Visual Analysis of Pedestrian Motion

PRS Transfer Report
Supervised by Dr Ian Reid

David Ellis
St John’s College

Robotics Research Group
Department of Engineering Science

Michaelmas 2009
Abstract

We describe progress towards visual analysis of pedestrian motion. While trajectories of humans on foot are stochastic in nature, in a constrained situation underlying patterns of motion can be identified. The work presented in this report focuses on movement of people through scenes which are under visual surveillance. In this case, analysis of continuous video footage provides a history of many observed trajectories which, to a human observer, readily reveals common properties of paths taken. We study the problems associated with automatically recognising such patterns, and also utilising this information to aid other automated tasks.

We begin with a review of previous attempts to model pedestrian activity in visual surveillance, highlighting strengths and weaknesses of current approaches. We also review algorithms which are key components of a visual system for tracking people from mobile cameras. The state of the art in pedestrian detection is reviewed, and details of our implementation of an algorithm designed for close to real time detection are discussed.

The main part of this report describes our novel representation of pedestrian motion patterns. The major strength of this method lies in our use of Gaussian processes for regression which allows us to be explicit about the uncertainty in pedestrian motion. The approach is non-parametric and so full use can be made of the vast quantities of data which surveillance footage provides. We exemplify the use of this model for long term prediction of target motion, and illustrate how the learned model produces more accurate predictions than a typical memoryless system.

Finally, we propose a plan for future work comprising both short term goals and long term research direction. In the short term, plans to extend the scene activity model are presented along with applications in visual surveillance. In the long term, we propose to extend the analysis of human motion to less constrained situations where a mobile camera is used.
Contents

1 Introduction ................................................................................................................. 3
  1.1 Outline ...................................................................................................................... 4

2 Background and Literature Review ................................................................. 6
  2.1 Computer Vision in Visual Surveillance ............................................................. 6
    2.1.1 Scene Activity Modelling ............................................................................... 7
  2.2 Mobile Vision Platforms ...................................................................................... 11
    2.2.1 Pedestrian Detection ...................................................................................... 12
    2.2.2 Mobile Platforms and Visual SLAM ............................................................. 17
  2.3 Conclusion .............................................................................................................. 19

3 Modelling Pedestrian Trajectories ............................................................... 21
  3.1 Gaussian Processes for Regression ..................................................................... 21
    3.1.1 Preliminaries: Properties of the Gaussian Distribution ............................... 22
    3.1.2 Inference of Functions .................................................................................... 23
  3.2 Modelling Pedestrian Trajectories with GPs ................................................... 30
    3.2.1 Modelling Clusters of Trajectories ............................................................... 31
    3.2.2 Long Term Prediction .................................................................................... 34
    3.2.3 Implementation .............................................................................................. 36
    3.2.4 Evaluation ...................................................................................................... 38
    3.2.5 Extending the Model ...................................................................................... 39
    3.2.6 Conclusion ...................................................................................................... 41
4 Conclusions and Future Work

4.1 Summary

4.2 GP Trajectory Modelling Extensions and Applications

4.2.1 Model Extensions

4.2.2 Active, Intelligent Re-acquisition

4.2.3 Semantic Path Classification

4.3 Moving Platforms

4.3.1 SLAM and Visual Surveillance

4.3.2 Open Environments

4.4 Plan

4.5 Acknowledgements

A Courses and Seminars Attended
Chapter 1

Introduction

The demand for intelligent, automated vision systems continues to increase as readily available cameras used in areas such as visual surveillance installations, driver assistance systems, and sports broadcasting continue to produce vast quantities of video data. In many situations the detection, tracking and behaviour analysis of people is an important aspect, or often the primary role, of the visual system.

In this report we focus on the problem of the analysis of pedestrian motion using live video data from both surveillance systems and cameras mounted on mobile platforms. The paths taken by humans on foot are inherently stochastic, with different people inevitably making different decisions about the best route to take in a particular situation. However, people move with the intention of reaching some goal, and these are often common between pedestrians in a given scene. Movement is therefore not completely random, and we argue that it is possible to learn an underlying model of motion through a scene which explains both the shared properties of pedestrian trajectories, as well as how they might differ. Figure 1.1a shows how pedestrian trajectories, while being individually unique, form clear patterns through a scene due to shared intentions.

As a pertinent example of how such knowledge, or lack thereof, affects a vision system consider the overhead surveillance camera in the information engineering building, an image from which is shown in Figure 1.1b. Now consider a person entering the atrium via the bridge on the left. To a human observer, there are two obvious paths to be taken: one carrying straight on, and the other a sharp turn to the left down the stairs. For a typical visual tracker, this higher
level understanding of the scene does not exist, and any prior information on the trajectory that will be taken is usually in the form of a parametric motion constraint such as a constant velocity model. This then causes problems when a target does turn left, since the sudden change in motion is not accounted for by the model, and tracking can fail.

A person with no prior knowledge of the scene would still be able to identify these typical paths given the trajectory data shown. Therefore, we argue that the trajectory history is sufficient for a system to automatically develop a scene specific model of pedestrian motions.

Figure 1.1: History of pedestrian trajectories in (a) an open urban environment, and (b) the atrium of a building. Displaying these trajectories makes common paths clear to a human observer.

1.1 Outline

The next chapter begins with a review of previous attempts to represent pedestrian activity for visual surveillance applications, outlining the strengths and weaknesses of the various approaches. We then discuss developments in two low level vision processes which are particularly vital to the task of tracking pedestrians from more challenging video sequences than encountered using static surveillance cameras. Pedestrian detection is studied with particular attention given to a fast algorithm. We describe our implementation of this algorithm and present results evaluating its performance. We also discuss advances in ego-motion estimation, emphasising the high robustness of the current state of the art.
In chapter 3 we describe our novel approach to modelling pedestrian trajectories in a surveillance setting, which is to be published in [23]. The required background on Gaussian processes is given, as this forms a crucial part of our probabilistic representation. We show how the model can be learned using a trajectory history in an unsupervised fashion. We also derive a method for long term prediction using the model and demonstrate this on two datasets.

Finally in chapter 4 we discuss both immediate and long term future work to be completed in the remainder of the DPhil. The short term work is focused on extensions and application of our model, while longer term goals discuss the use of mobile cameras both in visual surveillance and driver assistance systems.
Chapter 2

Background and Literature Review

This chapter begins with a summary of work carried out by others to date on the modelling of pedestrian motion patterns in a visual surveillance context. Vision algorithms applying to the detection and tracking of people from moving cameras are then briefly reviewed, with a particular focus on fast pedestrian detection, followed by recent advances in ego-motion estimation.

2.1 Computer Vision in Visual Surveillance

The widespread adoption of closed circuit television has created a strong demand for the development of intelligent, automated systems which attempt to maximise the effectiveness of surveillance installations. With an ever increasing number of cameras being introduced, the ratio of human operators to video feeds often becomes so low that much of the useful information from the cameras is missed by the human observers. In the review of Dee [20], installations covering public areas are found to have as few as 3% of screens likely to be monitored by a human at any one time. Much effort has been expended by the computer vision community to create systems that aid the human operatives.

However, there is no reason to assume that the performance of a computerised system is upper bounded by the capability of a human for the same task. Many applications of computer vision and machine learning tackle problems which are beyond the capabilities of a person, and so it is important not to fall into the mindset of simply attempting to mimic human strategies to solve a problem. In particular Dee cites an example in visual surveillance where the suspi-
cious movement of a rubbish bin by a metre in a crowded train station went undetected by 50 operatives, but was easily flagged by a simple background subtraction vision algorithm.

Already visual surveillance has seen the use of a wide spectrum of techniques from the computer vision and machine learning literature. The often hierarchichal nature of vision systems means that low level video processing such as motion detection, background modelling and visual tracking are all vital components to higher level algorithms such as those which analyse behaviour. Since the focus of this report is the analysis of human motion through scenes, the only low level operation reviewed in detail is that of pedestrian detection in section 2.2.1. The next section section reviews work that focuses on the ‘middle layer’ representation of all pedestrian trajectories in a scene after they have been collected, and how these can be further abstracted to a representation of behaviours.

2.1.1 Scene Activity Modelling

The goal of scene activity modelling is to use observed trajectories of pedestrians to encode a high level understanding about the characteristics of target motion through the area under surveillance, where targets are typically people but could be any mobile entities (such as a cars). Typical algorithms require a set of complete tracks as inputs, although some can work on only partially complete trajectories (tracklets) with missing observations. Once this high level representation of motion patterns has been established a few tasks particularly useful to surveillance can be automated. Trajectory classification is the process of labelling a test trajectory as being a member of a particular pattern, a process useful for automatically determining traffic flow, for example. Depending upon the representation of trajectories, this may also give rise to the ability to detect anomalous or suspicious behaviour if the the observed trajectory deviates from any expected pattern. Furthermore, it may also be possible to use the model to make predictions of object motion far into the future. For example given that a car has stopped in a box junction it is likely that the future motion will involve a right turn. We now review previous attempts to create useful scene models.

In their pioneering work Johnson and Hogg [32] began to investigate how a probability distribution over object trajectories could be learned from image sequences in a visual surveillance
context. They describe object motion in a 4 dimensional feature space containing object position and velocity in the image plane. A large set of these flow vectors are collected by tracking pedestrians from a static surveillance camera. This set is then subjected to vector quantisation, performed unsupervised using a neural network, to produce a codebook of representative prototypes where each prototype represents approximately the same number of vectors from the training set. This codebook therefore represents a point density estimate of the pdf over the input space. The temporal relationship between prototypes is maintained by a layer of ‘leaky neurons’ allowing sequences of flow vectors to be represented. A final neural network layer quantises these sequences leaving a set of prototype trajectories, i.e. the common paths taken through the scene.

Although the possibility of applying this method to trajectory classification and motion prediction is discussed, this is not demonstrated until later work. Bulpitt and Sumpter [12] show motion predictions using this neural network framework recursively, with the closest prototype being chosen at each time step. This method of prediction is akin to choosing the nearest neighbour trajectory in the quantised dataset to the observed motion so far. The major weakness of such an approach is that there is no uncertainty in this prediction, and it is unrealistic to predict that all objects will follow a particular set of paths exactly.

Stauffer and Grimson [60] also build upon Johnson and Hogg’s method by using a vast amount of surveillance data from multiple cameras to then classify activities. They replace the second quantisation layer with a hierarchical clustering technique. This enables sequences of prototypes to be separated into groups of varying levels of abstraction. For example the inclusion of object size in the feature vector allows prototypes corresponding to cars and pedestrians to be separated. Further down the hierarchy divisions between directions of traffic flow can be seen, and levels below that separate individual roads taken. However, like the predictions produced by Bulpitt the classifications made are non-probabilistic and no notion of uncertainty is given.

Other attempts to model scene activity have often followed an approach which is more focused on the geometry of trajectories. Often the motion of targets through real scenes is constrained and this can be used as prior information which can lead to simpler representations than
those of [32]. For example, some scenes are merely thoroughfares where motion is expected to be observed between only certain entry and exit points. Makris and Ellis [39] identify entrances and exits by clustering trajectory start and termination points, fitting a Gaussian mixture model (GMM) via the expectation-maximisation (EM) algorithm [10]. The resulting nodes are used along with the training trajectories to fit linear splines to represent the paths between them, as shown in Figure 2.1b. These splines also include variances to indicate the typical deviation of targets from the path in a direction perpendicular to each spline section. Baiget et al. [2] show that more flexible and compact B-splines can also be used to model paths between entry and exit points, however they incorporate no notion of path width. They cluster trajectories by termination points but use quality threshold clustering rather than fitting a mixture model. This has the advantage of not requiring a number of clusters to be specified a priori, the only initial parameter required defines the maximum cluster diameter.

Piciarelli et al. [47] use a simple geometric distance between trajectories to further decompose the routes between entrances and exits into single paths with junctions where ever they merge or split. This (asymmetric) distance is defined as the average Euclidean distance between all points in a trajectory and the closest neighbouring points in the other. After clustering the paths are related by a sparse graph with nodes representing clusters and edges representing transitions between them.

The more recent work of Hu [30] uses a scene representation that shares properties of the approaches of both Johnson and Makris, with the target application of modelling traffic flows. Trajectories are represented by a series of flow vectors with the addition of a size feature as in [60]. Two layers of clustering are performed, first spatially to identify different routes and then temporally to identify different speeds taken along the routes. Although fuzzy k-means is used for clustering in both cases it seems that a GMM/EM approach would be just as applicable. Hu’s key contribution is that groups of trajectories are then represented by a chain of probability distributions, with each node containing the distribution over flow vectors at a particular index along the trajectory. This allows for probabilistic reasoning of higher level behaviour. For example Hu can classify an observed trajectory as belonging to a route by analysing the posterior probability of route membership given only a partial track. The system can also detect
Figure 2.1: Two approaches to modelling pedestrian trajectories through scenes. In (a) Johnson and Hogg [32] estimate the distribution over ‘flow vectors’ $[x, y, \delta x, \delta y]^T$ shown by the arrows. Each prototype arrow in the codebook represents approximately the same number of vectors from the training set, hence a higher density of arrows correspond to more frequently observed motions. In contrast, (b) shows the geometric approach taken by Makris and Ellis [40]. Linear splines joining start and end points are shown along with the path width.

...anomalies by flagging paths taken than do not fit any previously observed patterns. The major drawback of such an approach is that motion patterns are discretised in time so that a finite number of Gaussians can be used for the chain of distributions.

While the majority of literature reviewed so far in scene and pedestrian motion modelling is targeted towards visual surveillance, work targeted towards other applications has produced some interesting approaches. Dapper et al. [17] are concerned with modelling pedestrian motions in attempt to produce humanoid behaviour from a robot. Their method is similar to a classic potential field approach used in path planning [31] where moving objects are assigned a positive potential while goals, or scene exits, are assigned a negative potential. Other objects and obstacles are also assigned positive potentials such that moving targets tend to move towards goals while avoiding each other and obstacles. While intuitive to set up, this method fails to model the stochastic nature of human behaviour since a given starting state will always produce exactly the same predicted path. Bennewitz et al. [8] are also interested in human motion from a robot’s point of view and track pedestrians with a laser range finder in an office environment. They use methods similar to those of Makris, showing the wide applicability of these techniques.

In visual tracking, Ali and Shah [1] have shown how knowledge of crowd flow can be used...
to aid tracking in particularly high crowd densities. They use optical flow among many other cues to produce a probabilistic ‘flow field’ showing the expected movement of targets in the image plane given image position. Their method however is highly tailored to situations where there is one dominant motion pattern in a busy crowd and is not suited to scenes with relatively few people who have more freedom over their choice of movements.

Despite the wide range of approaches reviewed so far, no work combines the efficiency of systems using a spline representation with the power of a probabilistic approach to describe the stochastic nature of human motion. In chapter 3 we present a novel representation based on Gaussian processes which moves towards this goal.

### 2.2 Mobile Vision Platforms

While much research effort has been made towards modelling of human motion using surveillance video data, work on the analysis of behaviour using video data from mobile platforms is much less common. There are undoubtedly some useful applications, such as intelligent cars detecting pedestrians crossing in front of them and other robot to human interaction in general, however this area is relatively underdeveloped. This is because the low level computer vision operations required to obtain trajectories are much more challenging than in the case of static cameras. For example, background subtraction and optical flow are heavily relied upon in surveillance to detect and track pedestrians, yet these both fail immediately when the background itself is no longer static relative to the camera.

However, the continual development of object classification from single images has begun to produce algorithms of sufficient quality that tracking of pedestrians by detection is achievable. The state of the art is briefly reviewed in section 2.2.1. Also recent developments in calculating the ego-motion of a camera are reviewed, as this allows the motion of pedestrians and camera motion to be decoupled.
2.2.1 Pedestrian Detection

Over the last decade, visual object recognition algorithms have advanced rapidly [25] and continue to do so. People are among the more challenging of objects to identify because of their highly varying appearance due to differing clothing, size and limb articulation, for example. While person detection algorithms are providing increasingly improved detection rates, relatively little effort has been made to lower their computational demands, and so state of the art detectors typically require over 10 seconds to process a single image. To be of practical use in visual surveillance and mobile robotics applications a system with close to real time (30Hz) performance is required. In this section, we focus on approaches which have shown promise of providing fast pedestrian detection.

The most common approach to detection involves sliding a fixed size classification window over all image positions and scales, and then making a binary decision as whether each window contains a person. The only other methods of note are parts based detectors [37, 26] which aggregate evidence based on the appearance of body parts in likely relative positions. While these potentially have the ability to better classify humans in different articulations, they are inherently slow since they require classification of many types of objects (arms, heads, legs etc.).

Within the sliding window methods there are two factors that differ between techniques. First is the choice of feature(s) extracted from the window and second is the choice of classifier used to distinguish the feature vector(s) between positive and negative cases. The complexity of features used range widely. The most simple with practical use in people detection are Haar wavelet rectangular features, as used by [46, 63, 64], which can be rapidly evaluated and have been used for real time face detection. Of greater complexity, but showing better performance on pedestrian detection tasks, are histograms of oriented gradients (HoG) [16]. These require computation of per pixel image gradients which are then placed into local histograms of orientations over 16x16 pixel blocks, giving a 36 dimensional feature vector. All features over the whole classification window are concatenated producing a 3780 dimensional feature vector for classification when using a 64x128 pixel window. While more complicated features have been developed (e.g. [53]), recent surveys [22, 67] of state of the art show that HoG features remain
competitive with, and often outperform, more complex systems.

The most successful classification algorithms are discriminative, with support vector machines (SVM) [10] being the most popular for high dimensional feature spaces. These, however, require exhaustive evaluation of the full feature vector over all classification windows in the image. An alternative is to use a boosted cascade of simple features, where windows which are unlikely to contain a person are discarded early after evaluation of only a few features. The famous Viola and Jones face detector [63] uses such an approach with Haar wavelet features to reliably detect faces in real time. However, Haar wavelets proved less accurate for person detection [64], and so we now briefly describe Zhu and Avidan’s approach [68] which uses a boosted cascade but with HoG features. This is followed by an analysis of our implementation of the full detection scheme.

The Boosted HoG Cascade

The methods of [63] and [68] use adaptive boosting (‘Adaboost’ [27]), which learns a strong classifier, $H(x)$, as a linear combination of $T$ weak classifiers, $h(x)$

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x).$$  \hspace{1cm} (2.1)

Where the weak classifiers are any functions such that $h_t(x) : \mathbb{R}^d \rightarrow \{0, 1\}$, and $H(x)$ is classified as true if above a threshold. The values of $\alpha_t$ are learned during the training phase, where typically thousands of positive and negative training examples are used. The classifiers $h_t(x)$ are weak in the sense that they only have to perform better than random guessing for boosting to operate. Thus they can be extremely simple and rapid to evaluate. Zhu and Avidan ([68]) use HoG blocks of various sizes and aspect ratios giving a pool of over 5000 features, along with a linear SVM classifier for $h(x)$. The key benefit of Adaboost is its selection of the most discriminative features from this vast pool, for classification of the current training data. During boosting, each training point (image) has an associated weight that is increased if it is misclassified by the already selected weak classifiers. By choosing the feature that minimises the weighted error, subsequent rounds of boosting are more likely to select features which correctly
classify previously misclassified examples.

To further improve evaluation speed, an attentional cascade of boosted classifiers is used (see Figure 2.2). This is a degenerate decision tree composed of $N$ Adaboost trained strong classifiers where an image window is tested sequentially by each classifier, being discarded if it fails at any stage. By using strong classifiers with only a few features early on in the cascade, with moderate false positive rates ($\sim 50\%$), but near 100% detection rates, as many negative examples as possible are rejected quickly. Later stages in the cascade are only trained using the negative examples which have erroneously passed through preceding stages, and boosting automatically produces more complex classifiers necessary to achieve a maximum allowable false positive rate per stage, and a minimum allowable detection rate.

**Implementation**

We now describe our implementation of Zhu’s algorithm. The detection algorithm is written in C++ for efficiency, while the training stage uses Matlab. Our implementation is identical to Zhu’s with the exception of the choice of linear classifier. We use a weighted form of Fischer’s linear discriminant, as described by Laptev [36], rather than an SVM. This allows the decision hyperplane to be set based on the weighted training points encountered during boosting, whereas an SVM chooses the decision boundary irrespective of weights.

Figure 2.3d shows the composition of our final trained cascade, specifically showing the cumulative number of weak classifiers evaluated by each stage. The first 10 stages have relatively few weak learners, but the curve steepens as more weak classifiers are required to classify the harder examples correctly. At the final (51st) stage of the cascade, close to 2500 features have
been evaluated.

Figures 2.3 (a), (b) and (c) show the detector in operation on an example image. The raw detections are shown in (a), with multiple detections occurring over different scales on the people to the right. The pedestrians in the centre of the image are more challenging since the are viewed from the side, and the detections are poorly localised in space. The ability of the attentional cascade to discard negative windows early is shown by Figure 2.3b. Notice how at this scale, most of classification windows are discarded after the first strong classifier, with windows only progressing further in the highly textured areas of the image. This is further quantified in Figure 2.3c, which shows that for this particular image (now over all scales) almost 90% of windows are rejected by the first stage, and 99.9% of windows do not pass the 13th cascade stage.

Table 2.1 shows some key figures relating to the performance of our implementation of the HoG cascade. While the average evaluation time per image is much lower for the cascade, it is still far from being real time, and can only reliably run between 2-3Hz. Notice also that there is a significant classification accuracy drop when using the boosted cascade. The HoG features used are dependent on integral histograms for rapid evaluation. This means that the trilinear interpolation on histogram bins, and the Gaussian mask applied to each HoG block in Dalal’s original work could not be implemented [68]. A performance decrease is inevitable for the simplified features. When trained for a lower miss rate, this manifests itself by producing more false positives for the cascade.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Evaluation Time</th>
<th>Miss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dalal and Triggs HoG</td>
<td>7s</td>
<td>0.10</td>
</tr>
<tr>
<td>HoG cascade</td>
<td>350ms</td>
<td>0.26</td>
</tr>
<tr>
<td>fastHoG [48]</td>
<td>64ms</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Table 2.1: Evaluation time is the average for a 640x480 RGB image. Miss rate, evaluated on the INRIA dataset[14], is calculated for a detector calibrated at a false positive rate of $10^{-4}$ per window as in [15].

This can be mitigated by analysing the temporal consistency of detections. We implemented a simple Kalman filter tracker[11] for multiple targets[3] under constant velocity models, using the HoG cascade detections as measurements. Figure 2.4 shows the system in action using footage from a static surveillance camera. While most targets are being tracked correctly, false
positive detections can lead to erroneous track formation. Track 34, a false detection in the centre of the image due to the bicycle, is detected as erroneous by the system since the detections are too infrequent and noisy. On the other hand, a frequent false positive in the shop window on the left causes the system to falsely identify a static target.

Due to the evaluation time of the cascade detector, these measurements are pre-computed offline. However, recent developments [48, 66] that exploit readily available multi-core graphics hardware have shown that by parallelising Dalal’s original HoG algorithm, close to real time performance can be achieved without a drop in classification accuracy. The performance of Prisacariu’s implementation is compared to the HoG cascade in table 2.1. While the cascade approach is still valuable if graphics hardware is not available on a platform, Prisacariu’s fastHoG algorithm is currently the most viable detector when real time detections are required.
2.2.2 Mobile Platforms and Visual SLAM

While aggregating detections over time is relatively simple for a static camera, when the viewpoint is on a moving platform several problems must be overcome. The motion of camera must be decoupled from the motion of pedestrians it is observing since the rapidly changing camera dynamics can mask the slower, more consistent motion of people, making tracking difficult.

Simultaneous localisation and mapping (SLAM) is the process of estimating the structure of an unknown scene while also estimating the ego-motion of the sensor. General SLAM has been
almost exhaustively studied in the robotics literature [61], and fully operational visual SLAM systems have been developed for both monocular [19, 33] and stereo [45] camera systems. Of particular note is the system developed at Oxford by Mei et al. [43], which allows accurate estimation of ego-motion in real time using a short baseline stereo camera.

Once the ego-motion has been determined, there still remains the problem of data association and the overall representation used for the dynamic objects in the scene. The most successful system to date is that of Ess et al. [24], who use an integrated approach to jointly estimate camera position, stereo depth, object detection and tracking. An example of this system in operation is shown in Figure 2.5. Although it is claimed that the tracking inference operates in near real time, pedestrian detections are acquired via a slow parts based method [37], and so this initial step is performed offline.

An initial experiment for the combination of our HoG cascade pedestrian detector with the SLAM system of Mei et al. using a moving camera is shown in Figure 2.6. Again, pedestrians are tracked in the image plane using constant velocity model Kalman filters, and here rotation estimates from the SLAM system are used to assist in tracking.

Dependable visual SLAM and pedestrian detection are both vital underlying components of
a system to detect and track people. The work of Ess et al. is a strong demonstration that the technology has reached a stage where such a system is feasible. It therefore seems reasonable that higher level interpretations of pedestrian motion, as investigated extensively in the visual surveillance literature, could now be applied to ego-centric vision. The possibility of this is discussed in chapter 4.

Figure 2.5: Pedestrian detection and tracking by Ess et al. [24], and Leibe et al. [38]

Figure 2.6: Initial experiments using a HoG cascade person detector, Kalman filter tracker in the image plane, and ego-motion estimates from [43], on video footage from a camera mounted on a mobile robot. In this system object depth is not estimated, and so only rotation estimates are extracted from the ego-motion data which is then used to approximately correct object position in the image on a frame by frame basis.

2.3 Conclusion

In this chapter we have reviewed in detail previous attempts to model pedestrian trajectories in scenes under visual surveillance. There is a division between approaches which use all of the available data to compute predominantly probabilistic models, and those which use more
efficient parametric representations such as splines but without a notion of uncertainty. In the next chapter we describe a representation which aims to exploit the strengths of both of these methods. We have also discussed recent advances in pedestrian detection and visual SLAM, which are recognised as two key components to tracking pedestrians from more challenging non-stationary viewpoints. In chapter 4, where we explain plans for future work, short term goals are discussed for applications of our motion representation to visual surveillance problems. In the longer term, we propose analysis of human motion from moving platforms.
Chapter 3

Modelling Pedestrian Trajectories

In this chapter we detail our novel representation of pedestrian trajectories observed in visual surveillance footage. The necessary background on Gaussian processes is given before an explanation of our model, a description of application to long term prediction, and then evaluation on a real dataset.

3.1 Gaussian Processes for Regression

Regression is a supervised learning task where given a data set \( D = \{(y_i, x_i)|i = 1, ..., n\} \) of \( n \) example pairs of continuous input and output variables, \( x \) and \( y \), we wish to estimate the output value \( y \) at a test point \( x^* \). Gaussian processes (GPs) provide a principled, probabilistic approach to regression without some of the computational complexity issues that plague many other Bayesian methods. GPs have been used widely throughout the machine learning community [52] as they are highly amenable to applications where quantification of uncertainty plays a crucial role. A full review of Gaussian Process regression is given here since in section 3.2 GPs are used in the modelling of pedestrian trajectories through scenes under visual surveillance. Before describing GP regression in detail some properties of multivariate Gaussian distributions which will become useful later are reviewed.
### 3.1.1 Preliminaries: Properties of the Gaussian Distribution

An $N$-dimensional Gaussian distribution over the vector $x \in \mathbb{R}^N$ takes the form

$$p(x) = \mathcal{N}(\mu, \Sigma) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) \right\},$$  \hspace{1cm} (3.1)

where $\mu \in \mathbb{R}^N$ is the mean vector and $\Sigma \in \mathbb{R}^{N \times N}$ is the symmetric, positive definite covariance matrix. The entry in row $i$, column $j$ of $\Sigma$ is given by $\Sigma_{ij} = \mathbb{E}[(x_i - \mu_i)(x_j - \mu_j)]$ and so each element gives the covariance between two individual variables in $x$.

Now consider partitioning the random variables in $x$ into two sets, $x_a$ and $x_b$. The resulting joint distribution is

$$p(x_a, x_b) = p\left( \begin{bmatrix} x_a \\ x_b \end{bmatrix} \right) = \mathcal{N}\left( \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix} \right).$$  \hspace{1cm} (3.2)

Since $\Sigma = \Sigma^T$ then $\Sigma_{aa} = \Sigma_{aa}^T$ with the same for $\Sigma_{bb}$, while for the off diagonal blocks $\Sigma_{ab} = \Sigma_{ba}^T$. If only one set of variables, for example $x_a$, are of interest then to calculate the distribution of only $x_a$ it is possible to integrate over all values of $x_b$, a process known as marginalisation.

$$p(x_a) = \int p(x_a, x_b) \, dx_b$$  \hspace{1cm} (3.3)

Remarkably, when the integrand is a Gaussian the resulting distribution is also a Gaussian [10]. When the joint distribution is as given in equation 3.2, then the resulting marginal is

$$p(x_a) = \mathcal{N}(\mu_a, \Sigma_{aa}).$$  \hspace{1cm} (3.4)

Thus when considering only a subset of variables over which we have a joint multivariate Gaussian distribution only the values of the covariance matrix relating to these variables need to be calculated. Intuitively, this is because the covariance matrix entries are the expected value of the product of pairs of variables, and omitting a variable from $x_b$ has no effect on the product of two variables in $x_a$.

It will also be useful to determine $p(x_a | x_b)$ which denotes the conditional probability of $x_a$
given $x_b$. This is given by

$$p(x_a | x_b) = \frac{p(x_a, x_b)}{p(x_b)}$$ \hspace{1cm} (3.5)

The result of conditioning upon a multivariate Gaussian is another Gaussian. This is perhaps more obvious than for marginalisation since, for a fixed value of $x_b$, the above equation is simply a re-normalisation of the joint distribution such that it is a valid distribution over $x_a$. For the partitioned Gaussian the resulting conditional mean and covariance are

$$p(x_a | x_b) = \mathcal{N}\left(\mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b), \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}\right),$$ \hspace{1cm} (3.6)

where the conditional covariance matrix is the Schur complement of the joint covariance matrix.

3.1.2 Inference of Functions

Formally, a Gaussian process is a collection of random variables where any chosen subset of the variables has a joint distribution that is a multivariate Gaussian. To explain GP regression, consider an N-dimensional vector of random variables which are jointly Gaussian and where each variable corresponds to the value of a function $f(x)$ at a different input point,

$$[f(x_1), f(x_2), \ldots, f(x_N)]^T \sim \mathcal{N}(\mu, \Sigma).$$ \hspace{1cm} (3.7)

Now consider extending the vector to have infinite dimension, then the random variables $f(x_i)$ can now represent all values of the function over its whole continuous domain. Rather than having a distribution over a few variables, we have a probability distribution over possible functions. Of course this would require the mean vector $\mu$ and covariance matrix $\Sigma$ to also have infinite dimensionality. However, the marginalisation property of the Gaussian distribution means that when considering only a finite collection of function values only the corresponding blocks of $\Sigma$ and values in $\mu$ need to be calculated. Thus when performing a regression task with observed training points in the dataset $\mathcal{D}$ and a point $x_*$ at which an estimate of the output is required, only the values of $\mu$ and $\Sigma$ associated with these variables need to be considered.

These values are provided by the mean $m(x)$ and covariance $k(x, x')$ functions which to-
gether fully define a GP, and the resulting function \( f(x) \) being distributed as such is denoted

\[
f \sim \mathcal{GP}(m(x), k(x, x')).
\] (3.8)

This is a distribution over functions and can subsequently be used as a prior for inference using Bayes’ theorem, just as with any other probability distribution. The properties of the possible functions are specified by \( m(x) \) and \( k(x, x') \), and the choice of both should reflect the prior information available on \( f \). The only condition on the covariance function is that it must produce positive semi definite covariance matrices for all input values \( x \).

A prototypical choice of such a function is the squared exponential (SE)

\[
k(x, x') = \sigma_f^2 \exp \left\{ -\frac{(x - x')^2}{2l^2} \right\}.
\] (3.9)

The SE covariance promotes smooth functions since points that are close together in input space are assigned a higher correlation than those that are distant, with the correlation tending to zero as the distance between them tends to infinity. \( \sigma_f \) and \( l \) are hyperparameters\(^1\) which provide finer control over the process. Specifically, \( l \) is often referred to as the lengthscale as decreasing it decreases the correlation between more distant points which has the effect of allowing the function to vary more rapidly. \( \sigma_f \) is a scaling parameter which controls the maximum range of the function.

With the covariance function defined it is possible to randomly sample functions from the GP prior. Figures 3.1a and 3.1b show five draws each from a GP with \( m(x) = 0 \) and an SE covariance function as in equation 3.9 but with different values of the lengthscale hyperparameter. Notice how the decreased lengthscale produces distinctly different functions. It is worth noting that there are many other possible choices for \( k(x, x') \), and valid covariance functions can be combined with others to form new alternatives to model a variety of effects. However, the focus here is on the squared exponential since it is particularly useful for the application discussed later.

\(^1\)‘hyper’ since they parameterise the covariance matrix, which itself is a parameter of the final distribution.
Figure 3.1: (a) 5 random draws of functions from a GP prior with zero mean, SE covariance and parameters shown. (b) 5 random draws when $l$ has been decreased, producing functions that vary more rapidly. (c) The effect of conditioning on two data points, shown by the red dots. The solid line shows the predictive mean and the shading indicates 2 standard deviation confidence bounds. The dotted lines show random draws from the posterior distribution. (d) shows an example of data taken from a noisy sine wave (crosses) with missing values. The predictive mean and confidence intervals are shown along with the underlying sine wave (dotted line).
Conditioning on Observations

Now we consider the problem of how to make predictions given some observed values of \( f(x_i) \) which we stack into the vector \( f \). We also take this opportunity to extend the definition of \( f \) to allow multidimensional inputs such that \( f : \mathbb{R}^d \rightarrow \mathbb{R} \). The values of the \( d \)-dimensional training inputs \( x_i \) are then concatenated into the matrix \( X = [x_1, ..., x_n] \). If we wish to predict the value of the function \( f_\ast \) at a particular input \( x_\ast \) then by the definition of a Gaussian process the joint distribution can be written

\[
\begin{bmatrix}
    f \\
    f_\ast
\end{bmatrix}
\sim \mathcal{N}
\left(
\begin{bmatrix}
    m \\
    m_\ast
\end{bmatrix},
\begin{bmatrix}
    K & K_\ast^T \\
    K_\ast & k_{\ast\ast}
\end{bmatrix}
\right),
\tag{3.10}
\]

where the following shorthand for the covariance sub-matrices has been used

\[ K_\ast = [k(x_\ast, x_1), k(x_\ast, x_2), ..., k(x_\ast, x_n)], \quad k_{\ast\ast} = k(x_\ast, x_\ast). \]

while \( K = K(X, X) \) is an \( n \times n \) matrix containing all covariances between the training points themselves. The goal of regression is to obtain \( p(f_\ast|f, X) \), the posterior probability of the test value given the training data. This is easily evaluated by using the conditioning result in equation 3.6 to give

\[
p(f_\ast|f, X) = \mathcal{N}(m_\ast + K_\ast K^{-1}(f - m), k_{\ast\ast} - K_\ast K^{-1}K_\ast^T) \tag{3.11}
\]

\[
= \mathcal{N}(GP_\mu(x_\ast, D), GP_\Sigma(x_\ast, D)) \tag{3.12}
\]

This gives a predicted mean value along with a variance describing the uncertainty in this prediction. The notation adopted in equation 3.12 references the point at which the prediction is being made, \( x_\ast \), along with the data that is being conditioned upon, \( D \). Equation 3.11 demonstrates the non-parametric nature of GP regression since, studying the expression for the predictive mean \( GP_\mu \), it can be seen that the predicted value is a linear combination of all of the training vectors.

Figure 3.1c shows the result of conditioning the prior in Figure 3.1a on two data points. The predictive mean and variance (indicated by 95% confidence intervals) is shown for an evenly spaced range of test points covering the \( x \) axis. Notice how the variance decreases to zero near
the training points and increases away from them. Also shown are three random draws from the posterior. In effect, by conditioning on the two training values functions that disagree with the data are rejected and only those that are feasible while considering the data remain.

It is also possible to make predictions when only noisy observations of the function values are available,

\[ y = f(x) + \epsilon. \]  \hspace{1cm} (3.13)

Provided \( \epsilon \) is Gaussian noise that is independent between observations, the process is still Gaussian but with a modified covariance function \( \text{Cov}(y) = K(X, X) + \sigma_n^2 I \) where \( \sigma_n \) is the standard deviation of the added noise. This expression then replaces \( K(X, X) \) in equation 3.11.

Figure 3.1d shows an example of GP regression with noisy training data. The training set was created by adding Gaussian noise to points along a sine wave and then removing points between \( x = -0.5 \) and 0. Again notice how the uncertainty in the prediction increases when there are no data points. The confidence bounds shown are relating to the observed process \( y \). It is also possible to perform prediction of the underlying process \( f \) only, in which case the uncertainty bound would be smaller. It is important to note that the form of GP predictions is sensitive to the choice of hyperparameters. A reliable approach to inferring optimal values of the hyperparameters from the data is now described.

**Hyperparameter Selection**

For a set of noisy observed values \( y \) being produced by a process as in equation 3.13 from the definition of a GP we have

\[ p(y | X, \theta) = \mathcal{N}(m, K + \sigma_n I). \] \hspace{1cm} (3.14)

Here any hyperparameters of the covariance function, hyperparameters in the mean function and \( \sigma_n \) have been collected into \( \theta \). The above is the marginal likelihood \(^2\), the probability of the observed data values given the hyperparameters. Taking the logarithm leaves the quadratic

\[^2\text{By jumping straight to equation 3.14 we have implicitly marginalised out the values of the underlying process } f: p(y | X) = \int p(y | f, X)p(f | X)df\]
exponent of the Gaussian plus the normalisation term

\[ L(\theta) = \log p(y|X, \theta) = -\frac{1}{2} \log |K| - \frac{1}{2} (y - \textbf{m})^T K^{-1} (y - \textbf{m}) - \frac{n}{2} \log(2\pi). \] (3.15)

The log likelihood, \( L \), can be maximised w.r.t. \( \theta \) to give the maximum likelihood values of the parameters

\[ \hat{\theta}_{ml} = \arg \max_{\theta} p(y|X, \theta) = \arg \max_{\theta} L(\theta) \] (3.16)

Analysing equation 3.15, only the middle term refers to \( y \) and this rewards a process that closely fits the data. On the other hand the first term acts as a complexity penalty, with \(|K|\) increasing for more complex functions (for example shorter length scales with a SE covariance function).

Thus by optimising the marginal likelihood a mechanism for the prevention of over fitting arises naturally. Derivatives of \( L \) w.r.t. \( \theta \) can be easily calculated allowing the use of well established, rapidly converging numerical optimisation routines. For example for the case of a zero mean process, the partial derivative w.r.t. a single parameter \( \theta_j \) is

\[ \frac{\partial L(\theta)}{\partial \theta_j} = -\frac{1}{2} \text{tr}(K^{-1} \frac{\partial K}{\partial \theta_j}) + \frac{1}{2} y^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1} y \] (3.17)

where matrix derivatives are elementwise. To evaluate the gradient it must then be possible to differentiate \( k(x, x') \) w.r.t. the hyperparameters. A brief glance at 3.9 reveals that these derivatives are easily calculated for the squared exponential covariance function. A similar expression can be obtained for the non-zero mean case, with the added condition that the mean function must also be differentiable with respect to its own hyperparameters.

For most datasets the maximisation of \( L \) is non-convex and some degree of care has to be taken to avoid local maxima. These are most prevalent when there are multiple explanations of the data corresponding to different parameter values. Figure 3.2 shows an example of this. Although one maximum will have larger value of \( L \), either set of hyperparameters could be plausible depending on prior information about the form of the signal. In applications where this information is available, it may be possible to avoid bad maxima by simply starting optimisation in the region of a good maximum for example by choosing an appropriate starting value of \( \sigma_n \). Otherwise, random starting points can be used and the solution with largest \( L \) can be chosen.
Figure 3.2: Two differing explanations of data given by two sets of hyperparameters with a zero mean, SE covariance GP. (a) is a contour plot of the log likelihood when varying $l$ and $\sigma_n$. (b) corresponds to the maximum with shorter lengthscale and smaller noise strength. (c) corresponds to the case when most of the short term variation is accounted for by noise. Figure taken from [52].

This ambiguity can be solved if more data is available. This is easily seen by considering Figure 3.2, where the addition of data at previously unobserved input values would demonstrate whether noise is predominantly responsible for variation in the $y$ values.
3.2 Modelling Pedestrian Trajectories with GPs

Visual surveillance systems often observe scenes through which pedestrians follow common motion patterns. We now propose a new scene representation allowing a generative model of motion given a set of observed trajectories. Previous approaches, reviewed in section 2.1.1, have used parametric methods such as spline fitting to model common pedestrian paths. However, such models are often too restrictive since the trajectories of actors are inherently stochastic, with varying degrees of uncertainty depending on factors such as physical scene structure, the presence of other people and the time of day. The use of Gaussian Processes (GPs) allows us to be explicit about such uncertainties and to adapt to the various complexities of different scenes and situations online.

Figure 3.3 shows a typical scene of interest. This scene has a number of entrance and exit points and various routes between them are commonly used, with some points being linked by more than one possible route. The aim is to describe the typical motion patterns using these already observed trajectories with a probabilistic model which can be used for a variety of

---

3 Thanks go to Makris et al. for allowing use of the trajectory database which appeared in [42]
purposes such as object tracking, semantic labelling of motions and anomaly detection. In this report we focus on long term motion prediction as a primary example application of the model.

We present a probabilistic approach that bridges the gap between the full joint pdf over trajectories estimated in Johnson and Hogg’s method [32] with the efficiency of the spline based models [2, 40]. At the core of the model is the use of Gaussian processes to estimate instantaneous velocity of an actor given its current position. We cluster the training trajectories based on the associated entry point into the scene and then build a separate model for each cluster. Therefore instead of estimating the joint probability distribution over object positions and instantaneous velocities as in [32], we estimate the conditional distribution over instantaneous velocities given the current position and cluster membership.

3.2.1 Modelling Clusters of Trajectories

We now focus on the problem of predicting changes in target position based on its current position and that it is following a previously identified cluster of similar trajectories (a path) through the scene. Trajectories are represented by a series of 2D-points \(x_t = [x_t, y_t]^T\) corresponding to measurements taken at discrete time steps \(t\). We wish to learn the prediction model \(f\) such that

\[
x_t = x_{t-1} + f(x_{t-1}) + \epsilon_t,
\]

where \(\epsilon\) is zero-mean white Gaussian noise. By treating the change in state \(\Delta x_t = x_t - x_{t-1}\) as the target variable and considering changes in \(x\) and \(y\) to be independent, we place a Gaussian process prior over \(f \sim GP(m(x), k(x, x'))\) leading to the approximate prediction model

\[
p(\Delta x_t|x_{t-1}, D_x) \approx N(GP_{\mu}(x_{t-1}, D_x), GP_{\Sigma}(x_{t-1}, D_x)).
\]

Here the dataset \(D_x = \{(x_t, y_t), \Delta x_{t+1}|t = 1, ..., n\}\) represents a set of trajectories containing \(n\) position and instantaneous velocity observations in total, and a similar but independent model is used to predict \(\Delta y\) using \(D_y\). The mean of the prior process is taken to be zero which is a reasonable assumption if predictions are only required at test points close to some input values in the training set. For the covariance function we use the squared exponential (SE) with an
The SE covariance function enforces smoothness since points that are close together in the input space have highly correlated outputs. This is intuitive for the current application where it would be expected that neighbouring points along a path tend to lead to movements in similar directions. $\Lambda$ is a diagonal matrix containing the length scale hyperparameters $l_x, l_y$. These dictate the rate at which the correlation decays as a function of distance between the inputs, and so quantify how close two points must be to have correlated responses. $\sigma_f$ determines the expected range of the output (velocities). The bias term $b$ has the effect of correlating all values to some degree, meaning that predictions well away from the data will tend to towards the average velocity rather than the value of $m(x)$ (zero in this case). Finally the noise term in equation 3.18 is incorporated into the model by the addition of $\sigma_n^2$ along the diagonal of $K$. This term allows for the variation between observed velocities for points that are close in the input space.

As an example, we consider a subset of the trajectories on the right hand side of Figure 3.3 where people enter the scene from the bottom right, move upwards, deviate either side of the lampposts and then continue upwards to the exit point. The actual data used to form $D_x$ and $D_y$ for the GP model is taken from the trajectories shown in Figure 3.4. The mean predictions for instantaneous velocity in Figure 3.4 follow the direction of the path even in the presence of noisy trajectories. More importantly, the uncertainty in the predicted velocity increases further away from the observed data. Notice also how the mean predictions are in a slightly upwards direction in these areas of high uncertainty. This is due to the bias parameter $b$, the value of which has been inferred, as with the rest of the hyperparameters, from the data.

**Learning the Hyperparameters**

Setting the values of the hyperparameters in the covariance function (equation 3.20) is a way of expressing some prior information about the expected form of the motion model. However it would be inconvenient to have to tune these parameters by hand for each scene being mod-
Figure 3.4: Motion model for the trajectory data shown by the solid lines. The arrows correspond to the mean predicted instantaneous velocity at the position of their tails. The background shading indicates the variance of the prediction from each point with darker shading indicating higher uncertainty.

elled. Instead, inferences about the hyperparameters are made from the observed data itself as described in section 3.1.2, page 27. The hyperparameters are collected into $\theta = \{\sigma_f, \Lambda, \sigma_n, b\}$. The maximum likelihood parameters $\hat{\theta}_{ml}$ are found by optimising 3.15 with respect to $\theta$ using iterative conjugate gradients [55].

For this particular application, we found that local maxima could be avoided by starting the optimisation with reasonable initial length scales, $\Lambda$. For example starting with a length scale of one metre when the measurement space is on the ground plane produced good estimates for $\hat{\theta}_{ml}$, irrespective of the choice of the remaining hyperparameters. This step is inevitable since GP regression is not suitable for learning a model without expressing any prior information at all.
3.2.2 Long Term Prediction

Given a current estimate of the pdf of the target position \( p(x_t) \), an estimate for the target position at the next time step \( x_{t+1} \) is

\[
p(x_{t+1}) = \int p(x_{t+1} | x_t) p(x_t) \, dx_t.
\] (3.21)

The prediction model used is calculated by GP regression as described in the previous section

\[
p(x_{t+1} | x_t) = N(x_t + GP_\mu(x_t, D), GP_\Sigma(x_t, D)).
\] (3.22)

To obtain a long term prediction of the position the above integral can be evaluated recursively. However, remembering that \( GP_\mu \) and \( GP_\Sigma \) are functions of \( x_t \) through the covariance function (see equation 3.11) then it can be seen that the integral has no analytic solution unless \( x_t \) is known exactly. In this section two possible approximations are discussed which allow long term prediction.

Using GP-Bayesfilters Approximations

If it is assumed that \( p(x) \) is Gaussian at all time steps then prediction can be performed using a Kalman filter. Ko and Fox [34] introduced the GP-EKF, an extended Kalman filter which uses Gaussian Process based prediction and measurement models. In the prediction step, the predicted mean \( \hat{\mu}_{t+1} \) is given directly by the GP mean function

\[
\hat{\mu}_{t+1} = GP_\mu(\hat{\mu}_t, D).
\] (3.23)

The additive process noise is given by the variance of the GP prediction

\[
Q_{t+1} = GP_\Sigma(\hat{\mu}_t, D).
\] (3.24)
Figure 3.5: (a) Prediction using a GP-EKF for a target moving from bottom right upwards. The ellipses show the $2\sigma$ bounds on the predicted position at successive time steps. The solid lines are a sample from the dataset used to train the scene model and are intended to show the possible paths to be taken. The red path is a trajectory from a test set, the starting point of which was used as the initialisation for the prediction. (b) Distribution of particles at four snapshots in time when starting from the estimate shown in the bottom right.

To estimate the predicted state covariance the Jacobian of the GP mean prediction with respect to the state is required

$$G_{t+1} = \frac{\partial \text{GP}_\mu(\hat{\mu}_t, \mathcal{D})}{\partial \mathbf{x}_t}$$

which can then be used to propagate the state covariance through the plant model

$$\hat{\Sigma}_{t+1} = G_{t+1} \hat{\Sigma}_t G_{t+1}^T + Q_{t+1}$$

Repeated application of equations 3.23 and 3.26 allows an estimate of target position to be calculated many time steps into the future. Figure 3.5a shows the propagation of the state through the motion model depicted in figure 3.4 with a starting point in the bottom right corner. This shows the covariance ellipses correctly distorting to express the uncertainty caused by changes in direction of the path. However, the use of a Gaussian state estimate is clearly too restrictive when paths diverge and take different routes since the unimodal estimate can only predict one of the possible tracks after a junction.
**Sequential Monte-Carlo Prediction**

The GP scene model we have described is capable of describing complex paths which can diverge and even reconverge, unlike other methods which separate diverging paths into smaller elements. To use the model effectively for prediction we must allow a multi-modal estimate of the position to be maintained. We use a sampling approach where the distribution $p(x_t)$ is represented by a set of particles. At each time step the distribution is updated by sampling $p(\Delta x_{t+1}|x_t)$, provided by the GP scene model, for each particle. This is effectively the prediction step of a particle filter.

Figure 3.5b shows how a distribution of 250 particles evolve over a number of time steps starting with a Gaussian estimate shown in the bottom right corner. The next distribution shows $p(x)$ as the target reaches the junction. In the next step a bimodal distribution begins to form as there are two possible hypotheses as to which route could be taken. Particles in the region between the two tracks disperse quickly because of the high predictive uncertainty in this area where no observations have been made. In the last snapshot shown there are clearly two modes to the distribution with a low probability of the target lying between the two tracks. Notice how also there is a fairly high variance in the direction of motion on the left hand track which is due to the variability in the time taken by pedestrians traversing the corner. This effect is seen again in the analysis of a more complex dataset in the next section.

### 3.2.3 Implementation

The previous section used the toy example of bifurcating trajectories extracted from a larger dataset to illustrate the potential power of the GP motion model. We now attempt to model a whole dataset extracted by an overhead camera covering a scene with many entrances and exits in the atrium of a building. Figure 3.6 shows the dataset under consideration. Since many of the paths cross in the central area, conventional tracking methods often lose targets when many actors are present. Long term prediction can therefore be used to aid in target reacquisition.

Firstly, the trajectories are grouped by start point using a simple mean shift clustering method[13], leading to the cluster assignment shown in Figure 3.6. The reason for clustering by start point is that the current GP model can only give a unimodal estimate for $p(x_{t+1}|x_t)$ (see
equation 3.22), and so can not model crossing trajectories. Clustering by start point exploits the fact that people tend not to walk back on themselves to provide a fairly general model, without introducing the problem of multi-modalities in predictions from a single point.

The trajectories are then subsampled such that position measurements at around 4Hz are used for the GP training data. This is to limit the number of data points used for prediction to around 1000, otherwise calculation of the predictive variance, which has complexity $O(n^2)$ in the number of data points, becomes computationally too expensive. For long term prediction this is not an issue since much of the discarded data provides little extra useful information. It is much more useful to have a larger sample of trajectories, rather than more points per trajectory, to cover the data space as well as possible with a representative set of common paths.

The hyperparameters for each cluster are then learned from the corresponding data sets. Following this, the model is ready to be used for prediction. As an example, consider a target which has been observed entering the scene on the left hand side of figure 3.6. The GP motion model for this cluster of trajectories is shown in Figure 3.7a, and long term prediction with 250 particles is shown in figure 3.7b. Notice how the distribution spreads to cover all three possibilities. The sharp left corner has the effect of spreading the estimate along the direction of the path due to differences in the time taken for targets to traverse the corner in the observed data. Although there are some particles in the area to the right of the corner, these are actually fairly sparsely distributed compared to the areas around the modes.

**On-line Adaptation**

At present each model can only store a few thousand data points due to the quadratic complexity in sampling the long term predictions. By only using the most recently acquired trajectories as a prediction dataset, the model can adapt to changes in the environment. Re-clustering is inexpensive and so can be carried out online while the recomputation of $K^{-1}$ requires one $O(n^3)$ operation when the dataset changes. The re-learning of hyper-parameters is slower, requiring recomputation of $K^{-1}$ at each optimisation step and is more suited to batch processing offline.

This is not a problem, since we would not expect the general properties of the trajectories controlled by the hyper-parameters to change for a given scene. However, actual deviations in
3.2.4 Evaluation

To evaluate the accuracy of the long term predictions made via the Monte Carlo sampling method, we performed leave-one-out cross-validation. For each of the 54 trajectories in the training set shown in Figure 3.7a a motion model is trained, excluding a trajectory from the training set. Long term prediction with 250 particles is used to predict the path after the \( t = 5 \) point along this trajectory. The observation space is divided into a grid of \( 0.5 \text{m}^2 \) cells. At each step \( N \) of prediction the fraction of particles residing in the correct cell is recorded. A standard constant velocity model Kalman filter [3], with the same parameters as used for collecting the trajectory data, is also left to perform open loop prediction. The predicted state estimate is integrated over the correct cell for direct comparison with the correct particle fraction.

Figure 3.8 shows the average value of the probability of being correct over all the trials. The Kalman filter predictions quickly become too vague as the prediction covariance rapidly
Figure 3.7: (a) Motion models for trajectories starting on the left of figure 3.6. As in Figure 3.4, note how the bias $b$ in equation 3.20 lets distant points assume average velocities. (b) Long term prediction showing the particle distribution estimate of $p(x_t)$ for three snap shots. 

increases without any new measurements. The GP model makes use of the previously observed trajectories to make a more informative prediction, leading to a higher probability being assigned to the correct cell. Even after 30 time steps (7.5s), on average 10% of particles still lie within the correct cell. However, the Kalman filter estimate becomes less informative than a uniform distribution over the area under surveillance.

### 3.2.5 Extending the Model

Analysis of individual trials in the experiments above showed that although long term predictions modelled uncertainty in change of directions well, variations between the speeds of targets caused large errors with the predictive cloud of particles falling ahead or behind of the true position. To account for varying velocities explicitly, rather than implicitly through the hyperparameters, the model can be extended to include information about the current velocity of the target to make predictions. The prediction model becomes

$$
\Delta x_t = f(x_{t-1}, \Delta x_{t-1}) + \epsilon_t. 
$$

(3.27)
Figure 3.8: Comparison of long term prediction performance of the GP scene model considering position only (red crosses), including velocity in the state vector (green dots), and a Kalman Filter with constant velocity dynamics modelled (dashed). $q$ is the cumulative distribution over the correct cell where the target actually resides, and the value shown is averaged over all trials.

We again place a GP prior on $f \sim \mathcal{GP}(m(z), k(z, z'))$ where the input to the process is now a 4-dimensional vector $z_t = [x_t, y_t, \Delta x_{t-1}, \Delta y_{t-1}]^T$. This use of GPs to estimate dynamical models is similar to that of Wang et al. [65], who suggested incorporating higher order features rather than making a first order Markov assumption about the dynamics. The approximate prediction model takes the form

$$p(\Delta x_t | x_{t-1}, \Delta x_{t-1}, D_x) \approx \mathcal{N}(\text{GP}_\mu(z_{t-1}, D_x), \text{GP}_\Sigma(z_{t-1}, D_x)),$$

with a similar expression for $\Delta y_t$. The amalgamation of current velocity into the training input requires little extra computation as the size of the covariance matrix $K$ is unaffected. The extra cost in evaluating each element of $K$ is negligible for the squared exponential covariance function, requiring only computation of the weighted squared distance in the higher dimension. The hyperparameter $\Lambda$ remains diagonal, with two extra elements controlling scaling in the velocity dimensions, and the hyperparameter learning proceeds as before. The Monte Carlo sampling approach is also easily implemented since a the newly sampled values of $\Delta x_t$ can be
used in the next sampling step along with the newly calculated position.

The same experiment described in section 3.2.4 was performed with the extended model, and the evaluation of the accuracy of long term prediction is shown in Figure 3.8. As expected, predicting future position based on both target position and velocity consistently provides more reliable estimates than the original model. The effect is particularly pronounced in the early stages of loss of tracking during the first 15 time steps of prediction, since this is where the inclusion of the initial velocity is most relevant. The higher variance on these results could in part because of a lack of training data. However, this could perhaps be an indication of overfitting of the hyperparameters, and this deserves further investigation.

3.2.6 Conclusion

In this chapter we have presented a novel method for modelling common pedestrian motions through a scene by regressing relative motion against current position. We have given examples of how the model can be used to perform long term target position prediction using data from two different datasets. The key benefit comes through the use of Gaussian processes to explicitly model uncertainty in predictions such that the characteristic unpredictability in human motion can be accurately represented. Two methods of prediction were investigated, the first making use of a GP-EKF to maintain a Gaussian estimate for the distribution over target position, and the second using a Monte Carlo sampling approach allowing a more flexible multimodal estimate. The advantages of the GP model over a standard Kalman filtering approach were demonstrated by comparing the accuracy of long term predictions on a dataset where pedestrians take diverging routes. An extension to the model was considered which utilises information about current velocity. Experiments showed that this increased the accuracy of long term predictions.
Chapter 4

Conclusions and Future Work

4.1 Summary

We have developed a novel representation of pedestrian motion through scenes which can be learned in an unsupervised fashion using a trajectory history provided by a visual surveillance system. The capability of this model to be used for long term predictions of target motion has been demonstrated using two methods. It was shown that using a Monte Carlo sampling approach for prediction carries the significant advantage that a multimodal distribution of the state estimate can be maintained, rather than the restrictive Gaussian state estimate when using a Kalman filter. However, the sampling approach is potentially more computationally expensive, and the cost of prediction per particle must carefully considered in any extensions to the model.

The remainder of this section details the work to be carried out in the remaining 2.5 years of the DPhil. We begin by exploring immediate extensions to the trajectory model discussed in the previous chapter with focus on applications in visual surveillance. In particular we examine the use of long term predictions to perform an active search for a target in an intelligent manner. We then discuss learning models of different clusters of trajectories to then be used to classify and label motion patterns. This is followed by a proposal of broader work which sees the combination of advances in mobile vision, visual SLAM, pedestrian detection and pedestrian motion modelling to perform high level reasoning about dynamic targets.
4.2 GP Trajectory Modelling Extensions and Applications

In this section we begin by identifying some possible weaknesses of the GP trajectory representation which may cause problems in certain applications and propose solutions to these issues. We then detail two key applications for which we wish to use the model. The first is the use of the particle distributions obtained during long term prediction to perform an active search for the target based on an information theoretic framework. The second involves using different path models (from different clusters of trajectories) to classify paths taken by targets and produce a semantic description of specific traffic flow through the scene.

4.2.1 Model Extensions

Limits on Training Data Size

In section 3.2 it was acknowledged that, due to the $O(n^2)$ cost of prediction for each particle, it was only feasible for the system to predict in real time for a few thousand training points (position velocity pairs). This computational intractability of GP models for large datasets has led to the development of algorithms to reduce this complexity, and these are often categorised as either global or local approximations.

Global approximations are based on using a small set of $m$ support points or ‘inducing inputs’ [50], $m << n$, to provide a sparse summary of the $n$ data points, reducing the cost per training step to $O(nm^2)$ and prediction to $O(m^2)$. The inducing inputs can be a subset of the data in a similar fashion to a support vector machine, or they can be placed in arbitrary locations which are often chosen to maximise the marginal likelihood during the same optimisation as when choosing hyperparameters. Global approximations are only accurate when the process lengthscale is large relative to the sample density[57], i.e. for cases where the data has been over sampled. Also, the re-learning of the positions of inducing inputs is costly and needs to be performed whenever new training data arrives.

Local approximations [57] exploit stationary covariance functions for which correlation decreases between more distant points such as the squared exponential. As the covariance between a test and training point tends towards zero, the training point has less influence on the final
prediction, and if the influence becomes negligible then the training point can be ignored completely. Thus only a small local subset of \( m \) training points are used with the cost of prediction again being \( \mathcal{O}(m^2) \). Algorithms where the subsets of points are defined at training time\[51\] are referred to as offline. For our scene model, an implementation would be to divide the ground plane into cells, each of which has its own GP and set of training points for prediction. This introduces problems of discontinuities at the cell boundaries.

An alternative proposed by Urtasun and Darrell \[62\] uses the \( M \) nearest neighbours to the test point such that discontinuities are less pronounced. This ‘online’ local approximation requires that \( K^{-1} \in \mathbb{R}^{M \times M} \) is computed for each test point, although \( M \) is often chosen to be sufficiently small such that this is not prohibitive. This approach also has the capability of modelling multi-modal outputs with a mixture of GPs. The \( N \) nearest neighbours to the input point each assign a GP prediction using their \( M \) nearest neighbours in the output space. The resulting prediction at \( x_* \) is then a mixture of Gaussians

\[
p(f_*|f, X) = \sum_{n=1}^{N} \pi_n \mathcal{N}(GP_{\mu}(x_*, D_n), GP_{\Sigma}(x_*, D_n)).
\] (4.1)

\( D_n \) is comprised of the \( M \) nearest points to \( f(x_n) \) in the output space. The mixing weights \( \pi_i \) are set to be a function of the inverse variance.

Figure 4.1 shows our implementation of the algorithm, showing a bimodal predictive distribution at \( x = 0.5 \). The adaptation of this algorithm to the GP scene model is relatively straightforward, and will allow more training data to be handled with the added benefit of the ability to represent multi-modalities, for example scenes where people are likely to move in more than one direction at a single position.

### 4.2.2 Active, Intelligent Re-acquisition

A major weakness of memoryless surveillance systems is that they operate in a state of continual surprise. They are heavily dependent upon persistent tracking since the loss of a tracked target is non-recoverable, and any following acquisitions of targets are all equally surprising, as far as the system is concerned, regardless of any previous activity that had been observed. This reliance
Figure 4.1: Local online approximation for regression using the datapoints shown by crosses. The prediction distribution $p(y|x = 0.5)$ is shown by the red curve. Here $N = 5$ local experts were used with each comprising of $M = 30$ neighbours.

on continued accurate tracking leads to inefficient use of active cameras because all resources are focused on tracking separate individuals, rather than observing the most informative aspects of the scene. A ‘more informative’ view could require zooming out to observe wider scale traffic in the environment, or zooming in on a target to achieve a high resolution image for facial identification. In either case, tracking of one individual may have to be sacrificed for the greater good of the system’s overall knowledge.

Sommerlade and Reid [58, 59] have approached this problem from an information theoretic viewpoint where an active camera’s attention is directed so as to minimise the conditional entropy of the scene model. The memory element of their system is the learning of rate of appearance of targets at different image locations, and the information gain of scene exploration (for example by zooming) is offset by the possible loss of information by failing to observe the appearance of a new target.

We propose to use the GP trajectory model and its long term predictions as a robust method of re-acquiring targets in a framework also based on information theoretic grounds. A key quantity required will be the Shannon entropy [54] of the target position,

$$H(x) = - \int p(x) \log p(x) dx. \quad (4.2)$$
A variety of methods exist to estimate the entropy of a distribution from a set of independent samples [4], which would be required when using the Monte Carlo method of prediction. Initial experiments show the Kozachenko and Leonenko estimate [35] based on nearest neighbours to be particularly efficient. It is envisaged that the entropy of the predicted distribution will be an indicator as to when the motion of the target has become uncertain, for example when reaching a junction. Therefore a system could automatically recognise that there is little information to be gained by tracking a target in the short term before it reaches the junction and resources can be focused elsewhere. The camera could then re-focus on the target at the time of the forseen ambiguity, and then determine which route was taken at the junction. Note also how if the target is not present in the expected area at this time then this result is significant as being anomalous, compared to a memoryless system.

Questions will also be raised as to the best strategy to search this space once a decision to do so has been made. This will relate to the work of Davidson [18], who presents a framework to analyse the value of making a measurement based on the mutual information, \( I(x, y) \) between the object state \( x \) and measurement \( y \), where

\[
I(x, y) = \int p(x, y) \log \frac{p(x|y)}{p(x)} \, dx \, dy. \tag{4.3}
\]

Davidson also introduces a cost in taking each measurement and balances this against the potential gain in information. A similar metric shall be particularly important in surveillance with slow maneuvering PTZ cameras.

This is the major application of the trajectory model which we wish to target, and the work in chapter 3 has shown that the framework for long term prediction is in place. It is hoped that this work will form the basis of a conference paper to be produced in the next 6 months.

### 4.2.3 Semantic Path Classification

With multicamera surveillance systems producing such large quantities of video data it is natural to consider some form of temporal compression such that footage can be reviewed quickly. Methods such as [49] attempt to provide a synopsis of a long video with a shorter one, which
can be somewhat confusing to interpret, especially for a non trained operative. We propose a system where natural language descriptions are given of scene activity as suggested, but not demonstrated in [41]. For example, the system operating in the IEB atrium could produce outputs such as “Person id:51 entered the Atrium via the bridge, proceeded slowly, and left via the downward stairs”. One principle novelty would be the combination with snapshots of image data, for example a close up image of the person’s face acquired from a PTZ camera.

In the modelling described in chapter 3 a convenient choice of dataset for GP prediction was obtained by clustering by start point. However, by clustering in different ways the trajectory history can be separated into groups which represent different properties of the observed motions. By clustering by each pair of start and end points, and creating a GP model for each pair, trajectories could be classified to give outputs as above. Furthermore, it is possible that by using hierarchichal clustering, different levels of semantic labelling will be able to be presented, from general patterns (“people tended to walk down the stairs today”) to specific incidents (“person x suspiciously stopped at position y”).

Existing techniques for path classification are often based on calculating a geometric distance between the observed trajectory and a prototype [56]. For the GP model, we can compare directly likelihoods of the observed data given each different model. We will therefore be able to provide uncertainties in the classification given, which may also prove useful in identifying anomalies where no previously defined models seem to fit the behaviour. Another advantage is that a complete trajectory observation is not required for classification, and so predictions about which exit may be taken, for example, could be updated online as a target is tracked.

The required infrastructure for learning scene models is already in place, and only the classification methods need to be developed upon. Therefore, semantic path classification is a relatively short project in comparison to active re-acquisition. However, it will usefully offer a direct method of comparison with other scene models where similar applications have been tested.
4.3 Moving Platforms

We now move onto discussing longer term directions for the DPhil which build upon work in visual surveillance and suggest applications where a mobile camera is the main sensor of interest.

4.3.1 SLAM and Visual Surveillance

Visual surveillance and visual SLAM systems take two very different approaches to the same problem of ascertaining the state of the world. Since a SLAM system often begins in a state of greater ignorance, the focus is on mapping a static environment, whereas surveillance cameras are mainly concerned with discovering the presence of dynamic objects such as people and monitoring their activity. Collaborative surveillance involves the use of multiple sensors, both fixed and mobile, to explore an environment while communicating between themselves to increase the efficiency or yield of the search. This would inevitably require the combination of SLAM and surveillance techniques.

The first example of collaborative surveillance involving mobile sensors was demonstrated a decade ago by Diehl et al. [21], where the focus was on a vision system for detecting people. Collaborative surveillance has recently become a particular area of interest in the defence industry, where the goal is to explore an unknown environment and identify threats using cooperating autonomous ground and air vehicles. The Ministry of Defence “Grand Challenge” competition [44], showcased the current state of the art, with a variety of vehicles exploring an artificial village. We propose cooperative surveillance of a smaller area, such as the information engineering building atrium, with the mobile sensor being a wearable camera attached to a human assistant. In this simpler situation the surveillance system already in place at Oxford [5] provides an ideal platform for development.

One particularly important benefit of using a mobile visual SLAM system is that it will allow mapping of the physical structure of the scene under surveillance. In our GP motion model described so far, the presence of obstructions is handled only implicitly. Since there are no observed trajectories through an obstruction, the model assigns a high uncertainty to the predicted motion through these areas, and in the case of Monte Carlo predictions particles
diffuse quickly and leave a low probability of the target being situated on the obstruction. This is an acceptable way for the current model to express its ignorance about what may happen where there has been no observed motion. However, if information about the scene structure is also available then this can be folded in to the prediction model. For example, if the SLAM system detects the presence of a wall during mapping, then the Gaussian distribution over predicted motion should be truncated at this boundary. Of course there may also be uncertainty in the mapping itself, but since the GP model is fully probabilistic this can be integrated into the final prediction proposal.

Another area of interest is the representation of dynamic objects in the SLAM system. Bibby and Reid [9], have already suggested a method using cubic splines. It is important that the parameterisation of dynamic objects in a SLAM system is compact, to avoid adding complexity to the state estimation procedure. Bibby prunes knots from the splines heuristically to keep the representation compact. The positions of the knots are estimated using a Kalman filter, and their representation is in fact a Gaussian process. It would be interesting to formalise this, especially with respect to choosing a principled sparsification method.

4.3.2 Open Environments

In section 2.2.2 we argued that detection and tracking of pedestrians from moving platforms has reached a level of maturity that should allow higher level analysis of pedestrian behaviour to be performed. We now suggest the use of a single mobile camera to detect and track pedestrians in large and open urban environments. This area of research is very active, being particularly focused on automotive driver assistance systems [28]. However, even state of the art systems [29] do not attempt to model high level pedestrian behaviour, but focus instead on identifying pedestrian velocities to predict possible collisions.

Relative to a car moving along a road, the motion of pedestrians along the pavement is fairly restricted. This immediately presents an opportunity for use of our GP motion model, which could be used to classify a pedestrian’s motion as either fitting a model where they walk along a pavement, or being a deviation from the expected path. We would however also like to investigate more general motion, and the inferences which can be made about future behaviour
by incorporating other visual cues into the system. For example, the gaze detection system of Benfold and Reid [7], could be used to determine whether pedestrians have seen the moving platform, or to determine future intentions (e.g. looking over the shoulder before crossing a road).

We are interested in the general ability of autonomous navigation through a pedestrian dense environment (such as Cornmarket street) as well as road safety, and so would like to collect some data to aid in working on the problem. We plan to mount a wearable stereo camera while cycling through Oxford to provide a dataset with pedestrians in close proximity to the moving platform. It is hoped that such challenging datasets such as this may also present the opportunity to model interactions between pedestrians in close proximity to one another.

### 4.4 Plan

We have stated short term goals of applications of our GP motion model in visual surveillance which it is envisaged will take approximately the next 9 months. Active re-acquisition is the more ambitious of the two proposed ideas, but provides the biggest scope for novelty. Semantic labelling is a smaller task but with potentially lower risk. We have also outlined two major directions in terms of long term future work. Both are concerned with the use of SLAM and moving cameras and there is some degree of overlap. The work in visual surveillance will require some collaboration with others who are already working on the system, and thus carries slightly higher risk than the open environments proposal, which is solely personal work for which initial experiments have already begun (Figure 2.6). Our plan for future work is summarised in the chart in figure 4.2.

### 4.5 Acknowledgements

Thanks go to Eric Sommerlade for his assistance in the work in [23], and to Dr Ian Reid for his helpful discussions throughout the year.
<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td>Q1</td>
<td>Intelligent Re-Acquisition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Entropy estimation</td>
</tr>
<tr>
<td>2011</td>
<td>Q2</td>
<td>Planning prototype</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Semantic Classification</td>
</tr>
<tr>
<td></td>
<td>Q3</td>
<td>Planning Implementation</td>
</tr>
<tr>
<td></td>
<td>Q4</td>
<td>Model Extensions</td>
</tr>
<tr>
<td>2010</td>
<td>Q1</td>
<td>Faster Implementation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SLAM and Visual Surveillance</td>
</tr>
<tr>
<td></td>
<td>Q2</td>
<td>Data collection</td>
</tr>
<tr>
<td></td>
<td>Q3</td>
<td>Driver Assistance</td>
</tr>
<tr>
<td></td>
<td>Q4</td>
<td>Write thesis</td>
</tr>
<tr>
<td>2009</td>
<td>Q1</td>
<td>Write thesis</td>
</tr>
</tbody>
</table>

Figure 4.2: Plan of future work
Appendix A

Courses and Seminars Attended

- Scientific computing for DPhil students, lectures and assignments, by Prof. Nick Trefethen, Michaelmas 2008.

- C4b Machine Learning, lectures by Andrew Zisserman, Hilary 2009.

- Managing your DPhil, training course, Michaelmas 2008.


I have regularly attended the Robotics Research Group seminars, and reading groups for the Active Vision group. I have also demonstrated in the 2nd year C++ coursework module, and given 3rd year tutorials on communications electronics.
Bibliography


