Site one is an integrated roundabout while site two is a separated roundabout. Sample frames are shown in Figure 5.3. Data from the first site is available online [4]. Video analytics were used to detect when there are bicyclist passing areas where there is a potential for interactions with the motor vehicles. For each detected cyclist a link containing information about its time position in the original video file was saved. This allowed a manually observer to quickly browse the video between the detections and make the behaviour analysis. The manual observer will then only have to look at at several small clips of the video and for each of those clips decide if it is of no interest or if it contains evasive manoeuvres or conflicts.

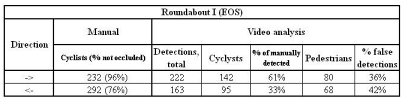
[4]http://www.lth.se/index.php?id=15823







Roundabout II (G:A Brandstation)						
Direction	Manual	Video analysis				
	Cyclysts	Detections, total	Cyclysts	% of manually detected	car/other	% false detections
incoming, öst	541	667	407	75%	260	39%
in the ring (syd till öst)	172	797	84	49%	713	89%
leaving, öst	832	1364	485	58%	879	64%



Roundabout I (EOS)						
Direction	Manual	Video analysis				
	Cyclists (% not occluded)	Detections, total	Cyclysts	% of manually detected	Pedestrians	% false detections
->	232 (96%)	222	142	61%	80	36%
<-	292 (76%)	163	95	33%	68	42%

Figure 5.4: The camera images projected onto blueprints of the roundabouts together with counts of both manual and automated bicycle detections.

To asses the quality of the automatic detections, 24 hour of video from each site was manually studied, and the same kind of detections as was desired from the automated system were made. Results are shown in Figure 5.4.

Chapter 6

Discussions

In this thesis we have present a probabilistic background foreground segmentation algorithm that for each pixel, or block of pixels, calculats the probability of this pixel currently viewing the static background or some moving object. The algorithm can handle the varying lighting of an outdoor traffic scene and is still fast enough to be executed embedded within an Axis network camera containing a 150 MHz ARM processor.

We have also presented a tracking algorithm based on the output of this segmentation algorithm. The output of the tracking algorithm is the trajectories of all objects moving in the scene. The resulting tracks are produced with a few seconds delay but are guaranteed to be consistent and respect any restrictions specified such as "objects may not occupy the same space at the same time" or "objects may only appear at the border of the image". Also, it can combine observations from several cameras viewing the same scene

Classically, tracking algorithms have been constructed to be strictly casual and only considers past frames when making a decision of the current state of the tracked objects. In many surveillance applications this is an unnecessary restriction though. When doing for example people counting or loitering detection there is typically no problem in allowing a delay of several seconds between an event happening in the scene and the detection made by the system. In those cases it is possible to let the decisions made depend not only of past frames but also on future frames.

Recent work on tracking have started to move from causal solutions into the non causal solutions. The work presented in this theses shows how to use classical hidden Markov models with this non casual approach, and how to do that in a way that guarantees consistency in the produced tracks, i.e. no jumping between different hypothesis. Furthermore, the algorithm can can assess whether a global optimum were actually found or not.

Experimental validation shows that this approach works very well for small scale applications where the area of interest only consists of for example an entrance or a corridor. But the solution do not scale very well and is currently not applicable for tracking several kinds of road users in an entire intersection using a normal desktop computer of today.

Some preliminary tests were presented on how this can be solved by using several overlapping hidden Markov models. In these tests the different models were optimised one by one with interactions in between. We do however believe that much can be gained by letting the different models interact. But this will require more research.

Bibliography

[1] Håkan Ardö and Rikard Berthilsson. Adaptive background estimation using intensity independent features. *Proc. British Machine Vision Conference*, 03:1069–1078, 2006.

[2] K. Ăström and A. Heyden. Stochastic analysis of image acquisition, interpolation and scale-space smoothing. *Advances in Applied Probability*, 4(31):855–894, 1999.

[3] Y. Bar-Shalom and E. Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975.

[4] Jérôme Berclaz, François Fleuret, and Pascal Fua. Robust people tracking with global trajectory optimization. In *In Conference on Computer Vision and Pattern Recognition*, pages 744–750, 2006.

[5] Andrew Blake, Michael Isard, and David Reynard. Learning to track the visual motion of contours. *Artificial Intelligence*, 78:101–134, 1995.

[6] M. Boninsegna and A. Bozzoli. A tunable algorithm to update a reference image. *SP:IC*, 16(4):353–365, November 2000.

[7] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(10):1042–1052, 1993.

[8] Marcelo G.S. Bruno and Jose M.F. Moura. Multiframe detector/tracker: Optimal performance. *IEEE Transactions on Aerospace and Electronic Systems*, 37(3):925–946, 2001.

[9] M.G.S. Bruno. Sequential importance sampling filtering for target tracking in image sequences. *IEEE Signal Processing Letters*, 10(8):246–249, 2003.

[10] R. S. Bucy and K. D. Senne. Digital synthesis of non-linear filters. *Automatica*, 7:287–298, 1971.

[11] Yizong Cheng. Mean shift, mode seeking, and clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(8):790–799, Aug 1995.

[12] S. J. Davey, D. A. Gray, and S. B. Colegrove. A markov model for initiating tracks with the probabilistic multi-hypothesis tracker. *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, 1:735–742 vol.1, 2002.

[13] A. Djouadi, O. Snorrason, and F. D. Garber. The quality of training sample estimates of the bhattacharyya coefficient. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1):92–97, Jan 1990.

[14] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1973.

[15] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. 1843/2000:751–767, 2000.

[16] D. Farin, P. H. N. de With, and W. Effelsberg. Robust background estimation for complex video sequences. *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, 1:–145, Sept. 2003.

[17] Nir Friedman. Image segmentation in video sequences: A probabilistic approach. pages 175–181, 1997.

[18] H. Gauvrit, J. P. Le Cadre, and C. Jauffret. A formulation of multitarget tracking as an incomplete data problem. *Aerospace and Electronic Systems, IEEE Transactions on*, 33(4):1242–1257, Oct. 1997.

[19] G. Gordon, T. Darrell, M. Harville, and J. Woodfill. Background estimation and removal based on range and color. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 2:–464, 1999.

[20] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[21] Peter J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

[22] Mei Han, Wei Xu, Hai Tao, and Yihong Gong. An algorithm for multiple object trajectory tracking. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, volume 01, pages 864–871, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[23] Michael Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pages 543–560, London, UK, 2002. Springer-Verlag.

[24] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, (57):97–109, 1970.

[25] E. Hayman and J-O. Eklundh. Background subtraction for a mobile observer. In *Proc. 9th Int. Conf. on Computer Vision, Nice, France*, 2003.

[26] David Hogg. Model-based vision: a program to see a walking person. *Image and vision computing*, 1(1):5–20, 1983.

[27] Wenze Hu, Haifeng Gong, Song-Chun Zhu, and Yontian Wang. An integrated background model for video surveillance based on primal sketch and 3d scene geometry. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[28] Stefan Huwer and Heinrich Niemann. Adaptive change detection for real-time surveillance applications. pages 37–45, 2000.

[29] C. Hydén. *The development of a method for traffic safety evaluation: The Swedish traffic conflicts technique*. PhD thesis, Institutionen fÃűr trafikteknik, LTH, Lund, 1987.

[30] M. Isard and A. Blake. A. visual tracking by stochastic propagation of conditional density. In *4th European Conf. Computer Vision*, 1996.

[31] Michael Isard and Andrew Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, August 1998.

[32] Michael Isard and Andrew Blake. ICONDENSATION: Unifying low-level and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[33] Michael Isard and John MacCormick. Bramble: A bayesian multiple-blob tracker. In *Proc. 8th Int. Conf. on Computer Vision, Vancouver, Canada*, pages 34–41, 2001.

[34] R. C. Jain and H. H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *PAMI*, 1(2):206–213, April 1979.

[35] Omar Javed, Khurram Shafique, and Mubarak Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *in IEEE Workshop on Motion and Video Computing*, pages 22–27, 2002.

[36] Andrew H. Jazwinski. *Stochastic processes and filtering theory*, volume 64 of *Mathematics in science and engineering*. Academic press, 111 Fifth Avenue, New York, 1970.

[37] Norman Lloyd Johnson. *Continuous univariate distributions. Vol. 2*. New York : Wiley, Cop., 1995.

[38] T. Kailath. The divergence and the bhattacharyya distance in signal selection. *IEEE Transaction on Communications*, COM-15:52–60, Feb. 1967.

[39] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME âĂŞ Journal of Basic Engineering*, pages 35–45, 1960.

[40] S. Kaneko, I. Murase, and S. Igarashi. Robust image registration by increment sign correlation. *Pattern Recognition*, 35:2223–2234, 2002.

[41] K. P. Karmann and A. v. Brandt. Moving object recognition using and adaptive background memory. In V. Cappellini, editor, *Time-Varying Image Processing and Moving Object Recognition 2*. Elsevier Science Publishers B.V, 1990.

[42] Jien Kato, T. Watanabe, S. Joga, Ying Liu, and H. Hase. An hmm/mrf-based stochastic framework for robust vehicle tracking. *IEEE Transactions on Intelligent Transportation Systems*, 5(3):142–154, 2004.

[43] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–26, 1996.

[44] P. Kohli and P. Torr. Effciently solving dynamic markov random fields using graph cuts. *ICCV 2005*, 2:922–929, 2005.

[45] P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(12):2079–2088, Dec. 2007.

[46] D. Koller, K. Danilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *Int. J. Comput. Vision*, 10(3):257–281, 1993.

[47] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. Technical Report UCB/CSD-93-780, Nov 1993.

[48] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81, 2002.

[49] P. Kumar, K. Sengupta, and A. Lee. A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system. *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 100–105, 2002.

[50] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[51] B. Leibe, K. Schindler, N. Cornelis, and L. Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(10):1683–1698, Oct. 2008.

[52] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289, May 2008.

[53] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, pages 259–289, May 2008.

[54] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2004.

[55] V. Mahadevan and N. Vasconcelos. Background subtraction in highly dynamic scenes. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–6, June 2008.

[56] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 873 – 889, 2001.

[57] Anurag Mittal and Nikos Paragios. Motion-based background subtraction using adaptive kernel density estimation. pages 302–309, 2004.

[58] Anurag Mittal and Nikos Paragios. Motion-based background subtraction using adaptive kernel density estimation. *cvpr*, 02:302–309, 2004.

[59] Javier Movellan, John Hershey, and Josh Susskind. Real-time video tracking using convolution hmms. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, 2004.

[60] Eva Möller, Gert Grieszbach, Bärbel Schack, Herbert Witte, and Pilu Maurizio. Statistical properties and control algorithms of recursive quantile estimators. *Biometrical journal*, 42(6):729–746, 2000.

[61] P. Noriega and O. Bernier. Real time illumination invariant background subtraction using local kernel histograms. In *Proc. British Machine Vision Conference*, page III:979, 2006.

[62] R. Pless, J. Larson, S. Siebers, and B. Westover. Evaluation of local models of dynamic backgrounds. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2:–73, June 2003.

[63] A. Prati, I. Mikic, C. Grana, and M. M. Trivedi. Shadow detection algorithms for traffic flow analysis: a comparative study. In *IEEE Intelligent Transportation Systems*, pages 340 – 345, 2001.

[64] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.

[65] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transaction on Automatic Control*, 24(6):843–854, 1979.

[66] D.B. Reid. An algorithm for tracking multiple targets. *IEEE Transactions Automatic Control*, 24(6):843–854, December 1979.

[67] Sylvia Richardson and Peter J. Green. On bayesian analysis of mixtures with an unknown number of components. 1994.

[68] T. Schoenemann and D. Cremers. Globally optimal shape-based tracking in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, June 2008.

[69] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1778–1792, Nov. 2005.

[70] R. W. Sittler. An optimal data association problem in surveillance theory. *IEEE Transactions on Aerospace and Electronic Systems*, MIL-8(Apr):125–139, 1964.

[71] Martin Sköld. *Computer Intensive Statistical Methods*. Center for Mathematical Sciences, Lund University, 2005.

[72] Xuefeng Song and Ram Nevatia. Detection and tracking of moving vehicles in crowded scenes. *wmvc*, 0:4, 2007.

[73] S. Spanne. *Konkret Analys*. KFS AB, Sölvegatan 22, Lund, Sweden, 1997.

[74] Henry Stark and John Woods. *Probability, random processes, and estimation theory for engineers*. A Paramount Communications Company, Englewood Cliffs, New Jersey 07632, 2 edition, 1994.

[75] Chris Stauffer. Adaptive background mixture models for real-time tracking. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[76] R. L. Streit and T. E. Luginbuhl. Probabilistic multi-hypothesis tracking. Technical Report 10428, NUWC, Newport, RI, 1995.

[77] Roy L. Streit and Tod E. Luginbuhl. Probabilistic multi-hypothesis tracking. 1995.

[78] J. Sullivan, M. Blake, M. Isard, and J. MacCormick. Bayesian object localisation in images. *International Journal of Computer Vision*, 44(2):111–135, 2001.

[79] T. N. Tan, G. D. Sullivan, and K. D. Baker. Model-based localisation and recognition of road vehicles. *Int. J. Comput. Vision*, 27(1):5–25, 1998.

[80] Luke Tierney. Markov chains for exploring posterior distributions. *Annals of Statistics*, pages 1701–1728, 1994.

[81] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Int. Conf. on Computer Vision*, pages 255–261, 1999.

[82] R. Tsai. An efficient an accurate camera calibration technique for 3D machine. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[83] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. Conf. Computer Vision and Pattern Recognition*. IEEE Computer Society Press, 2001.

[84] F. W. Magnus, Harry Bateman, and Arthur Erdaelyi. *Higher transcendental functions. Vol. II.* McGraw-Hill, New York, 1953.

[85] Rasmus Waagepetersen and Daniel Sorensen. A tutorial on reversible jump mcmc with a view towards applications in qtl-mapping. pages 200–1, 2001.

[86] P. Wayne, Power Johann, and A. Schoonees. Understanding background mixture models for foreground segmentation. *Proceedings Image and Vision Computing*, 2002.

[87] Greg Welch and Gary Bishop. An introduction to the kalman filter.

[88] P. Willett, Y. Ruan, and R. Streit. Pmht: problems and some solutions. *Aerospace and Electronic Systems, IEEE Transactions on*, 38(3):738–754, Jul 2002.

[89] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780–785, 1997.

[90] T. Fortmann Y. Bar-Shalom and M. Scheffe. Joint probabilistic data association for multiple targets in clutter. In *Conf. on Information Sciences and Systems*, 1980.

[91] Li Zhang, Yuan Li, and Ram Nevatia. Global data association for multi-object tracking using network flows. 2008.

[92] Tao Zhao and Ram Nevatia. Bayesian human segmentation in crowded situations. *cvpr*, 02:459, 2003.

[93] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2:28–31, Aug. 2004.

Index