Tree-Structured Graphical Models and Image Analysis

Stephen N. Felderhof

Doctor of Philosophy
Division of Informatics
University of Edinburgh
2003

Abstract

The thesis examines tree-structured graphical models within the domain of image and image sequence analysis. It looks at three issues: firstly, a comparison of generative and discriminative approaches to the use of tree-structured belief networks for static image segmentation; secondly, an approach to image sequence coding and segmentation in which information is incorporated in a temporal manner using an extension to the dynamic tree model of Williams and Adams (1999) and Storkey and Williams (2002); and finally, an investigation of the products of experts architecture in which each individual expert is a tree-structured Gaussian process.

With regard to the first problem, in Bayesian image analysis the problem of image segmentation involves each pixel in an image being assigned one of a predefined finite number of labels. This requires the fusion of local predictions for the class labels with a prior model for the label images. A number of authors have taken a generative approach using Gaussian mixture models to model the probability of the image given the labels thereby connecting the label field to the image data. This is compared to the discriminative approach in which a multilayer perceptron (MLP) is used to approximate the probability of the pixel classification given the image data, and the scaled-likelihood method is then used to fuse this with the prior label model. An evaluation of the classification results obtained is presented. Both the *maximum a posteriori* segmentation, and also the uncertainty, as evidenced e.g. by the pixelwise posterior marginal entropies, are examined. The results show that the discriminative approach performs better.

The second issue addresses the proposition that given a temporal sequence of images, having information about (or having performed inference for) previous frames should provide useful knowledge about the current frame. Position encoding dynamic trees (PEDT) have been demonstrated to be useful models for individual images. Here they are extended so as to allow them to model image sequences. In the theory and experiments presented, it is shown that the PEDT model can be used for the analysis of ordered sequences of images. A theoretical framework is provided within which the current prior model is updated sequentially using the variational approximation to the posterior probability distribution from the previous time frame. Experimental results are presented demonstrating improved coding and segmentation performance. It is also shown that substantial improvements in image coding cost can be achieved by coding the changes in the highest variational posterior probability tree structures between frames using a method presented.

Thirdly, the thesis looks at the products of experts (PoE) model which was introduced by Hinton (1999). It is examined using a PoE model in which each expert is a Gaussian. This gives rise to a product model that is also Gaussian. However, if each expert is constrained to be a tree structured Gaussian process (TSGP) the product of these then has a more complex structure than the individual trees. By way of comparison, a framework within which the resultant process is constructed from the *sum* of tree-structured Gaussian processes is also considered; the result of this method is also a Gaussian process. There is an investigation of the approximation of various target stationary processes with these Product of Experts and Sum of Experts models. The results show that the preferred choice between the two models depends on the type of target process. We also show that for AR(1) and MA(1) target processes, an exact representation of these processes using only two component TSGPs can be found.

Acknowledgements

I am endebted to Dr. Chris Williams who has been a fantastic supervisor. Many thanks are due for all the inspiration, motivation and gentle encouragement to persevere with the tasks in hand. If the work has any merits then they will be in no small part due to his efforts and leadership. Of course, any weaknesses are solely my responsibility and I apologise for anything which fails to live up to his high standards.

Dr. Amos Storkey has given me invaluable help. The discussions, ideas, support and programming problem solving have been vital. The work for Chapter 3 could not have been completed without the use of his DPT code and his help with its further development. Cheers Amos, I owe you many pints of beer.

Thanks also to Dr. Xiaojuan Feng for the useful help at the start of the project. The use of her CBN code was essential to work for the Chapter 2.

The work presented here was supported by EPSRC project grant GR/L78161 *Probabilistic Models for Sequences*. I thank EPSRC for funding my study and thanks also to British Aerospace for the additional financial contributions. In particular, thanks to Dr. Andy Wright and Dr. Gareth Rees of the Advanced Image Processing Department at the Advanced Technology Centre - Sowerby, BAE for useful discussions.

My parents have not only provided support and encouragement but also free room and board during my impecious final period of study, I am very grateful.

And finally, thanks to Birmingham and St. Andrews Universities which have on occasion allowed me to work remotely on the computers in Edinburgh via their generous, though albeit unwitting, provision of broadband internet access.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Stephen N. Felderhof*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image analysis, or computer vision, is a diverse field which addresses a wide range of practical problems in a number of domains of application. The overriding theme in image analysis could probably be characterised as an attempt to infer a description of a scene in terms of recognisable objects from some sort of visual representation of that scene. Given the typical nature of images, the desired description may well be hierarchical and describe how objects relate both to one another and to their constituent parts for instance in terms of their positions and orientation. Sensors can include devices such as: digital cameras, medical scanners such as X-ray machines or MRI machines, radar, and remote sensing devices.

To focus more closely, specific challenges include [67]:

- Texture analysis in which the statistics of regions of an image are measured. For instance, shading and changes in coarseness with range have been shown to help the determination of the shapes of objects. See for example, [34, 85].

- Texture synthesis which deals with the generation of different types of texture found in natural images, [18, 30, 87].

- Motion analysis in which the motion between frames of an image sequence are measured using calculations such as optical flow, [35].

- Surface orientation estimation in which the 3D shape is inferred from the 2D image using information such as the edges in images and the boundary of surfaces, e.g. [38].

- Stereo matching in which correspondences are made between multiple images

of the same scene viewed at different angles which allows the range of objects in the images to be estimated, see for instance [5].

- Segmentation in which an image is divided into regions, the cues on which this operation can be based include a number of the above sources of information, examples include [58, 20].

These methods often take an engineering approach in which the main aim is to provide practical solutions to the problem at issue.

On the other hand, probabilistic models for images are interesting because they provide a principled framework within which to perform image analysis. Two typical statistical pattern recognition tasks that we can undertake within a probabilistic framework are: image segmentation where each pixel is assigned one of a predefined finite number of class labels; and image coding where we can use the probability distribution over images to define an image coding scheme.

The advantage of a probabilistic approach is that it gives us a theoretical understanding of the calculations that are being performed and it provides quantitative measurements of performance. For instance with regard to segmentation, we do not have to rely on *ad hoc* segmentation heuristics but instead we can treat the problem as the calculation of the *maximum a posteriori* (MAP) image labelling. Needless to say, a fully Bayesian approach would eschew recourse to quantities such as maximum likelihood parameters and MAP solutions preferring instead to deal only with the probability distributions themselves. However, sometimes practicality must prevail especially when dealing with large problems such as realistic images and image sequences. Moreover, MAP configurations of variables can be useful for visualisation purposes. That said, a further advantage of employing probabilistic models is that not only can we make predictions but we can also understand the certainty or uncertainty with which those predictions are made.

The use of graphical models with discrete variables as a model for the prior probability distributions over image labels requires the fusion of local predictions for the class labels with the prior over the label field. This approach allows the tasks of recognition and segmentation to be performed simultaneously. Some recognition systems, in particular those concerned with handwritten character recognition, although demonstrating impressive performance, have often relied on objects being separately segmented and processed into feature vectors before classification is carried out (for

instance, see [44]). This ignores the fact that in more general problems each of these two steps is heavily reliant on the other. It is difficult to divide an image into regions representing individual objects without first recognising what the image contains. Similarly the process of recognising objects within an image will be affected by the context in which those objects occur.

Much work has been done employing Markov random fields (MRF) as image models (e.g. [11, 12]). In MRFs, correlations in the label field are introduced through nearest neighbour interactions between nodes in the graph. Although unfortunately calculations with this type of model involve an intractable partition function, approximate solutions can be found by adopting Gibbs sampling [25] or variational [61] methods. However a serious problem with this type of approach is that the model does not represent the fact that the structure of images is intrinsically hierarchical. As Marr [52] points out, how finegrained should a description of what constitutes an object in an image be? It is apparent that any description of the structure of an image should analyse the image over a range of lengthscales and so describe objects in the image as being composed of sub-objects in which turn themselves can be described in terms of their constituent parts. This motivates tree structures as a natural framework in which to consider the structure of images. This thesis will address three issues arising from the use of tree-structured graphical models as image and image sequence models. Chapter 2 will focus on fixed tree structures and will consider the tree-structured belief network as a model for static images. Chapter 3 will look at flexible tree structures as improved image models and consider their applicability to image sequences. Chapter 4 will consider the products of experts architecture as an alternative way to improve the representative power of tree-structured graphical models.

There has been some work using fixed tree-structured graphical models as models for images, e.g. [9, 50, 36, 24]. Fixed tree-structured belief networks have the pleasing property that inference can performed in a time which scales linearly with the number of nodes in the graph using the Pearl message-passing algorithms, [62]. **Chapter 2** makes a comparison of generative and discriminative approaches to the fusion of local image model predictions with a tree-structured belief network as a prior model for the image labels. A number of authors (for instance, see [9]) have taken a generative approach using Gaussian mixture models to model the probability of the image data given the labels thereby connecting the label field to the image data. This is compared to the discriminative approach in which we use a multi-layer perceptron (MLP) to

approximate the probability of the pixel classification labels given the image data, and then use the scaled-likelihood method to fuse this with the prior label model. An evaluation of the classification results obtained is presented. Both the *maximum a posteriori* segmentation, and also the uncertainty, as evidenced e.g. by the pixelwise posterior marginal entropies, are examined. The results show that the discriminative approach performs better.

An objection to the use of fixed tree-structured models to represent images is that the structure of model does not reflect the composition of the image. The fixed tree structure is the same for all images and as such the connectivity of graph does not represent the structure of the image. It cannot therefore be used as a parse tree for the image. (In this context, we are referring to a parse tree as a structure which is analogous to the tree-structured hierarchical decomposition of a sentence which is the product of grammatical parsing.) If the structure of the tree was reconfigurable, then we could infer a tree structure from the image which could then be used as a representation of the hierarchical composition of objects in the image. Furthermore, a practical implication of the use of a fixed quad-tree structured graphical model as the prior label model is that it results in higher probabilities being assigned to configurations of the image labels in which boundaries between regions in the image fall between the high-level blocks of the quad-tree structure. This causes the blocky artifacts that are observed in the image segmentations inferred using this prior model (as highlighted in Chapter 2).

The use of flexible tree-structured graphical models as alternative image models has been employed in the work on dynamic trees [1, 2, 72] and credibility networks [33]. These approaches define distributions over tree structures and show improved coding cost for images modelled with them. They are not troubled by the block boundary artifacts which affect the fixed tree image models. A price that has to be paid for this extra flexibility is that linear time inference is no longer possible and approximate solutions need to be found using either variational methods or sampling.

The position encoding dynamic tree (PEDT) [73] is an extension to the dynamic tree in which the nodes in the graph have variable positions relative to the positions of the pixels in the image. In **Chapter 3** we exploit this extra feature to deal with images which occur in temporal sequences. The position variable in the PEDT model allows associations between nodes in the tree structures for consecutive time frames to be made more easily which in turn allows objects to be tracked from one frame to the next in a more straightforward manner. Chapter 3 addresses the proposition that given

a temporal sequence of images, having information about (or having performed inference for) previous frames should provide useful knowledge about the current frame. Two approaches are considered.

Firstly, a method of generating a mapping between the HVPP[1] tree structures for two consecutive images is discussed. By coding some high-level information from this mapping, it is shown that substantial reductions in the image sequence coding cost can be achieved under the PEDT image model. This effectively codes the changes in the images between frames as high-level node movements in the HVPP tree structures. A further implication of this is that it demonstrates that the HVPP tree structure of an image in the PEDT framework is a useful way of representing the underlying structure in the image. In other words, we can regard it as a useful parse tree for the image. We make this assertion because when the mapping between time frames is made, the only information that we consider is the HVPP structure of the PEDT distribution for each frame. Using only this information, we can successfully code a mapping which reduces the image sequence coding cost. If the HVPP tree structure were not a useful representation of the underlying image structure this would not be possible.

The second approach considered in this chapter involves extending the PEDT framework itself so as to allow it to model image sequences. In the theory and experiments presented, it is shown that the PEDT model can be used for the analysis of ordered sequences of images. A theoretical framework is provided within which the current prior model is updated sequentially using the variational approximation to the posterior probability distribution from the previous time frame. Experimental results are presented demonstrating improved coding and segmentation performance.

Moving from fixed trees to flexible dynamic tree-structured models, is one method designed to improve the performance of the probabilistic models for images. Another method may be to move away from single tree representations of images and consider the combination of a number of individual models. **Chapter 4** presents an investigation of the products of experts architecture in which each individual expert is a tree-structured Gaussian process. The chapter looks at the products of experts (PoE) model which was introduced by Hinton (1999), [31]. The PoE model in which each expert is a Gaussian gives rise to a product that is also Gaussian. However, if each expert is constrained to be a tree-structured Gaussian process (TSGP) the product of

[1]The tree structure which has the highest probability under the variational approximation to the posterior probability distribution for the PEDT model, we term the *highest variational posterior probability* tree structure or HVPP tree (see Section 3.2 for more details).

these then has a more complex structure than the individual trees. By way of comparison, a framework within which the resultant process is constructed from the *sum* of tree-structured Gaussian processes is also considered; the result of this method is also a Gaussian process. There is an investigation of the approximation of various target stationary processes with these Product of Experts and Sum of Experts models. The results show that the preferred choice between the two models depends on the type of target process. We also show that for AR(1) and MA(1) target processes, an exact representation of these processes using only two component TSGPs can be found.

Finally, **Chapter 5** discusses the conclusions that we can draw from the work presented and suggests avenues for further research. Note that since each of the substantive chapters has a different theme, there will be no individual literature review chapter. However, the introduction and theory sections of each of the individual chapters will present a review of the related work relevant to that chapter.

Chapter 2

A Comparison of Discriminative and Generative Approaches to Image Segmentation with Tree-Structured Belief Networks

Initially we examine the application of fixed tree-structured belief networks (TSBNs) to image analysis. Specifically, in this chapter we describe an experiment to compare the performance of generative and discriminative approaches to the fusion of pixel image information with the spatial prior over image labels provided by the TSBN. Section 2.1 provides an introduction to the problem and an outline of the theory behind the calculations performed. In Section 2.2, there will be a detailed description of the experiments performed. The results of the experiments are presented in Section 2.3. The chapter concludes with a discussion of the results in Section 2.4.

This chapter is based on work presented in Feng, Williams and Felderhof (2001), [22]. The TSBN framework, including the ML and CML learning procedures, was developed by Drs. Feng and Williams. It is presented here to provide the background for the work carried out. The experiments involve the use of the `cbn` code library which was written by Dr. Feng. It is worth noting that the work in this chapter precedes and to some extent motivates the work on Dynamic Trees done by Williams and Adams (see the comments in Chapter 3) consequently, use of these methods is not made in this chapter. However, the next chapter *is* an extension of the Dynamic Tree framework.

2.1 Introduction

As discussed in Chapter 1, early work on Bayesian image modelling involved the use of Markov random fields. However, the fact that these models are noncausal and typically have a flat nonhierarchical structure combined with their associated high computational complexity, make them unattractive as practical images models especially as images themselves are intrinsically multiscale in nature.

The TSBN provides a generative model for the image labels in the form of a directed tree-structured graph. For the purposes of this chapter, the graph will take the form of a fixed tree structure in which all the nodes above the leaf layer have four children except the root node which will have six children in order to maintain the aspect ratio of the images. The hierarchical nature of this model allows for the straightforward encoding of the long-range correlations between leaf nodes that we might expect to find in image data. Inference in the TSBN can be performed in a time proportional to the number of nodes in the graph using Pearl's message-passing scheme [62]. This is a generalisation of the standard Baum-Welch forward-backward algorithm for inference in Hidden Markov Models [64] and as such, it is also referred to as the upward-downward algorithm, [66, 47].

TSBN models have been used by a number of authors for image analysis tasks. Bouman and Shapiro [9] introduced such a model using discrete labels in the nodes for an image segmentation task. Pérez *et al* [63] have discussed MAP and MPM (maximum posterior marginal) inference on TSBNs for image processing tasks. Laferté *et al* [42, 43] have extended this model using a multiscale feature pyramid image decomposition. They used the EM algorithm for parameter estimation. Cheng and Bouman [13] have investigated trainable multiscale models, using decision trees to represent compactly conditional probability tables (CPTs) in their model.

TSBN models can also be used for continuously-valued Gaussian processes in one and two dimensions. These are the generalisation of Kalman filter models from chains to trees. They have been studied by a number of groups, notably Prof. Willsky's group at MIT, who have developed the theory and derived fast algorithms for problems such as optical flow estimation, surface reconstruction and texture segmentation [6, 50, 23, 49]; see also [40]. Crouse *et al* [16] have used a multiscale TSBN to model wavelet coefficients. De Bonet and Viola [18] also model wavelet coefficients using a tree-structured type dependence in which the wavelet coefficients at a particular length-

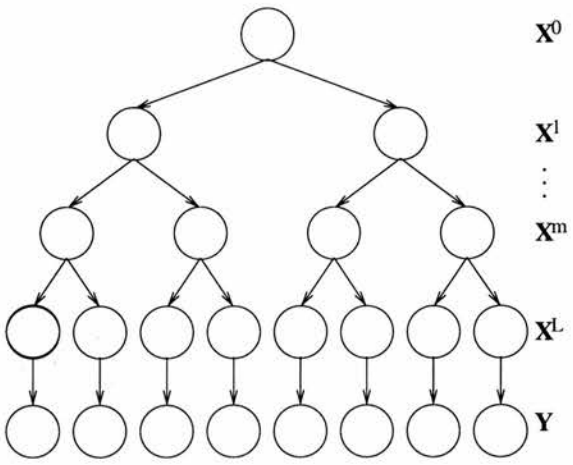


Figure 2.1: A 1-D Graphical Model Illustrating a Small Tree-Structured Belief Network. The layers in $\mathbf{X}$ are denoted $\mathbf{X}^0$, $\mathbf{X}^1$ and so on, down to $\mathbf{X}^L$; in this case $L = 3$. $\mathbf{Y}$ denotes the raw image information.

scale are dependent on all of the longer length-scale coefficients rather than just those in the layer above. They estimate the parameters in the model using a Parzen window density estimator. However, since we require a prior over class-label images, we will focus on TSBNs with discrete-valued nodes.

The model for the data is illustrated in Figure 2.1. The observed data $\mathbf{Y}$ is assumed to have been generated from an underlying process $\mathbf{X}$, where $\mathbf{X}$ is a TSBN. The network is arranged in layers. At the highest level (level 0) there is one node $\mathbf{X}^0$, which has children in level 1. The lower levels are denoted $\mathbf{X}^1, \mathbf{X}^2$ and so on, down to $\mathbf{X}^L$ in level L.

Each node is a multinomial variable, taking on one of C class labels. These labels are those used for the segmentation, e.g. road, sky, vehicle etc[1]. The links between the nodes are defined by conditional probability tables (CPTs) and the root node has an unconditional prior distribution instead of a CPT.

The nodes in $\mathbf{X}^L$ have a one-to-one correspondence with the observed data $\mathbf{Y}$. (The observed data in our case will be features derived from blocks of pixels, rather than individual pixels in the raw image.) The model for the observations illustrated in Figure 2.1 is that the observation Y_i is dependent only on the corresponding variable X_i^L,

[1]Note that it is not necessary for the hidden nodes to be C-class multinomial variables. We have used this for convenience, and because it gives rise to a simple initialisation of the TSBN-training, as described in Section 2.2.5.

i.e. $P(Y_i = y_i|\mathbf{X} = \mathbf{x}) = P(Y_i = y_i|X_i^L = x_i^L)$. The overall model for $P(\mathbf{X}, \mathbf{Y})$ is

$$P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{X}) \prod_{i=1}^{N} P(Y_i|X_i^L). \tag{2.1}$$

where N is the number of pixels in the image. This describes a fully generative model for $\mathbf{Y}$, defined by CPTs from the root downwards.

In the experimental section (Section 2.2), we will compare the performance of the TSBN using maximum likelihood (ML) parameters with the performance of the TSBN using the conditional maximum likelihood (CML) parameters, which are the parameters that maximise $P(\mathbf{X}|\mathbf{Y})$. In the latter situation, we assume that the local image model $P(Y_i|X_i)$ is fixed and maximise $P(\mathbf{X}|\mathbf{Y})$ with respect to the parameters in the TSBN. We take these two approaches rather than maximising the joint distribution $P(\mathbf{X}, \mathbf{Y})$ with respect to the parameters in both the tree and the local image models because in this way we can better separate out the influence of the TSBN and that of the local image models.

With regard to the local image models, since each pixel is composed of a number of components in feature space, $P(Y_i = y_i|X_i^L = x_i^L)$ is a multivariate density function. One way to model this distribution is to use a Gaussian mixture density, as used in [9] and described in Section 2.2.4. An alternative to using a fully generative approach for the local image model makes use of the *scaled likelihood* method and uses a multi-layer perceptron (MLP) to model the probability of the pixel labels given the raw image information.

In the following section (Section 2.1.1), the scaled likelihood method is outlined. Section 2.1.2 describes the training of the TSBN in terms of both the calculation of the ML and the CML parameters. The training of the local image models are discussed in the Experimental Details section, Section 2.2.4 looks at training a GMM in the generative paradigm and Section 2.2.3 looks at training an MLP in the discriminative paradigm. There is a discussion of the relative benefits of the "sampling" (generative) and "diagnostic" (discriminative) paradigms in the context of medical diagnosis in [17].

2.1.1 Scaled Likelihood Method

The database of images being used provides the raw data $\mathbf{Y}$ and the labelled segmentations $\mathbf{X}^L$, we can train a classifier to predict X_i^L given Y_i [65]. The resulting estimate of

the posterior probability $P(X_i|Y_i)$ can be used to calculate $P(Y_i|X_i)$, which is required for the fusion with the TSBN prior distribution, through the application of Bayes' theorem.

From Equation (2.1) we can see that the local probabilities of the image information given the pixel class value, $P(Y_i|X_i)$ which we obtain from our generative Gaussian mixture model for the data, are an intrinsic part of the probabilitistic model. However we are also allowed an alternative. Using Bayes theory we can rewrite the conditional probability of the data under the generative model as follows,

$$P(Y_i|X_i^L) = \frac{P(X_i^L|Y_i)P(Y_i)}{P(X_i^L)}. \tag{2.2}$$

When performing inference on $\mathbf{X}$, the data $\mathbf{Y}$ remains fixed and so the terms $P(Y_i)$ in the equation above are merely constant factors.

We define the *scaled likelihood* $L(X_i^L, Y_i)$ at pixel i as

$$L(X_i, Y_i) \equiv \frac{P(Y_i|X_i^L)}{P(Y_i)} = \frac{P(X_i^L|Y_i)}{P(X_i^L)} \tag{2.3}$$

which means we can rewrite Equation (2.1) as:

$$P(\mathbf{X}|\mathbf{Y}) = \frac{P(\mathbf{X})}{P(\mathbf{Y})} \prod_i P(Y_i|X_i^L) \tag{2.4}$$

$$\propto P(\mathbf{X}) \prod_i L(X_i^L, Y_i) \tag{2.5}$$

This leads to a principled method with which we can use discriminative techniques to perform inference on the image labels. Rather than using this local scaled likelihood, $L(X_i^L, Y_i)$, we use a local prediction of the scaled likelihood which we define as follows:

$$\hat{L}(X_i^L, Y_i) = \frac{\hat{P}(X_i^L|Y_i)}{\hat{P}(X_i)}. \tag{2.6}$$

This is calculated using the local discriminative predictions $\hat{P}(X_i^L|Y_i)$ and also an estimate of the local prior leaf-node probability $\hat{P}(X_i^L)$. This is an approach which was first suggested for use with HMMs [69, 54].

As it stands this formulation requires a separate model to be trained for $P(X_i^L|Y_i)$ at each pixel position i. This is undesirable since with the number of free parameters involved, the generation of suitable estimates for their values would require both a large data set and a massive amount of computation. Instead one local pixel model is used but the position coordinates of the pixel in the image are given as one of the features on which to base classification.

To compute $\hat{L}(X_i^L, Y_i)$, the second quantity that we need to estimate is the prior distribution over the pixel labels, $\hat{P}(X_i^L)$. The value of this quantity is dependent on the position of the pixel in the image. It for example encodes the fact that the pixel label 'sky' generally occurs more frequently in the top half of pictures in the training set than it does in the bottom half.

However, in the following experiments (see Section 2.3.2), we compare an estimate for $\hat{P}(X_i^L)$ which is spatially invariant with one derived from the pixelwise marginals of the ML-trained TSBN; the results obtained were very similar. When we estimate $\tilde{P}(X_i^L)$, the prior probabilities of the labels in the reduced MLP training set (as described in Section 2.2.3), we use a spatially invariant estimate. Initially, a comparison was made of the performance of various methods for computing a suitable estimate. The methods studied were:

(i) the use of a radial basis function network with softmax outputs which was trained to estimate the label prior given the pixel position;

(ii) the use of the relationship $P(X_i^L) = \int P(X_i^L|Y_i)P(Y_i)dY_i$ and approximating the integral with an average over appropriate feature vectors;

(iii) to take a region-based average of the label frequency in the training set;

(iv) to take an overall average of the label frequency.

It turns out that the use of the spatially invariant estimate $\tilde{P}(X_i^L)$ (method iv) is the method which is most robust to problems due to the sparsity of data. It is often the case that the estimates for $P(X_i|Y_i)$ and $P(X_i)$ are both very small though the ratio can be large. The more sophisticated estimates often confidently underestimate the probability of certain classes in regions where few training examples have been seen which results in broad areas of the images being biased towards one or two classes and many misclassifications being made. The spatially invariant estimate gives the best pixelwise classification accuracy when fused with the pixelwise predictions from the ML-trained TSBN.

With this framework in place there are two basic tasks that we will want to perform. Firstly we want to use a training database which consists of a set of images and image labels $\{\mathbf{X}^L, \mathbf{Y}\}$ to learn the model parameters, see Section 2.1.2. The second basic task is to use the learnt parameters to perform inference on new previously unobserved images. The full Bayesian approach to this problem is to calculate the posterior

$P(\mathbf{X}^L|\mathbf{Y})$, however this is highly expensive as $\mathbf{X}^L$ has C^K states. In real images, the number of pixels K can easily be of the order 10^5. However, calculations that are computationally feasible include the computation of the posterior marginals $P(X_i^L|\mathbf{Y})$ from which we can construct the maximum posterior marginal (MPM) segmentation and also maximum a posteriori (MAP) configuration of the data, that is the segmentation which maximises the posterior probability distribution, $\text{argmax}_{\mathbf{x}} P(\mathbf{X}|\mathbf{Y})$. These calculations can be performed in a time proportional to the number of nodes using Pearl's λ-π message-passing schemes[62].

2.1.2 Training the TSBN

The probability distribution over the classes of each node in the TSBN, $P(\mathbf{X})$, is parameterised by the entries of the conditional probability table which defines the link between parent and child nodes in the tree and the prior probabilities of the root node. We will use $\boldsymbol{\theta}$ to represent the complete set of parameters for the TSBN. Specifically θ_{ikl} will represent $P(X_i = x_{il}|Pa_i = pa_{ik})$ which is the probability that the class X_i of node i will take value x_{il} where $l = 1,\ldots,C$ given that the class Pa_i of the parent of node i has value pa_{ik} where $k = 1,\ldots,C$.

We distinguish between the parameters in the TSBN and the parameters of the likelihood model which we will denote as $\boldsymbol{\phi}$. In the training phase we have a number of observed images $\mathbf{y}^m$ and associated labelled images $\mathbf{x}^{Lm}$ where $m = 1,\ldots,M$, is the index of the images in the training set. The likelihood model is thus denoted as $P(\mathbf{y}^m|\mathbf{x}^{Lm},\boldsymbol{\phi})$.

Even for small examples and certainly for TSBNs corresponding to realistic size images it is not possible to hand code the parameters to represent all the correlations between pixels in a training database. In order to perform meaningful inference these parameters will need to have been learnt in a suitable fashion.

Ideally one would learn the probability distribution over these parameters from the training set, then all subsequent calculations would be performed as weighted integrations over the parameter space. However compromises need to be made for the purposes of practicality and tractability. Work done here makes use of estimates of the parameters. We will compare the performance of maximum likelihood (ML) parameter estimates and the conditional maximum likelihood (CML) estimates.

2.1.2.1 Maximum Likelihood

We estimate the TSBN parameters $\boldsymbol{\theta}$ such that they maximise $P(\mathbf{x}^{Lm}|\boldsymbol{\theta})$. Thus,

$$\hat{\boldsymbol{\theta}}^{ML} = \text{argmax}_{\boldsymbol{\theta}} \prod_{m=1}^{M} P(\mathbf{x}^{Lm}|\boldsymbol{\theta}). \tag{2.7}$$

$P(\mathbf{x}^{Lm}|\boldsymbol{\theta})$ can be calculated using Pearl message-passing. The calculation consists of a single upwards pass through the tree collecting the lambda messages and then marginalising the root node using the prior root probabilities. The optimisation can be performed using the EM algorithm. For details of the EM update equations and their derivation, please refer to [22].

2.1.2.2 Conditional Maximum Likelihood

In the CML procedure, the object is to predict correctly the labels $\mathbf{x}^L$ associated with the evidence $\mathbf{y}$. The parameters are estimated by maximising the probability of the correct labelling given the evidence. As mentioned earlier, we assume that the local image model $P(\mathbf{y}^m|\mathbf{x}^{Lm},\boldsymbol{\phi})$ which is parameterised by $\boldsymbol{\phi}$ is fixed. The local image model can take the form of either a GMM or an MLP. The training of these models is discussed in Sections 2.2.4 and 2.2.3 respectively. The conditional maximum likelihood parameters are given by,

$$\hat{\boldsymbol{\theta}}^{CML} = \text{argmax}_{\boldsymbol{\theta}} \prod_{m=1}^{M} P(\mathbf{x}^{Lm}|\mathbf{y}^m,\boldsymbol{\theta},\boldsymbol{\phi}). \tag{2.8}$$

$$= \text{argmax}_{\boldsymbol{\theta}} \prod_{m=1}^{M} \frac{P(\mathbf{x}^{Lm},\mathbf{y}^m|\boldsymbol{\theta},\boldsymbol{\phi})}{P(\mathbf{y}^m|\boldsymbol{\theta},\boldsymbol{\phi})} \tag{2.9}$$

The numerator of Equation (2.9) can be further decomposed into $P(\mathbf{y}^m|\mathbf{x}^{Lm},\boldsymbol{\phi})P(\mathbf{x}^{Lm}|\boldsymbol{\theta})$. As indicated by Equation (2.1), $P(\mathbf{y}^m|\mathbf{x}^{Lm},\boldsymbol{\phi})$ is the product of the pixelwise likelihoods. $P(\mathbf{y}^m|\boldsymbol{\theta},\boldsymbol{\phi})$ is calculated using message-passing in the same way as $P(\mathbf{x}^{Lm}|\boldsymbol{\theta})$. Unfortunately, the EM algorithm is not applicable to CML estimation because the CML criterion is expressed as a rational function [28]. The optimisation is instead carried out using the scaled conjugate gradients algorithm, see [22] for more details.

We compare the performance of the ML and CML approaches because whilst the ML calculation is easier to do and only requires the label data, it seems that this throws away the extra information we receive from the fact that a pixel label is always (or mostly) associated with the corresponding pixel image data. The task that we will

essentially always want to perform is that of inferring the pixel label given the pixel image data, so it seems natural that this task will be performed best in a general sense when using the parameters calculated in the CML regime.

2.2 Experimental Details

2.2.1 Data

Colour images of outdoor scenes from the Sowerby Image Database[2] of British Aerospace are used in our experiments. The database contains both urban and rural scenes. They feature a variety of every-day objects from roads, cars and houses to lanes and fields in various places near Bristol, UK. All the scenes were photographed using small-grain 35mm transparency film, under carefully controlled conditions. Each image of the database has been digitised with a calibrated scanner, generating a high quality 24-bit colour representation.

Both colour images and their corresponding label images are provided in the database. The label images were created by over-segmenting the images, and then hand labelling each region produced. There were 92 possible labels, organised in a hierarchical system. We combined labels to produce seven classes, namely "sky", "vegetation", "road markings", "road surface", "building", "street furniture" and "mobile object". For instance, the class "street furniture" is a combination of many types such as road sign, telegraph pole and bounding object.

Figure 2.2a shows two scenes from the test image set of the database, one rural and one urban. Figure 2.2b shows their hand-labelled classifications; different colours in the label image correspond to the seven different possible classes. The original 104 images were divided randomly into independent training and test sets of size 61 and 43 respectively. The full-resolution colour images of size 512 by 768 pixels were downsampled into 128 by 192 regions of size 4 by 4 pixels. The label of the reduced region was chosen by majority vote within the region, with ties being resolved by an ordering on the label categories. From now on we will refer to the reduced label images as label images because the original label images will no longer be used. Close

[2]This database can be made available to other researchers. Please contact Dr. Andy Wright or Dr. Gareth Rees, Advanced Information Processing Department, Advanced Technology Centre - Sowerby, BAE SYSTEMS Ltd, PO Box 5 Filton, Bristol, BS34 7QW, UK, email: `gareth-s.rees@baesystems.com` for details.

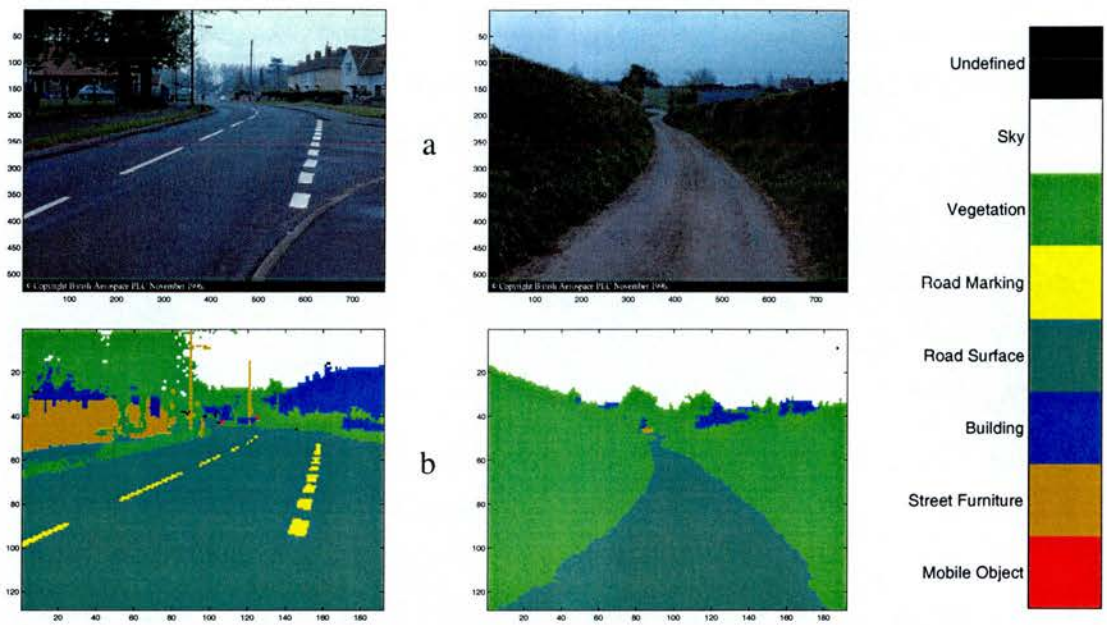


Figure 2.2: A Rural and an Urban Scene and their Hand-Labelled Classification. The subfigures show: (a) the original images; (b) the hand-labelled classifications. On the right is a key describing the labels used.

inspection of the example images and their labels will reveal that the labelling that we have to use as the ground truth is not always entirely accurate. Note also that the labellings do occasionally contain "undefined" pixels. Both these factors combine to make the task of image modelling more challenging.

2.2.2 Feature Extraction

An important step in classification is that of feature selection. Initially forty features were extracted from each region. Among them, six features were based on the R, G, B colour components, i.e., the mean and variance of overall intensity in the region, the colour hue angle (sine and cosine) [4], (R-B) and (2G-R-B)/2 (as used by [59]), where R, G and B indicate the means of the red, green and blue components respectively. The texture features are the grey-level difference vectors (GLDV) textural features [78, 76] of contrast, entropy, local homogeneity, angular second moment, mean, standard deviation, cluster shade and cluster prominence. The GLDV features were extracted based on the absolute difference between pairs of gray levels at a distance $d = 1$ apart at four angles $\theta = 0°, 45°, 90°, 135°$. The (x, y) location of the region was also included

in the feature space, as described in Section 2.1. Each feature was normalised by using a linear transformation to have zero mean and unit variance over the training set.

It is useful to limit the number of features used because increasing the number of features increases the free parameters which need to be optimised in the MLP training phase. A linear model using the normalised features as inputs and softmax outputs [7] was used in feature selection. For each input, the sum of the absolute values of the weights coming out of that input in the trained model was calculated, and the twenty-one features for which this sum was larger than unity were retained. This selection procedure is based on the idea that more important features will tend to give rise to larger weights (*cf* the Automatic Relevance Determination idea of MacKay and Neal [56]).

2.2.3 MLP Training

Multi-Layer Perceptrons (MLPs) were used for the task of predicting $P(X_i^L|Y_i)$. As explained in Section 2.1, these probabilities were estimated by a MLP that takes as input both the non-positional feature vector and position of pixel i. The retained features produced a feature vector for each region and were fed to a MLP with 21 input nodes, 7 output nodes and one hidden layer which was trained to classify each region into one of the seven classes. The activation functions of the output nodes and hidden nodes were the softmax function and tanh sigmoid functions respectively. The error function used in the training process was the cross-entropy function for multiple classes (see [7]). A scaled conjugate gradient algorithm was used to minimise the error function. The training is performed using over 51,000 regions extracted from the training image dataset, and validating on an independent validation dataset of over 15,000 regions. The validation dataset was used in order to choose the optimal number of hidden nodes in the MLP; eventually the best performance on the validation set is obtained for a MLP with 30 hidden nodes.

The training dataset for MLP training was formed by choosing randomly up to 150 regions for each class from each single image. By doing so we tried to use equal numbers of regions from each class in the training set of the MLP. The aim of this rebalancing of the training set is to give the net a better chance to learn about the infrequent classes (see [7], p 224). The probabilities for each class in the training set of the MLP, denoted by $\tilde{P}(c_k), k = 1, \cdots, C$, were estimated simply by evaluating the fraction of the training set data points in each class. The corresponding probabilities for all

pixels in the whole of the training set images are denoted by $P(c_k), k = 1, \cdots, C$. These turned out to be $\tilde{P}(c_k) = (0.2089, 0.2122, 0.0453, 0.2122, 0.1549, 0.1140, 0.0525)$ and $P(c_k) = (0.1468, 0.3665, 0.0023, 0.4205, 0.0491, 0.0078, 0.0070)$. The ordering of the classes is "sky", "vegetation", "road markings", "road surface", "building", "street furniture" and "mobile object". These two sets of prior probabilities are very different; $\tilde{P}(c_k)$ is almost uniformly distributed over all classes, but $P(c_k)$ is biased towards classes two and four, corresponding to "vegetation" and "road surface" respectively.

Since the training set for the MLP reweights the classes according to $\tilde{P}(c_k)$, it is necessary to consider what effect this will have on the scaled likelihood. In fact, following [7] (p. 223) we find that

$$P(c_k|y_i) = \frac{1}{Z}\frac{\tilde{P}(c_k|y_i)}{\tilde{P}(c_k)}P(c_k), \tag{2.10}$$

where y_i is the input to the network at pixel i, $\tilde{P}(c_k|y_i)$ is the network output for class k, $P(c_k|y_i)$ is the compensated network output and Z is the normalising factor used to make $\sum_k P(c_k|y_i)$ sum to one. Hence we see that the scaled likelihood $P(c_k|y_i)/P(c_k)$ is equal to $\tilde{P}(c_k|y_i)/\tilde{P}(c_k)$ up to an unimportant constant. We call $\tilde{P}(c_k|y_i)$ the raw MLP prediction, and $P(c_k|y_i)$ (as given by Equation 2.10) the *compensated* MLP prediction. A segmentation can be obtained by choosing the most probable class at each pixel independently. We call these the raw MLP and compensated MLP segmentations, when using the uncompensated and compensated predictions respectively.

2.2.4 Gaussian Mixture Model Training

In Section 2.2.3 we have described how MLPs were trained to relate the image features to labels. The alternative approach is to build class-conditional density estimators for each class, and to use this along with Bayes' rule to make predictions. Following [9, 43] we have used Gaussian mixture models (GMMs) for this task.[3] The same training set was used as for the MLP. We considered three different feature sets (i) the average R, G, B values in a region, (ii) the 21 features used to train the MLP and (iii) all 40 features. In addition two different settings of the `cluster` program were used, allowing for either diagonal or full covariance matrices for the Gaussians. We describe the combination of the use of the feature vectors of length 3, 21 and 40, with full and diagonal covariance matrix models as: 3-full, 3-diag, 21-full, 21-diag, 40-full, 40-diag

[3] Specifically the `cluster` program available from `http://www.ece.purdue.edu/~bouman` was used.

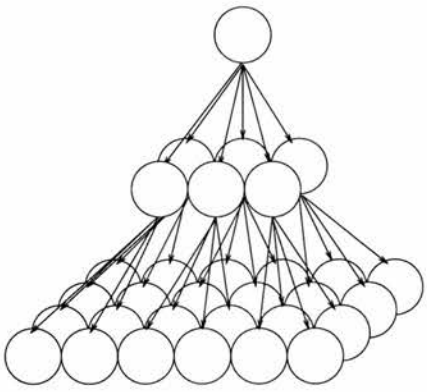


Figure 2.3: A Diagram of the Basic Fixed Quadtree Architecture. This is an illustration of the tree structure of the first two layers of the TSBN used for the work in this chapter. Note that there are 6 nodes in the second layer, this ensures that the layout of the nodes in the layers below fit to the shape of the images which have a 3:2 aspect ratio.

respectively. The program selects the number of mixture components automatically using a MDL criterion; the recommended initialisation of starting with three times as many components as features was used.

The GMMs for each class were combined with the prior probabilities for each class (the $P(c_k)$'s given in Section 2.2.3) to produce pixelwise classifications. The overall classification accuracies were 68.72%, 49.76%, 75.95%, 71.38%, 77.45%, 71.92% for the 3-full, 3-diag, 21-full, 21-diag, 40-full, 40-diag models respectively on the test set. The GMM model with highest pixelwise performance (namely 40-full) was then used in further experiments (see Section 2.3.2 for further details). The number of mixture components in the 40-full model for each of the seven classes was 9, 9, 4, 9, 8, 7, 6 respectively.

We have found that the trained GMM sometimes makes very confident misclassifications and that this can cause underflow problems when evaluating the conditional probability $P_{TSBN}(\mathbf{x}^{L*}|\mathbf{y})$, the conditional probability of the "ground truth" labelling $\mathbf{x}^{L*}$ given the image $\mathbf{y}$ (see Section 6.4). For this reason we replaced the likelihood term $P(Y_i|X_i^L)$ with $\max(P(Y_i|X_i^L), \varepsilon)$, where ε was set to 10^{-29}, the minimum value needed to avoid underflow in the ML-TSBN.

2.2.5 TSBN Training

The TSBN used was basically a quadtree (see Figure 2.3), except that there were six children of the root node to take into account the 2:3 aspect ratio of the images. For a down-sampled image with a total of $128\times$ 192 pixels, we took each pixel in the down-sampled image as a leaf node in a belief network and built up an eight-level TSBN with a total of 32756 links between nodes at adjacent levels. If each link had a separate CPT, a very large training set would be needed to ensure the CPTs were well determined, and this in turn implies that huge computational resources could be needed in order to find a suitable minimum of the CML objective function. In practice such an approach is clearly impractical. One technique for dimensionality reduction in this case is to tie CPTs. In our experiments all of the CPTs on each level were constrained to be equal, except for the transition from level 0 to level 1, where each table was separate. This flexibility allows knowledge about the broad nature of scenes (e.g. sky occurs near to the top of images) to be learned by the network, as is indeed reflected in the learned CPTs (see Section 2.3.1).

For training the ML-TSBN, the network parameters $\boldsymbol{\theta}$ were initialised in a number of different ways. It was found that the highest marginal likelihood in the training data was obtained when the initial values of $\boldsymbol{\theta}$ were computed using probabilities derived from downsampled version of the images. The sparse-data problem appeared in the initial values of the CPTs because some pairings do not occur in the training data[4]. We dealt with the problem by adding a small quantity η to each conditional probability $p(x_{il}|\mathrm{pa}_{ik})$ for $l = 1, \cdots C$, then normalising the modified probabilities. We have used $\eta = min_k(p(x_{il}|\mathrm{pa}_{ik}))/100$ for the case that at least one $p(x_{il}|\mathrm{pa}_{ik}) \neq 0$, otherwise $\eta = 1/C$. The plot of likelihood against iteration number levelled off after 30 iterations. In the database some pixels are unlabelled; assuming these values are "missing at random", we treated them as uninstantiated nodes, which can easily be handled in a belief network framework.

The CML training was initialised at the ML-TSBN solution. The plots of conditional likelihood against iteration number had levelled off after 44 iterations of CML training by scaled conjugate gradient optimisation for both the GMM and MLP predictors.

[4]The reason why it is important to consider this is that if a CPT entry is set to zero, the EM algorithm will not move it away from zero during training.

2.2.6 Combining Pixelwise Predictions with Trees

We now have GMM and MLP local predictors, and ML and CML trained TSBNs. This gives rise to a large number of possible combinations of pixelwise predictors with trees. The ones that we have investigated are

1. raw GMM pixelwise predictions
2. compensated GMM pixelwise predictions (spatially-uniform compensation)
3. compensated GMM pixelwise predictions (using marginals of ML-TSBN)
4. GMM likelihood + ML-TSBN
5. GMM likelihood + CML-TSBN
6. raw MLP pixelwise predictions
7. compensated MLP pixelwise predictions (spatially-uniform compensation)
8. compensated MLP pixelwise predictions (using marginals of ML-TSBN)
9. MLP scaled-likelihood + ML-TSBN
10. MLP scaled-likelihood + CML-TSBN

The TSBN methods calculated the scaled likelihoods as described in Section 2.2.3, and MAP inference was used for the pixelwise predictions. For entries 3 and 8 (compensation using the marginals of the ML-TSBN) note that different compensation probabilities are used in the six regions of the image defined by the six CPTs from the root to level 1. The performance of these methods is investigated in Sections 2.3.1 and 2.3.2.

2.3 Results

2.3.1 TSBN Training

In this section we describe the results from training the TSBN using both ML and CML training. We first discuss label-image coding results using the ML-trained tree, and then inspect the learned CPTs from the ML-trained TSBN and the CML-trained TSBN.

2.3.1.1 The Learned CPTs

The CPTs derived using ML training are shown in Figure 2.4. Note that six separate CPTs were used for the transition from the root node to level 1, as explained in Section 2.2.5.

We can also calculate the prior marginals at each node in the tree, by simply taking the prior from the root node and passing it through the relevant CPTs on the path from the root to the node under consideration[5]. The fact that there are six CPTs for the root to level 1 transition means that there are, in effect, six different prior marginals in levels 1 to 7, defined by the 2:3 aspect ratio of the image. These prior marginals are shown in Figure 2.5. It may not be easy to interpret the CPTs/marginals as a permutation of the state labels at a node and corresponding permutations of the incoming and outgoing CPTs would leave the overall model unchanged; however, it appears that the downsampling initialisation means that this is not a large problem.

Analysing Figures 2.4 and 2.5, we see that (1) The prior marginals at level 7 reflect the overall statistics of the images. The sky, vegetation and road surface classes are the most frequently occurring, the sky class is more likely to be found in the top half of the images and road surface in the bottom half. Similar patterns are detectable at level 1 of Figure 2.5, although the vegetation label is less prevalent in the upper half at this level. (2) The trained CPTs in levels 1 to 7 exhibit a strong diagonal structure, implying that the children are most likely to inherit their parent's class. (3) The level 0 to level 1 CPTs need to be read in conjunction with the root's prior distribution to provide a good explanation of the level 1 prior marginals. (4) Although the CPTs superficially look similar for layers 2-7, close inspection reveals that they do differ somewhat. Features can be observed. For instance, if we look at the fifth row of the CPTs which correspond to the probabilities of each class given that the parent is of class 5 (building). We can see that high up in the tree there are significant probabilities that the child nodes are a different class whereas at the bottom of the tree the child is almost certainly also of class 5 (building). This corresponds to the fact that in our dataset buildings tend to be far off in the distance and hence tend to be quite small. (5) Similar scale effects can be observed in the prior marginals. At the top levels the classes which correspond to large scale objects (sky, vegetation, road) are the most probable by far. Whereas, at lower levels small-scale objects are more probable.

[5]This can also be achieved using Pearl's propagation scheme outlined in Appendix A, with every leaf node uninstantiated.

Although Laferté *et al* [43] have carried out EM training of a TSBN, we note that they only estimated CPTs that were tied on a layer-by-layer basis. For our data Figures 2.4 and 2.5 show that relaxing this constraint can be useful.

The CPTs and prior marginals obtained with CML training were similar to those shown in Figures 2.4 and 2.5 respectively. This is probably due to the fact that CML training was initialised at the ML-TSBN solution for both GMM and MLP predictors.

2.3.2 Segmentation Results and Performance Evaluation

We now turn to the classification of the test images. Often classification performance is evaluated on pixelwise accuracies. However, for complex real-world classification task, such as ours, this does not tell the whole story. There are a number of other factors that concern us, most notably the fact that we are predicting the labels of pixels in an image, and that spatial coherence is important. We also note that the fractions of pixels from different classes are tremendously different, and that the ground-truth labels used in assessing performance are not 100 percent correct because of both the downsampling process, and also because of inaccuracies in the hand-labelling process. Therefore, it is a difficult task to assess the quality of classification derived from the various methods, which may also depend on the uses to which the classification will be put. An early reference to assessing the quality of segmentations is [45]. More recently, there has been some realisation that the aim of segmentation may not be to return just a single segmentation, but multiple solutions [57], or a probability distribution over segmentations $P(\mathbf{x}^L|\mathbf{y})$. This posterior distribution can be explored in many ways; below we describe two, namely (i) posterior marginal entropies and (ii) the evaluation of the conditional probability $P(\mathbf{x}^{L^*}|\mathbf{y})$, where $\mathbf{x}^{L^*}$ is the ground truth image for given input data $\mathbf{y}$.

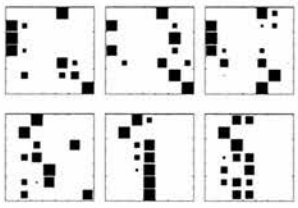
In this section we compare the performance of classification based on the smoothness of the segmented image in Section 6.1, the pixelwise prediction accuracies in Section 6.2, the marginal entropies in Section 6.3 and the conditional probability in Section 6.4.

2.3.2.1 Smoothness

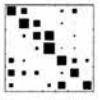
For the rural scene in Figure 2.2, Figure 2.6 shows classifications using most of the combinations outlined in Section 2.2.6. The classifications obtained from the single-

(a) Prior for root node:

(b) From root node to Level 1:



(c) From Level 1 to Level 2:



(d) From Level 2 to Level 3:

(e) From Level 3 to Level 4:

(f) From Level 4 to Level 5:

(g) From Level 5 to Level 6:

(h) From Level 6 to Level 7:

Figure 2.4: Estimated Prior Probabilities for the Root and CPTs (ML training) of an Eight-Level Belief Network after being Trained using the Training Images. The area of each black square is proportional to the value of the relevant probability. (a) The prior probabilities at the root node. (b) The six independent CPTs on the links from root node to its six children on the first level. (c)-(h) CPTs for the links between adjacent levels from Level 1 to Level 7 respectively. The seven labels are 1-Sky, 2-Vegetation, 3-Road marking, 4-Road surface, 5-Building, 6-Street furniture, 7-Mobile object. The CPTs have entry (1,1) in the top left-hand corner, and are read with "from level l" indexing the rows and "to level $l+1$" indexing the columns.

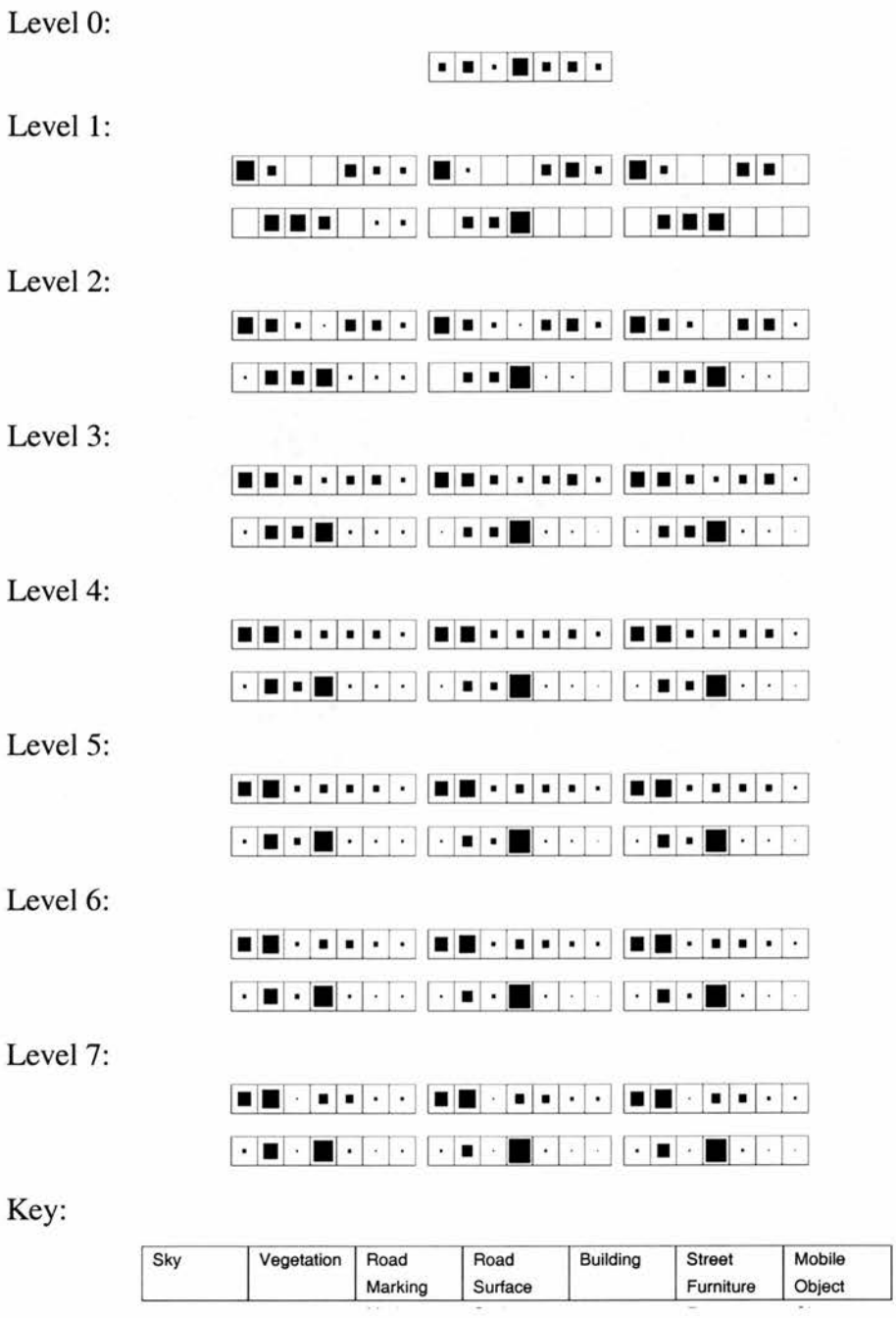


Figure 2.5: The Prior Marginals after Training with the ML Algorithm. The area of each black square is proportional to the value of the relevant probability. See text for further details.

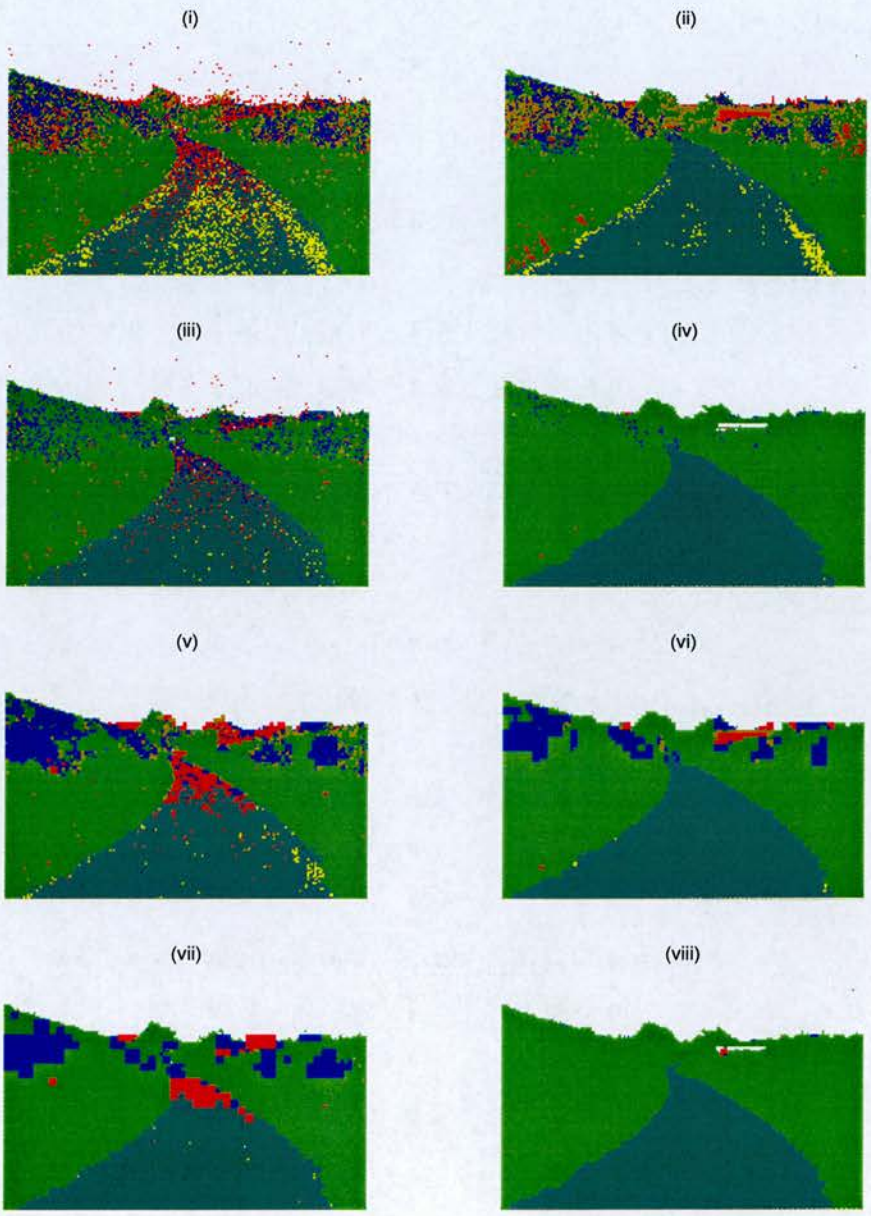


Figure 2.6: Classification of a Rural Scene. (i) raw GMM pixelwise predictions, (ii) raw MLP pixelwise predictions, (iii) compensated GMM pixelwise predictions, (iv) compensated MLP pixelwise predictions, (v) MAP segmentation for GMM + ML-TSBN, (vi) MAP segmentation for MLP + ML-TSBN, (vii) MAP segmentation for GMM + CML-TSBN, (viii) MAP segmentation for MLP + CML-TSBN.

pixel methods typically have a lot of high-frequency noise, due to locally ambiguous regions. Both the ML- and CML-trained trees tend to smooth out this noise.

A similar smoothing can be obtained using a majority filter [46], where one simply chooses the most common class within a window centered on the pixel of interest. However, one drawback with this is that majority-filter smoothing with a reasonably-sized window tends to remove fine detail, such as road markings; in contrast it seems that the TSBN methods yield something like an adaptive smoothing, depending on the strength of the local evidence. Also, note that majority-filtering does not return a probability distribution over segmentations.

2.3.2.2 Pixelwise Classification Accuracy

Table 2.1 shows the pixelwise classification accuracy for each class, and the overall pixelwise accuracy (weighted average of the classes) for each of the ten methods listed in Section 2.2.6. For the TSBN methods the MAP segmentation result is reported; the MPM results were similar, although they were generally worse by a few tenths of one percent. The most noticeable feature is that the performance obtained with the MLP methods is superior to that from the GMM methods.

Looking at the results in detail, we notice that the raw results for both the GMM and MLP (columns 1 and 6) are improved by compensation (columns 2 and 7 resp). The compensated methods simply give more weight to the more frequently occurring classes, as can be seen by comparing columns 1 and 2, and 6 and 7. There are only small differences between the spatially uniform compensation of columns 2 and 7, and the ML-TSBN marginal compensation scheme of columns 3 and 8.

Columns 4 and 8 combine pixelwise evidence with the ML-TSBN. For both GMM and MLP local models, it is perhaps surprising that the performance decreases as compared to columns 3 and 7 respectively. (This is a fair comparison as the methods of columns 3 and 7 use the marginals of the ML-TSBN, but not the correlation structure.)

Columns 5 and 10 show the performance of the GMM and MLP local models combined with trees trained using the CML method on the relevant data. In both cases the performance is better than fusion with the ML-TSBN and for the MLP this method obtains the best overall performance.

For comparison, we note that McCauley and Engel [53] compared the performance of Bouman and Shapiro's SMAP algorithm against a pixelwise Gaussian classifier on a remote sensing task, and found that the overall classification accuracy of SMAP was

a 3.6% higher (93.4% vs 89.8%).

Whilst the reasons for the superior performance of the MLP are not easy to ascertain clearly, it is pleasing to note that the results are compatible with our intuition that the general strategy of discriminative training should be more suitable for the task in hand. We can see that the test images are a relatively diverse set of images (although drawn from the same distribution as the training images). It may be that features that are important to the MLP classifier are similar in the training and test images, while other features (whose distribution has to be modelled by the GMMs) do vary between training and test sets. In contrast, some other evaluations in the literature (e.g. [53]) use only a single test image with training data drawn from a subset of the pixels; in this case the issue of inter-image variability does not arise. We also note that the comparison of the GMM and MLP classifiers was carried out using a training set of a particular size and composition; different results might be obtained if these factors were varied.

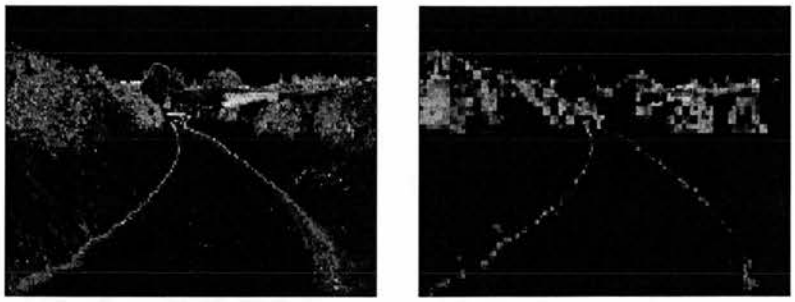
Table 2.1: Pixelwise Classification Accuracy Results for the TSBN Image Segmentation Experiments. The table shows the performance of the 10 methods, showing the percentage correct for each class and overall. The second column in the table gives the overall percentage of each class in the test images.

Class	Overall percentage	1	2	3	4	5	6	7	8	9	10
sky	12.10	91.85	94.72	94.87	93.74	92.74	95.90	95.66	96.19	94.69	96.30
vegetation	40.12	61.04	78.92	77.46	67.92	75.72	79.32	92.41	90.40	81.67	90.45
road markings	0.17	55.14	42.26	44.33	43.09	27.36	78.61	68.97	67.91	70.04	67.91
road surface	39.5	62.14	78.18	80.45	70.10	73.45	94.52	97.16	96.26	94.99	96.85
building	6.11	44.19	46.98	49.67	63.70	74.92	67.69	44.43	52.73	79.40	64.60
street furniture	1.35	28.58	14.05	13.77	20.97	8.58	24.89	3.96	4.62	10.07	6.63
mobile object	0.57	58.85	43.05	43.10	72.76	76.23	49.13	28.83	32.15	78.89	44.74
overall	100.00	63.71	77.45	77.94	71.00	75.85	85.72	90.16	89.38	87.38	90.68

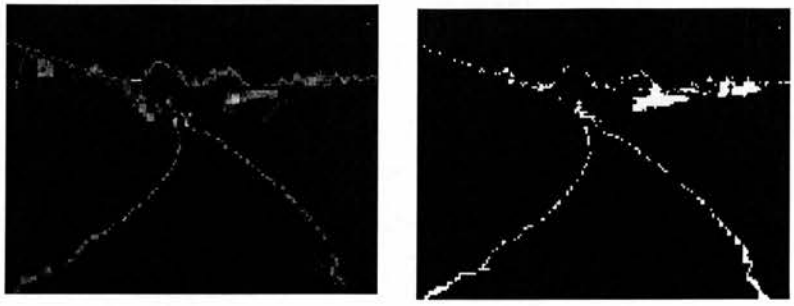
2.3.2.3 Pixelwise Entropy

We are interested in understanding the uncertainty described by $P(\mathbf{x}^L|\mathbf{y})$. Posterior marginal entropies are readily computable from the posterior marginals $P(x_i^L|\mathbf{y})$, i.e. $H(x_i^L|\mathbf{y}) = -\sum_{k=1}^{C} P(x_i^L = k|\mathbf{y}) \log_2 P(x_i = k|\mathbf{y})$[6]. Images displaying these posterior marginal entropies are shown in Figure 2.7, pertaining to the original image shown on

[6]Pérez *et al* [63] also use the calculation of posterior marginal entropies in their determination of "confidence maps".



(a) 0.2021 (b) 0.1094



(c) 0.0487 (d)

Figure 2.7: Posterior Marginal Entropies for the MLP Predictor. (a) compensated pixelwise predictions, (b) ML-TSBN, (c) CML-TSBN. The greyscale is such that black denotes zero entropy, white denotes 2.45 bits. The number underneath each plot is the average pixelwise posterior entropy. (d) Binary image showing misclassified labels (bright) and correctly-classified labels (dark).

the right in Figure 2.2. As expected, the pixelwise entropy is reduced by the use of the TSBN; this is particularly effective for the CML trees. Notice that the pixels which have significant posterior marginal entropy are good indicators of the pixels that are misclassified; this is especially true of the CML-TSBN + MLP combination in Figure 2.7. This property could well be useful information for a later stage of processing.

2.3.3 Conditional Probability

If the model we have developed is a good one, then $P(\mathbf{x}^L|\mathbf{y})$ should ascribe high probability to the ground truth labelling $\mathbf{x}^{L^*}$. Different image models can be compared in terms of the relative values of $P(\mathbf{x}^{L^*}|\mathbf{y})$. In particular we compare the TSBN image models against independent pixel models.

Ignoring spatial correlations we obtain $P_{MLP}(\mathbf{x}^{L^*}|\mathbf{y}) = \prod_i P(x_i^{L^*}|y_i)$ for the MLP

local prediction, and similarly for GMM local prediction. For a TSBN $P(\mathbf{x}^{L^*}|\mathbf{y})$ can be calculated as follows:

$$P_{TSBN}(\mathbf{x}^{L^*}|\mathbf{y}) = \frac{P(\mathbf{x}^{L^*},\mathbf{y})}{P(\mathbf{y})} = \frac{P(\mathbf{x}^{L^*})\prod_i P(y_i|x_i^{L^*})}{\sum_{\mathbf{x}} P(\mathbf{x})\prod_i P(y_i|x_i^L)} \tag{2.11}$$

$$= \frac{P(\mathbf{x}^{L^*})\prod_i L(x_i^{L^*},y_i)P(y_i)}{\sum_{\mathbf{x}} P(\mathbf{x})\prod_i L(x_i^L,y_i)P(y_i)} \tag{2.12}$$

$$= \frac{P(\mathbf{x}^{L^*})\prod_i L(x_i^{L^*},y_i)}{\sum_{\mathbf{x}} P(\mathbf{x})\prod_i L(x_i^L,y_i)}. \tag{2.13}$$

Equation (2.12) follows from Equation (2.11) as $P(y_i|x_i^{L^*}) = L(x_i^{L^*},y_i)P(y_i)$. $P(\mathbf{x}^{L^*})$ and the denominator can be evaluated by the methods outlined in [22].

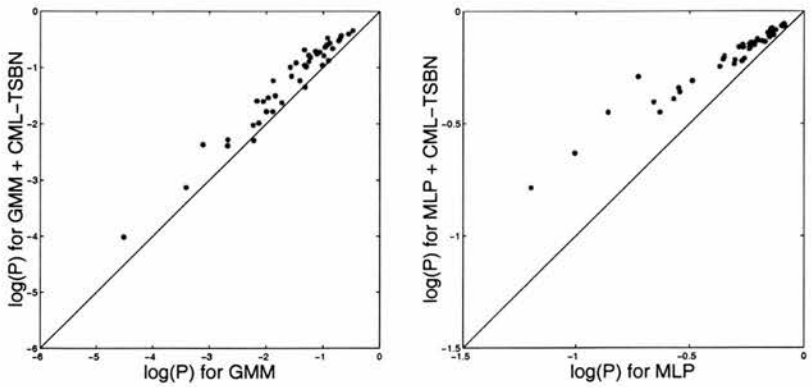
A complexity arises in this calculation when there are pixels which do not have a label. For $P_{MLP}(\mathbf{x}^*|\mathbf{y})$, these pixels were simply ignored. For $P_{TSBN}(\mathbf{x}^{L^*}|\mathbf{y})$, the unlabelled pixels were ignored in both the numerator and denominator of Equation 2.13. As the likelihood is calculated using the Pearl message passing scheme, this is achieved by setting the λ-vector at the appropriate nodes to be the vector of ones. Pearl explains that if there is no evidence at a node then this is equivalent to saying that the probability of an empty set for the node is one, (see [22] for details).

In Figure 2.8 we plot $\frac{1}{N}\log P(\mathbf{x}^{L^*}|\mathbf{y})$ under various models. Panel (b) shows that in all 43 test images, the posterior probability under the MLP+CML-TSBN method is larger than that under the compensated MLP using an independent-pixel model. Panel (a) shows that for a similar comparison using the GMM predictor, the CML-TSBN method is better in 41 out of 43 cases. Notice also the relative scales of the plots, and especially that the GMM model makes confident mistakes on some pixels, thereby dragging down the average posterior probability.

There are a large number of similar plots that can be made. A comparison of the posterior probabilities under the ML-trained TSBN and independent models comes up with roughly equal numbers being better coded under the two models, for both GMM and MLP predictors. The CML-trained TSBN with MLP prediction is better than all other methods on 39 out of the 43 test images; on the remaining four the ML-TSBN with MLP wins.

2.4 Discussion

The results presented here illustrate a number of points:



(a) (b)

Figure 2.8: A Comparison of the Image Coding Costs. The figure shows a comparison of $\log P(\mathbf{x}^{L^*}|\mathbf{y})/N$ for (a) compensated GMM vs GMM + CML-TSBN (b) compensated MLP vs MLP + CML-TSBN.

- We have compared the performance of GMM and MLP pixelwise classifiers on a sizable real-world image segmentation task. The performance of the discriminatively-trained MLP was found to be superior to the class-conditional GMM model. We have also shown that the scaled-likelihood method can be used to fuse the pixel-wise MLP predictions with a TSBN prior.

- We have compared conditional maximum likelihood (CML) training for the tree against maximum likelihood (ML) training on a number of dimensions including classification accuracy, pixelwise entropy and the conditional probability measure $P(\mathbf{x}^L|\mathbf{y})$.

- We have used the EM algorithm to train the ML-TSBN and have observed that the learned parameters do reflect the underlying statistics of the training images.

The problem of evaluating segmentations is an old one, and a full answer may well depend on a decision-theoretic analysis which takes into account the end-use of the segmentation (e.g. for an automated driving system). However, one attractive feature of the TSBN framework is that some aspects of the posterior uncertainty can be computed efficiently, e.g. the posterior marginal entropies discussed in Section 2.3.2.3.

Chapter 3

Position Encoding Dynamic Trees for Image Sequence Analysis

In the first chapter we looked at issues to do with the segmentation of static images. They are static in the sense that they are single images which, although they form part of a set of similar images, do not form a meaningful sequence of images, as do for instance, consecutive frames in a piece of video. In the latter type of dataset, we might expect that having inferred some structure in the previous image, this might provide information about the structure of the current image.

In this chapter we look at how the incorporation of information about previous time frames in a sequence can help us perform image coding and image segmentation. We will tackle the problem within a framework provided by Position Encoding Dynamic Trees.

Some of the work in this chapter is based on work reported in Felderhof, Storkey and Williams (2002), [21]. The static PEDT model and its variational approximation were developed by Drs. Storkey and Williams. The sequential PEDT model was developed in collaboration with Dr. Storkey. The `dpt` code library was used as the basis for the variational calculations in the experiments reported here. The further development of the `dpt` code was done in conjunction with Dr. Storkey.

3.1 Introduction

Position Encoding Dynamic Trees (PEDTs) are a form of hierarchical graphical model with adaptable structure in which nodes of the graph have both class and position

variables. They were developed by Storkey and Williams [73, 71] and are described in Section 3.2.1.

The PEDT model was developed with image sequences specifically in mind. The extra flexibility compared to the Dynamic Tree model was introduced in order to allow it to represent images evolving through time without considerable changes to the tree structure. Despite this original aim, so far the PEDT has only been assessed as a model for static images.

The main objective for this chapter is therefore to develop and analyse the PEDT as a model for image sequences. There are two tasks which we will consider when making this assessment: image segmentation, and image coding. To segment sequences of images we will need to develop a sequential extension of the PEDT model. However, on the other hand we will show that efficient image coding can be done in the original PEDT framework as it stands. As a reflection of this, the chapter looks at these two distinct topics.

In the first half of the chapter (Section 3.4) we address the issue of trying to code image sequences as efficiently as possible using the static PEDT model. This is based on the fact that although coding an image given the highest variational posterior probability (HVPP) tree costs very little, it is too expensive to code the actual HVPP tree. The coding cost in bits per pixel is estimated by calculating the negative log likelihood of the image per pixel under the model. This can be achieved on average over a dataset using an arithmetic coding scheme. When coding a sequence, we calculate the HVPP tree for both the current frame at time t and for the previous frame at time $t-1$. We compare the two tree structures and generate a mapping which describes the high-level node movements between the two frames and which allows us to reconstruct an approximately correct HVPP tree for the image at time t given the HVPP tree at time $t-1$. We code this mapping and code the image under the distribution defined by the approximately correct HVPP tree for the image at time t. Thus in order to decode, the process involves taking the image at $t-1$ which has already been decoded, calculating the HVPP tree for this image, and then reconstructing the approximately correct HVPP tree for the current image which can then be used to decode the current image. We will show that we can achieve estimated coding costs 30% lower than the next best image coding scheme that we could find.

It should be noted that in the experiments in this chapter, the data takes the form of sequences of ray-traced images and their corresponding ground truth label images that

indicate to which of a finite number of classes each pixel belongs. Our main focus will be on the coding cost of the image labels. We concentrate on this because whereas in the previous chapter we were interested in finding the best way to combine the image tree structure with the local image models, we are now more interested in how well the graphical model represents the image structure and how this propagates through time. The label image coding cost is a direct way of measuring this and is not influenced by the specific effects of particular local image models. However, whilst the image coding presented here is restricted to label images, it would be straightforward to extend this to a create a coding scheme for real video image sequences by using an appropriate local image model.

In the second half of the chapter (Section 3.5), we show that we can directly incorporate time series information into an extension of the PEDT model by allowing nodes the possibility of connecting to nodes in the previous time frame. Standard probabilistic models for sequences include Hidden Markov models (HMM), see e.g. [64], and Linear Dynamical systems (LDS) or Kalman Filters, see e.g.[77]. These two models have attractive polynomial-time algorithms but have a relatively simple latent variable structure. Our aim is to develop more powerful multiple-cause latent variable graphical models for sequential problems, particularly image sequence analysis.

The key idea in the sequential modelling is that information at time $t-1$ should be predictive of the information at time t. Within the probabilistic framework, this means that the model for an image at time t is influenced by previously observed evidence. As with the HMM and LDS models we use a Markov-type model for the evolution of the state variables. Within the PEDT framework it is not possible to compute exactly the predictive distribution of the model at time t given data up to time $t-1$, but a variational approximation can be used.

We will assess the sequential PEDT as a model for image sequences in two ways. Firstly we will calculate the pixelwise classification accuracy on an image segmentation task and compare the performance of the static and the sequential PEDT models and also that of simpler pixelwise classifiers on this task. Secondly, we measure how predictive the sequentially updated prior PEDT distribution is for unseen ground truth image labels by calculating the estimated coding cost of the unseen ground truth labels at time t given all the raw images up to time $t-1$.

The sequential PEDT model is, in itself, a model for a sequence of images and can be used as a valid coding scheme. Indeed we show that it would perform better

than all but one of the image coding formats that we could find as a comparison[1]. However, here we are really interested in the image coding cost as a measure of how good the model is. The better that a probabilistic model represents the true underlying distribution from which the data is generated, the lower the estimated coding cost[2]. If we were seriously interested in trying to code the image sequence we would not use this approach as it does not use all the information available; the image labels are not used in the calculation of the variational approximations for each frame, they are just used to measure how good the sequentially updated PEDT prior distributions are as a model for the image label sequence.

There is of course a considerable literature on the analysis of image sequences in computer vision. Our work has some similarities to work on tracking (see, e.g. [8]). The work of Blake and Isard and others has essentially developed non-linear Kalman filtering methods for the tracking problem. Usually this is carried out using a contour-based shape representation. While contour-based methods are useful for the tracking of geometrical objects, they require good initialisation for the contour and are less well-suited to problems where the objects have highly variable geometries. In contrast we use a region-based approach which should be robust to objects whose shape changes continually and those which merge and split into discontinuous components, for instance, clouds. Note also that contour-based tracking needs to represent explicitly the number of objects moving; again, the PEDT framework does not require this.

The static PEDT was developed as an alternative to Markov Random Field (MRF) image model. The generalization of the PEDT to sequential data mirrors the similar generalization of MRFs, as carried out e.g. by [55, 29, 39]. The problem of making inference calculations in graphical models which are linked through time has been considered by a number of authors. The work on the Helmholtz machine through time is one example, [32]. However, see also [26, 51, 68, 84]. Essentially, each of these approaches propose approximation schemes for dealing with the exponential complexity of correct inference in factorial HMMs.

There is much work on motion estimation in the literature, see e.g. [75]. This differs from our work in that it is involved with grouping local motion estimates derived from

[1]The only method that performs better (excluding the HVPP Tree Mapping method outlined above) consists of making a `tar.gz` archive of the image bitmaps. This is clearly not a method which allows for images to be streamed unlike the sequential PEDT and the HVPP Tree Mapping approach.

[2]Since we estimate the coding cost by calculating the negative log likelihood of the data, this is equivalent to the fact that the KL-divergence between the model and the underlying distribution has to be greater than zero.

optical-flow computations. In order to incorporate optical-flow observations into our model it would be necessary to have a velocity variable on each node. We also note that both motion and intensity cues can be used for segmentation, see e.g. [74].

The structure of the remainder of the chapter is as follows: Section 3.2 will present the background theory, in Subsection 3.2.1 we describe the static PEDT model and in Subsection 3.2.2 we go on to motivate and discuss further the HVPP tree mapping method and the sequential PEDT model as two possible methods of including sequential information in the PEDT framework; the data that is used for the experimental work is discussed in Section 3.3; the experiments using the HVPP tree mapping algorithm are described in Section 3.4; and the experiments using the sequential PEDT model are described in Section 3.5; we conclude with a discussion in Section 3.6.

3.2 Theory

As just mentioned, this section is divided in two. The first part outlines the basic PEDT model for single images and the second part explores how we can use this framework to deal with sequential information.

3.2.1 Static PEDT Model

This is an introduction to the static PEDT. In 3.2.1.1, we outline a variational approximation which we will use to perform inference within the framework and Section 3.2.1.2 describes the KL-divergence minimisation update equations for the variational procedure. This is a review of the technique and full details can be found in [70, 71].

Firstly, we look at the overall generative model for the position encoding dynamic tree. There are four parts to it, the network nodes, the dynamic structure, the node classes and the node positions.

The model will be described using the belief network formalism. The graphical model will have a number of layers of nodes. The set N of n network nodes is denoted by $\{1,2,\ldots,n\}$. The nodes are organised into layers $\{1,2,\ldots,h\}$, where 1 denotes the top layer and h, the leaf node layer. We use L^H to denote the set of nodes in layer H. The network nodes are connected together in a way that represents the structural relationships in the image. We generally use the superscript notation to represent a set

of random variables. For example X^B represents the set $\{X_i | i \in B\}$. For the state of all nodes we drop the N: $X^N = X$. We will also use the shorthand X' to denote X^{L^B} where $B = \{1, 2, \ldots, h-1\}$. In other words X' represents the set of X_i corresponding to all nodes i not in the bottom layer.

The network connectivity is denoted by a matrix Z of indicator variables z_{ij}. Setting $z_{ij} = 1$ represents the existence of the connection between child i and parent j. We also include the variable z_{i0}, which is set to 1 if and only if i is a root node (i.e. it has no parent). Because Z must be a tree structure, i can have at most one parent, and so given i, $z_{ij} = 1$ for one and only one value of j and is zero otherwise. Furthermore, the constraint that all nodes can only have parents in the layer above means z_{ij} can only equal 1 if node j is in a layer one higher than node i (or $j = 0$).

The variable Z can represent all the possible tree structures which are of interest. We define a prior distribution over tree connectivity, which is taken to be of the form

$$P(Z) = \prod_{i=1,j=1}^{n} \gamma_{ij}^{z_{ij}}, \qquad \sum_j z_{ij} = 1, z_{ij} \in \{0, 1\}. \tag{3.1}$$

γ_{ij} is the probability of i being the child of j (we will say i chooses parent j with probability γ_{ij}). Generally it is assumed that this probability is uniform across all parents $j \neq 0$, expressing node interchangeability. Interchangeability is not a theoretical necessity (and so a different form of γ_{ij} could be used). However, because we will allow each node to take any position, interchangeability does make the later computational optimisations easier as it introduces a symmetry which means that any one of the symmetric optima can be found, rather than one particular optimum.

Each node i has a value $\mathbf{r}_i$ which represents the position of the centre of the node. For the root nodes, we suppose the position is taken from a broad Gaussian distribution. Gaussian distributions are also used to specify the centres of the child objects given the parent. Formally,

$$P(\mathbf{r}_i | \mathbf{r}_j, z_{ij} = 1) = \frac{1}{|2\pi\Sigma_{ij}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{r}_i - \mathbf{r}_j)^T (\Sigma_{ij})^{-1} (\mathbf{r}_i - \mathbf{r}_j)), \tag{3.2}$$

where $d = 2$ is the dimensionality of the space, and Σ_{ij} is a covariance used to represent the order of magnitude of the object size. The overall distribution is given by

$$P(R|Z) = \prod_{ij|z_{ij}=1} \frac{1}{|2\pi\Sigma_{ij}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{r}_i - \mathbf{r}_j)^T (\Sigma_{ij})^{-1} (\mathbf{r}_i - \mathbf{r}_j)\right). \tag{3.3}$$

At the lowest level (layer h) we will need to connect the nodes to the pixels. For this reason we set the positions of the bottom layer of nodes to be given by the pixel

positions, and then use the distribution $P(Z,R'|R^{L^h})$ as the prior over positions and connectivity.

The other remaining important concept is the class label of each of the nodes. This represents the type of object or texture which the node is representing. Suppose there are C possible classes to which nodes can belong. We denote the class of a node i by X_i and use the indicator $x_i^k = 1$ to represent the state that X_i is in class k, (note that $x_i^j = 0$ for all $j \neq k$). We represent the probability of a node i being in state k given its parent j is in state l by a conditional probability table P_{ij}^{kl}. We make a simplifying assumption that this conditional probability is independent of the positions of the parent and child nodes. This is not entirely realistic, but is a useful first approximation. Hence $P(X|Z)$ can be written as

$$P(X|Z) = \prod_{ijkl} (P_{ij}^{kl})^{x_i^k x_j^l z_{ij}}. \tag{3.4}$$

The intensity of each pixel will depend on the class label of the node it is connected to. For each bottom layer node i there is a pixel in position $\mathbf{r}_i$, we assume its value is entirely dependent on the node state X_i through the distribution $P(Y_i|X_i)$, where Y_i denotes the pixel RGB intensity. The overall pixel model is

$$P(Y|X,R) = \prod_{i \in L^h} P(Y_i|X_i). \tag{3.5}$$

The full definition of the prior distribution is given by the following. First $P(Z)$ is defined using (3.1) and $P(R|Z)$ using (3.3) to give us $P(R,Z)$. We then impose the condition fixing the bottom layer nodes to the pixel positions to get $P(Z,R'|R^{L^h})$. Using Equation (3.4) gives $P(X,R',Z|R^{L^h}) = P(Z,R'|R^{L^h})P(X|Z)$. Lastly $P(Y|X)$ is of the form (3.5), giving the final joint distribution $P(Y,X,R',Z|R^{L^h})$. This fully specifies a position encoding dynamic tree model.

The position encoding dynamic tree model gives the prior probability distribution for the image through the use of a large hierarchical latent variable model. To use it, we need to find out what the posterior probability distribution of the latent variables is given a specific image. In other words we condition on the fact that we have pixel values Y, and try to discover the distribution over Z,X and R' that results. This distribution will give us a set of possible good interpretations of the connectivity, content and positioning of objects in the image.

3.2.1.1 Variational Inference in the PEDT

To use the PEDT model, some approach for doing Bayesian inference is needed. Were this model a tree structured belief network, we could use techniques such as belief propagation [62] to calculate exactly the required posterior distribution $P(X, R', Z|Y, R^{L^h})$. However because the PEDT is not of this form, and because other approaches such as the junction tree method are inappropriate because of the large clique size of the triangulated network, exact calculations are not feasible as there is an exponential explosion in the number of tree structures that need to be considered [3]. In one-dimensional problems, such as context-free grammars, this problem can be overcome using dynamic programming techniques however unfortunately in cases such as this where the branching factor is higher this approach does not work. Instead the graphical structure of the distribution is used to develop variational approximation methods to the posterior.

Variational methods are one of a number of approaches for approximate inference in networks. Other approaches include Markov chain Monte-Carlo methods [27], where a Markov chain is constructed which has the desired posterior as the limit distribution. Samples from this Markov chain are used as an approximation for the posterior. Monte-Carlo and annealing approaches for dynamic trees were investigated in [79]. Unfortunately obtaining Monte-Carlo samples for a network of this size would take longer than is affordable. The variational approach, on the other hand, fits an approximating distribution to the true posterior. The variational distribution must be chosen so as to allow tractable calculations so that a variational approximation to the posterior can be obtained in a reasonable time. At the same time the fit should be as good as possible so that the approximation is not a poor one. The choice of variational distribution is a non-trivial exercise, further discussion of this issue can be found in [68].

Below we give an outline of the variational method and the form of variational distribution which is used. The resulting update equations used to fit the distribution to the posterior are given in Section 3.2.1.2.

The variational approach involves approximating the posterior distribution with a factorising distribution of the form $Q(Z)Q(R)\ Q(X|Z)$, where $Q(Z)$ is the approximating distribution over the Z variables, $Q(X|Z)$ is the approximating distribution over the states, and $Q(R)$ is an approximating distribution over the node positions. The KL

divergence between the approximation and the true posterior is of the form

$$KL(Q||P) = \int dR' \sum_{Z,X} Q(Z)Q(X|Z)Q(R') \log\left(\frac{Q(Z)Q(X|Z)Q(R')}{P(Z,R',X|R^{L^h},Y)}\right). \tag{3.6}$$

To choose good forms for the Q's the Kullback-Leibler divergence $KL(Q||P)$ between the $Q(Z)$ $Q(R)Q(X|Z)$ distribution and the true posterior should be minimised. In fact the approximate distributions which are used take the form of a dynamic tree model, and give propagation rules which are efficient and local.

Introducing Θ to represent the conditional probability table parameters, we define

$$\begin{aligned} L_{var}(Y|\Theta) &\stackrel{def}{=} -\int dR' \sum_{Z,X} Q(Z)Q(R')Q(X|Z) \log \frac{Q(Z)Q(R')Q(X|Z)}{P(Z,R,X,Y|\Theta)} \\ &= -KL_\Theta(Q||P) + \log P(Y|R^{L^h},\Theta) + \log P(R^{L^h}) \end{aligned} \tag{3.7}$$

in which we use the fact that $P(Z,R,X,Y|\Theta) = P(Z|\Theta)P(R|Z,\Theta)P(X|Z,\Theta)P(Y|X,\Theta)$. $L_{var}(Y|\Theta)$ is the variational log likelihood. The fact that $KL_\Theta(Q||P) \geq 0$ implies the log likelihood $\log P(Y|R^{L^h},\Theta)$ is lower bounded by the variational log likelihood minus the additive constant $\log P(R^{L^h})$. Minimising the KL divergence maximises the variational log likelihood.

The forms of each of the approximating distributions needs to be finalised. We use a $Q(Z)$ and $Q(R')$ of the form

$$Q(Z) = \prod_{ij} \alpha_{ij}^{z_{ij}} \text{ and } Q(R') = \prod_{i \in N \setminus L^h} \frac{1}{(2\pi)^{d/2}|\Omega_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{r}_i^T - \boldsymbol{\mu}_i^T)\Omega_i^{-1}(\mathbf{r}_i - \boldsymbol{\mu}_i)\right) \tag{3.8}$$

where the α's are probabilities and where $\boldsymbol{\mu}_i$ and Ω_i are position and covariance parameters respectively, all of which need to be optimised. Ω_i is assumed to be diagonal. Lastly the $Q(X|Z)$ is a dynamic tree distribution of a form identical to that used in [70]:

$$Q(X|Z) = \prod_{ijkl} (Q_{ij}^{kl})^{x_i^k x_j^l z_{ij}}. \tag{3.9}$$

Again Q_{ij}^{kl} are parameters to be optimised.

This distribution is chosen because local update propagations can be obtained through KL divergence minimisation. Hence the model can be tuned to the posterior efficiently. Also the marginal values of this distribution can be obtained straightforwardly. In addition this variational form is guaranteed to give better approximations than simpler ones such as a mean field approximation [70]. The difference between

this approach and the mean field approach stems from the fact that dependence on Z is allowed in (3.9), and that dependence of a node i on parent nodes is allowed, rather than forcing $Q(X|Z)$ to take a factorised form. Because there are significant dependencies between child and parent in the prior we would expect some dependence to remain in the posterior. The mean field approach ignores that dependence, whereas this more structured approximating distribution allows some of those dependencies to be captured.

3.2.1.2 Update Equations

We want to minimise the KL divergence (3.6), with the forms of approximate distribution given in the last section. This must be done subject to the constraints that $\sum_k Q_{ij}^{kl} = 1$ and $\sum_j \alpha_{ij} = 1$ since these quantities are probabilites. A set of Lagrange multiplier terms corresponding to these constraints are added to Equation (3.6). We set the derivatives to zero. Solving this gives a set of update equations. The full derivations are given in [73]. Below, the update equations for the conditional probability tables, position variables and tree connectivity of the variational distribution are given.

Minimising the KL divergence gives us a set of update equations. Given α, let m_i^k be given recursively from the top down by

$$m_i^k = \sum_{jl} \alpha_{ij} Q_{ij}^{kl} m_j^l. \tag{3.10}$$

Then m_i^k is the marginal probability of node i being in class k under the variational distribution. Again given α we find that minimisation of the KL divergence gives

$$Q_{ij}^{kl} = \frac{P_{ij}^{kl} \lambda_i^k}{\sum_{k'} P_{ij}^{k'l} \lambda_i^{k'}} \text{ where } \lambda_i^k = \prod_{c \in c(i)} \left[\sum_g P_{ci}^{gk} \lambda_c^g \right]^{\alpha_{ci}}, \tag{3.11}$$

where $c(i)$ is used to denote the possible children of i, in other words the nodes in the layer below that containing node i. Given α, all the Q values can be updated by making a single pass up the tree to calculate the λ values, and then calculating the Q values.

The update equations for the positions given the α values, take the following forms

$$\boldsymbol{\mu}_i = \sum_k \alpha_{ki} (\Sigma_{ki})^{-1} \boldsymbol{\mu}_k + \sum_j \alpha_{ij} (\Sigma_{ij})^{-1} \boldsymbol{\mu}_j, \tag{3.12}$$

$$(\Omega_i)_{pp} = \frac{1}{\sum_j \left(\alpha_{ij} (\Sigma_{ij})_{pp}^{-1} + \alpha_{ji} (\Sigma_{ji})_{pp}^{-1} \right)} \tag{3.13}$$

where we have assumed that both Σ and Ω are diagonal. The equations for $\boldsymbol{\mu}$ involve a sum across all possible parents and a sum across all possible children, and hence need to be iterated until suitably converged.

Lastly the connectivity needs to be considered. For fixed parameters in $Q(X|Z)$ and $Q(R')$ of the forms given above, we obtain

$$\begin{aligned}\alpha_{ij} &\propto \gamma_{ij}\exp(-\Psi_{ij})\exp(-\Phi_{ij}) \\ \text{where } \Psi_{ij} &= \sum_l m_j^l[\log\sum_k P_{ij}^{kl}\lambda_i^k] \\ \text{and } \Phi_{ij} &= \frac{1}{2}(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)^T\Sigma_{ij}^{-1}(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)+\frac{1}{2}\mathrm{Tr}(\Sigma_{ij}^{-1}\Omega_i)+\frac{1}{2}\mathrm{Tr}(\Sigma_{ij}^{-1}\Omega_j).\end{aligned} \tag{3.14}$$

The constant of proportionality is found by normalisation as $\sum_j \alpha_{ij} = 1$.

3.2.1.3 Optimisation Process

The above equations give all the necessary update rules. The whole optimisation process involves an outer loop optimising the $Q(Z)$ values and an inner loop containing up and down passes of the $Q(X|Z)$ optimisation and a number of passes of the $Q(R)$ optimisation. The KL divergence can be calculated up to an additive constant, and so can be used as an explicit objective function and be monitored accordingly. A termination criterion based on the change in the KL divergence can be set. However the calculation of the KL divergence itself is one of the most costly parts of the optimisation, so in the experiments we used a fixed number of iterations, and only calculated the KL divergence at the end of each inferential run.

3.2.1.4 Highest Variational Posterior Probability

The tree $\mathbf{Z}^*$ which maximises $Q(Z)$ after the variational distributions have been optimised is called the tree with the highest variational posterior probability (HVPP). It is effectively the maximum a posteriori (MAP) tree under the variational approximation. Likewise we can pick the HVPP solution for any set of random variables, such as the classification labels or positions. This is useful for visualisation.

3.2.2 Introducing Sequential Information

Having introduced the PEDT framework in the previous section, we can see that we are now in the position to perform inference and learning with our model for individual

images. The question that we really want to address is how can we make use of the flexibility of the model to allow sequential information to help to improve the overall performance of the model on images which form part of a sequence?

We will look at two approaches here. The first is specifically designed to reduce the coding cost of a sequence of images and involves making mappings between the HVPP trees generated by the static PEDT for consecutive images. The second is slightly more general in that it includes the sequential nature of the data directly into the probabilistic model and as such allows it to be used for both image coding and also image classification. Both of these approaches are explored in the experiments which were carried out.

3.2.2.1 Motivating the HVPP Tree Mapping Algorithm

The aim is to use the fact that the images occur in a sequence to improve the cost of coding them. The HVPP tree mapping algorithm generates a mapping between the nodes in HVPP tree structures for consecutive frames. It then codes the movement of the mapped nodes between frames in order to give a cheap method of coding the images themselves. It is fully described in Section 3.4.2.

The coding cost is estimated by calculating the negative log likelihood of the image under the model. We use this because by employing an arithmetic coding scheme we should be able to achieve a coding cost arbitrarily close to this with the probabilities given by the model, see [15]. Arithmetic coding is a practical, though somewhat computationally intensive, method with which to code images. One would code the first pixel under the model, instantiate the node with its coded value, propagate the changes to calculate the distribution given the value of pixel 1, then code the next pixel given the first pixel and so on. Since, $\log P(X_1, X_2, \ldots, X_n) = \log P(X_1) + \log P(X_2|X_1) + \ldots + \log P(X_n|X_1, X_2, \ldots, X_{n-1})$, we can see that the negative log probability of the image is an estimate for the cost of coding the image one pixel at a time using the arithmetic scheme.

The HVPP tree mapping approach is inspired by the results of some preliminary experiments described in the experimental section. As will be shown later, on the sequence of 50 test images the estimated coding cost of the images given the HVPP

tree structure for the images[3] was approximately 0.007bpp[4]. This is low. For four class images, such as these, we could naïvely expect the coding cost to be around 2bpp. In the experimental data section (Section 3.3), we will show the coding costs of the images using a variety of different image formats and algorithms, the lowest of which is about 0.14bpp which would thus be the target that we are aiming to beat. The problem with coding the images using the prior CPTs and the HVPP tree structure is that it also requires coding the tree structure which is shown to cost 2.19bpp. This is far too expensive. However, since the images occur in a sequence, if when decoding the images we have already reconstructed the previous frame at time $t-1$ then we can calculate the HVPP tree structure for this frame at no cost at all. If we use this *one step late* HVPP tree structure rather than the correct one, then the coding cost of the image sequence comes out at 0.291bpp.

This shows us a way forward. We can take advantage of the fact that the variational posterior distribution of our model is a meaningful representation of the structure in the image and that the structure of two consecutive images in a sequence is similar. When coding the image sequence we calculate the variational posterior distribution for both the image at time $t-1$ and at time t. If we compare the HVPP tree structure and class labels between the two frames to generate a mapping between matching nodes in the two frames and then calculate the way that the matching nodes move between the two frames, then we can then code the node movements for all the nodes in a certain number of the highest layers in the tree. We would hope that this closely represents the way that objects in the two images move between frames. Thus when we come to reconstruct the image sequence, we can calculate the HVPP tree for the image that we already have at time $t-1$ and use the high-level coded node movements to propagate the tree structure forward into the current time frame.

We can take advantage of the fact that the coding cost under the PEDT model is reasonably insensitive to the exact connectivity of the tree. As long as distinct objects in the image all connect into the same tree structures, the internal connectivity of the tree has a relatively unimportant effect on the overall coding cost. Mistakes where

[3]This is the coding cost given the variational prior distribution for the class labels which is fixed for all the nodes in each layer. It is also possible to measure the coding cost given the HVPP tree structure using the variational posterior probability distribution which is neglible (5×10^{-19}bpp). However this isn't helpful since the use of the posterior CPTs would require having to code them as well and this needs K real numbers per node in the graph where K is the number of classes in the image. There are more nodes in the graph than there are pixels so needless to say, this is prohibitively expensive.

[4]*bpp* is an abbreviation for *bits per pixel*.

a particular tree structure connects to areas of different image label class type are, however, expensive.

We will show that using the coded node movements we can build from the one-step late tree structure, a tree structure which is closer to the correct HVPP tree structure than the one-step late tree structure itself. We will demonstrate that this is the case because if it is indeed so, then we will be able to code the image more efficiently using the mapped tree structure than we can do with the one-step late tree structure. We expect that the coding cost using the actual HVPP tree structure from time t will be a lower bound on what we can achieve using this approach.

The aim of the experiments will therefore to be show that we can incorporate the sequence information in this fashion to generate a useful image coding algorithm. We would expect to improve on the one-step late image coding cost and hopefully be able to code the images more cheaply than the coding cost that we have set as a target (0.14bpp). Note that full details of the HVPP mapping algorithm can found in Section 3.4, the discussion here is only intended to provide a motivation for the work.

3.2.2.2 PEDT Sequence Model Background

Another way to introduce sequence information is to extend the PEDT model itself. Position encoding dynamic trees have been designed to provide the type of positional and structural information needed for modelling sequences, but there are still a number of forms that sequence models can take and a number of sequence-specific issues, such as occlusion, which will impact on the choice of model.

We will consider the requirements of a PEDT sequence model. The form the model should take is described along with some model related issues and difficulties. This eventually leads to the final form of the model described and defined in Section 3.5.1.

Because of the large size of any model of general image forms, most methods for image sequence modelling focus on the filtering problem. They are interested in developing and working with a conditional latent model of the next time frame given the previous one, and given the current image information. We will also be focussing on the filtering problem here, and will need to specify a predictive model of the PEDT at time t given the state at time $t-1$. We also need to specify how an image at time t would be generated from the PEDT, and hence how the PEDT at time t is updated when we observe the image at time t.

The basic characteristics of the PEDT are node labels, positions and tree struc-

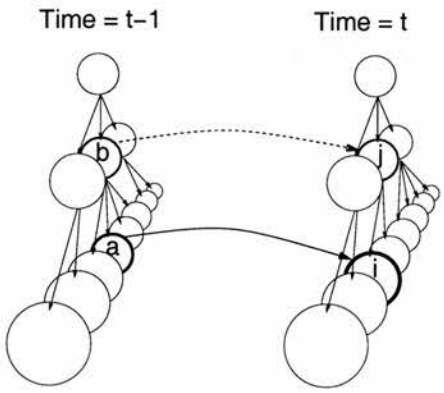


Figure 3.1: A Schematic Diagram Illustrating Possible Node Connections Through Time in the Sequential PEDT Model. The prior probability distribution, P_{ij}^{kl}, at time t is updated by allowing node i to connect to nodes in the previous time frame. The update also depends on the probability that node j, the parent of node i, connects to nodes in the previous frame. The indexes, i, j, a, b correspond to the labels used in equation 3.21 which describes the sequential update scheme in more detail.

tures, and these are used to describe the image contents. As we progress through the sequence, the positions of objects will change. Some objects might start to become occluded by other objects, parts of some objects might appear from behind others etc. We can be sure of a significant continuity of the images and so nodes in certain positions in one frame are likely to have the same class labels as nodes in similar positions in the next time frame. What we need is some method of making a loose correspondence between nodes in the two time frames, which will then allow information to be shared through time. This correspondence will map a node which represents a certain object or part object to a node in the next time frame which corresponds to the same object or part object. The correspondence needs to be loose and flexible because certain parts of objects might be occluded or have disappeared from the field of view, or might have got larger or smaller. There is no requirement for a one to one mapping. Also it is important that the model allows all areas of the image to be represented. Variational methods for the PEDT can result in only a small number of all the possible nodes in a layer being distinct from one another if no other objects need to be represented. It is important that such a contraction in representation does not limit the representation at the next time step. The distribution over maps used here and given in Section 3.2.2.2 satisfies these requirements.

(a) (b) (c)

Figure 3.2: A Training Image. Subfigure (a) shows one image from the training dataset, Subfigure (b) shows the corresponding image labels, (c) shows the key for the image label colours.

Once a map between nodes at two adjacent time frames is obtained, a predictive distribution for the PEDT at the later time frame can be defined taking into account not only the usual static PEDT prior information but also the information from the previous frame. As nodes represent object parts, we would expect class label information to be passed from the past node to its corresponding node. For example, the node at the second time frame will generally have the same class label, and its position (relative to its parent if it has one), should be similar in both time frames, representing the fact that typically inter-frame movements should be small. Details of the implementation of the sequential PEDT model are described in Section 3.5.

3.3 Experimental Data

The discussion of the experimental work is divided into three sections. In this section, we will introduce the data on which the experiments are performed. Section 3.4 will describe the detail of the experiments using the HVPP mapping algorithm to minimise the coding costs of an image sequence. Section 3.5 will describe the use of the sequential PEDT model for image coding and classification.

We use sequences of ray-traced images in the following experiments. It is a difficult and time-consuming procedure to produce reliably labelled real world images. This is true of single individual images. The task is even more cumbersome when dealing with large sequences of images.

The use of artifically generated image sequences gives us access to an unlimited supply of training and test data for which it is straightforward to generate precisely the ground truth image labels.

We have a set of 50 training and 50 test images showing two balls rolling around on a flat plane. Each 8-bit colour image is 80×120 pixels in size. Figure 3.2 shows

Image format or algorithm	Program used for coding	coding cost /bpp
ppm	povray	24.102
tga	povray	24.015
multiframe gif	imagemagick	9.1921
bmp	imagemagick	1.3020
average pixelwise entropy		1.1531
jpg	imagemagick	1.0609
mng	imagemagick	0.8438
png	povray	0.5744
simple dynamic pixel model		0.2738
jpeg-ls	locoe	0.2492
bmp.gz	imagemagick/gzip	0.2116
tar.gz of bmp	imagemagick/tar/gzip	0.1378

Table 3.1: The Cost of Coding the 50 Images in the Test Set Using a Variety of Image Formats and Algorithms.

an example image and its corresponding ground truth labelling. The labels are "background", "floor", "ball 1" and "ball 2". The image is one frame of the training sequence. Animations of the full training and test sequences and their corresponding ground truth labels can be found online[5].

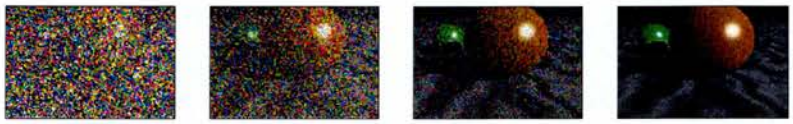
We will be using the coding cost of the label images as a measure of performance. In order to provide a sense of scale for these measurements, Table 3.1 shows the average coding cost of the 50 test images using a number of image formats or coding algorithms. There are a number of points to note from the table. The image formats included in the table are all, with the exception of the jpg format, lossless methods. The images can be perfectly reconstructed from the encoding. The jpg format is included because it is a common format which usually achieves high compression rates. Some of the numbers in the table seem actually just to serve to illustrate that the programs generating the image encodings are probably not performing optimally. For instance, the lossless jpg images are somewhat smaller than the lossy version. Also, the mng format is supposedly a version of png which has been specially adapted for image sequences and yet it performs worse in compression terms than its static counterpart.

[5]Go to: http://www.dai.ed.ac.uk/homes/stephenf/pedt

All the schemes in the table except for the `tar.gz` images allow for the data to be streamed. The images can be reconstructed on an image-by-image basis given the data already sent without having to wait for the full dataset to arrive. The `tar.gz` image coding cost is included because it provides the best compression rate for the images found. The average pixelwise entropy of the test images is fairly self-explanatory, it is the entropy of the distribution over the classes found by averaging the frequency of occurrence of each class over all the pixels in every image in the test set. For this test set the frequencies of class occurence are (0.085,0.767,0.075,0.073) for classes 1 to 4 respectively. Finally, the simple dynamic pixel model is a straightforward model for the data which says that the value of each pixel label is generated by it either keeping the same value as it had in the frame before with probability p_{same} or with probability $1 - p_{\text{same}}$ it choses a new value which it does by sampling from the overall class label frequency distribution. The coding cost shown is the negative log likelihood of the test set under this model using the best value of p_{same}.

In the coding cost experiments we are only be interested in using the sets of image labels. However, when it comes to the image classification we will need to deal with the ray-traced data itself. In this situation a disadvantage of using ray-traced images is that with simple examples the pictures lack the complexity and detail of real world images. Moreover, when we are trying to ascertain the benefits of using the PEDT framework, if the problem is too simple then a purely local image model may perform very well on pixel-wise classification and it will be hard to assess the benefits of a more sophisticated approach. This is even more pronounced when we come to deal with assessing the fusion of temporal information within the PEDT framework in comparison with both an ordinary PEDT model and simple local models. It is difficult to determine the effects of the extra work if the simpler models already perform well. In order to address these issues, we perform all the experiments with a number of levels of independent Gaussian noise added to each pixel. The performance of each of models can thus be assessed in situations of varying degrees of difficulty.

Figure 3.3 shows the same training image as illustrated in Figure 3.2 but to which various amount of noise have been added. Each image has RGB values on a scale from 0 to 1. The independent Gaussian noise is added to each pixel with variances of 1, 10^{-1}, 10^{-2}, 10^{-3}. This operation is performed using the `imnoise` function from the image processing toolbox in MATLAB. Noise added to a pixel value that takes it outside the RGB colour cube is truncated.



(a) (b) (c) (d)

Figure 3.3: A Training Image with Added Gaussian Noise. In each of the subfigures respectively, the added Gaussian noise has a variance of: (a) 1; (b) 10^{-1}; (c) 10^{-2}; (d) 10^{-3}.

3.4 HVPP Tree Mapping Image Coding Cost Experiments

The aim of the experiments in this section is to demonstrate that the variational posterior distribution has useful structure of which we can take advantage. More specifically, the structure of the variational posterior distributions for consecutive frames is similar enough and in some sense representative enough of the underlying structure in the images that a comparison of the differences between the two distributions allows us to reduce the coding cost of the image labels.

Before making at a detailed examination of this, we will first look at a preliminary experiment which, as mentioned in Section 3.2.2.1, motivates the rest of this experiment. The preliminary investigation shows that coding images given the HVPP tree structure for that image is very cheap but coding that tree structure is too expensive. It also shows that the cost of coding images using the HVPP tree structure for the previous frame (the one step late tree) is high but in the range that we are interested in. This is points the way to the HVPP mapping algorithm (Section 3.4.3) in which we find and code a mapping between the tree structures for the current frame and the previous frame which allows us to construct an approximately correct tree structure which we can use to efficiently code the image without having to pay the cost of coding the tree structure itself. The casual reader may want to skip the tedious details of the description of the experiment and move straight to the results which are presented in Section 3.4.4.

3.4.1 Preliminary Coding Cost Calculations

We trained the PEDT with 40 iterations of the training cycle as described in Section 3.2.1.3 using the test set of 50 image labels to give us the prior distribution, $Q_{\text{prior}}(X|Z)$, for the sequence. This distribution is uniform in the sense that the CPTs

$-\log(Q^t_{\text{post}}(\mathbf{X}^*_t\|\mathbf{Z}^*_t))/N$	$-\log(Q_{\text{prior}}(\mathbf{X}^*_t\|\mathbf{Z}^*_t))/N$	$-\log(Q_{\text{prior}}(\mathbf{X}^*_t\|\mathbf{Z}^*_{t-1}))/N$	$-\log(Q^t(\mathbf{Z}^*_t))/N$
5.07×10^{-19}bpp	7.91×10^{-3}bpp	2.91×10^{-1}bpp	2.19bpp

Table 3.2: Preliminary Coding Cost Results. The table shows the average coding cost of 49 images from the test sequence. From the left to right the entries are:(i) the cost of coding the image labels under the full variational posterior distribution for X at time t given the HVPP tree structure at time t;(ii) the cost of coding the image labels under the variational prior distribution for X given the HVPP tree structure at time t;(iii) the cost of coding the image labels under variational prior distribution for X given the HVPP tree structure for the frame at time $t-1$;(iv) the cost of coding the HVPP tree structure under the variational distribution for Z at time t.

are held fixed across each layer. With this initialisation, we calculate the static PEDT variational approximation to posterior probability distribution for each image. We compare the cost of coding various quantities using the computed distributions. The results are displayed in Table 3.2.

The first quantity that we look at is the log likelihood of the ground truth image labels at time t which is denoted by $\mathbf{X}^*_t$ under the variational posterior distribution for that frame, $Q^t_{\text{post}}(X|Z)$, given the HVPP tree structure for the frame, $\mathbf{Z}^*_t$. As is to be expected, this is neglible. However, it is not of any practical use from a coding point of view as it requires knowledge of both the CPT for each link in the variational posterior distribution for the class labels, $Q^t_{\text{post}}(X|Z)$ and the HVPP tree structure for that frame. Neither of these can be coded efficiently. The second codes the ground truth image labels at time t under the uniform prior distribution for the sequence given the HVPP tree structure for time t. Thirdly, we calculate the coding cost of the image labels at time t under the variational prior distribution for the class labels given the HVPP tree structure for time $t-1$ which is available for no extra cost given that we have already reconstructed the previous image in the sequence. The final result is an estimate of the cost of coding of the HVPP tree structure. It is given by the log probability of the HVPP tree structure $\mathbf{Z}^*_t$ under the variational posterior distribution for the structure for that image, $Q^t(Z)$. This cost demonstrates that coding the tree structure is not something that we can afford to do.

Note that the sequential coding schemes (i.e. those which make use of information from having already reconstructed the previous image) require the first image of sequence to have been coded by some other means already and when a PEDT calcula-

tion is involved there is also a cost for coding the CPTs for variational prior distribution $Q(X|Z)$. Neither of these quantities has been included in the results above because the costs are fixed for the sequence and as such their relative significance in terms of the number of bits per pixels that they cost will vary depending on the length of test series. However, for instance, for this test sequence we can see that we could always code the first image as a gzipped bitmap in which case we can expect that it might cost 0.2116 bpp (as shown in Table 3.1) and we can represent the required CPTs with $h \times k$ real numbers where h is the number of layers of nodes in the tree and k is the number of classes. In this case, the cost is $8 \times 4 \times 16$ bits for the whole sequence. Thus as an indication of the magnitude of the total extra cost, if you had a sequential method which coded the 49 images of the test set excluding the first in 0.1 bpp then the full cost of coding the sequence including the fixed extra costs of coding the first image and prior CPT information would be 0.103 bpp.

Without a method to make correspondences between nodes in two different frames it doesn't make sense to consider measurements such as $-\log(Q^t_{\text{post}}(\mathbf{X}^*_t|\mathbf{Z}^*_{t-1}))/N$, the log likelihood of the image labels under the variational posterior given the one-step late HVPP tree structure or $-\log(Q^t(\mathbf{Z}^*_{t-1}))/N$, the log likelihood of the one-step late HVPP tree structure under the variational posterior for the current frame. The trivial mapping in which nodes which are initialised in the same position in the two frames are associated is not very useful because there is no guarantee that nodes which were initialised in the same position in two frames will end up representing corresponding regions in the image after the variational approximation has been calculated for the two frames.

3.4.2 HVPP Tree Mapping Algorithm

The coding cost of the image labels under the variational posterior given the one-step late tree structure is too expensive to be useful on its own but it is the right order of magnitude. However we can use the information that consecutive HVPP tree structures provide to code efficiently.

This experiment describes: a way to make a mapping in a hierarchical manner between the nodes in two consecutive time frames (Section 3.4.2.1), the cost of coding this mapping (Section 3.4.2.2), how to construct a new approximately correct tree structure for the current time frame using the HVPP tree structure for the previous frame and the coded mapping (Section 3.4.2.3), and finally

We can calculate the HVPP tree structure at time $t-1$ for no cost since we assume that we have already reconstructed that image. We can construct a new tree structure for the image at time t by moving the high-level nodes in the tree according to the coded movements in order to generate an approximation to the correct tree structure. This process of tree construction is described in detail Section 3.4.2.3. In outline, this procedure involves: moving the tree structures corresponding to moving objects in the images according to their coded node movements; using as much of the tree structures corresponding to static objects in the image at time $t-1$ as possible in the tree structure at time t; growing up new trees above regions in the image where gaps have been left allowing these to connect into either object or background trees where appropriate.

The approximate tree structures generated, whilst being better representations of the correct tree structures than the corresponding one step late trees, will nevertheless have some connections which are not optimal. The coding cost is improved if we smooth the distribution under which we code. This reduces the coding cost penalty that we make an incorrect connection. We do this by allowing nodes in the tree a probability of disconnecting. This has the effect of making the CPTs on the links in the tree less diagonal. Of course, there is a trade-off here because if we add too much smoothing to too many connections then it increases the cost of coding all the correctly connected nodes. The schemes that we use for applying this smoothing are described in more detail in Section 3.4.2.4. We can then code the image at time t using the prior variational distribution $Q_{\text{prior}}(X|Z)$ given this newly mapped HVPP tree structure. The experiments measuring the coding cost performance are described in Section 3.4.3.

Having discussed the general outline of the method, we shall now look at the detail of the algorithm. It is worth bearing in mind that there are many ways of implementing this scheme. We are not proposing that the particular approach presented here is either the correct way or even the best way of tackling the task. We will merely show that given the method we use, it is possible to make substantial gains in terms of the coding cost of the test image sequence. The approach can clearly be modified and improved, in fact at the end of the chapter we will point out specific ways in which some development would lead to further coding benefits.

Before attempting to match the HVPP tree structures between frames, we first simplify the tree structures. It is noted in Section 3.5 that in the variational approximation there is a tendency if we start with too many nodes in a layer, for nodes to aggregate in the same position. To simplify the mapping procedure and to make it more robust we

do a pass up through each tree structure recalculating the HVPP structure and merging nodes which are less than a pixel spacing apart.

3.4.2.1 Node Mapping

In this section, we describe the method used to make a mapping between nodes the HVPP tree structures for consecutive images in a sequence. For an image at time t, we make a mapping between the nodes in the distributions at time $t-1$ and at time t so that the HVPP tree structure and class values are as closely matched as possible. We define the class of a node as the class value with highest marginal probability in the variational posterior distribution.

We take advantage of the fact that in general images are composed of static background regions and objects which move around in front of the background. The tree structure for background regions which appear in both images will be largely similar and any differences in the connectivity we will presume to be largely irrelevant. Typically, nodes of the same class value tend to group together hierarchically in tree structures however it is possible that subtrees of one class can connect into trees of another class. We will match the nodes in object class trees. Since objects in the images may undergo arbitrary transformations between frames such as translations, scaling, rotations or even general changes of shape, the tree structures corresponding to objects in the images will almost certainly vary in both position and connectivity between two frames. We specify which of the image label classes represent background type regions and which represent object classes. In the test sequence, classes 1 and 2 are background whilst classes 3 and 4 are objects.

The first parameter N_s that our mapping algorithm requires will be used to define the distance over which to search for matching nodes in each layer. We specify a unit distance, d_h for each layer as $\sqrt{1/N_h}$ where N_h is the total number of nodes in the layer. This is roughly proportional to the average nearest neighbour spacing between nodes in the layer. Two nodes will only be matched if the distance between the nodes in the two frames at layer h is less than $N_s d_h$. The distance between the nodes is measured as either the difference in the position relative to the matching parent or the difference in absolute position of the nodes depending on whether the matching nodes are the children of a matched parents or not. A node at time $t-1$ will be matched to the closest available node at time t. We take this greedy approach as other suitable methods would be too computationally expensive.

Starting from the top of the two trees, we work downwards considering only the object class nodes in each frame. For each layer, we first try to match the children of matched parents. For each child node at time $t-1$ that has a parent for which a mapping has already been found, we look for the closest node of the same class which is a child of the mapped parent at time t and whose difference in relative position from its parent is less than the maximum node search distance $N_s d_h$.

Next we try to match the root nodes in the layer. For each root node at time $t-1$ in turn we look for the closest root node at time t which doesn't yet have a node mapped to it and is within the minimum search distance. Then we deal the leftover nodes in the two time frames, we try to map any unmatched non-root nodes at time $t-1$ to the closest unmatched root node at time t, then we try to map unmatched root nodes at time $t-1$ to the closest unmatched non-root nodes at time t, and finally we try to match the children of parents which aren't mapped to one another if there are within the minimum search radius.

On the face of it, it may seem that doing anything which doesn't purely match root nodes to root nodes and or match the children of matching parents together is in danger of confusing the relationships between the nodes in the two frames. However, the internal connectivity of tree structures representing the same object in two consecutive images can be quite different and we want our mapping to be as robust as possible to this variability. We hope that by working in a top down fashion and by primarily trying to match the children of matching parents we maintain as much of the significant structure as possible whilst not being so selective that we only end up generating a mapping for a small fraction of the nodes.

The pruning of excess object nodes in the tree takes place automatically in that when we come to construct a new tree for time t using the HVPP tree from time $t-1$ and the mapping, then any object nodes which aren't mapped will be omitted. However, we also need to be able to grow new nodes in order to accommodate objects that get larger from frame to frame. To help do this, we keep a record of the object nodes in the HVPP tree at time t which weren't mapped by nodes at time $t-1$. We will add these nodes as new nodes when constructing a new tree for time t.

Figure 3.4 gives an illustration of the node mapping process in action. The figure shows three layers of the HVPP trees for two consecutive frames in the test sequence. The bottom four layers are omitted from the figure because the number of nodes in each of these is too great and it becomes difficult to see clearly what is being shown.

The top layer is omitted because all the nodes in the second layer disconnect and so the node at the top is redundant. All the nodes in layer 2 are background and so no attempt is made to make mappings between these nodes. In layer 3, we can see an example of the children of unmatched parents being mapped to one another. In layer 4 there is a root node in each frame which are mapped together (the green nodes) and also there is an example of the children of matching parents being mapped (the blue nodes).

3.4.2.2 Estimating the Coding Cost of the Node Mapping

In this section, we look at the calculation of the cost that we have to pay for coding the node mapping and the movement of the mapped nodes between frames.

For mapped nodes which have mapped parents, we record the change in their position relative to their parents between the two time frames and for all the other mapped nodes we record their absolute change in position between frames. We also record the positions of the new nodes (object nodes in the HVPP tree at time t which were not mapped by anything at time $t-1$). Node movements which are smaller than the pixel spacing are unimportant so all the node movements and new node positions are discretised to multiples of the pixel spacing in each direction.

To use this information within a coding scheme, there are three contributions which will add to cost of coding the sequence. We first need to indicate which of the object nodes in the HVPP tree at time $t-1$ have a mapping, this will cost at most 1 bit per object node in the frame at time $t-1$.

We need to code the discretised positions of all the new nodes that will be added at time t. We assume that there is a uniform distribution of new nodes added over the area of the image and so this will cost $\log_2 N$ per new node where N is the number of pixel positions in the image. The test sequence images are 80×120 pixels in size so there are 9600 possible pixel positions.

Finally we need to code the discretised mapped node movements themselves. Since potentially a node could be mapped from one side of the image to the other in either direction, there are up to $2N_x + 1$ possible node movements in the horizontal direction if the image is N_x pixels wide in that direction and similarly up to $2N_y + 1$ possible node movements in the vertical direction. However, the distribution of mapped node movements in each layer is strongly peaked about zero (see Figure 3.5). We code the node movements in each layer using a distribution which we estimate empirically from the independent training sequence of images.

For each layer, we count the frequency of occurence of the mapped node movements over all the pairs of consecutive images in the training sequence. We assume the distribution of the movements is factorised and treat movements in each direction separately. We initalise the count by adding a single count to each bin to ensure that when we come to analyse the movements of the nodes in the test set there are no movements which have zero probability. The log likelihood of the node movements for each pair of images in the test set under this distribution provides our estimate of the cost of coding the discretised mapped node movements.

Figure 3.5(a) and (b) show the histograms of the node movements used to code the mapped node movements. The histograms as just described are measured on the training data set and include an extra count in each bin so that there are no zero probability movements. Only layers 3 to 8 are displayed in the figure as no nodes are mapped in the top two layers and so the distributions are uniform in these layers. Columns (c) and (d) of the figure show the positions at which new nodes in the training set are placed. As just mentioned, these are not used to code the new node positions as we assume the distribution over new node positions is uniform across the image, as we can see from the histograms plotted in Columns (c) and (d) this assumption is not unreasonable.

The total cost of the mapping increases as more layers in the trees are mapped. We will only be able to afford to code the mapping for a certain number of layers at the top of the tree. We specify a parameter of the algorithm, N_d, which defines the depth in the tree to which the mapping is coded. All the mapping information for nodes in the layers below the layer N_d is discarded.

3.4.2.3 New Tree Construction

In this section, we now consider in more detail how we construct the HVPP mapped tree for the image at time t given the mapping we have encoded.

Using just the HVPP tree for the frame at time $t-1$, the discretised movements of the mapped nodes for the top N_d layers and the positions of the new nodes to add to the top N_d layers of the HVPP tree that we have just calculated, we now want to construct a tree structure for the frame at time t which we can use to code the image and which will be a better approximation of the HVPP tree structure at time t than the one-step late tree.

Firstly, all the mapped object nodes in the top N_d layers of the $t-1$ HVPP tree are moved according to the discretised mapped node movements. A node is moved

relative to its parents position or relative to its current position in the frame depending on whether the node is a child of a mapped parent or not. The children and further descendents of the mapped object nodes in the $h - N_d$ layers at the bottom of the tree are moved so that they dangle down in the same position relative to their parents as in the HVPP tree at time $t - 1$.

We now start with an empty grid of pixel positions to which we want to attach the leaf nodes of our new tree structure. When all the mapped object nodes and their descendents have been moved, the descendents of the mapped object nodes in the leaf layer are one by one fixed to their nearest free pixel position. Next, all the background leaf nodes in the HVPP tree at time $t - 1$ which aren't in areas covered by the new object nodes are placed in the same positions as in the $t - 1$ frame along with all the nodes and tree structure from which they descend.

We are left with some empty pixel positions and some gaps in the tree structure above. The empty regions are where there were object leaf nodes in the $t - 1$ frame but from where the object nodes have moved away due to the high-level mapped node movements.

To fill these regions, we will grow new parts of the tree structure up from the empty pixel positions. However, we first need to add some nodes in the layers above the leaf layer to which the nodes below in the empty region may want to connect. First, we add all the new nodes from the mapping in top N_d layers. There will still be fewer nodes in each layer than are used in the variational approximation due to the node mergers, the object nodes which found no mapping and the background nodes their ancestors which were obscured by the moved object nodes. We add extra new nodes to each layer by creating a grid of evenly spaced node positions. The number of available node positions equals the number of nodes used in variational approximation. For each layer, we allow the existing nodes to find their closest available grid position one by one. Each grid position which is occupied by an existing node is discarded and when all the existing nodes have claimed a position, the extra new nodes will be added at the remaining grid positions in the layer.

Now we can grow the remaining parts of the tree up from the leaf layer in order to connect up the nodes at the empty pixel positions. Nodes in a layer choose to connect to the nearest parent in the layer above. They can either choose to connect to mapped forward object nodes, existing background nodes, new nodes added by the mapping or extra new nodes added to the layer above. However, the added nodes from the mapping

are only allowed to connect to object node parents or to other added nodes from the mapping in the layer above since they correspond to object nodes in the time t HVPP tree. The extra new nodes can connect to any sort of node in the layer above. New added mapped nodes and the new extra nodes to which nothing decides to connect are discarded.

When this upward pass through tree is complete, we have our new tree structure which we will use to code the image at time t.

3.4.2.4 Smoothing the CPTs

It is apparent that neither the one step late tree nor the newly created mapped tree will be exactly like the HVPP tree structure for the image at time t and neither will be represent the structure of the image as well as the HVPP tree. This lack of accuracy in the connectivity of the approximate trees results in a higher coding cost for the image. The penalties in terms of coding cost occur where the ancestors of nodes representing different objects in the image join together in the same tree where the CPTs don't allow transitions of this sort with a high probability. In other words, where the structure of the tree isn't representative of the underlying image structure.

We can, however, address this problem. We can make the distribution more accommodating to these mistakes by smoothing the CPTs. Where the CPTs strongly predict the class of the node given the class of the parent, we can reduce the certainty of this relationship by effectively adding a probability that the node doesn't in fact connect to its parent but instead disconnects. This softens the CPTs. The parameter π_d will be used to describe this probability. If $P(X_i|X_j)$ is the CPT and $P(X_i)$ are the prior probabilities of each class then the smoothed CPTs, $\tilde{P}(X_i|X_j)$ will be given by:

$$\tilde{P}(X_i|X_j) = \pi_d P(X_i) + (1-\pi_d)P(X_i|X_j) \tag{3.15}$$

In the experiments we will look at two ways of applying this smoothing. In the first, which we will term *full* smoothing, the probability of disconnection, π_d, is applied to every CPT in the tree.

The second approach that we take is slightly more complicated and we will call it *reduced* smoothing. We compare the leaf layer nodes which are descendents of the mapped object nodes in our new tree with the object regions in the image at time t that we are trying to code. This will indicate the regions where the tree structure makes mistakes. Next, we take the boundary between the object regions and the background

regions in the image at time $t-1$ and we measure the minimum distances from either side of the boundary which define a region that encloses all the mistakes made that the mapped tree structure.

When we come to reconstruct the image sequence we will have already decoded the image at $t-1$ and so the boundary will already be available. The two numbers, R_{in} which gives the minimum distance from the boundary to the furthest pixel within the object regions whose leaf node connects into background tree and R_{out} which conversely gives the minimum distance from the boundary to the furthest pixel outwith the object regions whose leaf node connects into an object tree, allow us to calculate the region within which all the uncertainties occur. These radii have discrete values as they are measured in units of pixels furthermore, $R_{\text{in}}+R_{\text{out}}$ must be less than the length (in pixels) of the main diagonal across the image. For the purposes of the experiment, we restrict them to values up to the width of the longest edge of the image (120 pixels) and we assign 14 bits per image as the cost of coding this information. Reduced smoothing consists of applying smoothing to the CPTs on just the links between the leaf nodes and their parents in this uncertain region.

Since in some frames it will be cheaper to use full smoothing and in others reduced smoothing, we can also treat the choice of smoothing as a parameter and use 1 bit per image to indicate the cheaper choice. We will refer to the use of this approach as using the *best* smoothing method.

3.4.3 HVPP Tree Mapping Experiments

We have now fully described the HVPP mapping algorithm and we are in a position to test its performance. In Section 3.4.3.1, we look at a preliminary investigation of smoothing which illustrates the effect that it has. Section 3.4.3.2 presents the coding costs of the HVPP mapping algorithm both in absolute terms and in comparison to coding using the one step late tree. Finally, Section 3.4.3.3 looks at the coding costs of the HVPP mapping algorithm as the time difference between consecutive frames increases. The coding costs are compared to the cost of coding using the N step late tree.

For each of these experiments, the first step is to train the PEDT and calculate the variational posterior distribution for each image in the sequence in exactly the way that we did for the preliminary coding cost experiments.

3.4.3.1 Preliminary Smoothing Experiment

As an illustration of the effect of smoothing on the coding cost of images, Figure 3.6 shows the results of a preliminary experiment which measures the variation in the image coding cost with the probability of disconnection when applying full smoothing and using the one step late tree to code the images. For comparison, the figure also shows the coding cost of the images given the actual HVPP tree structure. There are a number of features that are worth noting. In general, unsurprisingly the coding cost of the images given the one step late tree is higher than given the actual HVPP tree however we can see that as the probability of disconnection increases, the coding cost given the one step late tree decreases. As the probability of disconnection gets too high the coding cost starts to increase dramatically. This is due to the trade off that occurs, we add the smoothing to reduce the effect of the mistakes in the tree connectivity on the image coding cost however as it gets too high we get penalised for adding unnecessary smoothing to all the correct connections. As the probability of disconnection goes to 1 the cost of coding the image given both the one step late HVPP tree and the correct HVPP tree converge to the average pixelwise entropy of the test sequence which is reassuring because this is the entropy of the distribution that assumes that all the pixels are independent of one another. As there are no mistakes in the HVPP tree structure for the frame, adding smoothing to the CPTs only acts to increase the image coding costs. The optimum probability of disconnection for the one step late tree is 0.03, this gives an average test image coding cost of 0.1088bpp.

3.4.3.2 HVPP Mapped Tree and the One Step Late Tree Comparison

As noted above, there are three parameters to set for the HVPP mapping: the search distance, N_s, which controls the distance over which a node looks for a match; the mapping depth, N_d, which defines the depth to which the mapping is coded; and the probability of disconnection, π_d, which controls the amount of smoothing added to the CPTs.

In general, we will only allow a fixed number of settings for each of these parameters. The search distance will have 10 possible settings (0.1,0.5,1,2,3,4,5,6,7,8). Remember that this is scaled so that it roughly represents multiples of the average nearest node spacing in each layer. We will consider 4 possible mapping depths of (3,4,5,6). In practice, there are no node mappings in the top two layers so there is

Coding Method	Optimum Parameters		
	π_d^*	N_d^*	N_s^*
HVPP mapped tree, full smoothing	0.022	3	0.1
HVPP mapped tree, reduced smoothing	0.125	4	2
1 step late tree, full smoothing	0.022	-	-
1 step late tree, reduced smoothing	0.177	-	-

Table 3.3: Optimum Fixed Parameter Values as Calculated on the Training Image Sequence. The table shows the combination of parameter values of the HVPP tree mapping algorithm and the smoothing method which give the lowest image coding cost averaged over the training sequence.

no point in considering these and coding the mappings to below layer 6 is too expensive. The probability of disconnection will be allowed to take 15 possible values, $(2^{-0.5}, 2^{-1}, 2^{-1.5}, \ldots, 2^{-8})$.

We will take two approaches to setting these parameters. In the first, we calculate the average image coding cost over every pair of consecutive images in the independent training image sequence described in the experimental data section and find the best parameter settings. Then we take these fixed parameters and use them to calculate the coding cost of every image in the test sequence. Table 3.3 shows the optimum parameter values calculated in this way. When we use these parameter values, they will be referred to as *fixed* parameters. The second method for setting the parameters is to find the best parameters for coding the images on an image-wise basis and code these parameter values. We will refer to these as *optimised* parameters.

There are a number of interesting features to note from the Table 3.3. We can see that in the optimum probability of disconnection when using the full smoothing method is considerably lower than when using the reduced smoothing. This is not unexpected. The full smoothing adds the probability of disconnection to every CPT in the tree whereas the reduced smoothing only adds the disconnection probability to the CPTs at the leaf layer in a specific region. We can expect the more focussed approach to need a higher probability of disconnection in a small region. Secondly, the reduced smoothing needs a higher probability of disconnection when used with the one step late tree as compared to when used with the HVPP mapped tree. This is an indication that in general the HVPP mapped tree is more accurate and requires less smoothing.

Finally, we can see that for the HVPP mapped tree with full smoothing the optimum values of the search distance, N_s, and the mapping depth, N_d, are 0.1 and 3 respectively which means that on average over the sequence the best mapping is minimal. Nodes are only mapped down to the third layer and are only allowed to search 0.1 of the average nearest node distance in that layer for a match. This is actually symptomatic of the fact that the best parameter values vary considerably from image to image and the difference in coding cost between the optimum parameter setting and the other settings can be significant whereas the most minimal mapping is never particularly bad as it is pretty close to the one step late tree.

This motivates the second method for choosing the parameter values which will involve finding the parameters for each image and paying a cost to code which parameters to use. π_d has 15 possible values so can be coded with less than 4 bits per image, N_s has 10 possible values and can similarly be coded with less than 4 bits per image, and N_d has 4 possible values and can be coded with 2 bits per image.

Overall we will compare the total coding cost of the test image sequence using the following approaches:

- HVPP mapped tree, full smoothing, fixed parameters
- HVPP mapped tree, reduced smoothing, fixed parameters
- HVPP mapped tree, full smoothing, optimised parameters
- HVPP mapped tree, reduced smoothing, optimised parameters
- HVPP mapped tree, best smoothing, optimised parameters
- one step late tree, full smoothing, fixed parameters
- one step late tree, reduced smoothing, fixed parameters
- one step late tree, full smoothing, optimised parameters
- one step late tree, reduced smoothing, optimised parameters
- one step late tree, best smoothing, optimised parameters.

3.4.3.3 HVPP Mapped Tree and the N Steps Late Tree Comparison

As a follow up to the previous experiment, we will also look at how robust the HVPP mapping algorithm is to lower frame rates. In other words, what happens to the performance as the previous image to the one that we are trying to code becomes less predictive of the content of the current image? We compare how HVPP mapping algorithm and the N steps late tree perform when coding a test image sequence as the number of time steps between the frames that are used increases.

When we calculate the coding cost for an image at time t we use the image at time $t-N$ as the basis for the HVPP mapped tree and N step late tree. The coding cost is measured for the same 16 images selected from the test sequence but the images at time, $t-N$, which are used for coding vary as we increase N from 1 to 10. When the frame rate is reduced by a factor 10, consecutive images are sufficiently different that it isn't necessary to reduce it further.

3.4.4 Results

Here, we will look at the results for the two main experiments described above.

3.4.4.1 HVPP Mapped Tree and the One Step Late Tree Comparison

Table 3.4 shows the results of the coding cost experiments comparing the performance coding images given the HVPP mapped trees and given just the one step late tree structure. The table shows the average negative log likelihood of the images given the particular tree structure used, $-\log(Q_{\text{prior}}(\mathbf{X}_t^*|\mathbf{Z}))/N$. This is the coding cost of the images given the structure $\mathbf{Z}$. For the HVPP mapped trees, the table also shows the average cost of coding the mapping used. This has been calculated as described in Section 3.4.2.2. For each of the methods compared, the table includes the cost of the parameters which need to be coded for each approach. The final column gives the overall coding cost for each method. The best coding cost result of 0.0937bpp is obtained through the use of the HVPP mapping algorithm with optimised parameters for each frame and using the best of the choice of full or reduced smoothing for the frame. It is worth paying the cost of coding the parameters and the choice of smoothing on a frame-wise basis as the best HVPP mapped tree coding cost with fixed parameters is 0.1050bpp. (In this case the fixed parameters have been chosen using an independent training set of images). Both of these results compare favourably with the best coding

Coding Method			Average Coding Cost/bpp							
Tree	Smooth	Param.	nllh	mapping	π_d	N_d	N_s	B_{map}	R_{in},R_{out}	Total
map	full	fixed	0.1030	0.0019	-	-	-	-	-	0.1050
map	red.	fixed	0.1266	0.0060	-	-	-	-	1.5×10^{-3}	0.1341
map	full	opt.	0.0920	0.0041	4.2×10^{-4}	2.1×10^{-4}	4.2×10^{-4}	-	-	0.0971
map	red.	opt.	0.1055	0.0047	4.2×10^{-4}	2.1×10^{-4}	4.2×10^{-4}	-	1.5×10^{-3}	0.1127
map	best	opt.	0.0881	0.0038	4.2×10^{-4}	2.1×10^{-4}	4.2×10^{-4}	1.0×10^{-4}	6.1×10^{-4}	0.0937
1 step	full	fixed	0.1120	-	-	-	-	-	-	0.1120
1 step	red.	fixed	0.2305	-	-	-	-	-	1.5×10^{-3}	0.2320
1 step	full	opt.	0.1104	-	4.2×10^{-4}	-	-	-	-	0.1108
1 step	red.	opt.	0.2176	-	4.2×10^{-4}	-	-	-	1.5×10^{-3}	0.2195
1 step	best	opt.	0.1100	-	4.2×10^{-4}	-	-	1.0×10^{-4}	3.0×10^{-5}	0.1106

Table 3.4: Results of the Comparison of the Image Coding Cost using the HVPP Mapping Algorithm and the One Step Late Tree. The table shows the average coding cost of the images in the test sequence using different methods of coding. The column labelled *nllh* shows the average negative log likelihood of the image under the variational distribution, $-\log(Q_{\text{prior}}(\mathbf{X}_t^*|\mathbf{Z}))/N$. The second column, *mapping*, gives the cost of using the HVPP mapping where appropriate. The next five columns show the cost of using the various parameters where appropriate. The last column gives the overall coding cost for the method.

cost given the one step late tree which costs 0.1106bpp and is achieved by using the one step late tree and coding the probability of disconnection and the choice of smoothing method on a framewise basis.

It is worth noting that on average the reduced smoothing method produces disappointing results. It is the case that for particular images the reduced smoothing method produces lower coding costs than the full smoothing method for both the HVPP mapped trees and the one step late trees, however, when it gets it wrong, it pays a heavy price. This is indicative of an underlying problem with HVPP mapping algorithm and is an area with scope for future improvement. The problem is that both when making the mapping between object nodes and when finding the regions in which the reduced smoothing method needs to apply the probability of disconnection, the nodes are only distinguished as either object nodes or background nodes. If the regions of an image representing two different types of object in the image get mixed together in the same tree structures then this results in a penalty to the coding cost. However, unfortunately the HVPP tree mapping allows this possibility to happen and the reduced smoothing method will not notice where mistakes of this kind take place. If the mappings for

object type tree structures were to be made for each image object type at a time, then this would certainly result in further reductions in the image coding costs.

We can look in more detail at the results for the HVPP mapping algorithm with optimised parameters by inspecting the frequencies with which each of the parameter are chosen to code the sequence most efficiently. Figure 3.7 shows the best values of the smoothing parameter, that is to say the probability of disconnection, π_d. It illustrates again that in general the full smoothing method requires a smaller amount of smoothing to achieve the best results than the reduced smoothing method. Figure 3.8 shows the best values of the two parameters for the mapping algorithm, the search distance N_s and the mapping depth N_d. We can see that there is a tendency to choose minimal mappings, that is to say mappings which only search over a small distance for matching nodes and which only code the mapping for the top layers of the tree. This is probably partly explained firstly by the fact that if it is difficult to find a good mapping then it pays to use a coding scheme which has the lowest overhead costs, but secondly it may also be due to the fact that there are a number of images in the test sequence which change little from frame to frame and in this case very little is required to code the differences between the frames. However, in general this observation also fits with our assumptions that we should be able to represent the changes between images by a small number of high level node movements. Nevertheless, we can also see that on a small number of occasions when there are substantial differences between the consecutive images, it does actually pay to code the mapping all the way down to layer 6. The extra cost that is required to code the mapping down to this level is outweighted by the benefit to the coding cost of having the more accurate representation of the tree structure for the image that is generated by the HVPP tree mapping.

We can also examine how average coding cost of the test sequence varies as the parameters which are fixed for the whole sequence are varied as shown in Figure 3.9. The figure shows the dependence of the coding cost of the test sequence on the parameters using both the full smoothing, row (i), and the reduced smoothing, row (ii). Apart the bias in the full smoothing results towards minimal mappings, we can see that the lowest coding costs are within the allowed parameter ranges which is reassuring.

3.4.4.2 HVPP Mapped Tree and the N Steps Late Tree Comparison

Having looked at the relative performance of the one step late tree and HVPP mapped tree, we now compare the performance of the HVPP mapped tree with the N step late

tree. Figure 3.10 shows the coding cost of a selection of 16 images from the sequence given the HVPP mapped tree derived from the image at time $t-N$ (when coding an image at time t) and compares it to the coding cost the same images at time t given just the HVPP mapped tree at time $t-N$ which is how we define the N step late tree.

We can see from the graph that the use of the HVPP tree mapping algorithm is most beneficial in comparison to the use of the N step late tree when the gap between the frames is small, i.e. for low N. In this case, the image at time $t-N$ provides good information about the structure of the image at time t and the HVPP tree mapping algorithm is able to take advantage of it.

When there is a large gap between the images in the sequence so that the image at time t bears little relationship to the image at time $t-N$, it is difficult to make a mapping between the tree structures of the two frames and there is little to be gained from doing it. In this situation, we can see that the coding cost of the test sequence given the HVPP mapped tree and given the N step late tree is similar.

3.4.5 Summary

In the experiments in this section, we have looked at how we can use the PEDT to code a sequence of test images.

We take advantage of the fact that given a correct or an approximately correct tree structure for an image, we can code that image cheaply. However, coding the tree structure itself is expensive. The HVPP mapping algorithm generates a mapping which can be applied to the HVPP tree structure for the image at time $t-1$ to give a better approximation of the tree structure for the image at time t which would then be used to code the image at time t.

We compared the performance of the HVPP mapping algorithm with the use of the one step late tree structure for coding, for both approaches we used a variety of smoothing methods and parameter settings.

The method with the lowest estimated coding cost was found to be the HVPP mapping algorithm with the parameters optimised and coded for each frame and for which the best of the smoothing methods was found and coded on a framewise basis. The coding cost of the test image sequence using this approach was 0.0937bpp. If we add into this the fixed costs[6] for the sequence as described in Section 3.4.1 then the total

[6]The fixed costs are the cost of coding the first image in the sequence and the cost of coding the prior CPTs for the test sequence.

coding cost would be 0.0971bpp. This represents a reduction of 29.5% on the cost of coding the images in `tar.gz` format which was the cheapest other method found, (see Table 3.1).

The approach which involves coding the images given the one step late tree structure is the same as that which involves the calculation of the HVPP mapped tree except that the one step late approach makes no attempt to code any sequence information. The best result for the HVPP mapped tree coding cost is a 15.3% improvement on the estimated cost of coding the images using the best one step late method which costs 0.1106bpp. This illustrates the value of including sequence information.

3.5 Sequential PEDT Experiments

In the experiments in the previous section, we used the static PEDT and took advantage of the HVPP structure of variational posterior distribution to incorporate sequential information via the HVPP tree mapping. The aim was purely to achieve as low a coding cost as possible.

As mentioned when discussing the theory in Section 3.2.2.2, we can introduce sequential information directly into the PEDT framework taking a filtering approach and updating the prior distribution at time t based on the posterior at time $t-1$ by setting a probability that nodes make connections back to nodes in the earlier frame. We have referred to this as the sequential PEDT. In this section we will be investigating the behaviour of the sequential model. First we will look at the sequence model in more detail, (Section 3.5.1). Then, we will examine its performance when using the image coding cost as a measure of how predictive the sequential updates are of the image at time t, (Section 3.5.2). Following this, we examine the performance of the sequential PEDT model as measured by the pixelwise classification accuracy, (Section 3.5.3).

3.5.1 Sequence Model Implementation

The basic sequence model needs to include a method of generating a correspondence between nodes at two time frames and then generating the predictive distribution at time t given the information at time $t-1$. We define that form of model here. First we define a correspondence model. Then we define a temporal model for the node states and connectivity. It turns out that this temporal model produces a predictive model at

time t which has the same PEDT prior but with modified parameters. Hence all the variational machinery already derived is directly applicable for inference in the model at this new time frame t.

Let S^t denote a set of connections from nodes at time $t-1$ to nodes at time t. S^t consists of indicator variables s^t_{ij} where $s^t_{ij} = 1$ denotes that node i of the PEDT at time t corresponds to node j from the PEDT at time $t-1$. We also allow a connection to a null node 0 denoted $s^t_{i0} = 1$ representing the fact that i has no corresponding node at time $t-1$. $s^t_{ij} = 1$ for one and only one value of j, and is zero otherwise.

In order to define the node correspondence, we will define a set of reference positions for each node i, denoted by $\mathbf{v}_i$.

We can define a probability distribution

$$P(s^t_{ia}|\mathbf{v}_i, \mathbf{r}^{t-1}_a) \tag{3.16}$$

which depends on the positions of nodes at the past time frame and the chosen reference positions $\mathbf{v}_i$ of the nodes at the new time frame. This distribution gives the through time connection probabilities. The initial node centres $\mathbf{v}_i$ are set *a priori*. In order to ensure that, at each time step, reasonable coverage of the space is obtained, we set the initial node centres to take a uniform quadtree-like distribution[7]. Thus we use

$$P(s^t_{ia}|\mathbf{v}^t_i, \mathbf{r}^{t-1}_a) \propto \exp(-\frac{1}{2}(\mathbf{r}^{t-1}_a - \mathbf{v}_i)\Lambda_l^{-1}(\mathbf{r}^{t-1}_a - \mathbf{v}_i)). \tag{3.17}$$

which can alternatively be seen as the responsibilities of each node a for each point i under a mixture of Gaussian distributions centred at nodes a with covariance Λ_l dependent on layer l. The probability of a node being a new node $P(s^t_{i0})$ is set *a priori*.

The actual map S^t from nodes at time t to nodes at time $t-1$ are drawn from the distribution

$$P(S^t) = \prod_{ij} P(s^t_{ij}|\mathbf{v}_i, \mathbf{r}^{t-1}_j). \tag{3.18}$$

Having obtained the connection probabilities, the predictive distribution of the nodes at time t must be specified. The simplest approach is to allow the class variables and connectivity to be passed forward. If a node is connected through the past we would expect the class variable and parent to be identical to its state in the previous

[7]Each layer is a grid suitably positioned so that, where possible, four nodes of a layer are equidistant from the closest node in the layer above.

time frame. We write this as

$$P(z_{ij}^t = 1 | s_{ia}^t = 1, s_{jb}^t = 1, z_{ab}^{t-1} = 1) = 1 \text{ and} \quad (3.19)$$

$$P(X_i^t = k | X_i^{t-1} = l, s_{ij}^t = 1) = \delta_{kl}. \quad (3.20)$$

If a node i is considered to be a new node (i.e. $s_{i0} = 1$), then its class is entirely dependent on its parent at time t through the usual prior CPTs, and if the parent of a node does not propagate through to time t, or a node is a new node, then the node has the usual uniform connection probabilities with the parents in the layer above.

This is the simplest form of dynamic model we can use. It encodes the fact that at the next time step we expect to see similar sorts of objects and structures in our image. The uncertainty in the dynamic model comes from the uncertainty in the position that a node moves to through time, and in whether or not the node is still represented at the next time step.

The PEDT uses approximate variational inference in order to obtain a representation of the posterior distribution. When a PEDT model is used in a sequence it would be correct to do variational updates on the whole sequence. However the computational cost of this is prohibitive (even on smaller images there are tens of thousands of nodes per frame). In almost all practical circumstances models of the filtering process make an assumption that the approximate representation of the posterior at time $t-1$ could be taken as suitably close to the true posterior for practical purposes. For example in [37] the Monte-Carlo sample from time $t-1$ is presumed practically exact when propagation to the next time frame is done. We make the same assumptions here, assuming that the variational distribution at time $t-1$ can be presumed to be good enough without repeated iteration of the variational method through time.

The first step, then is to find the predictive PEDT at time t given the variational posterior at time $t-1$. It turns out that this propagation can be done exactly. This relies on the fact that the variational posterior $Q(X_i^t | X_j^t, z_{ij}^{t-1} = 1)$ at time $t-1$ takes the same values whatever parent node j is being considered. The implication of equations (3.19) and (3.20) for the class labels is considered first, followed by the implication for the Z prediction.

We have a predictive distribution taking the PEDT form at time t with the CPTs

given by

$$P(X_i^t = k|X_j^t = l, z_{ij}^t = 1) = P(s_{i0}^t)P_s(X_i^t = k|X_j^t = l, z_{ij}^t = 1) + \sum_a P(s_{ia}^t)P(s_{j0}^t)P(X_a^{t-1} = k) + \sum_{ab} P(s_{ia}^t)P(s_{jb}^t)P(X_a^{t-1} = k|X_b^{t-1} = l, z_{ab}^{t-1} = 1), \quad (3.21)$$

where P_s denotes the standard static PEDT prior conditional probabilities. The three contributions to this term correspond to the following three situations respectively: node i is a new node at time t; node i at time t corresponds to node a at time $t-1$ but node j is a new node at time t; node i corresponds to node a and node j corresponds to node b.

If we make $P(s_{j0}^t) = (1 - \Gamma^h)$ identical for all nodes j in the same layer h and at every time t, and use the fact that $P(X_a^{t-1} = k|X_b^{t-1} = l, z_{ab}^{t-1} = 1) = Q_a^{kl,t-1}$, the variational posterior CPTs from the previous time frame we obtain

$$P(X_i^t = k|X_j^t = l, z_{ij}^t = 1) = P(s_{i0}^t)P_s(X_i^k|X_j^k, z_{ij}^t) + \sum_a P(s_{ia}^t)(1-\Gamma^h)m_a^{k,t-1} + \sum_a P(s_{ia}^t)\Gamma^h Q_a^{kl,t-1} \quad (3.22)$$

where $m_i^{k,t-1}$ denotes the marginal variational posterior probability of node i being in state k at time $t-1$, and h denotes the layer that node j is in.

Looking now at the predictive prior over the Z variables we have the distribution

$$P(z_{ij}^t = 1) \propto \left(P(s_{i0}^t) + \sum_a P(s_{ia}^t)P(s_{j0}^t)\right) P_s(z_{ij}^t = 1) + \sum_{ab} P(s_{ia}^t)P(s_{jb}^t)Q(z_{ab}^{t-1} = 1) \quad (3.23)$$

for $j \neq 0$ and

$$P(z_{i0}^t = 1) \propto P(s_{i0}^t)P_s(z_{i0}^t = 1) + \sum_a P(s_{ia}^t)Q(z_{a0}^{t-1} = 1). \quad (3.24)$$

This gives us the predictive prior $P(Z^t|Q^{t-1})$ and $P(X^t|Z^t, Q^{t-1})$ at time t given the variational posterior Q^{t-1} at time $t-1$. Again P_s denotes the prior under the static PEDT model.

These equations for the predictive model take the form of another PEDT, with modified CPTs and modified connection probabilities. Because it is a PEDT, the variational posterior at time t can be calculated using the usual PEDT variational optimisation. Then this can be fed forward to time $t+1$ etc. Hence we have a way of implementing filtering in a chain of PEDTs. The implementation of this method involves running a standard static PEDT for the first image of the sequence. Then equations (3.22) and

(3.23) are used to calculate the effective CPTs and predictive prior over Z for time t given what we know at time $t-1$. Once these calculations are done we effectively have a static PEDT again (but with modified CPTs etc.).

The overall form of the optimisation involves initialising the positions of the node's variational centres to the reference positions $\boldsymbol{\mu}_i = \boldsymbol{\nu}_i$, calculating the responsibilities $P(s_{ia})$ of each node a at time $t-1$ for each node i at time t, propagating forward the conditional probabilities, and then conditioning on the positions of the root nodes and the pixel data to get a final posterior at time t. This posterior is calculated using the usual variational optimisation given the initialisation described above.

The model for mapping nodes in this time frame to those in the past appears to depend on a characteristic of the variational method. However the precise initialisation can be set *a priori*, and so is not dependent on the variational optimisation. Furthermore correspondence between nodes is somewhat arbitrary due to node exchangeability (relabelling the nodes makes no difference to the predictive distribution we define for the Z and X variables). The reason we use the variational centres to set the correspondence is in order to avoid local minima problems which can be encountered if the initial positions of nodes are far from their positions at the previous time frame, and to avoid node clumping problems (another form of local minimum), where a large number of the nodes are used to represent a small part of the overall space. This is achieved because at each stage the past nodes are associated with new nodes which are evenly or uniformly distributed through space. Other model forms which say generate node positions dependent on the positions in the past can easily suffer from node clumping at later time stages.

3.5.2 Sequential PEDT Image Coding Cost Experiments

The experiment takes the following approach. We train the PEDT on the 50 image test sequence in exactly the same way as in the previous experiment to give us the CPTs for the prior PEDT distribution. We instantiate the leaf nodes in the PEDT at time $t = 1$ with the image labels for the first image in the sequence and calculate the variational posterior distribution.

For the next image and each subsequent image in turn, we update the usual PEDT prior distribution using the sequential updates as described in Section 3.5.1. There are two parameters which can be varied: the probability of back connection Γ which determines the significance of the posterior distribution for the previous frame in com-

parison to the current prior distribution; and the update depth, N_d, which specifies the depth in the tree down to which the updates are made. The parameters are held fixed for all the images in the sequence.

We want to measure how informative the updated prior distribution is of the image labels at time t. Unfortunately, the variational log likelihood of the data under the full PEDT distribution is not useful because we can only calculate it up to an unknown additive constant which is due to the fact that the distribution over node positions is conditioned on the leaf nodes being fixed on the pixel positions. As an alternative estimate of the coding cost we calculate the negative log likelihood of the image labels at time t under the product of the pixelwise marginal probabilities for the updated prior distribution.

After having calculated this estimate of the coding cost, we instantiate the leaf nodes with the image labels, calculate the variational posterior and move to the next image in the sequence. The coding cost is calculated as the average over images 2 to 50 in the sequence. The experiment is repeated for the following values of $\Gamma = 0, 0.1, 0.2, \ldots, 1.0$ and repeated with the update depth parameter, N_d, taking the values $3, 4, \ldots, 8$.

3.5.2.1 Results

Figure 3.11 shows a graph of the estimated coding cost of the test image sequence as the probability of back connection, Γ, varies. (The update depth is fixed at $N_d = 3$.) The lowest average coding cost is found when the update depth is 3 and the probability of back connection is 0.8. This gives an estimated coding cost of 0.2033bpp. Note that when Γ is 0, there are no sequential updates and we are effectively coding under the prior PEDT distribution, in this case the coding cost is close to the entropy of the overall class frequencies.

To give us a sense of scale, we can again compare this result with the table of results from other image coding formats (Table 3.1). Before we do this, we should again as before, add the fixed coding costs for the sequence. These are the cost of coding the first image and the cost of coding the CPTs for the prior distribution. We could, of course, code the first image of the sequence using just the prior PEDT distribution without any sequential updates however it is cheaper to code it using a more efficient image format. Using the same values as for the fixed costs in the previous experiment, the estimated coding cost of the sequence increases to 0.2046bpp. Note that this com-

pares favourably to all the coding costs in the table of comparisons other than the most efficient `tar.gz` format. It is also worth bearing in mind when comparing it to the `tar.gz` method that this is the only method in the table of comparisons that doesn't allow the images to be streamed and processed one by one. All the images have to be put in one big archive for the Lempel-Ziv algorithm to achieve its full efficiency. The scheme that we are measuring here would allow images to be coded and decoded one by one in a streamed manner.

The coding cost of the sequential PEDT on the test images is not lower than those presented for coding using the HVPP mapped trees or using the one step late trees. There are two reasons for this. First of all, the coding costs for these experiments are estimated from the pixelwise marginals which ignore the correlations between the pixels. Secondly, in this experiment we are using the coding cost as a measure of how predictive the sequential PEDT model for test images to which it has never had access. Whereas, in the HVPP tree mapping experiments, we allowed full access to the test image being coding so as to try to find the best coding scheme possible. This is a considerable advantage.

3.5.3 Sequential PEDT Image Classification Experiments

Here we demonstrate the application of the sequential PEDT model to image sequence segmentation. We evaluate how the incorporation of sequential information, as described above, into our PEDT model affects the performance of the model on the segmentation task. The local pixel-wise image model that we employ is described in 3.5.3.1 and the PEDT training set-up is briefly reported in Section 3.5.3.2. Section 3.5.3.3 describes the investigations that were carried out into the behaviour of the algorithm with respect to the various parameter settings of the PEDT. In Section 3.5.3.4, we discuss the various ways in which we can measure the performance of the approach.

3.5.3.1 Local Image Model

To make inferences about a particular previously unseen test image, we need to make an estimate of the likelihood, $p(y_i|x_i)$, that pixel i has value y_i given it has class label x_i for each of the pixels in that frame. For the purposes of these experiments we use a simple local image model which we derive from a histogram of the training set class

frequencies.

To construct the histogram, the RGB colour cube is divided into 27 bins of equal volume and the frequency with which the training set pixels fall into each of these bins is counted for all the pixels of each class. For a particular test pixel whose pixel value falls into bin b of the colourcube, our estimate for the likelihood is given by the measured frequency of bin b for each of the four possible class labels.

The local image model depends on the level of the noise added to the training set. Figure 3.12 shows an example image from the test image database to which the various amounts of noise have been added and then the appropriate local model used to construct both the maximum likelihood segmentation under the local pixelwise model and the MAP segmentation.

For the highest two levels of noise, the MAP local segmentation just picks the class with the highest prior probability for all the pixels. Moreover, although the PEDT experiments were carried out with the four noise levels described above, the noisiest images (with noise variance 1) were classified as the highest prior class for all pixels under all the model parameter settings tried. As these results are rather uninteresting, they are omitted from the subsequent discussion.

3.5.3.2 Training the PEDT

We learn the PEDT parameters by carrying out 40 iterations of the variational optimisation as described in Section 3.2.1.3 for 3 images chosen at random from the training set. During training the probability of backconnection Γ is set to zero.

3.5.3.3 Investigating the Parameters of the PEDT

We perform three sets of experiments, in each case we update the PEDT prior model as described in Equation (3.22). We fix the probability that a node connects to any node in its level at the previous time step to be same for every node in all the layers down to a specified depth j and set $\Gamma^k = 0$ for $k > j$. (The implications of this assumption are discussed in Section 3.5.3.5.2.) We examine how the algorithm performs on the segmentation task for a range of values for Γ (0, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00).

In the first experiment, we vary the levels of noise added to the image. In the second, we examine how varying the depth in the tree to which the updates are performed affects the performance.

3.5.3.3.1 Noise Levels We add independent Gaussian noise to each pixel as described in the experimental data section (Section 3.3) with a variance of $0.1, 0.01, 0.001$ and measure the segmentation accuracy. (We also used noise with variance 1, however this is too much noise, all segmentations predict the most probable class in the prior for all pixels. This is uninteresting and this setting is ignored for the rest of the discussion.) The prior updates were performed down to, and including, the fifth layer from the root.

3.5.3.3.2 Update Depth For a fixed noise level (variance 0.1), we experiment with varying the depth in the tree to which the updates are performed. We look at all possible update depths. However, in the results section, the results for the updates down to only layers 1 and 2 are omitted for convenience because with this setting little information is propagated forward between time frames and the results show little difference from having Γ set to zero and are thus uninteresting.

3.5.3.4 Performance Measures

As mentioned in the theory section (Section 3.2), in all the following experiments, it is possible to inspect the highest variational posterior probability tree and the segmentations that it generates, for a particular combination of parameters. See for example the segmentation shown in Figure 3.15. Although this is just a qualitative assessment, it can be very useful as an aid to understanding what the algorithm is doing.

As well as this, we use two quantitative assessments of the performance. Firstly we measure the pixelwise classification accuracy, that is to say, the percentage of pixels correctly classified by the variational approximation to the true posterior probability distribution. Secondly, we compare the average variational log likelihood of the test images under the static model and under the temporal model. The variational log likelihood for the temporal model is the same as for the static model, but using the modified CPTs given in Equation (3.21) instead of the static CPTs. The additive constant $\log P(R^{L^H})$ is the same in both cases, and so this term disappears when one term is subtracted from the other.

3.5.3.5 Results

In this section we will discuss the quantitative results obtained from the sequential PEDT classification experiments. The tables of results for the above experiments can

be found in Appendix A. However, before looking at the details, as an illustrative example refer to Figure 3.15 which displays the PEDT that produces the HVPP segmentation for one of the test images. It allows a comparison between the segmention produced by a static PEDT with $\Gamma = 0$, (see Figure 3.15m) and that produced by a PEDT using temporal updates, $\Gamma = 0.15$, (see Figure 3.15k). Figure 3.15o highlights where the use of the temporal updates has improved the segmentation (white) and where it has made matters worse (black). The areas which are either correctly or incorrectly classified in both segmentations are coloured grey.

3.5.3.5.1 Noise Levels Table A.1a shows the classification accuracy of the PEDT model with updates using varying amounts of past information. Figure 3.13 illustrates these results graphically. The points to note are:

- The use of the PEDT model gives rise to improvements in classification accuracy of 9.2, 13.2 and 5.3 percent over the simple pixelwise local model for noise levels of 10^{-1}, 10^{-2} and 10^{-3} respectively.
- The less noise added, the higher the classification accuracy.
- When the added noise has variance 10^{-1}, there is improved classification accuracy over a broad range of Γ with a maximum at around 0.8 to 0.9.
- For the test images with less noise, there is no clear trend in the classification results as Γ varies. It could be that in these low noise regimes, it is hard to improve the performance of the static PEDT so the temporal information is redundant.
- The average variational log likelihood is improved for all values of $\Gamma > 0$ for all three noise levels.

3.5.3.5.2 Update Depths Table A.2 and Figure 3.14 show how the pixel-wise classification performance depends on Γ and the update depth for a fixed amount of noise.The points to note are:

- For shallow updates (down to layers 1-4), the sequential information has little effect.
- When the updates go down close to the leaf layers, the performance can be heavily penalised especially at high Γ.

- The best performance occurs for 5 layer updates, as seen for both the classification and variational log likelihood measures.

To understand the poor performance when the Γ updates do down to near leaf level, bear in mind that each level has an associated scale over which correspondences are made, and this decreases as we approach the leaves. To yield good motion predictions, this scale should match the scale of inter-frame motion that we wish to capture, and enforcing Γ updates at low levels makes this scale too small.

The strong effect of the update depth parameter suggests that fixing Γ^k to be the same for all layers is perhaps not a good approach. Varying Γ from layer to layer would perhaps allow the influence of the previous timeframe posterior to depend on length scale.

3.5.3.5.3 Demonstration As well as the quantitative evaluation of the performance of the algorithm presented above, it may be instructive to look at an example image sequence segmentation for a short test image set. An mpeg movie clip can be found on the technical report website[8]. It shows the PEDT segmentation and an inferred image structure which illustrates the way in which the probability distribution over image segmentations is constructed.

Figure 3.16 shows a sample of frames from the sequence. We can see how the inferred tree structure changes as the balls in the test image sequence move and as ball 2 starts to occlude ball 1. The PEDT tree provides a plausible hierarchical parse structure for the image. (It is worth bearing in mind that the cross-section drawn is of the HVPP tree and that in fact the PEDT framework generates a full probability distribution over node positions, tree structures and node labels in the form of the variational approximation to the true posterior probability distribution).

3.6 Discussion

In this chapter, we have looked at the extension of the PEDT architecture to deal with sequences of images. We have been interested in two tasks: image coding and image segmentation.

With the aim of coding sequences of images as efficiently as possible, we examined the use of the static PEDT with the HVPP tree mapping alogrithm. We have showed

[8]See: `http://www.dai.ed.ac.uk/homes/stephenf/pedt/`

that we can achieve significant reductions in the coding cost of test images with respect to all our comparitive image coding formats and also with respect to the cost of coding using the static PEDT and the one step late tree which we can think of as the analogue of the former method but with the most simple method of incorporating the sequence information.

The HVPP mapping algorithm is not put forward as the only way to propagate the sequence information for image coding nor even as the correct way to do it. However, we do show that using the PEDT framework and the HVPP mapping algorithm, it is possible to incorporate the sequential information effectively and this results in improved coding costs. It is certainly possible to improve the results further still. We have pointed out specific improvements which can be made.

A pleasing side-effect of the success of the HVPP mapping algorithm at image sequence coding is that it provides an illustration that the highest variational posterior probability configurations of the tree structure and class variables are a useful method of visualising the structure of images. We can say this because the HVPP tree mapping algorithm uses only the HVPP structure of the variational distributions for a pair of images and from this it distills a mapping which allows us to represent efficiently the changes in the images between the frames. This would only be consistently possible over a full set of images if the HVPP structure was somehow representative of the underlying structure of the images.

In this chapter, we have also developed the Sequential PEDT model and evaluated its performance on a test image sequence. We have shown that adding temporal constraints can lead to increases in performance. When we use the image coding cost as a measure we have shown that significant gains can be made.

On the other hand, when we consider the image labelling task and use the pixelwise classification accuracy as a measure of performance, whilst we see some positive effects, the benefits may not always be very large. We have already argued that the pixelwise accuracy is a poor measure of performance for a model of what is essentially the latent structure of the image as the effects of the local image model also play a large part in the results. Another reason for the difficulty in assessing the model's benefits in the labelling task may be that there are a relatively small number of labels, and thus the predictive gain one can achieve through the use of temporal context (in bits/pixel) is relatively small.

A possible future direction of work could be to replace the discrete-state structure

of the X variables with real-valued variables; for example one could use a Gaussian distribution. This would extend the fixed-tree Gaussian TSBN models of Willsky *et al* [6, 50] to temporal dynamic tree architectures.

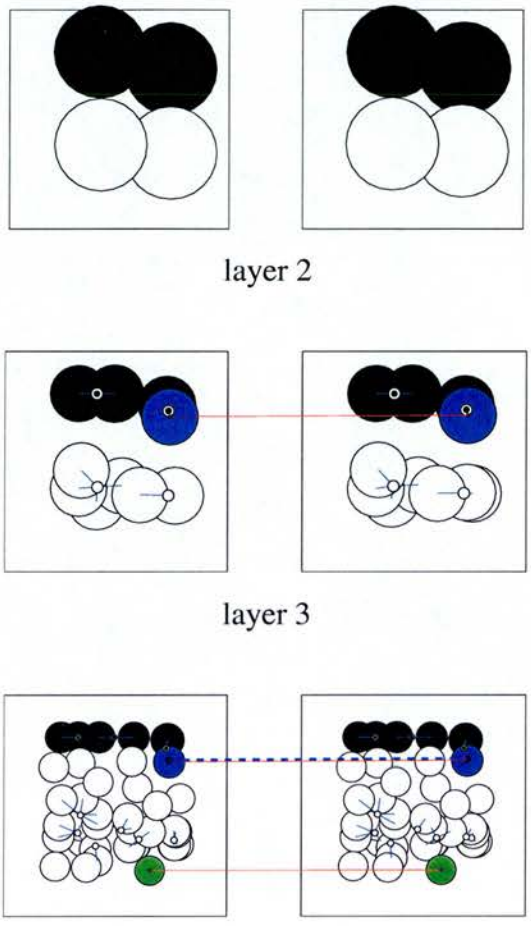
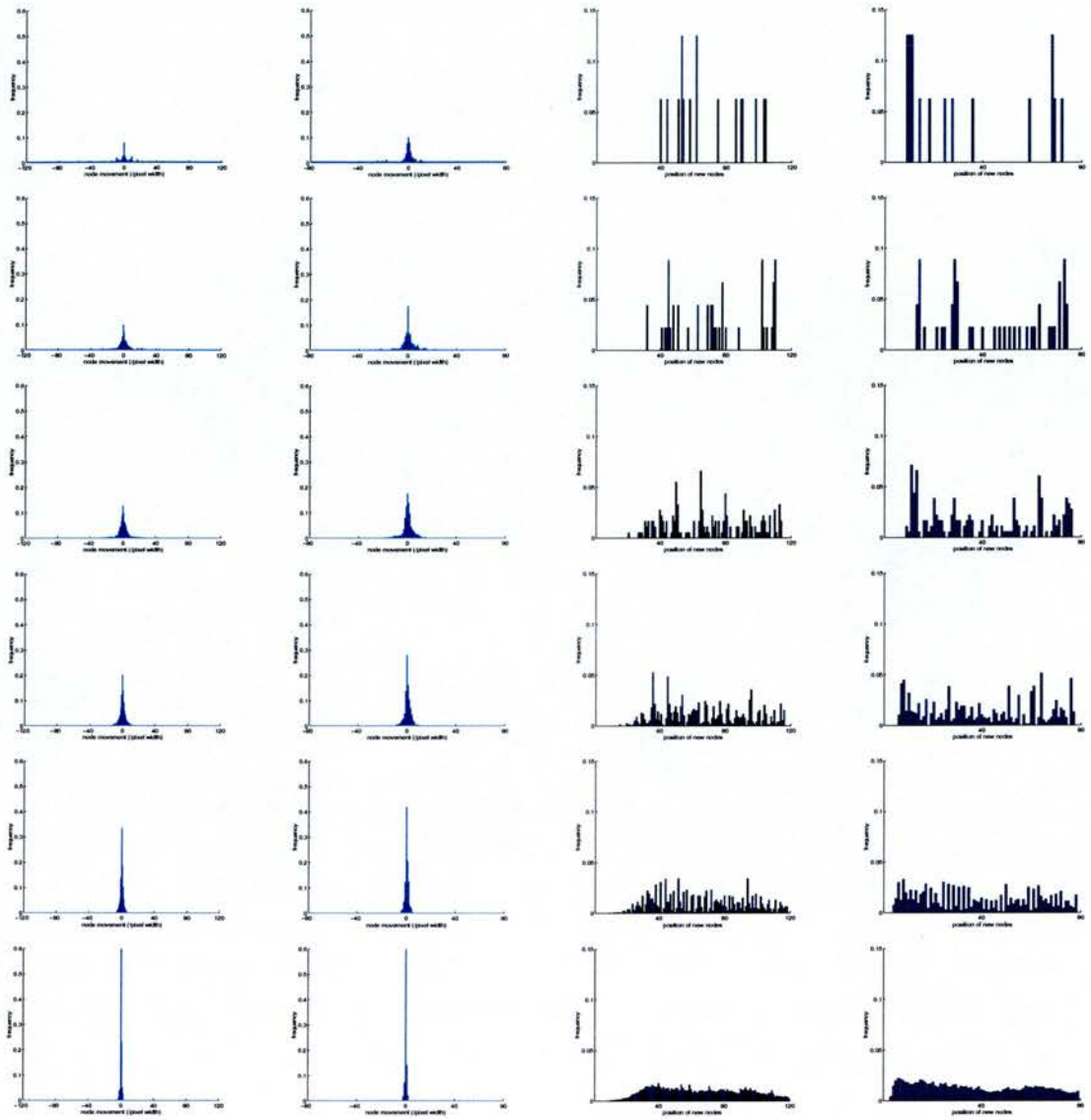


Figure 3.4: Example of Mapping Object Nodes. The figure shows three layers from the HVPP trees of the first image from the test set (LH column) and the second image from the test set (RH column). The large circles represent the nodes belonging to that layer, their HVPP class value is indicated by their colour. The smaller circles and the lines connecting them to the larger nodes show the positions of the parent nodes for the nodes in the layer. Thick dashed lines indicate mappings in the parent layer and the red lines show the mappings in the current layer.



(a) (b) (c) (d)

Figure 3.5: Frequencies of Node Movements and New Node Positions in Each Layer. The histograms for layer 3 are at the top and they go down in order so that the histograms for layer 8 are in the bottom row. The figure shows the frequency of occurence of the distances of node movement between frames (column (a) is the horizontal direction in the images and (b) is the vertical direction), and the frequency at which new nodes are placed in each pixel position by the mapping (column (c) is the horizontal direction in the images and (d) is the vertical direction). The data was generated by making mappings between all the pairs of consecutive images in the 50 image training sequence.

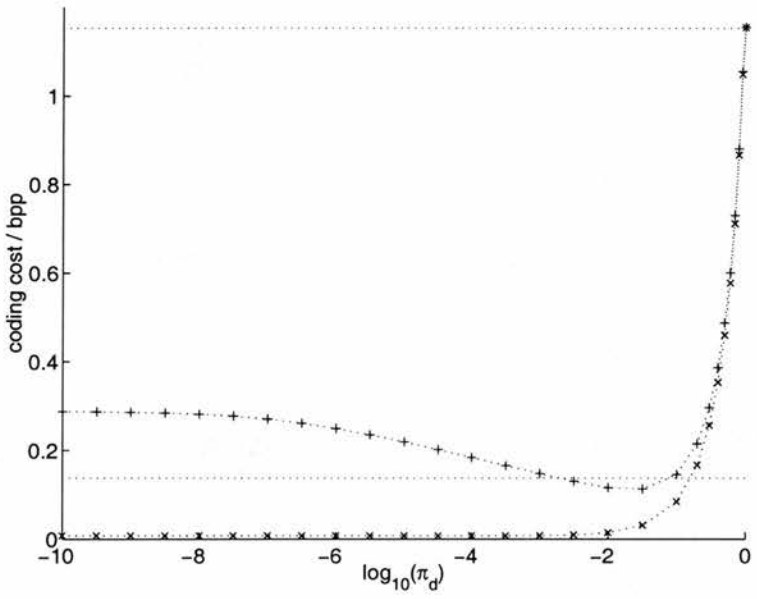


Figure 3.6: Preliminary Experiment Showing the Effect of Smoothing on Image Coding Costs. The figure shows the cost of coding the test image sequence using the CPTS from the variational prior distribution and given the one step HVPP tree structure with a varying amount of smoothing as indicated by the probability of disconnection ($+$ points). For comparison, the figure also shows the cost of coding the test image sequence using the same CPTs from the variational prior distribution and given the actual HVPP tree structure for the image with a varying amount of smoothing ($\times$ points). Also, to give a sense of scale, the upper dotted line at 1.153bpp shows the average pixelwise entropy of the test sequence and the lower dotted line at 0.138bpp shows the coding cost of the test sequence in `tar.gz` format.

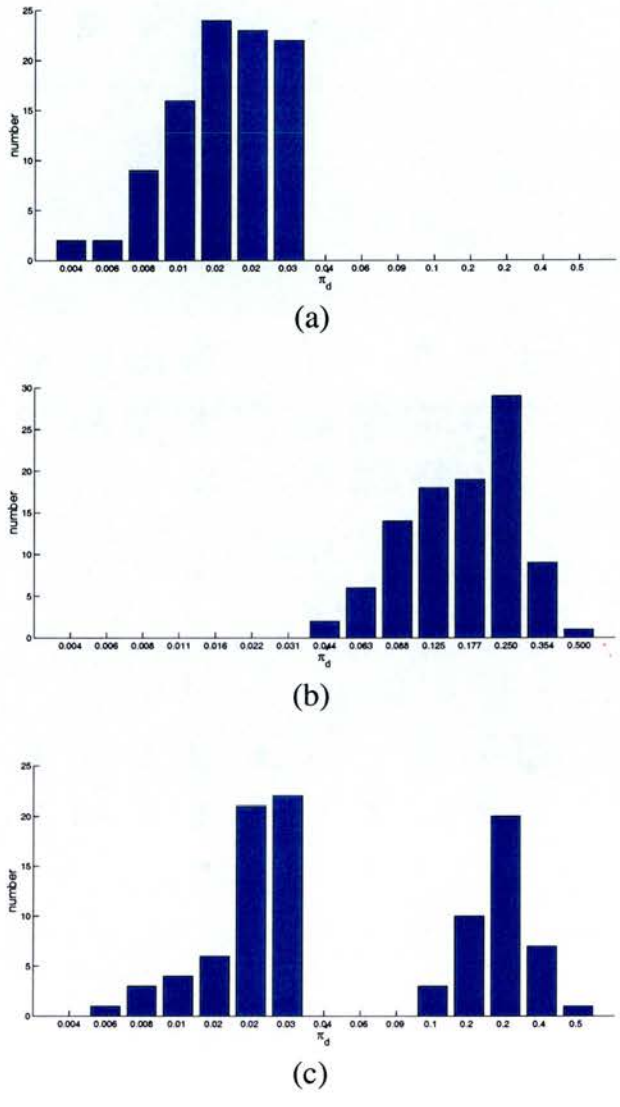
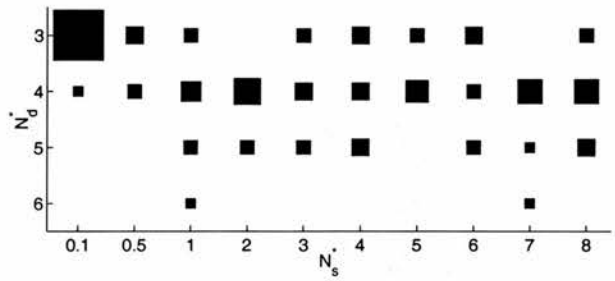
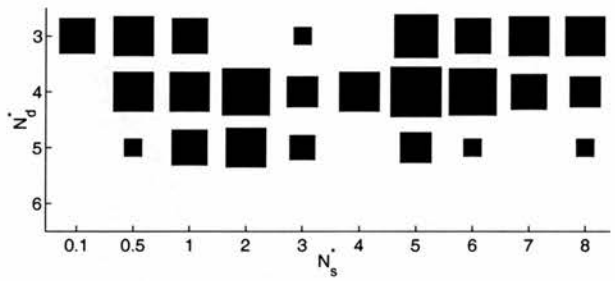


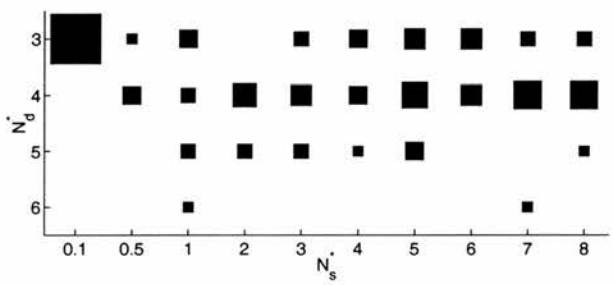
Figure 3.7: Optimum Values of Parameter, π_d, for the HVPP Mapping Algorithm. The figure shows the frequency with which each of the possible values of the probability of disconnection are chosen as the best value for coding the test sequence using the HVPP Mapping Algorithm. Histogram (a) shows the frequencies when using full smoothing, histogram (b) shows the frequencies when using reduced smoothing and (c) shows the frequencies when the best of the two methods is chosen.



(a)



(b)



(c)

Figure 3.8: Optimum Values of Parameters, N_s and N_d, for the HVPP Mapping Algorithm. The figure shows the frequency with which each of the possible values of the search distance, N_s and the mapping depth, N_d are chosen as the best value for coding the test sequence using the HVPP Mapping Algorithm. The plots are Hinton diagrams in which the area of the squares is proportional to the frequency. Plot (a) shows the frequencies when using full smoothing, plot (b) shows the frequencies when using reduced smoothing and (c) shows the frequencies when the best of the two methods is chosen.

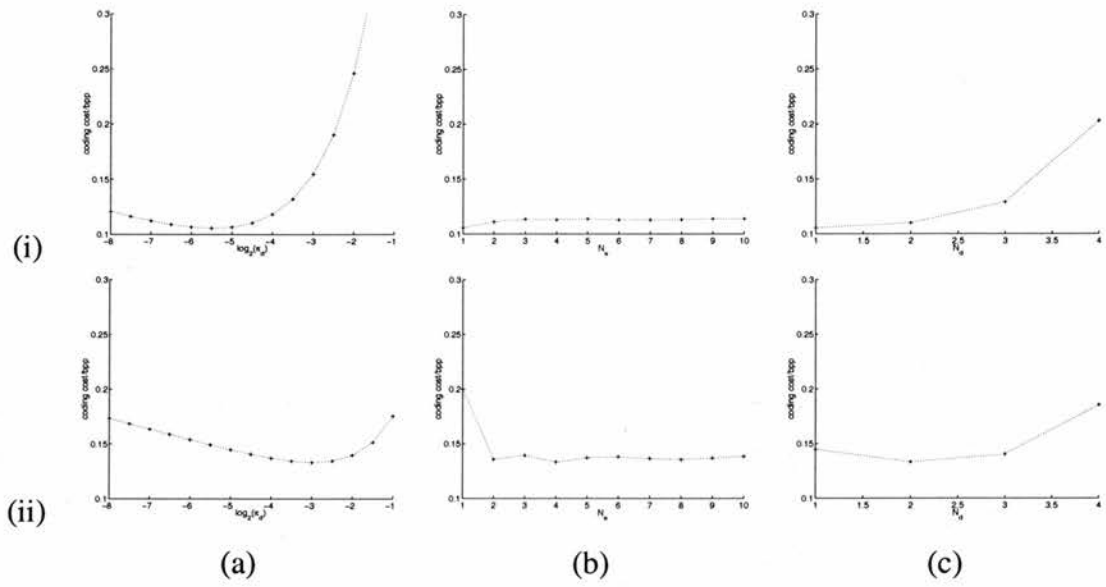


Figure 3.9: Average Coding Cost of the Test Image Sequence with Fixed Parameter Values. The figure shows how the average coding cost of the test sequence varies for different values of the parameters which are fixed for the whole sequence. Row (i) shows the coding costs for the HVPP mapping algorithm with full smoothing. Graph (i)a. shows the coding cost varying with different values of the probability of disconnection, (i)b. shows the variation with N_s and (i)c. shows the variation with N_d. For each of these graphs the parameters which aren't changed are held fixed at $\pi_d = 0.02$, $N_s = 0.1$ and $N_d = 3$ (the best parameter values as chosen on an independent training set). Similarly row (ii) shows the coding costs for the HVPP mapping algorithm with reduced smoothing. Again graphs (ii)a, (ii)b and (ii)c show the variation with π_d, N_s and N_d respectively. The non-varying parameters are held fixed at $\pi_d = 0.125$, $N_s = 3$ and $N_d = 4$.

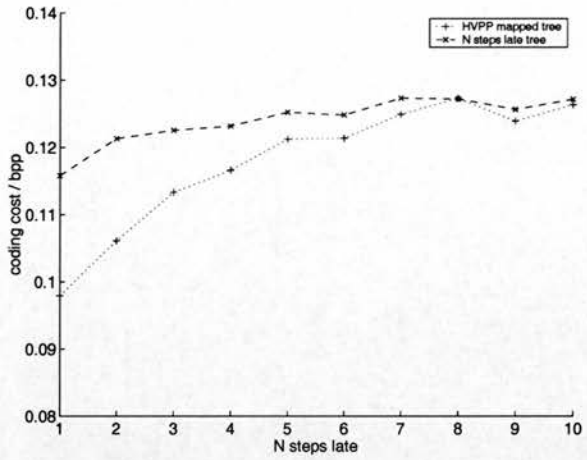


Figure 3.10: Comparison of Image Coding Costs using the HVPP Map Tree and the N Steps Late Tree. The points plotted with "$+$"s show the average coding cost of the image at time t given the HVPP mapped tree derived from the image at time $t-N$. The points plotted with "$\times$"s show the average coding cost of the image at time t given the HVPP for the image at time $t-N$.

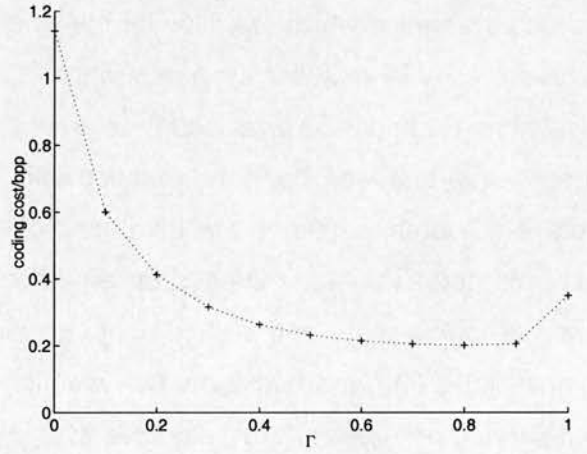


Figure 3.11: Coding Cost of the Test Sequence using the Sequential PEDT with Varying the Probability of Back Connection, Γ. The figure shows the estimated coding cost for the test image sequence, the update depth parameter is set at $N_d = 3$.

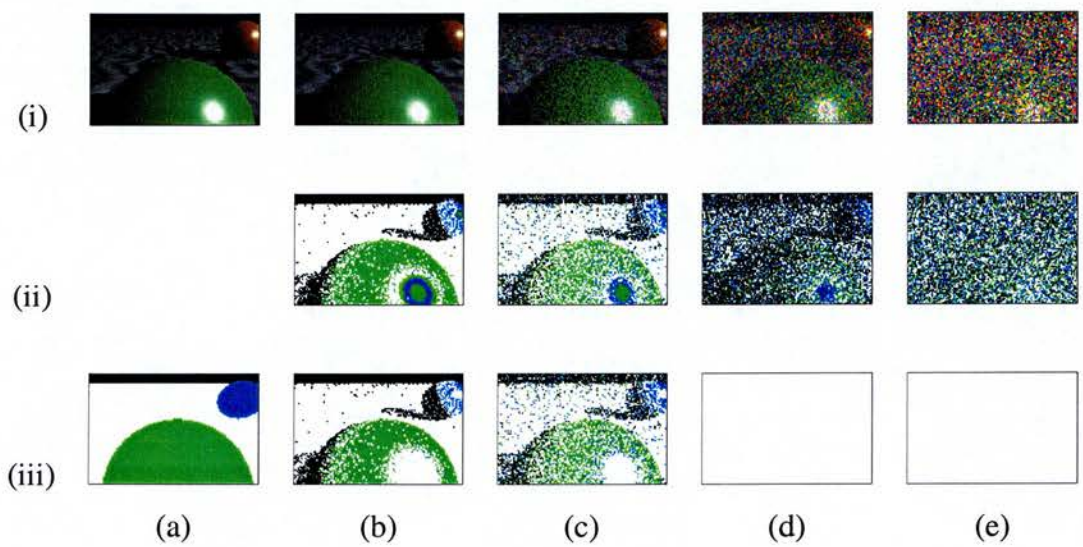
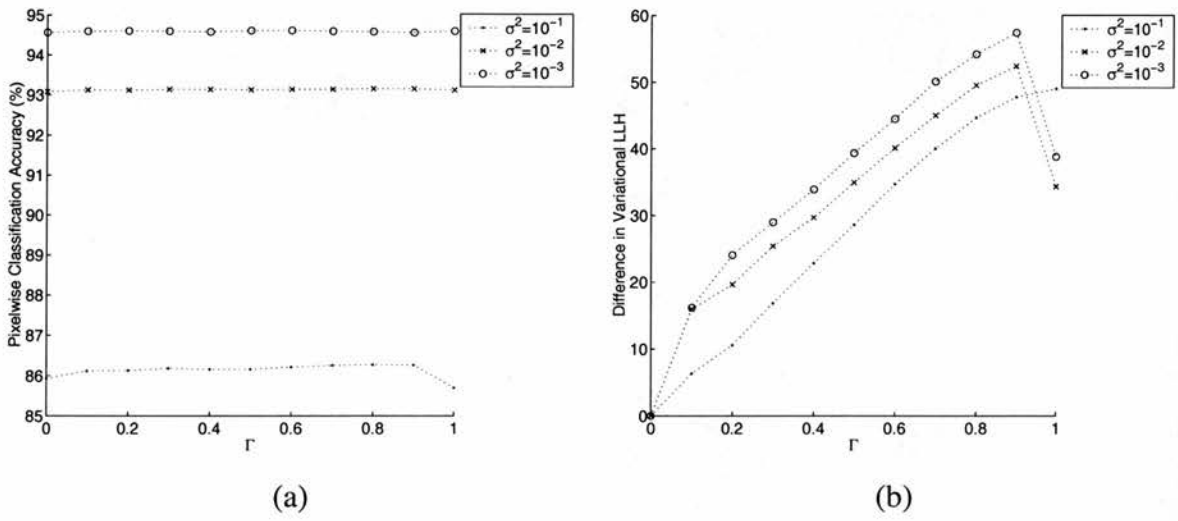
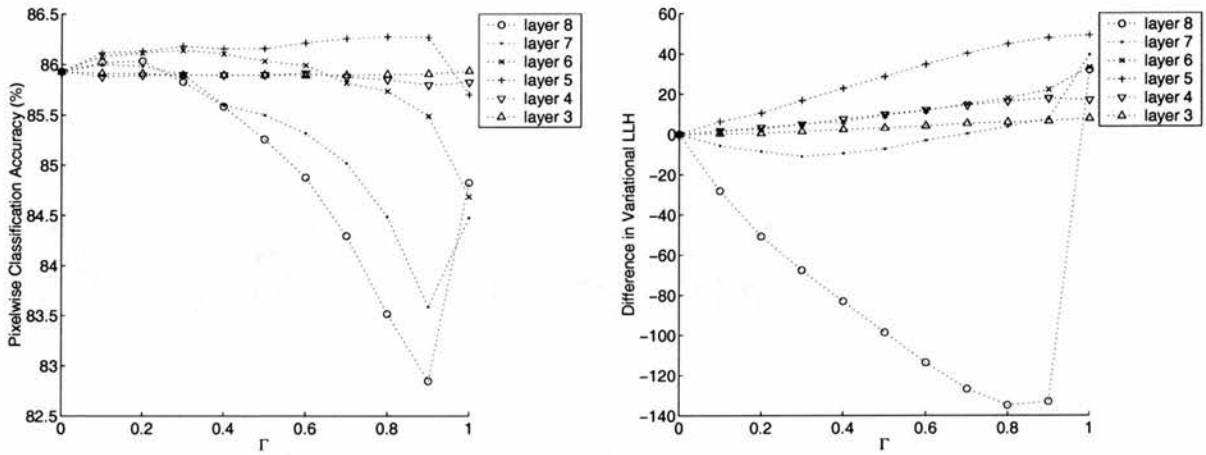


Figure 3.12: The Local Segmentations of a Test Image. Subfigure (a-i) shows the example test image and Subfigure (a-iii) shows its corresponding ground truth labelling. Excluding column (a), row (i) shows the test image with various amounts of added noise, row (ii) shows the maximum likelihood segmentation under the local image model and row (iii) shows the MAP segmentation under the local image model. The variance of the Gaussian noise added is 10^{-3} in column (b), 10^{-2} in column (c), 10^{-1} in column (d) and 1 in column (e). The key for the colour of image labels is the same as that found in Figure 3.2.



(a) (b)

Figure 3.13: Results of Sequential PEDT Performance Experiments with Varying Noise Levels. Subfigure (a) shows the pixelwise classification accuracy for each of the levels of image noise over a range of values for Γ, Subfigure (b) shows the improvement in the average variational log likelihood of all the images in the test set for a range of values of Γ and for each of the noise levels, over the average variational log likelihood of all the test images under the static PEDT model ($\Gamma = 0$). The average values of L_{var} for the static PEDT model are 2706.1, 2531.2 and 2334.4 for noise levels of 10^{-1}, 10^{-2} and 10^{-3} respectively.



(a) (b)

Figure 3.14: Results of Sequential PEDT Performance Experiments with Varying Update Depths. Subfigure (a) shows the pixelwise classification accuracy for each of the update depths over a range of values for Γ, Subfigure (b) shows the improvement in the average variational log likelihood of all the images in the test set for a range of values of Γ and for each of the update depths, over the average variational log likelihood of all the test images under the static PEDT model ($\Gamma = 0$).

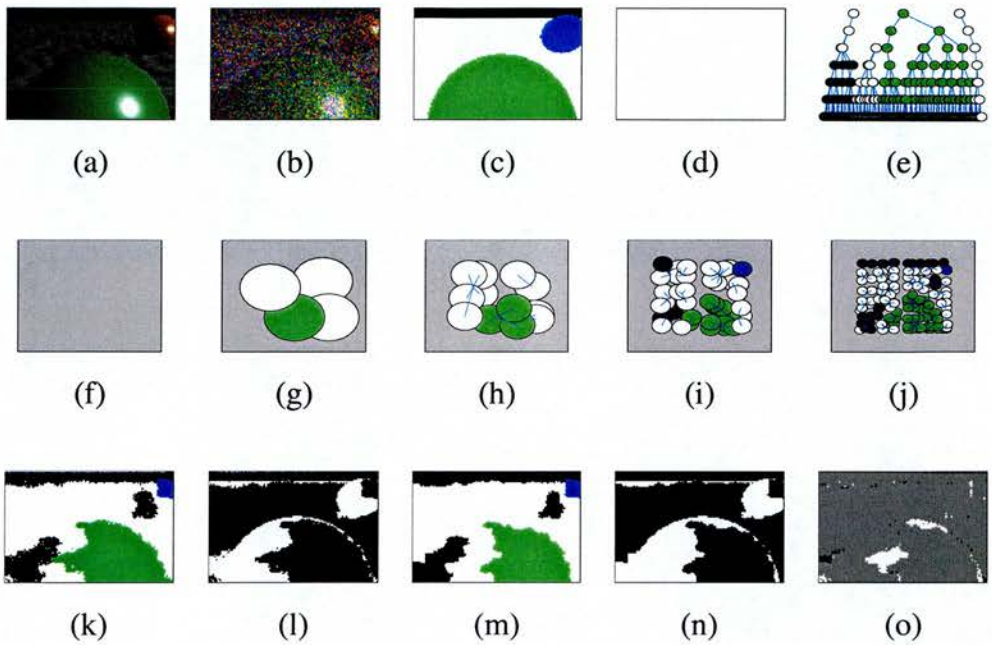


Figure 3.15: The HVPP Position Encoding Dynamic Tree for an Image in the Test Sequence. The subfigures display: (a) the test image; (b) the noisy image on which the inference is performed; (c) the ground truth target image; (d) the HVPP segmentation under the local image model; (e) a cross-section through the HVPP PEDT (the leaf nodes in the 70th row of the image are shown along with their hierarchy of ancestors); (f)-(j) the highest five layers of nodes (the nodes in Subfigure (g). form root nodes so there are no nodes present in the top layer), the bottom three layers of nodes have been omitted as they are difficult to read; (k) the HVPP PEDT segmentation with $\Gamma = 0.15$; (l) the pixels misclassified by the PEDT with $\Gamma = 0.15$ (white); (m) the HVPP segmentation with the static PEDT ($\Gamma = 0$); (n) the pixels misclassified by the static PEDT; (o) the gain between using the static PEDT and the PEDT with $\Gamma = 0.15$, (the pixels wrongly classified by the former and correctly by the latter are white, the pixels correctly classified by the former and wrongly by the latter are black, the rest are grey).

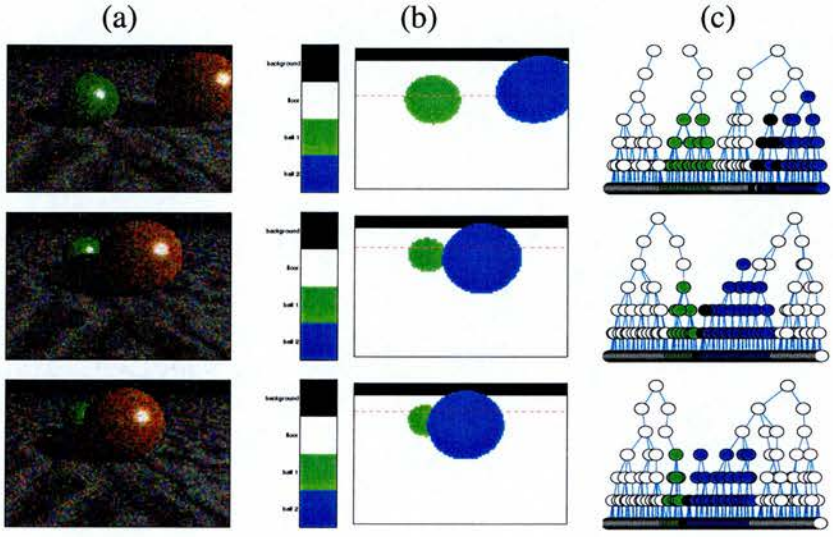


Figure 3.16: Cross-section of the HVPP Tree Structure for Frames 15, 19 and 20 from the Test Image Sequence. Column (a) shows the test images (with variance of added noise 10^{-1}); (b) shows their ground truth segmentation (key: black = background, white = floor, green = ball 1, blue = ball 2). The red dotted line on the ground truth images indicates the line on which lie the leaf-nodes of the cross-section through the highest variational posterior probability (HVPP) tree that is shown in column (c). The line of the cross-section is chosen such that it is the horizontal line that passes through the widest part of ball 2 in the ground truth display. (Note that although this line is chosen based on the ground truth image in order hopefully to show a part of the tree with interesting features, the tree itself and the probability distribution from which it is drawn was generated with no reference to the ground truth labels.)

Chapter 4

Products and Sums of Tree-Structured Gaussian Processes

This chapter looks at a different aspect of hierarchical graphical models. We shall study ways that individual tree structures of continuous Gaussian random variables can be combined and how this improves their representative power. The structure of the chapter is as follows. In Section 4.2 we discuss the theory of tree-structured Gaussian processes (TSGPs), products of TSGPs and sums of TSGPs. In Section 4.3 we present the results of experiments of matching the products and sums of TSGPs to target stochastic processes. Section 4.4 provides an analysis of why an exact representation using only two component TSGPs can be found for AR(1) and MA(1) target processes. We conclude with a discussion of the issues raised. This chapter is based on work done in Williams and Felderhof (2001),[83].

4.1 Introduction

Recently Hinton (1999) has introduced the Products of Experts (PoE) model in which several individual probabilistic models for data are combined to provide an overall model of the data. In this chapter we consider a PoE model in which each expert is a Gaussian. It is easy to see that in this case the product model will also be Gaussian. However, if each Gaussian has a simple structure, the product can have a richer structure. Below we constrain each expert to be a tree-structured Gaussian process (TSGP), for which efficient inference can be carried out in time linear in the number of leaves. Our aim is then to investigate how products of TSGPs can be used to approximate more

general (non tree-structured) Gaussian processes. Such product-of-Gaussians models may be useful in applications such as image analysis, where it is not feasible to deal with the very large matrix calculations that might otherwise be involved.

Using Gaussian experts is attractive as it permits a thorough analysis of the product architecture, which can be difficult with other models, e.g. models defined over discrete random variables.

An alternative way of generating more general Gaussian processes from component TSGPs is to consider the process which arises from summing the data generated by each of the individual TSGPs. It is interesting to compare the types of stochastic process which can be effectively modelled by each approach.

4.2 Theory

We consider in turn a single tree-structured Gaussian process (TSGP), the product of permutations of TSGPs, the sums of permutations of TSGPs and the fitting of products and sums of TSGPs to target processes.

4.2.1 Tree-Structured Gaussian Processes

Consider m nodes with indices $1,\ldots,m$. A Gaussian process over these nodes is defined by the mean vector (which without loss of generality we take to be zero) and the covariance matrix. Tree-structured Gaussian processes are a subset of Gaussian processes in which the distribution over the m leaf nodes is constructed by summing independent random variables organised in a tree-like manner, as illustrated in Figure 4.1(a). Below we consider one-dimensional TSGPs, although it is easy to extend this to higher dimensions; for example, in two dimensions a quad-tree branching structure can be used.

To sample from a TSGP we start at the root node and work downwards, using the equation

$$x_i^{(j)} = a x_{i-1}^{Pa(j,i)} + b_i w_i^{(j)}. \tag{4.1}$$

Here $x_i^{(j)}$ denotes the jth node in the ith layer, $Pa(j,i)$ is a function which returns the index of the parent of the node $x_i^{(j)}$, a and $\{b_i\}$ are constants, and the w's are a set of independent $N(0,1)$ Gaussian random variables. The root node takes the value $x_1^{(1)} = b_1 w_1^{(1)}$. The process that we are interested in is $\mathbf{x}_L$, the set of x variables at level

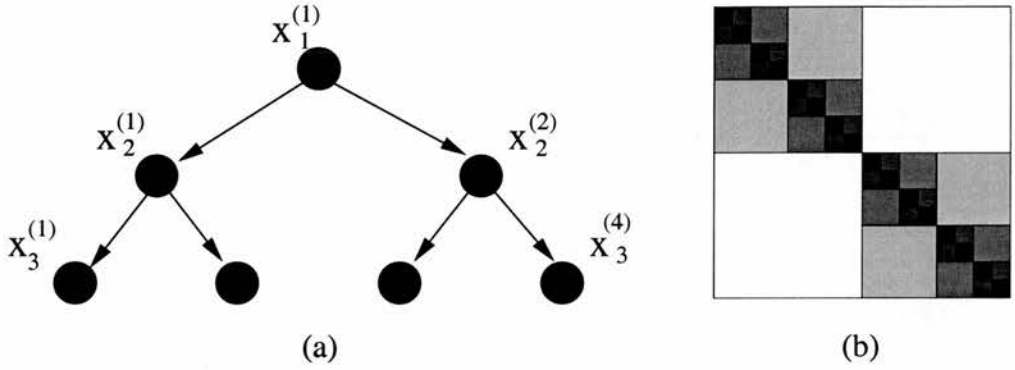


Figure 4.1: (a) A 3-layered Tree-Structured Gaussian Process. (b) A Schematic Representation of the Covariance Matrix for the Leaves of a 5-Layer Tree. Areas of constant shading represent equal covariance. Since the covariances are all positive, the darker shades of grey represent higher covariance.

L. By repeatedly applying Equation 4.1 we find that

$$x_L^{(j)} = \sum_{i=1}^{L} a^{L-i} b_i w_i^{Pa^{L-i}(j,L)}, \tag{4.2}$$

where $Pa^k(j,L)$ is a function that returns the value of the kth ancestor of the node $x_L^{(j)}$. The variance of a leaf node is easily calculated as

$$var(x_L^{(j)}) = \sum_{i=1}^{L} a^{2(L-i)} b_i^2 \stackrel{def}{=} \sum_{i=1}^{L} \sigma_i^2, \tag{4.3}$$

using the reparameterization $\sigma_i = a^{L-i} b_i$. The covariance of any two leaf nodes turns out to be the variance of their common parent. Note that allowing extra freedom in the parameterization of a (e.g. allowing a to depend on the level i) does not increase the flexibility of the TSGP as it is the σ_i which defines the covariance structure. The form of the covariance matrix for the tree-structured process is schematically represented in Figure 4.1b. As there is a σ_i parameter for every layer there are $\log_2 m + 1$ parameters for a binary tree. Since σ_i^2 is added at the ith node, every entry in the TSGP covariance matrix is constrained to be non-negative.

Tree-structured Gaussian processes have been studied by a number of authors including the group led by Prof. Willsky at MIT [14, 48]. In [48] a more constrained type of hierarchical stochastic process is considered in which the b_is decay exponentially: $b_i = 2^{-\delta i/2}$, (δ is a constant), leaving only two free parameters.

It is also useful to consider relaxing the form of Equation 4.1. For example different values of a might be used for the left and right branches at each non-leaf node; this kind

of construction is discussed in Section 4.4. It is also possible to generalize Equation 4.1 so that $x_i^{(j)}$ is vector valued, and a and the b's then become matrices (see e.g. [14]). It is perhaps worth noting that an equivalent way of constructing a tree-structured Gaussian process is as the linear combination of a set of eigenfunctions, see Appendix B for more details.

A significant feature of the tree-structured Gaussian process is that the process $\mathbf{x}_L$ at the leaf nodes of a single Gaussian tree is a non-stationary process, i.e. the covariance between $x_L^{(i)}$ and $x_L^{(j)}$ does not depend only on $|i-j|$. In the next two sections, we will look at ways which in which we can use cyclic permutations of the single tree-structured process to create new stationary processes.

4.2.2 Products of Tree-Structured Gaussian Processes

We will only consider cyclic permutations which involve moving the whole tree structure relative to the nodes, wrapping the structure around at the edges. We will not consider more general permutations which would involve changing the ordering of the nodes with respect to one another.

Let $\mathbf{P}_i$ denote a cyclic permutation matrix, (a matrix of zeros with a single "1" in each row and column), such that $\mathbf{P}_i\mathbf{y}$ is the ith cyclic permutation of vector $\mathbf{y}$. (The 0th permutation is the identity). If $\mathbf{C}_0$ denotes the unpermuted covariance matrix, the covariance of $\mathbf{P}_i\mathbf{y}$ is $\mathbf{C}_i = \mathbf{P}_i\mathbf{C}_0\mathbf{P}_i^T$.

Let the product of n zero-mean Gaussians have covariance matrix $\mathbf{C}_{model}$. We then have $\mathbf{C}_{model}^{-1} = \sum_{i=0}^{n-1} \mathbf{C}_i^{-1}$. Note that if $\mathbf{x}_L$ has length m, then the product of all m cyclic permutations of a TSGP will give rise to a stationary Gaussian process with a Toeplitz covariance matrix.

4.2.3 Sums of Tree-Structured Gaussian Processes

In this construction, the process that the model generates, $\mathbf{x}$, is formed by summing permuted TSGPs, $\mathbf{x}_L^{(i)}$. In other words, $\mathbf{x} = \sum_i \mathbf{x}_L^{(i)}$. The overall covariance matrix is the sum of the covariance matrices of all the component processes[1], $\mathbf{C}_{model} = \sum_i \mathbf{C}_i$. A special case of this technique is the cyclostationary process, described in [60] (section 11-4). The cyclostationary construction with a discrete process of period m would produce an overall covariance matrix which is the average of m shifted copies of $\mathbf{C}_0$.

[1] $\mathbf{C}_{model} = \left\langle (\mathbf{x}_L^{(1)} + \ldots + \mathbf{x}_L^{(n)})(\mathbf{x}_L^{(1)} + \ldots + \mathbf{x}_L^{(n)})^{\mathbf{T}} \right\rangle = \sum_{i=1}^{n} \left\langle \mathbf{x}_L^{(i)} \mathbf{x}_L^{(i)\mathbf{T}} \right\rangle$.

Notice that for the products of TSGPs (PTSGPs) we multiply the probability distributions of the component processes, $P(\mathbf{x}_L) = \prod_i P(\mathbf{x}_L^{(i)})$. The overall process is a conjunctive representation of its experts. A particular data pattern $\mathbf{x}_L$ only has a high probability if it has high probability under each of the permuted TSGPs. On the other hand, the Sum of Tree-Structured Gaussian Processes (STSGP) is generated by adding together the component processes which represents a convolution of the component densities, [80]. In this case the covariance of the model is the sum of the permuted covariance matrices whereas for the PTSGP, the *inverse* covariance of the model is the sum of the permutations of the *inverse* covariance matrix of the TSGP.

4.2.4 Target Processes

As we have indicated above, a problem with a single TSGP is its non-stationarity. Our aim will be to find out how well products of TSGPs can model a stationary target process. Our target processes are (a) first order autoregressive processes, AR(1), (b) second order moving average processes, MA(1), (c) second order autoregressive processes, AR(2). We construct each of these processes on a circle. If there are m discrete points in the process, then the mth point wraps around to be adjacent to the first point. For AR(1), the value of the process at the ith point $X_i = \alpha X_{\mathrm{mod}(i-1,m)} + Z_i$ where Z_i is a zero mean Gaussian random variable with variance v and $i \in \{0,\ldots,m-1\}$. In this case the inverse covariance matrix has a tridiagonal circulant structure. The diagonal elements have value $(1+\alpha^2)/v$ and the off-diagonal elements are $-\alpha/v$. Similarly for the AR(2) process, $X_i = \beta X_{\mathrm{mod}(i-1,m)} + \alpha X_{\mathrm{mod}(i-2,m)} + Z_i$. Now the inverse covariance matrix has a pentadiagonal circulant structure and reading from the central diagonal outwards the lines have values $(1+\alpha^2+\beta^2)/v$, $(\alpha\beta-\beta)/v$, $-\alpha/v$. A general MA(l) process on a discrete lattice is a linear filter applied to $\{Z_i\}$, of the form $X_i = \sum_{j=0}^{l} \beta_j Z_{\mathrm{mod}(i-j,m)}$ where by convention $\beta_0 = 1$. The process has covariance $\gamma(k) = \sum_{i=0}^{l-k} \beta_i \beta_{i+k}$ for $k = 0,\ldots,l$ and zero for $k > l$ [10].

4.2.5 Fitting Products and Sums of TSGPs to Target Processes

Through the construction of either the PTSGP or the STSGP we arrive at the covariance matrix for the model $\mathbf{C}_M$, which depends on the $\{\sigma_i\}$ parameters. (We assume that all experts use the same parameters.) Our task is now to fit this to the target process with covariance matrix $\mathbf{C}_T$. A natural measure of discrepancy is the Kullback-Leibler diver-

gence $\int P_T(\mathbf{x})[\log P_T(\mathbf{x}) - \log P_M(\mathbf{x})]d\mathbf{x}$ between the target distribution $P_T(\mathbf{x})$ and the model distribution $P_M(\mathbf{x})$. Minimising this KL-divergence is equivalent to maximising the infinite data log likelihood. For the special case of two zero-mean Gaussians this is given in [41]:

$$D_{KL} = \frac{1}{2}\log\frac{|\mathbf{C}_M|}{|\mathbf{C}_T|} + \frac{1}{2}\mathrm{tr}\mathbf{C}_T(\mathbf{C}_M^{-1} - \mathbf{C}_T^{-1}). \tag{4.4}$$

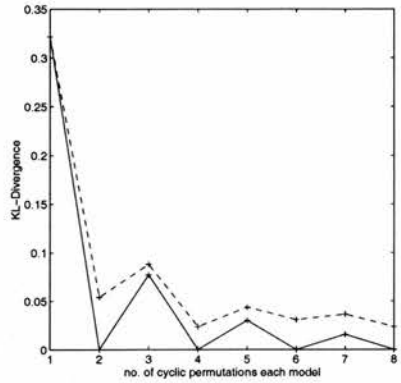
For convenience a simplex search (the `fmins` function in MATLAB) was used to carry out this optimization, although more sophisticated gradient-based methods could also be used. The possibility of local optima in the search space was addressed by initializing the search several times using different randomly-chosen starting points for several target processes. From this we concluded that there do not seem to be significant local optima problems in the search, so in the experiments below only one optimization is carried out per experimental configuration.

4.3 Experimental Results

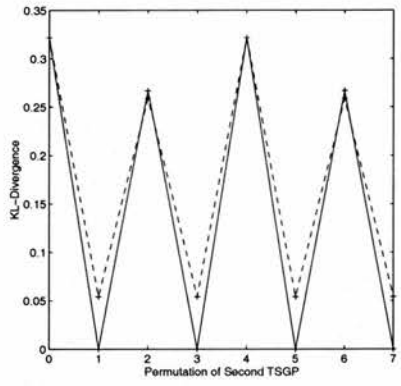
We have conducted experiments to investigate how well the PTSGP and STSGP models can be used to fit the target processes.

The first question to ask is: how many permutations of the single TSGP do we need to model the target processes effectively and which permutations are the best ones to choose? There are many possible ways of choosing which of the permutations to be included in the model, for each of the different target processes each of which can have different parameters settings. Figure 4.2 shows a selection of the results. The figure shows the results of trying to fit PTSGPs and STSGPs to the AR(1) target process with $\alpha = 0.5$ and $m = 8$.

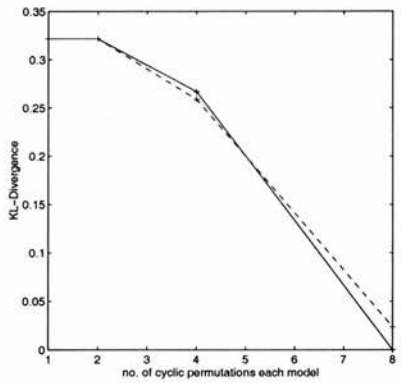
Firstly, note that having just a single TSGP is the worst fitting model and that combining one or more permutations of the TSGP either in the STSGP or PTSGP frameworks always reduces the KL-divergence between target and model (except when the TSGP which is shifted halfway round the circle so that it has the same covariance structure as the original TSGP, then using just this and the original TSGP in a model gives the model no extra flexibility and KL-divergence doesn't decrease). The use of all of the possible permutations, in either the PTSGP or the STSGP, produces the best fitting model possible. This may also be achieved with a smaller number of permutations. For the AR(1) target, the PTSGP with two TSGPs ($\mathbf{C}_0$ and $\mathbf{C}_1$) fits the target exactly, i.e. the KL-divergence is zero. On the other hand, the STSGP isn't able to fit the target



(a)



(b)



(c)

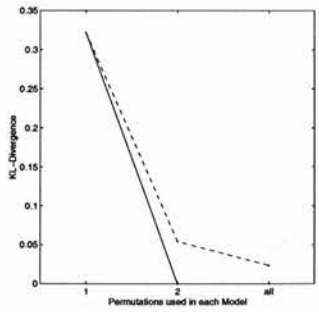
Figure 4.2: Combining the Permuted TSGPs. These graphs show the KL-divergence between the target process (in this case AR(1)) and the model process which is either the sum (dashed line) or product (solid line) of TSGPs. Each graph shows a different way that one might use to choose the permutations of the TSGP to include in the sum/product: (a) add extra permutations one at a time, the new permutation added is adjacent to the last one added ie $\{\mathbf{C}_0\}$, $\{\mathbf{C}_0, \mathbf{C}_1\}$, $\{\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2\}$, etc; (b) the unpermuted TSGP and one other at each different shifted position, $\{\mathbf{C}_0, \mathbf{C}_0\}$, $\{\mathbf{C}_0, \mathbf{C}_1\}$, $\{\mathbf{C}_0, \mathbf{C}_2\}$, etc; (c) doubling the number of TSGPs keeping them evenly spaced around the circle, $\{\mathbf{C}_0\}$, $\{\mathbf{C}_0, \mathbf{C}_4\}$, $\{\mathbf{C}_0, \mathbf{C}_2, \mathbf{C}_4, \mathbf{C}_6\}$, etc.

exactly with any number of permutations. However, using just two permutations in the STSGP allows it to do nearly as well as the best it does which is when it uses all the possible permutations. With these considerations in mind, we will subsequently only present results for the cases where (i) only the single TSGP has been used (ii) two TSGPs have been combined (the original TSGP and a TSGP shifted by one) and finally (iii) all the possible TSGP permutations have been combined.

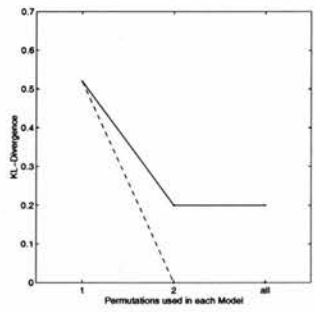
Figure 4.3 shows plots of the KL-divergence between the selected target processes and the PTSGPs and STSGPs each with one, two and all permutations as just described, for $m = 64$. Figure 4.3(a) shows again that a PTSGP can exactly fit an AR(1) process with just two permutations and that the STSGP can't with any number. Figure 4.3(b) shows that the STSGP can exactly match an MA(1) process with just two permutations and in this case the PTSGP doesn't have sufficient flexibility to fit the target. Figure 4.3(c-e) show the performance of the two approaches when the target process is AR(2). The different cases correspond to different parameter values[2] α and β. We can see that although the different methods perform best in different settings, neither actually manage to reduce the KL-divergence to zero.

Figure 4.4(a) shows the structure of the inverse covariance matrix of the product of two TSGPs which have been adjusted so that the product fits the AR(1) target (see Figure 4.5(c)) and similarly Figure 4.4(b) shows the structure of the covariance matrix of the sum of two TSGPs which have been adjusted so that the sum fits the MA(1) target process (Figure 4.4(d)). Notice that in both cases the fitted TSGPs only have covariance in the bottom two layers of the tree (see the right panels of figures 4.4(a) and (b)). This can be understood by the inspection of Figure 4.5. Figure 4.5(b) shows the structure of the inverse covariance matrix of a TSGP which only has covariance in the bottom two layers. Using the products construction, we can see that if we add the values in this matrix to those in an identical matrix which has been permuted by one then we will be able to get a inverse covariance matrix in the form of the target AR(1) process. Likewise Figure 4.5(a) shows the structure of the covariance matrix of a TSGP which only has covariance in the bottom two layers. With the summation construction, if we add the values in this matrix to those in an identical matrix but which has been permuted by one then we will be able to get a covariance matrix in the form of the

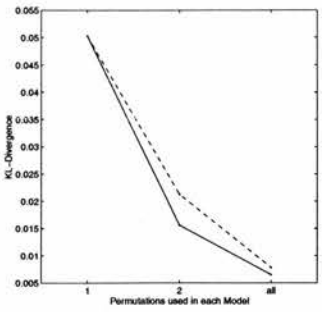
[2]Analysis in [19] suggests that we might expect different regimes of behaviour depending on the values of α and β. For the AR(2) process, we solve the equation for the covariance, $\gamma_k = \beta\gamma_{k-1} + \alpha\gamma_{k-2}$ by assuming a solution of the form $\gamma_k = \lambda^k$ which gives a quadratic in λ. Whether $\beta^2 - 4\alpha$ is less than, equal to or greater than zero affects the solution. Note that the specific values that we use for each case are the same as those used in the examples in [19].



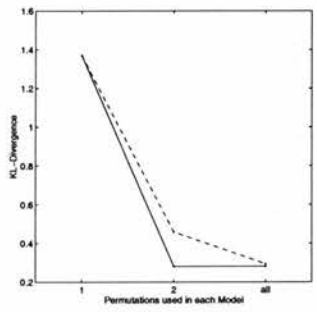
(a) AR(1)



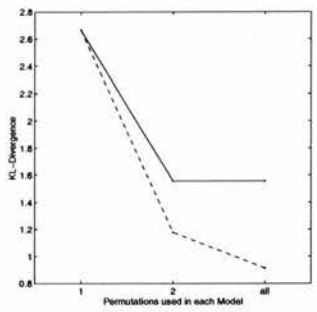
(b) MA(1)



(c) AR(2) case 1



(d) AR(2) case 2



(e) AR(2) case3

Figure 4.3: Fitting the Models to Target Processes. The graphs show the KL-divergence between the fitted models (PTSGP solid line, STSGP dashed line) and various different target processes. The three points on each line represent the KL-divergence for the cases where in each model there is: just a single TSGP; the single TSGP and a TSGP which has been shifted by one; and finally where there are all the possible permutations included in the sum/product. In (a) the target process is AR(1), (b) MA(1) with $\beta_0 = \beta_1 = 1$, (c) AR(2) with $\alpha = 0.1$, $\beta = 0.2$, (d) AR(2) with $\alpha = -0.25$, $\beta = 1$, (e) AR(2) with $\alpha = -0.5$, $\beta = 1$. In these experiments, the target and model processes have 64 discrete points corresponding to tree structures with 7 layers.

target MA(1) process. This works because we know the AR(1) process is defined by the distance-two interactions in the inverse covariance matrix and similarly the MA(1) process is defined by distance-two interactions in the covariance matrix. The above construction allows all the appropriate distance-two interactions to be generated and the processes are fitted exactly. The individual TSGPs required to achieve the fit have all their variance in the bottom two layers of the tree corresponding to the distance-two interactions. In Section 4.4 below we analyse the nature of the exact fit of the TSGPs to the AR(1) and MA(1) processes.

4.4 Characterisation of AR(1) and MA(1) Processes as Sums and Products of TSGPs

The results in section 4.3 show that the AR(1) process can be fitted exactly with a product of two TSGPs, and similarly that a MA(1) process can be fitted exactly as a sum of two TSGPs. Below we analyse this in more detail.

For both the AR(1) and MA(1) cases, Figure 4.4 shows that the variance is only non-zero at the leaves and for the layer above. This means that each "individual" tree is actually a forest of $m/2$ trees each of height 1 (see Figure 4.6). The covariance matrix for the leaves of a height-1 tree has the form

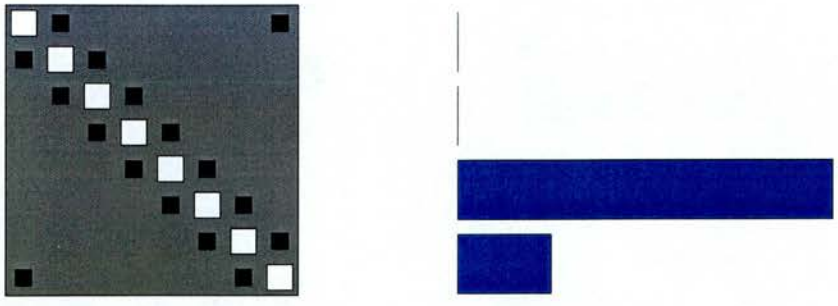
$$\begin{pmatrix} \sigma_1^2+\sigma_2^2 & \sigma_2^2 \\ \sigma_2^2 & \sigma_1^2+\sigma_2^2 \end{pmatrix} \tag{4.5}$$

using the notation of Equation 4.3 with $L=2$. Thus the covariance matrix of the sum of two such TSGPs (with the second displaced by one leaf node from the first) will have a tridiagonal circulant structure with diagonal elements of $2\sigma_1^2+2\sigma_2^2$ and off-diagonal elements of σ_2^2. The general MA(1) process has a tridiagonal circulant covariance structure with $\beta_0^2+\beta_1^2$ on the diagonal and $\beta_0\beta_1$ off. If $\beta_0\beta_1 > 0$ then we find the solution $\sigma_2^2=\beta_0\beta_1$ and $\sigma_1^2=(\beta_0-\beta_1)^2/2$. If $\beta_0\beta_1<0$ then we need a modification of the parameterization of the tree to allow a negative covariance. If $x_2^{(1)}$ and $x_2^{(2)}$ are the leaves of the height-1 tree, and $x_1^{(1)}$ is the root, then letting

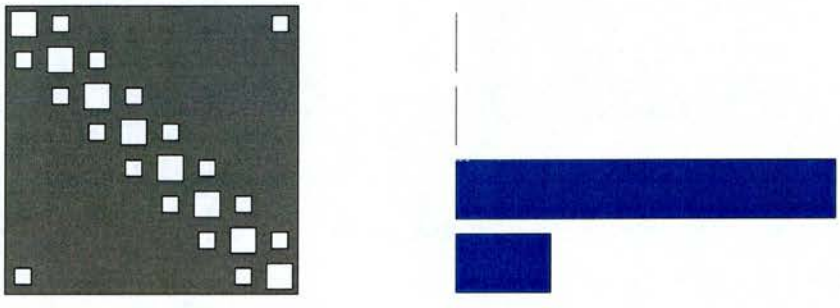
$$x_2^{(1)} = ax_1^{(1)}+b_1w_1^{(1)}, \tag{4.6}$$

$$x_2^{(2)} = -ax_1^{(1)}+b_1w_1^{(2)}, \tag{4.7}$$

(note the $-$ sign in the second equation) we find that their covariance matrix now has the same form as in Equation 4.5 but with the off-diagonal entry now being $-\sigma_2^2$. With

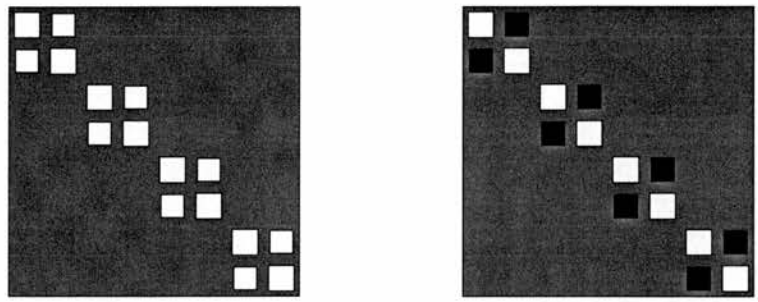


(a) Inverse PTSGP fitted to AR(1) process

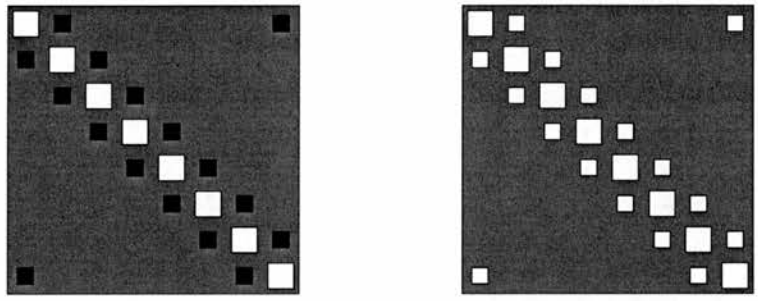


(b) STSGP fitted to MA(1) process

Figure 4.4: Results for the Target Processes and Models which Fit Exactly. Subfigure (a) shows how the PTSGP can be used to generate an AR(1) process and subfigure (b) shows how the STSGP can be used to generate an MA(1) process. Only two TSGPs are included in each model: the unpermuted TSGP and the TSGP which has been shifted by one. Within subfigure (a) the plot on the left shows a Hinton diagram of the structure of the inverse PTSGP covariance matrix (where the magnitude of an entry is represented by the area of the box, white represents positive values and black represents negative values), and the plot on the right shows the variance σ_i^2 added at each level of the single TSGP which is used in the model. Subfigure (b) shows the corresponding results for fitting the STSGP to the MA(1) process except the left plot shows the structure of the covariance rather than the inverse covariance matrix.

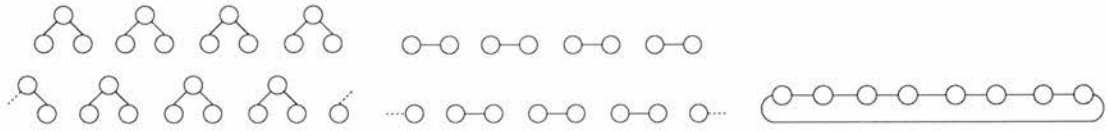


(a) TSGP (b) Inverse TSGP



(c) Inverse AR(1) (d) MA(1)

Figure 4.5: Hinton Diagrams Showing the Structure of the Covariance and Inverse Covariance Matrices. The figure shows the structure of the covariance and inverse covariance matrices for a single TSGP, the inverse AR(1) covariance and the MA(1) covariance. The TSGP is shown with variance due to only the last two layers of the tree. For representational convenience, processes with only 8 discrete points are shown.



(a) (b) (c)

Figure 4.6: A Schematic Diagram Indicating the Correlated Nodes. The subfigures show: (a) a forest of height 1 trees and a second forest of height 1 trees which have been permuted by one place; (b) a diagram showing nodes between which correlations have been induced by the two forests of trees in subfigure (a), the lines indicate correlated nodes; (c) a diagram showing the nodes between which correlations have been induced when taking the product or sum of the two forests of trees.

this modification we can obtain a match of the sum of two TSGPs to a general MA(1) process.

For the AR(1) process, we need to consider the inverse covariance matrices. The inverse of (4.5) has the form

$$\frac{1}{\Delta}\begin{pmatrix} \sigma_1^2+\sigma_2^2 & -\sigma_2^2 \\ -\sigma_2^2 & \sigma_1^2+\sigma_2^2 \end{pmatrix} \tag{4.8}$$

where $\Delta = \sigma_1^2(\sigma_1^2+2\sigma_2^2)$. Thus the overall inverse covariance of a product of two such TSGPs (with the second displaced by one leaf node from the first) will have tridiagonal circulant structure with diagonal elements of $2(\sigma_1^2+\sigma_2^2)/\Delta$ and off-diagonal elements of $-\sigma_2^2/\Delta$. For a general AR(1) process (with noise variance $Z_i \sim N(0,v)$ for all i), the diagonal entries in the inverse covariance are $(1+\alpha^2)/v$ and the non-zero off-diagonal entries are $-\alpha/v$. By equating these terms we obtain

$$\frac{1+\alpha^2}{v} = \frac{2}{\Delta}(\sigma_1^2+\sigma_2^2), \tag{4.9}$$

$$\frac{\alpha}{v} = \frac{\sigma_2^2}{\Delta}. \tag{4.10}$$

By dividing (4.9) by (4.10) we obtain $\sigma_1^2 = \sigma_2^2\frac{(1-\alpha)^2}{2\alpha}$. Substituting this back into (4.10) we obtain

$$\sigma_2^2 = \frac{4\alpha v}{(1-\alpha)^2(1+\alpha)^2}. \tag{4.11}$$

Thus for a given values of v and $\alpha > 0$ we can derive the parameters for the equivalent PTSGP. When $\alpha < 0$ the same modification is needed as for the MA(1) process with $\beta_0\beta_1 < 0$.

4.5 Discussion

Above we have fitted products and sums of TSGP models to a number of target processes by minimizing the KL divergence criterion. For the AR(1) process an exact representation of the target process was achieved using PTSGPs, and for the MA(1) process using STSGPs. In both cases we only needed two TSGPs in the model to achieve the exact match. This is illustrated schematically in Figure 4.6. In Figure 4.6a we can see two forests of height 1 trees based on eight nodes, Figure 4.6b indicates which leaf nodes are correlated by the respective forests, and Figure 4.6c shows that all the nodes can become correlated by combining the two models. In general if we have

models with inverse covariance matrices $\mathbf{C}_i^{-1}$, then the zeros in $\mathbf{C}_M^{-1} = \sum_i \mathbf{C}_i^{-1}$ will only occur when there are zeros in *all* of the component inverse covariance matrices.

Stationary processes on the circle have sinusoidal eigenfunctions so in general a stationary 1-d process on m nodes needs $m/2+1$ Fourier coefficients to describe it. (The sine and cosine components have the same variance, but there is no $\sin 0t$ component to match the $\cos 0t = 1$ basis function). With the current tied parameterization, a sum or product of TSGPs has only $\log_2 m + 1$ free parameters. Thus there are some target processes which cannot be exactly matched using either approach. To answer this one can add some extra flexibility to the models' parameter structure. Examples of extra flexibility include (i) increasing the number of parameters in a given tree (see, e.g. section 4.4), (ii) using vector rather than scalar valued processes in the individual TSGPs and (iii) removing the tying of parameters across TSGPs; if only a small number of permutations were used this would not lead to an explosion in the number of parameters to be fitted.

Chapter 5

Conclusions and Further Work

In this chapter, we will look at a summary of the conclusions that can be drawn from the work presented in the previous chapters and at an outline of directions for future work.

5.1 Conclusions

This thesis makes the following original contributions:

- Chapter 2 presents a comparison between generative and discriminative approaches for the fusion of the local image information with a fixed tree-structured belief network prior image model. The generative approach uses Gaussian mixture models to model the class conditional probability distributions for the pixel information. The discriminative approach uses a multi-layer perceptron to approximate the probability of the class-label given the pixel values. The discriminative approach is shown to achieve better performance. In the segmentation task, the use of the MLP combined with the conditional maximum likelihood trained TSBN gives a pixelwise classification accuracy rate on an independent test set of images of 90.68%. This compares to 75.85% for the Gaussian mixture model with the conditional maximum likelihood trained TSBN. It is also shown that the test images have lower image coding costs with the MLP compared to the GMM.

- Chapter 3 considers how to deal with sequences of images. As a first challenge, we look at the coding cost of a sequence of image labels. With this in mind,

the HVPP mapping algorithm is introduced. The aim is to use the fact that the images occur in a sequence to find the lowest possible coding cost. A mapping is generated between the HVPP structures of two consecutive images. Coding the mapping allows a tree structure to be constructed for the image being coded. The estimated coding cost for the test image sequence is 0.0937bpp for the four class images that are used in the experiment. This is 14% smaller than the estimated coding cost given the one step late tree which uses the most basic sequence information since for this method the images are coded given the HVPP tree structure for the previous frame in the sequence. It is shown that the HVPP mapped tree coding cost is useful in absolute terms as it is shown that it is 29% smaller than the cost of the best standard image coding format found.

In effect, this is a quantitative illustration that the HVPP structure is a good representation of the structure of an image and hence might be a useful parse tree for that image. Only the HVPP structure for the images is used to make the mapping between consecutive frames and yet the algorithm consistently reduces the image coding cost for the images in the sequence.

- Also in chapter 3 an extension to the position encoding dynamic tree model is proposed in order to incorporate sequential information in a principled manner. A filtering approach is adopted, the prior distribution for a time frame is updated given the variational posterior distribution for the previous time frame by allowing each node in the tree a probability of connecting to nodes in the previous frame rather than parents in the layer above. The experiments show improved image coding cost and pixelwise classification accuracy results.

- Chapter 4 presents an investigation the products of experts architecture in which each of the experts is a tree-structured Gaussian process. We show that the AR(1) and MA(1) processes can be represented by products and sums of the tree-structured Gaussian processes respectively.

Overall the thesis has looked at issues surrounding tree-structured graphical models in particular with regard to their use as models for images and image sequences. There is still obviously plenty of scope for further work.

5.2 Further Work

As far as Chapter 2 is concerned, the two fundamental criticisms of the use of fixed tree-structured belief networks as models for images are: firstly, that the structure cannot be regarded as a parse tree for the image as it remains fixed for all images; and secondly that practically this results in blocky artifacts in the segmentations inferred under this model. To a large extent, these shortcomings have been addressed by the work done by others on dynamic trees, position encoding dynamic trees and credibility networks as discussed earlier.

With regard to the use of the position encoding dynamic tree as an image sequence model, there are a number of avenues for further research. For instance, a primary focus for future work could be the introduction of a motion model into the PEDT framework. In general, the work presented here makes the assumption that incorporating an unchanged posterior distribution from the previous frame improves the model for the current image. This is a reasonable first approximation however, it is a little naïve. The assumption works best if there are no changes between consecutive images, in which case there is no need for a sequence model. The assumption works increasingly less well if there are considerable changes between consecutive images which is exactly when sequential information could potentially be most useful. One approach to this problem would be to add a velocity variable to each of the nodes in graph to complement the position variables already in the model. This could allow inference and learning in the model to involve motion and image dynamics in a more informative way. Alternatively, if one had access to blackbox estimates for the hierarchical motion of different regions in the image between frames (such as for instance optical flow estimates) these could be used to vary the probability with which a node might want to connect to particular nodes in the previous frame.

From a practical point of view, a considerable problem with the use of the PEDT model with image sequences is the availability of labelled image data. A database of labelled images such as the Sowerby database of roadside scenes is difficult and expensive to compile. However this problem is magnified when considering image sequences. This is why sequences of ray-traced images were used for the experiments in Chapter 3. In this regard, it might be fruitful to investigate the robustness of learning in the TSBN and PEDT in the face of an increasing percentage of unlabelled pixels in the data. If methods for handling sparsely labelled data could be introduced, it would

greatly improve the practicality of the TSBN and PEDT as image models.

The HVPP tree mapping algorithm presented is not an optimal method for coding a tree for the next frame in a sequence. Indeed, there is no apparent and computationally acceptable method of finding the mapping which minimises the coding cost of an image given the HVPP tree structure of the previous image in the sequence. As a software engineering problem, there is plenty of scope for improving the mapping generated. A first improvement to be made would be to deal with each of the object-type image classes individually both when making the node mapping and when calculating the regions in the image to which smoothing needs to be applied. In the experiments presented in Chapter 3, the implementation of mapping algorithm did not distinguish between objects nodes of different classes with the result that tree structures representing different object-type classes in the image could get confused resulting in higher image coding costs. Resolving this problem would inevitably lead to improved coding cost performance.

Finally with respect to Chapter 4, when considering a products of experts architecture in which each of the individual experts is a Gaussian, there is no need to restrict the experts to be tree-structured Gaussian processes. Less constrained Gaussian distributions can also be considered. Further work on this topic can be found in [82, 81].

Appendix A

Tables of Results for the Sequential PEDT Image Segmentation Experiments

(a) Pixelwise Classification Accuracy

Variance of added Gaussian noise	local	PEDT										
		$\Gamma = 0$	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
$\sigma^2 = 10^{-1}$	76.66	85.93	86.12	86.13	86.18	86.16	86.16	86.21	86.26	86.27	86.27	85.7
$\sigma^2 = 10^{-2}$	79.86	93.08	93.13	93.12	93.14	93.15	93.14	93.15	93.15	93.16	93.16	93.13
$\sigma^2 = 10^{-3}$	89.21	94.56	94.6	94.61	94.6	94.59	94.62	94.62	94.6	94.59	94.57	94.6

(b) Difference in the Average Variational Log Likelihood of the Test Images between a Static PEDT and a PEDT with Γ Updates

Variance of added Gaussian noise	Γ									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
$\sigma^2 = 10^{-1}$	6.36	10.6	16.8	22.8	28.6	34.7	40.1	44.7	47.8	49.1
$\sigma^2 = 10^{-2}$	16	19.6	25.4	29.7	34.9	40.1	45.1	49.6	52.5	34.4
$\sigma^2 = 10^{-3}$	16.2	24.1	29	33.9	39.4	44.5	50.2	54.3	57.5	38.9

(c) Percentage of Images with an Improved Variational Log Likelihood over the Static PEDT

Variance of added Gaussian noise	Γ									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
$\sigma^2 = 10^{-1}$	93.88	97.96	97.96	100	100	100	100	100	100	95.92
$\sigma^2 = 10^{-2}$	97.96	95.92	100	100	100	100	100	100	100	77.55
$\sigma^2 = 10^{-3}$	87.76	97.96	97.96	97.96	97.96	97.96	97.96	95.92	100	77.55

Table A.1: Tables of Results for the Sequential PEDT Image Segmentation Experiments with Fixed Update Depth and Width Parameters and Varying Image Noise Levels. (Update depth = 5, width parameter=1.)

(a) Pixelwise Classification Accuracy

Update Depth to:		PEDT										
	local	$\Gamma = 0$	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
layer 8 (leaf)	76.66	85.93	86.02	86.03	85.83	85.58	85.26	84.88	84.29	83.52	82.85	84.82
layer 7	76.66	85.93	86	85.98	85.9	85.6	85.5	85.32	85.02	84.48	83.59	84.47
layer 6	76.66	85.93	86.08	86.11	86.14	86.11	86.03	85.99	85.81	85.73	85.49	84.68
layer 5	76.66	85.93	86.12	86.13	86.18	86.16	86.16	86.21	86.26	86.27	86.27	85.7
layer 4	76.66	85.93	85.88	85.89	85.9	85.9	85.9	85.91	85.88	85.85	85.79	85.82
layer 3	76.66	85.93	85.91	85.91	85.89	85.89	85.9	85.89	85.89	85.9	85.9	85.93

(b) Difference in the Average Variational Log Likelihood of the Test Images between a Static PEDT and a PEDT with Γ Updates

Update Depth to:	PEDT									
	$\Gamma = 0.10$	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
layer 8 (leaf)	-28.1	-50.7	-67.6	-83.1	-98.6	-114	-127	-135	-133	32
layer 7	-5.64	-8.41	-10.9	-9.47	-7.31	-3.13	0.188	3.65	7.69	39.4
layer 6	1.58	2.7	4.97	6.41	9.39	11.7	15	17.7	22.1	33.2
layer 5	6.36	10.6	16.8	22.8	28.6	34.7	40.1	44.7	47.8	49.1
layer 4	1.78	3.35	5.07	7.59	9.82	12.1	14.3	16.2	17.8	16.9
layer 3	0.338	0.532	1.58	2.42	3.02	4.02	5.25	5.9	6.48	7.77

(c) Percentage of Images with an Improved Variational Log Likelihood over the Static PEDT

Update Depth to:	Γ									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
layer 8 (leaf)	2.041	0	0	0	0	0	0	2.041	4.082	87.76
layer 7	28.57	32.65	40.82	55.1	57.14	59.18	59.18	63.27	67.35	85.71
layer 6	57.14	61.22	65.31	63.27	67.35	65.31	69.39	71.43	73.47	77.55
layer 5	93.88	97.96	97.96	100	100	100	100	100	100	95.92
layer 4	77.55	85.71	89.8	97.96	97.96	97.96	97.96	97.96	100	95.92
layer 3	71.43	65.31	81.63	87.76	87.76	95.92	97.96	97.96	97.96	100

Table A.2: Tables of Results for the Sequential PEDT Image Segmentation Experiments with a Fixed Width Parameter, Fixed Levels of Image Noise and a Varying Update Width Parameter. (Noise variance added = 0.1, width parameter =1.)

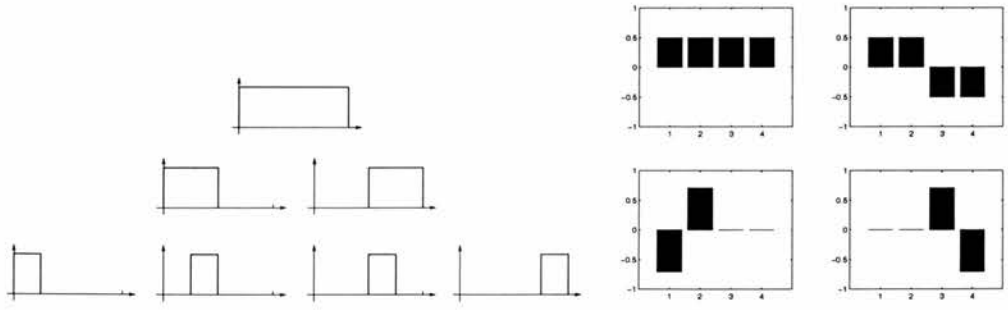
Appendix B

Tree-Structured Gaussian Process Basis Function Representation

An equivalent way of constructing this process is to consider a set of basis functions as shown in Figure B.1(a). These are scaled and translated "box-car" functions. In this representation we write the 1D process $x_L(t)$ as

$$x_L(t) = \sum_{k=1}^{2m-1} \beta_k \psi_k(t) \tag{B.1}$$

where ψ_i are the basis functions illustrated and β_i are zero-mean gaussian random variables with $\beta_k \sim N(0, \sigma^2_{l(k)})$, where $l(k)$ is the function that returns the layer to which basis function k corresponds. Here we regard the discrete process as a subsampling of the continuous process at regular points along the line. We are not aware of previous work taking this view of the TSGP.



(a) Tree basis (b) Eigenfunctions

Figure B.1: The Tree Basis Functions and the Eigenfunctions of a TSGP with 4 Leaf Nodes.

Note that the basis functions are not orthogonal. However, if we define the covariance function $K(t,t') = \sum_k \sigma^2_{l(k)} \psi_k(t)\psi_k(t')$, we can calculate the eigenfunctions of this kernel $\phi_i(t)$ as linear combinations of the basis functions, i.e. $\phi_i(t) = \sum_j \gamma^i_j \psi_j(t)$, giving rise to a matrix eigenproblem to determine the γs. For the 4-node example of Figure B.1(a) the four eigenfunctions with non-zero eigenvalues are plotted in Figure B.1(b).[1]

The multiscale nature of the basis function decomposition is reminiscent of wavelets; see e.g. [86] for further discussion and analysis of this kind of construction.

[1] The parameters for this tree were $\sigma^2_1 = 1.0, \sigma^2_2 = 0.5, \sigma^2_3 = 0.25$.

Bibliography

[1] N.J. Adams. *Dynamic Trees: A Hierarchical Probabilistic Approach to Image Modelling*. PhD thesis, Division of Informatics, University of Edinburgh, 2002.

[2] N.J. Adams, A.J. Storkey, Z. Ghahramani, and C.K.I. Williams. MFDTs: Mean field dynamic trees. In *Proceedings of the International Conference on Pattern Recognition*, 2000.

[3] N.J. Adams and C.K.I. Williams. DTs: Dynamic trees. In M.J. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999.

[4] D. H. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, 1982.

[5] S.T. Barnard. A stochastic approach to stereo vision. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 676–680, CA, 1986. Morgan Kaufmann.

[6] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky. Modelling and estimation of multiresolution stochastic processes. *IEEE Transactions on Information Theory*, 38:766–784, 1992.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[8] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, 1998.

[9] C.A. Bouman and M. Shapiro. A multiscale random field model for Bayesian image segmentation. *IEEE Transactions in Image Processing*, 3(2):162–177, 1994.

[10] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, London, 4th edition, 1989.

[11] R. Chellappa and S. Chatterie. Classification of textures using Gaussian Markov random fields. *IEEE Transactions in Accoustics, Speech and Signal Processing*, 33:959–963, 1985.

[12] R. Chellappa and A. Jain. *Markov Random Fields - Theory and Applications*. Academic Press, London, 1993.

[13] H. Cheng and C. Bouman. Trainable context model for multiscale segmentation. *Proceedings of International Conference on Image Processing*, 1:610–614, 1998.

[14] K.C. Chou, A.S. Willsky, and A. Benveniste. Multiscale Recursive Estimation Data Fusion, and Regularization. *IEEE Transactions on Automatic Control*, 39(3):464–477, March 1994.

[15] T.M Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley and Sons, New York, 1991.

[16] M. Crouse, R. Nowak, and R. Baraniuk. Wavelet-based statistical signal processing using Hidden Markov Models. *IEEE Transactions on Signal Processing*, 46:886–902, 1998.

[17] A.P. Dawid. Properties of diagnostic data distributions. *Biometrics*, 32:647–658, 1976.

[18] J.S. de Bonet and P.A. Viola. A non-parametric multiscale statistical model for natural images. In M.I. Jordan, M.J. Kearns, and S.A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 773–779. MIT Press, 1998.

[19] P.J. Diggle. *Time Series: A Biostatistical Introduction*. Oxford University Press, Oxford, 1990.

[20] J.O. Eklundh, H. Yamamoto, and A. Rosenfeld. A relaxation method for multispectral pixel classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:72–75, 1980.

[21] S.N. Felderhof, A.J. Storkey, and C.K.I. Williams. Position Encoding Dynamic Trees for image sequence analysis. Technical report, Division of Informatics, University of Edinburgh, 2002.
See `http://www.dai.ed.ac.uk/~stephenf/pedt/`.

[22] X. Feng, C.K.I. Williams, and S.N. Felderhof. Combining neural networks and belief networks for the segmentation of outdoor scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):467–483, 2002.

[23] P.W. Fieguth, W.C. Karl, and A.S. Willsky. Efficient multiresolution counterparts to variational methods for surface reconstruction. *Computer Vision and Image Understanding*, 70(2):157–176, 1998.

[24] P.W. Fieguth, A.S. Willsky, and W.C. Karl. Multiresolution stochastic imaging of satellite oceanographic altimetric data. *IEEE International Conference on Image Processing*, 2:1–5, 1994.

[25] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[26] Z. Ghahramani and M.I. Jordan. Factorial Hidden Markov Models. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.

[27] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, 1996.

[28] P.S. Gopalakrishan, D. Kanevsky, A. Nadas, and D. Nahamoo. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory*, 37(1):107–113, 1991.

[29] T.J. Hainsworth and K.V. Mardia. A Markov Random Field restoration of image sequences. In R. Chellappa and A. Jain, editors, *Markov Random Fields - Theory and Applications*. Academic Press, London, 1993.

[30] D.J. Heeger and J.R. Bergen. Pyramid-based texture analysis/synthesis. In *Computer Graphics Proceedings*, pages 229–238, 1995.

[31] G. E. Hinton. Products of Experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999.

[32] G.E. Hinton, P.S. Dayan, A. To, and R.M. Neal. The Helmholtz machine through time. In F. Fogelman-Soulie and P. Gallinari, editors, *Proceedings of the International Conference on Artificial Neural Networks*. EC2 et Cie, 1995.

[33] G.E. Hinton, Z. Ghahramani, and Y.W. Teh. Learning to parse images. In S.A. Solla, T.K. Leen, and K-R Muller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 463–469. MIT Press, 2000.

[34] B.K.P. Horn. Obtaining shape from shading information. In P.H. Winston, editor, *The Pyschology of Computer Vision*, pages 115–155. McGraw-Hill, New York, 1975.

[35] B.K.P. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[36] W.W. Irving, P.W. Fieguth, and A.S. Willsky. An overlapping tree approach to multiscale stochastic modeling and estimation. *IEEE Transactions on Image Processing*, 6(11):1517–1529, 1997.

[37] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings European Conference on Computer Vision*, pages 343–356, 1996.

[38] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[39] A.C. Kokaram and S.J. Godsill. A system for reconstruction of missing data in image sequences using sampled 3D AR models and MRF motion priors. In *Springer Lecture Notes in Computer Science: Computer Vision - ECCV '96*, volume II, pages 613–624. Springer-Verlag, April 1996.

[40] S. Krishnamachari and R. Chellappa. Multiresolution Gaussian Markov random field models for texture segmentation. *IEEE Transactions on Image Processing*, 6(2):251–267, 1997.

[41] S. Kullback. *Information Theory and Statistics*. Cover, New York, 1959.

[42] J-M. Laferté, F. Heitz, P. Pérez, and E. Fabre. Hierarchical statistical models for the fusion of multiresolution data. *Proceedings Fifth International Conference on Computer Vision*, 1995.

[43] J-M. Laferté, P. Pérez, and F. Heitz. Discrete Markov Image Modelling and Inference on the Quadtree. *IEEE Transactions on Image Processing*, 9(3):390–404, 2000.

[44] Y. Le Cun, B. Boser, J.S. Denker, S. Solla, R.E. Howard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[45] M. D. Levine and A. M. Nazif. Dynamic measurement of computer generated image segmentations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(2):155–164, 1985.

[46] T. M. Lillesand and R. W. Kiefer. *Remote Sensing and Image Interpretation*. Wiley, New York, third edition, 1994.

[47] H. Lucke. Bayesian belief networks as a tool for stochastic parsing. *Speech Comm.*, 16:89–118, 1995.

[48] M.R. Luettgen, W.C. Karl, and A.S. Willsky. Multiscale Representations of Markov Random Fields. *IEEE Transactions on Signal Processing*, 41(12):3377–3395, 1993.

[49] M.R. Luettgen, W.C. Karl, and A.S. Willsky. Efficient multiscale regularisation with applications to the computation of optical flow. *IEEE Transactions on Image Processing*, 3(1):41–64, 1994.

[50] M.R. Luettgen and A.S. Willsky. Likelihood calculation for a class of multiscale stochastic models, with application to texture discrimination. *IEEE Transactions on Image Processing*, 4(2):194–207, 1995.

[51] S.P. Luttrell. Partitioned mixture distributions: The dynamical case. In *Proceedings of the 5th IEE Conference on Artificial Neural Networks*, 1997.

[52] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman and Co., San Francisco, 1980.

[53] J.D. McCauley and B.A. Engel. Comparison of scene segmentations: SMAP, ECHO, and maximum likelihood. *IEEE Transactions on Geoscience and Remote Sensing*, 33(6):1313–1316, 1995.

[54] N. Morgan and H.A. Bourlard. Neural networks for statistical recognition of contiguous speech. *Proceedings of the IEEE*, 83(5):742–770, 1995.

[55] D.W. Murray and B.F. Buxton. Scene segmentation from visual motion using global optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):220–228, 1987.

[56] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, New York, 1996. Lecture Notes in Statistics 118.

[57] K. M. Nickels and S. Hutchinson. Textured image segmentation: returning multiple solutions. *Image and Vision Computing*, 15:781–795, 1997.

[58] R. Ohlander, K. Price, and D.R. Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[59] Yuichi Ohta. *Knowledge-based Interpretation of Outdoor Natural Colour Scenes*. Pitman Publishing Limited, London, 1985.

[60] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Singapore, 1991.

[61] G. Parisi. *Statistical Field Theory*. Addison-Wesley, Reading, MA, 1988.

[62] J. Pearl. *Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[63] P. Pérez, A. Chardin, and J-M. Laferté. Noniterative manipulation of discrete energy-based models for image analysis. *Pattern Recognition*, 33(4):573–586, 2000.

[64] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[65] M.D. Richard and R.P. Lippmann. Neural network classifiers estimate Bayesian a-posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[66] O. Ronen, J.R. Rohlicek, and M. Ostendorf. Parameter estimation of dependence tree models using the EM algorithm. *IEEE Signal Processing Letters*, 2(8):157–159, 1995.

[67] A. Rosenfeld. Image analysis: Problems, progress and prospects. *Pattern Recognition*, 17(1):3–12, 1984.

[68] L.K. Saul and M.I. Jordan. Exploiting tractable substructures in intractable networks. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*. MIT Press, Cambridge, MA, 1996.

[69] P. Smyth. Hidden markov models for fault detection in dynamical systems. *Pattern Recognition*, 27(1):149–164, 1994.

[70] A. Storkey. Dynamic trees: A structured variational method giving efficient propagation rules. In C. Boutilier and M. Goldszmidt, editors, *Uncertainty in Artifical Intelligence*, pages 566–573, 2000.

[71] A. J. Storkey and C.K.I. Williams. Dynamic positional trees for structural image analysis. In T. Jaakkola and T. Richardson, editors, *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 298–304. Morgan Kaufmann, 2001.

[72] A.J. Storkey. Dynamic trees: A structured variational approach giving efficient propagation rules. In *Uncertainty in Artificial Intelligence*, 2000.

[73] A.J. Storkey and C.K.I. Williams. Image modelling with position-encoding dynamic trees. *To appear in: IEEE Transactions in Pattern Analysis and Machine Intelligence*, 2002.

[74] W.B. Thompson. Combining motion and contrast for segmentation. *IEEE Trans. PAMI*, 2(6):549–543, 1980.

[75] Y. Weiss and E.H. Adelson. A unified mixture framework for motion segmentation: incorporating spatial coherence and estimating the number of models. *CVPR*, 1996.

[76] R. M. Welch, M. S. Navar, and S. K. Sengupta. The effect of spatial resolution upon texture-based cloud field classifications. *Journal of Geophysical Research*, 94:14767–14781, 1989.

[77] M. West and J. Harrison. *Forecasting and Dynamic Models*. Springer, 2nd edition, 1997.

[78] J. S. Weszka, C. R. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(4):269–285, 1976.

[79] C.K.I. Williams and N.J. Adams. DTs: Dynamic trees. In M.J. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, Cambridge, MA, 1999. MIT Press.

[80] C.K.I. Williams and F. Agakov. Products of Gaussians and probabilistic minor component analysis. Technical report, Division of Informatics, Edinburgh University, 2001.

[81] C.K.I. Williams and F. Agakov. Products of Gaussians and probabilistic minor component analysis. *Neural Computation*, 14(5):1169–1182, 2001.

[82] C.K.I. Williams, F. Agakov, and S.N. Felderhof. Products of Gaussians. In S. Becker, Z. Ghahramani, and T. Dietrich, editors, *Advances in Neural Information Processing Systems*, volume 13, Cambridge, MA, 2001. MIT Press.

[83] C.K.I. Williams and S.N. Felderhof. Products and sums of tree-structured gaussian processes. In *Proceedings of the 4th ICSC Symposium on Soft Computing and Intelligent Systems for Industry*, volume 4, The Netherlands, 2001. ICSC-NAISO Academic Press.

[84] C.K.I. Williams and G.E. Hinton. Mean field networks that learn to discriminate temporally distorted strings. In D.S. Touretzky, J.L. Elman, and T.J. Sejnowski, editors, *Connectionist Models: Proceedings of the 1990 Summer School*, San Mateo, CA, 1991. Morgan Kaufmann.

[85] A.P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.

[86] G. W. Wornell. A Karhunen-Loève-like Expansion for $1/f$ Processes via Wavelets. *IEEE Transactions on Information Theory*, 36(4):859–861, 1990.

[87] S.C. Zhu, X.W. Liu, and Y.N Wu. Exploring texture ensembles by efficient Markov chain Monte Carlo towards a "trichromacy" theory of texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 1999.