Accepted for publication in Neural Computation

Greedy Learning of Multiple Objects in Images using Robust Statistics and Factorial Learning

Christopher K. I. Williams* and Michalis K. Titsias

School of Informatics, University of Edinburgh, Edinburgh EH1 2QL, UK

c.k.i.williams@ed.ac.uk M.Titsias@sms.ed.ac.uk

10 October 2003

Abstract

We consider data which are images containing views of multiple objects. Our task is to learn about each of the objects present in the images. This task can be approached as a factorial learning problem, where each image must be explained by instantiating a model for each of the objects present with the correct instantiation parameters. A major problem with learning a factorial model is that as the number of objects increases, there is a combinatorial explosion of the number of configurations that need to be considered. We develop a method to extract object models

*http://anc.ed.ac.uk

sequentially from the data by making use of a robust statistical method, thus avoiding the combinatorial explosion, and present results showing successful extraction of objects from real images.

1 Introduction

In this paper we consider data which are images containing views of multiple objects. Our task is to learn about each of the objects present in the images. Previous approaches (discussed in more detail below) have approached this as a factorial learning problem, where each image must be explained by instantiating a model for each of the objects present with the correct instantiation parameters. By factorial learning we mean a situation where multiple causes (factors) are needed to explain the data (image)[1]. A serious concern with the factorial learning problem is that as the number of objects increases, there is a combinatorial explosion of the number of configurations that need to be considered. Suppose there are L possible objects, and that there are J possible values that the instantiation parameters of any one object can take on; we will need to consider $O(J^L)$ combinations to explain any image. In contrast, in our approach we find one object at a time, thus avoiding the combinatorial explosion.

In unsupervised learning we aim to identify regularities in data such as images. One fairly simple unsupervised learning model is clustering, which can be viewed as a mixture model where there are a finite number of types

[1]This is the same terminology as used in the factor analysis model from statistics, although that model uses linear/Gaussian assumptions.

of object, and data is produced by choosing *one* of these objects and then generating the data conditional on this choice. As a means of discovering objects in images standard clustering approaches are limited as they do not take into account the variability that can arise due to translations, rotations *etc* (the instantiation parameters) of the object. Suppose that there are m different instantiation parameters, then a single object will sweep out a m-dimensional manifold in the image space. Learning about objects taking this regularity into account has been called transformation-invariant clustering by Frey and Jojic (1999, 2003). However, this work is still limited to finding a single object in each image.

A more general model for data is that where the observations are explained by multiple causes; in our example this will be that in each image there are L objects. The approach of Frey and Jojic (1999, 2003) can be extended to this case by explicitly considering the simultaneous instantiation of all L objects (Jojic and Frey, 2001). However, this gives rise to a large search problem over the instantiation parameters of all objects simultaneously, and approximations such as variational methods are needed to carry out the inference. In our method, by contrast, we discover the objects one at a time using a robust statistical method. Sequential object discovery is possible because multiple objects combine by occluding each other and/or the background.

The general problem of factorial learning has longer history, see, for example, Barlow (1989), Hinton and Zemel (1994), and Ghahramani (1995). However, Frey and Jojic made the important step for image analysis problems of using explicit transformations of object models, which allows the incor-

poration of prior knowledge about these transformations and leads to good interpretability of the results. A related line of research is that concerned with discovering part decompositions of objects. Lee and Seung (1999) described a non-negative matrix factorization method addressing this problem, although their work does not deal with parts undergoing transformations. Other relevant work including that by Shams and von der Malsburg (1999) on learning parts and work from the computer vision community on layered representations of images are discussed in section 4.

The structure of the remainder of this paper is as follows. In section 2 we describe the model, first for images containing only a single object and then for images containing multiple objects. In section 3 we present experimental results finding objects appearing against static, moving and random backgrounds. We conclude with a discussion in section 4[2].

2 Theory

In section 2.1 we describe how to learn about an object when there is only a single object (plus background) in every image. In section 2.2 we discuss how this model can be robustified to deal with foreground and background occlusion caused by other objects being present in the images. Then in section 2.3 we describe a model that fully explains L objects in the images. An efficient greedy algorithm for training this model is described in section 2.4.

[2]This paper is a revised and extended version of Williams and Titsias (2003) which was presented at NIPS 2002.

2.1 Learning one object

In this section we consider the problem of learning about one object which can appear at various locations in an image. The object is in the foreground, with a background behind it. The problem is set up in terms of a generative model for the image $\mathbf{x}$ given the transformations of the foreground and background. The background can be one of three cases: (i) a static background that is fixed for all training images, (ii) a moving background which occurs for example when a moving camera captures a sequence of frames and (iii) random backgrounds where each image can have a completely different background.

The two key issues that we must deal with are the notion of a pixel being modelled as foreground or background, and the problem of transformations of the object and the background. We consider first the foreground/background issue and we assume that the background is static; cases (ii) and (iii) are discussed later in this section.

Consider an image $\mathbf{x}$ of size $P_x \times P_y$ containing $P \stackrel{def}{=} P_x P_y$ pixels, arranged as a length P vector. Our aim is to learn appearance-based representations of the foreground $\mathbf{f}$ and the static background $\mathbf{b}$. As the object will be smaller than $P_x \times P_y$ pixels, we will need to specify which pixels belong to the background and which to the foreground; this is achieved by a vector of binary latent variables $\mathbf{s}$, one for each pixel. Each binary variable in $\mathbf{s}$ is drawn independently from the corresponding entry in a vector of probabilities $\boldsymbol{\pi}$. For pixel p, if $\pi_p \simeq 0$, then the pixel will be ascribed to the background with high probability, and if $\pi_p \simeq 1$, it will be ascribed to the foreground

with high probability. We sometimes refer to $\boldsymbol{\pi}$ as a mask.

x_p is modelled by a mixture distribution:

$$x_p \sim \begin{cases} p_f(x_p; f_p) = N(x_p; f_p, \sigma_f^2) & \text{if } s_p = 1, \\ p_b(x_p; b_p) = N(x_p; b_p, \sigma_b^2) & \text{if } s_p = 0, \end{cases} \tag{1}$$

where σ_f^2 and σ_b^2 are respectively the foreground and background variances. Thus, ignoring transformations, we obtain

$$p(\mathbf{x}) = \prod_{p=1}^{P} [\pi_p p_f(x_p; f_p) + (1 - \pi_p) p_b(x_p; b_p)]. \tag{2}$$

Notice that the fact that each pixel follows a mixture distribution ensures that the foreground and background appearances strictly combine by occlusion and thus no transparency between them is allowed.

The second issue that we must deal with is that of transformations of the foreground object. Below we consider only translations, although the ideas can be extended to deal with other transformations such as scaling and rotation (see e.g. Frey and Jojic (2002)). Each possible transformation (e.g. translations in units of one pixel) is represented by a corresponding transformation matrix, so that matrix T_{j_f} corresponds to transformation j_f and $T_{j_f}\mathbf{f}$ is the transformed foreground model. In our implementation the translations use wrap-around, so that each T_{j_f} is in fact a permutation matrix. The semantics of foreground and background mean that the mask

$\boldsymbol{\pi}$ must also be transformed, so that we obtain

$$p(\mathbf{x}|j_f) = \prod_{p=1}^{P}[(T_{j_f}\boldsymbol{\pi})_p p_f(x_p; (T_{j_f}\mathbf{f})_p) + (\mathbf{1} - T_{j_f}\boldsymbol{\pi})_p p_b(x_p; b_p)]. \qquad (3)$$

where $\mathbf{1}$ denotes the P length vector that contains ones. Notice that the foreground $\mathbf{f}$ and mask $\boldsymbol{\pi}$ are transformed by T_{j_f}, but the static background $\mathbf{b}$ is not. In order for equation 3 to make sense, each element of $T_{j_f}\boldsymbol{\pi}$ must be a valid probability (lying in $[0, 1]$). This is certainly true for the case when T_{j_f} is a permutation matrix (and can be true more generally). To complete the model we place a prior probability P_{j_f} on each transformation j_f; this is taken to be uniform over all possibilities so that $p(\mathbf{x}) = \sum_{j_f=1}^{J_f} P_{j_f} p(\mathbf{x}|j_f)$.

So far the background $\mathbf{b}$ was considered to be static. However in many cases, as for example when a video camera follows an object, the background can change from frame to frame. Next we generalize our method to deal with moving backgrounds.

To model a moving background we assume an underlying static background which is typically much larger than the input images. We sometimes refer to this large static background as panorama. When we generate an image a part of this panorama scene is selected and used as the current background of the image, similarly to Rowe and Blake (1995). More specifically, we assume that the background $\mathbf{b}$ corresponds to an $M_x \times M_y$ image, where in general $M_x \geq P_x$ and $M_y \geq P_y$. $\mathbf{b}$ is represented as a M-dimensional vector with $M = M_x M_y$. We introduce a transformation variable j_b that explains how from the panorama $\mathbf{b}$ the background of a data image is selected. In our implementation we consider as possible backgrounds all the

$P_x \times P_y$ image blocks (aligned to the axes of the background image) taken from any possible location within the panorama $\mathbf{b}$[3]. Clearly, j_b takes on $J_b = (M_x - P_x + 1)(M_y - P_y + 1)$ total values and a certain value j_b is represented by a $M \times P$ transformation matrix T_{j_b}, so that $T_{j_b}\mathbf{b}$ selects the appropriate image $P_x \times P_y$ block from $\mathbf{b}$.

The conditional density of an image given the transformation variables now becomes:

$$p(\mathbf{x}|j_f, j_b) = \prod_{p=1}^{P} [(T_{j_f}\boldsymbol{\pi})_p p_f(x_p; (T_{j_f}\mathbf{f})_p) + (\mathbf{1} - T_{j_f}\boldsymbol{\pi})_p p_b(x_p; (T_{j_b}\mathbf{b})_p)]. \quad (4)$$

and the likelihood of an image $\mathbf{x}$ is $p(\mathbf{x}) = \sum_{j_f=1}^{J_f} \sum_{j_b=1}^{J_b} P_{j_f} P_{j_b} p(\mathbf{x}|j_f, j_b)$. Note also that when the background is static is expressed as a special case of the above model; by choosing the background $\mathbf{b}$ to have the same size as the data images there is only one possible value for j_b, so the background is static.

For random backgrounds we do not try to model the backgrounds explicitly, but simply use a large-variance Gaussian at each pixel, which can account for the large background variability. $\mathbf{b}$ is the mean of this Gaussian.

Given a data set $\{\mathbf{x}^n\}$, $n = 1, \ldots, N$ we can adapt the parameters $\theta = (\mathbf{f}, \boldsymbol{\pi}, \mathbf{b}, \sigma_f^2, \sigma_b^2)$ by maximizing the log likelihood $L(\theta) = \sum_{n=1}^{N} \log p(\mathbf{x}^n|\theta)$. This can be achieved through using the EM algorithm to handle the missing data which is the transformations j_f and j_b. However, an exact EM algorithm requires a search over $J_f J_b$ possibilities which can be very demanding even

[3]Of course the above model does not account for rotations or scaling of the background and it can only approximately model such kind of situations.

for small images. Our greedy training algorithm deals separately with each transformation by learning first the background and then the foreground object. This algorithm is presented for the more general case of L foreground objects in section 2.4 and also in the Appendix A.

2.2 Learning one object using robust statistics

Suppose that apart from the one foreground object being modelled the images can additionally contain some other objects. However, we consider these objects as "outlying" information and thus we do not wish to model their appearances. Our objective is to learn only the one object of interest and efficiently "ignore" all the other objects.

A way to learn one object under the above assumptions is to robustify the model described in section 2.1 so that foreground and background occlusion can be tolerated. More specifically, for a foreground pixel, some other objects may be interposed between the camera and our object, thus perturbing the pixel value. This can be modelled with a mixture distribution as

$$p_f(x_p; f_p) = \alpha_f N(x_p; f_p, \sigma_f^2) + (1 - \alpha_f)U(x_p), \tag{5}$$

where α_f is the fraction of times a foreground pixel is not occluded and the robustifying component $U(x_p)$ is a uniform distribution common for all image pixels. When a object pixel is occluded this should be explained by the uniform component. Such robust models have been used for image matching tasks by a number of authors, notably Black and colleagues (Black and Jepson, 1996).

Similarly for the background, a different object from the one being modelled may be interposed between the background and the camera, so that we again have a mixture model

$$p_b(x_p; b_p) = \alpha_b N(x_p; b_p, \sigma_b^2) + (1 - \alpha_b)U(x_p), \tag{6}$$

with similar semantics for the parameter α_b. Note that for random backgrounds the above robustification make less sense (since the Gaussian will have large variance σ_b^2) but it will apply to the static or moving background cases.

It is not necessary that the robustifying component be a uniform distribution, for example a broad Gaussian would also work. However, as pixels do have maximum and minimum values the uniform distribution is a natural choice, and is also the maximum entropy distribution.

Training this model is completely analogous to the non-robust case. In practice the above model can be used to learn multiple objects in images. By random parameter initializations, and on different runs we can find different objects. We denote such an algorithm as RANDOM STARTS.

2.3 Learning multiple objects

One way to learn multiple objects in images is by applying the RANDOM STARTS algorithm described in the above section. However, we have found (Williams and Titsias, 2003) that this is rather inefficient as the basins of attraction for the different objects may be very different in size given the initialization. Thus, in this section we describe a model that explicitly as-

sumes L foreground objects in the images and in section 2.4 we present the GREEDY algorithm that learns the background and the objects sequentially.

Assume that each image contains L foreground objects. Similarly to the single object case each object ℓ, with $\ell = 1, \ldots, L$ is modelled by a separate foreground appearance $\mathbf{f}_\ell$ and mask $\boldsymbol{\pi}_\ell$. The background can be thought as the $L+1^{th}$ object having a mask $\boldsymbol{\pi}_b = \mathbf{1}$, since the background is present everywhere. For each foreground object ℓ we assume a transformation variable j_ℓ representing all possible translations. Below we assume a moving background where the transformation variable j_b is defined in section 2.1, however all derivations also apply for static or random backgrounds by simply ignoring the variable j_b.

It will be instructive to introduce the model for the case there are only two foreground objects. Assuming $L = 2$, an image $\mathbf{x}$ is generated by instantiating the transformation variables j_1, j_2 and j_b and then drawing $\mathbf{x}$ according to

$$\begin{aligned} p(\mathbf{x}|j_b, j_1, j_2) &= \prod_{p=1}^{P} \{(T_{j_1}\boldsymbol{\pi}_1)_p p_{f_1}(x_p; (T_{j_1}\mathbf{f}_1)_p) + (\mathbf{1} - T_{j_1}\boldsymbol{\pi}_1)_p [(T_{j_2}\boldsymbol{\pi}_2)_p p_{f_2}(x_p; (T_{j_2}\mathbf{f}_2)_p) \\ &+ (\mathbf{1} - T_{j_2}\boldsymbol{\pi}_2)_p p_b(x_p; (T_{j_b}\mathbf{b})_p)]\}, \end{aligned} \tag{7}$$

where the p_{f_1}, p_{f_2} and p_b pixel densities are Gaussians given as in equation 1. Note that each image pixel follows a three component mixture distribution, so that with probability $(T_{j_1}\boldsymbol{\pi}_1)_p$ the pixel can belong to the first object, with probability $(\mathbf{1} - T_{j_1}\boldsymbol{\pi}_1)_p (T_{j_2}\boldsymbol{\pi}_2)_p$ to the second object and with the rest of probability to the background. The fact that the probabilities corresponding

to the second object's pixels are always multiplied by $(\mathbf{1} - T_{j_1}\boldsymbol{\pi}_1)_p$ implies an occlusion ordering between these two objects, so that the first object can occlude the second one, but the opposite is not allowed.

In the general case with arbitrary number of objects the model 7 becomes

$$p(\mathbf{x}|j_b, j_1, \ldots, j_L) = \prod_{p=1}^{P} p(x_p|j_b, j_1, \ldots, j_L) \tag{8}$$

where $p(\mathbf{x}|j_b, j_1, \ldots, j_L)$ is $L+1$-component mixture model

$$p(x_p|j_b, j_1, \ldots, j_L) = \sum_{\ell=1}^{L} \prod_{k=1}^{\ell-1} (\mathbf{1} - T_{j_k}\boldsymbol{\pi}_k)_p (T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_{f_\ell}(x_p; (T_{j_\ell}\mathbf{f}_\ell)_p) + \prod_{k=1}^{L} (\mathbf{1} - T_{j_k}\boldsymbol{\pi}_k)_p p_b(x_p; (T_{j_b}\mathbf{b})_p) \tag{9}$$

where if $\ell = 1$ then the term $\prod_{k=1}^{\ell-1}(\mathbf{1} - T_{j_k}\boldsymbol{\pi}_k)_p$ in 9 is defined to be equal to 1.

The order (from left to right) of the object models in 9 corresponds to the occlusion allowed between the objects. Particularly, the first object exists closest to the camera, thus it can never be occluded by any other object, the second object can only be occluded by the first object and so on. The background exists in the furthest distance from the camera.

Notice that in the above model there is an asymmetry between the objects because of the specified occlusion ordering. If the objects can arbitrarily occlude one another so that the occlusion ordering can change from image to image, then the above model is no longer appropriate. A principled way to deal with this situation is to consider all $L!$ possible rearrangements of the objects (using an additional hidden variable). An alternative way is to replace the foreground pixel densities p_{f_ℓ} by their robust counterparts given

by equation 5. This can allow for arbitrary occlusion between the objects without increasing the model complexity. Of course such a model will not be able to answer immediately what is the occlusion ordering in a given image, but that can be done in a post-processing stage.

From now on we will assume that both the foreground p_{f_ℓ} and background p_b pixels densities are robustified as described in section 2.2. This robustification is the key for our GREEDY algorithm to find the objects one at a time. However, bear in mind that robustifying p_{f_ℓ} makes also sense in terms of allowing arbitrary occlusion between the L foreground objects.

2.4 The GREEDY training algorithm

An exact EM algorithm (Dempster et al., 1977; McLachlan and Krishnan, 1997) for training the above model is highly intractable. This is because a full search over all transformations of the objects requires to consider $J_f^L J_b$ possibilities, which can be extremely large even for small L. An alternative is to consider approximations; Ghahramani (1995) suggests mean field and Gibbs sampling approximations and Jojic and Frey (2001) use approximate variational inference. Below we describe a different learning algorithm by finding the background and all the foreground objects sequentially one after the other.

Each component in the mixture distribution of equation 9 corresponds to an object model which is either one of the L foreground objects or the background. The key idea of our learning algorithm is to learn this mixture model (and thus the relation with the associated transformation variable) sequen-

tially, by learning the objects one at a time. An intuitive way to introduce this algorithm is that originally we constrain the mixture distribution so that the background takes all the probability and the masks of the foreground objects are zero. Since the background pixel densities are robustified according to equation 6 we can learn the background by "ignoring" all the foreground objects. When a pixel of the background is occluded by a foreground object that should be explained by the outlier component in 6, so that the pixel will not affect the estimation of the background. At each subsequent stage the mask of a foreground object is set free to get a non zero value and the corresponding object model is learned. Below we first describe learning the background in section 2.4.1, then discuss learning the first object in section 2.4.2 and further objects in section 2.4.3. We summarize the algorithm in section 2.4.4. Further details are given the Appendix.

2.4.1 Finding the background

The GREEDY algorithm starts by first finding the background. By constraining all the masks $\{\boldsymbol{\pi}_\ell\}_{\ell=1}^L$ to be zero, the mixture model 9 has only one component (corresponding to the background) and thus 8 takes the form $p(\mathbf{x}|j_b) = \prod_{p=1}^P p_b(x_p; (T_{j_b}\mathbf{b})_p)$. Assuming an uniform prior P_{j_b} for transformation j_b the log likelihood of the training images is
$L_b = \sum_{n=1}^N \log \sum_{j_b=1}^J P_{j_b} p(\mathbf{x}^n|j_b)$ which can be maximized with respect to $\{\mathbf{b}, \sigma_b^2\}$ by the EM algorithm. This algorithm searches over J_b possibilities and is tractable; details are provided in the Appendix A.1.

2.4.2 Finding the first object

At the second stage of the algorithm we learn the first foreground object. By allowing the mask $\boldsymbol{\pi}_1$ to take on non-zero values equation 8 becomes

$$\begin{aligned} p(\mathbf{x}|j_b, j_1) &= \prod_{p=1}^{P} (T_{j_1}\pi_1)_p p_{f_1}(x_p; (T_{j_1}\mathbf{f}_1)_p) + (\mathbf{1} - T_{j_1}\boldsymbol{\pi}_1)_p p_b(x_p; (T_{j_b}\mathbf{b})_p) \\ &= \prod_{p=1}^{P} p(x_p|j_b, j_1). \end{aligned} \tag{10}$$

The log likelihood of the training images is $L_1 = \sum_{n=1}^{N} \log \sum_{j_1,j_b} P_{j_1} P_{j_b} p(\mathbf{x}^n|j_1, j_b)$ and a direct maximization using the EM algorithm can be quite demanding, since inference involves searching over $J_f J_b$ possibilities. Our GREEDY algorithm drops the complexity of the search to J_f possibilities by applying a constrained EM algorithm (Neal and Hinton, 1998) that exploits the fact that we already know the background. In particular, for each training image $\mathbf{x}^n$ we introduce the distribution $Q^n(j_1, j_b) = Q^n(j_1|j_b)Q^n(j_b)$ over the transformations and we express a lower bound (based on the Jensen's inequality) of the log likelihood L_1:

$$F_1 = \sum_{n=1}^{N} \sum_{j_b,j_1} Q^n(j_1|j_b)Q^n(j_b) \left\{ \log \left(P_{j_1} P_{j_b} \prod_{p=1}^{P} p(x_p^n|j_b, j_1) \right) - \log Q^n(j_1|j_b)Q^n(j_b) \right\}. \tag{11}$$

This lower bound becomes tight by choosing $Q^n(j_b, j_1)$ to be the posterior $P(j_b, j_1|\mathbf{x}^n)$ for every image $\mathbf{x}^n$. Since we have learned the background we can use the posterior probability $P(j_b|\mathbf{x}^n)$ (computed as described in Appendix A.1) to find the most probable transformation j_b^n that best explains image $\mathbf{x}^n$

and then we approximate $Q^n(j_b)$ so that it gives probability one for $j_b = j_b^n$ and zero for the remaining values[4]. Thus, F_1 takes the form

$$F_1 = \sum_{n=1}^{N} \sum_{j_1=1}^{J_f} Q^n(j_1) \left\{ \sum_{p=1}^{P} \log p(x_p^n | j_b^n, j_1) - \log Q^n(j_1) \right\} + const, \qquad (12)$$

where *const* depends on the uniform probabilities P_{j_b} and P_{j_1}. Also the dependence of $Q^n(j_1)$ on j_b^n for simplicity has been omitted from our notation.

Maximization of F_1 can be carried by the EM algorithm, where in the E-step we maximize F_1 with respect to the Q^n distributions (see equation 22) and in the M-step with respect to the object parameters $\{\mathbf{f}_1, \boldsymbol{\pi}_1, \sigma_1^2\}$. Thus, the computational complexity for learning the object has been kept to a minimum since we have only to search over the J_f possible transformations of the object. Recall that the pixel densities p_{f_1} and p_b are robustified which allows us to deal with occlusion that can be caused by the remaining $L-1$ not-yet-discovered objects.

2.4.3 Learning further objects

The algorithm for learning the second and subsequent foreground objects is a bit more complicated as we subtract out the objects learned so far. We first describe how we learn the second object and then generalize to the case of the ℓ^{th} object.

[4]It would be possible to make a "softer" version of this, where the transformations are weighted by their posterior probabilities, but in practice we have found that these probabilities are usually 1 for the best-fitting transformation and 0 otherwise after learning.

To learn a second foreground object we first allow the mask $\boldsymbol{\pi}_2$ to take on non-zero values so that the conditional density of $\mathbf{x}$ given the hidden transformations becomes $P(\mathbf{x}|j_b, j_1, j_2) = \prod_{p=1}^{P} p(x_p|j_b, j_1, j_2)$ where $p(x_p|j_b, j_1, j_2)$ is given by equation 9. We learn the second object by maximizing a lower bound of the log likelihood $L_2 = \sum_{n=1}^{N} \log \sum_{j_b,j_1,j_2} P_{j_b} P_{j_1} P_{j_2} p(\mathbf{x}^n|j_b, j_1, j_2)$. Particularly, since we have learned the background and the first object we use the most probable transformations j_b^n and j_1^n that explain image $\mathbf{x}^n$ to lower bound the log likelihood L_2:

$$F_2 = \sum_{n=1}^{N} \sum_{j_2=1}^{J_f} Q^n(j_2) \left\{ \sum_{p=1}^{P} \log p(x_p^n|j_b^n, j_1^n, j_2) - \log Q^n(j_2) \right\} + const. \quad (13)$$

F_2 can be tractably optimized over $Q^n(j_2)$ and over the parameters of the second object $\{\mathbf{f}_2, \boldsymbol{\pi}_2, \sigma_2\}$. However, we can make the search for the second object much more efficient (ensuring that we will find a different object) by further constraining equation 13 so that all the pixels belonging to the first object are removed from consideration. First of all note that the values of the transformed mask $T_{j_1^n}\boldsymbol{\pi}_1$ will be close to 1 for all pixels of image $\mathbf{x}^n$ that are part of the first object. All these pixels should be removed from consideration unless there are occluded by other not-yet-discovered objects. Thus, we consider the vector $\boldsymbol{\rho}_1^n = (T_{j_1^n}\boldsymbol{\pi}_1) * \mathbf{r}^{j_1^n}$ where $\mathbf{y} * \mathbf{z}$ denotes the element-wise product of the vectors $\mathbf{y}$ and $\mathbf{z}$ and $r_p^{j_1^n} = \frac{\alpha_f N(x_p^n; (T_{j_1^n}\mathbf{f}_1)_p, \sigma_1^2)}{\alpha_f N(x_p^n; (T_{j_1^n}\mathbf{f}_1)_p, \sigma_1^2) + (1-\alpha_f) U(x_p^n)}$. Thus, $\boldsymbol{\rho}_1^n$ will roughly give values close to 1 only for the non-occluded object pixels of image $\mathbf{x}^n$, and these are the pixels that we wish to remove from consideration. Now considering $(\boldsymbol{\rho}_1^n)_p$ as the probability according to which the pixel p of image $\mathbf{x}^n$ is part of the first object we once again lower bound F_2

using the inequality $\log \sum_i y_i = \log(\sum_i \frac{y_i}{p_i} p_i) \geq \sum_i p_i \log \frac{y_i}{p_i}$ (obtained from Jensen's inequality) to obtain

$$F_2 = \sum_{n=1}^{N} \sum_{j_2=1}^{J_f} Q^n(j_2) \Big\{ \sum_{p=1}^{P} (\boldsymbol{\rho}_1^n)_p \log\{(T_{j_1^n} \boldsymbol{\pi}_1)_p p_{f_1}(x_p^n; (T_{j_1^n} \mathbf{f}_1)_p)\} + (\overline{\boldsymbol{\rho}}_1^n)_p \log\{(\mathbf{1} - T_{j_1^n} \boldsymbol{\pi}_1)_p \times$$
$$[(T_{j_2} \boldsymbol{\pi}_2)_p p_{f_2}(x_p^n; (T_{j_2} \mathbf{f}_2)_p) + (\mathbf{1} - T_{j_2} \boldsymbol{\pi}_2)_p p_b(x_p^n; (T_{j_b^n} \mathbf{b})_p)]\} - \log Q^n(j_2) \Big\} + const, \tag{14}$$

where $\overline{\boldsymbol{\rho}}_1^n = \mathbf{1} - \boldsymbol{\rho}_1^n$ and the *const* is a constant term containing the entropic term $-\sum_{n=1}^{N} \sum_{p=1}^{P} \{(\boldsymbol{\rho}_1^n)_p \log(\boldsymbol{\rho}_1^n)_p + (\overline{\boldsymbol{\rho}}_1^n)_p \log(\overline{\boldsymbol{\rho}}_1^n)_p\}$ plus terms involving the uniform probabilities $\{P_{j_b}, P_{j_1}, P_{j_2}\}$. Since the parameters of the first object are fixed, the above quantity is further written as

$$F_2 = \sum_{n=1}^{N} \sum_{j_2=1}^{J_f} Q^n(j_2) \Big\{ \sum_{p=1}^{P} (\overline{\boldsymbol{\rho}}_1^n)_p \log[(T_{j_2} \boldsymbol{\pi}_2)_p p_{f_2}(x_p^n; (T_{j_2} \mathbf{f}_2)_p) +$$
$$(\mathbf{1} - T_{j_2} \boldsymbol{\pi}_2)_p p_b(x_p^n; (T_{j_b^n} \mathbf{b})_p)] - \log Q^n(j_2) \Big\} + const. \tag{15}$$

Note that when for a pixel p of image $\mathbf{x}^n$ $(\overline{\boldsymbol{\rho}}_1^n)_p \simeq 0$, this pixel is removed from consideration (in a probabilistic fashion) according to equation 15.

Further objects are learned similarly to the two-objects case except that the pixels of all previously learned foreground objects should be removed from consideration. This is achieved by setting $\mathbf{z}_0^n = \mathbf{1}$ for all $n = 1, \ldots, N$ and using the recursion $\mathbf{z}_\ell^n = \mathbf{z}_{\ell-1}^n * \overline{\boldsymbol{\rho}}_\ell^n$. Note that $\mathbf{z}_1^n = \overline{\boldsymbol{\rho}}_1^n$. For object ℓ the objective function F_ℓ given in equation 16 is optimized to yield $\{\mathbf{f}_\ell, \boldsymbol{\pi}_\ell$ and $\sigma_\ell^2\}$.

Note that the GREEDY algorithm treats the background and the rest

L objects differently, since pixels ascribed to the non-occluded background are not removed from consideration as is the case for the foreground objects. We implemented an alternative greedy algorithm that treats the background similarly to the remaining objects (removing non-occluded background pixels from consideration). However, this algorithm did not work so well in practice as some pixels can wrongly be removed from consideration because their values happen to agree with pixels of the occluding object. For the background the number of such pixels can be large since the background is always occluded by all of the L objects. We observed experimentally that this can result in noisy estimates for some of the L foreground objects since many of their pixels are accidentally removed from consideration after the background is learned. On the other hand in the case of the foreground objects is not problematic since occlusion occurs only in some images and is typically partial.

2.4.4 Summary of the GREEDY algorithm

1. Learn the background and infer the most probable transformation j_b^n for each image $\mathbf{x}^n$.

2. Initialize the vectors $\mathbf{z}_0^n = \mathbf{1}$ for $n = 1, \ldots, N$.

3. For $\ell = 1$ to L

    - Learn the ℓ^{th} object parameters $\{\mathbf{f}_\ell, \boldsymbol{\pi}_\ell, \sigma_\ell^2\}$ by maximizing F_ℓ us-

ing EM algorithm, where

$$F_\ell = \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)\Big\{\sum_{p=1}^{P}(\mathbf{z}^n_{\ell-1})_p \log[(T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_{f_\ell}(x_p;(T_{j_\ell}\mathbf{f}_\ell)_p)+ (\mathbf{1}-T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_b(x_p;(T_{j^n_b}\mathbf{b})_p)] - \log Q^n(j_\ell)\Big\}. \quad (16)$$

- Infer the most probable transformation $\{j^n_\ell\}$ and update the weights $\mathbf{z}^n_\ell = \mathbf{z}^n_{\ell-1} * \overline{\boldsymbol{\rho}}^n_\ell$ where $\overline{\boldsymbol{\rho}}^n_\ell$ is computed as described in the text.

The update equations used at any stage of the above algorithm are given in appendix A.

3 Experiments

We describe five experiments extracting movable objects from images using static, moving and random backgrounds. In these experiments the uniform distribution $U(x_p)$ is based on the maximum and minimum pixel values of all training image pixels. In all the experiments reported below except experiment 5 the parameters α_f and α_b were chosen[5] to be 0.9; in experiment 5 α_b was set to 1. Also we assume that the total number of objects L that appear in the images is known, thus the GREEDY algorithm terminates when we discover the L^{th} object.

To apply the GREEDY algorithm we have to initialize the model parameters at each stage. We first describe how we initialize the background

[5]These parameters could be learned with some additional care but in our current implementation we do not do so.

parameters as the background is learned at the first stage of the algorithm. The background appearance $\mathbf{b}$ corresponds to an image that is larger than the input image size. We initialize the centred $P_x \times P_y$ block of $\mathbf{b}$ to be equal with the mean of the training images. The rest pixels of $\mathbf{b}$ are initialized by repeating the border-lines of pixels in the centered block of $\mathbf{b}$ and then adding a Gaussian noise to these pixels. The variance σ_b^2 is initialized to a large value (much larger value than the overall variance of all image pixels[6]). The parameters of an object learned at each subsequent stage of the GREEDY algorithm are always initialized in the same way. Each element of the mask $\boldsymbol{\pi}$ is initialised to 0.5, and the variance σ_ℓ^2 to a large value equal to the σ_b^2 initial value. To initialize the foreground appearance $\mathbf{f}_\ell$ we compute the pixelwise mean of the training images and add independent Gaussian noise with equal variances at each pixel, where the variance is set to be large enough so that the range of pixel values found in the training images can be explored.

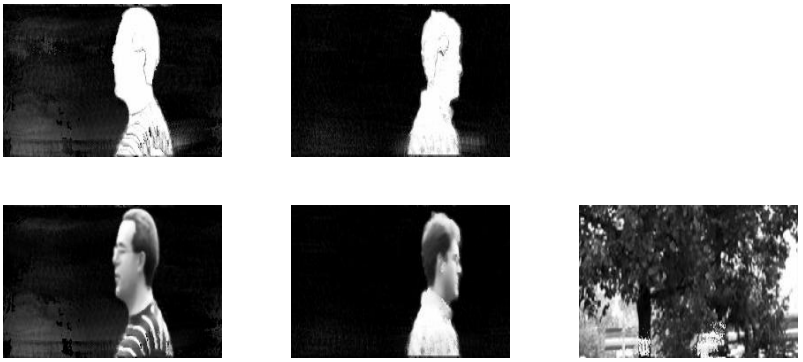
In all of the experiments described below the above initialization scheme proved to be effective and we obtained good results by performing one or two runs of the GREEDY algorithm. At each stage of the algorithm typically 100 iterations were sufficient to reach convergence.

[6]In our experiments the input image pixels are normalized to lie in $[0, 1]$ and the background variance σ_b^2 as well as the any foreground object variance σ_ℓ^2 is initialized to 2.

Experiment 1. Figure 1 illustrates the detection of two objects against a static background[7]. Some examples of the 44 118×248 training images (excluding the black border) are shown in Figure 1(a) and results are shown for the GREEDY algorithm in Figure 1(b). For both objects we show both the learned mask and the elementwise product of the learned foreground and mask. In most runs the person with the striped shirt (Frey) is discovered first. It is interesting to comment on how the GREEDY algorithm operates in case the person with the lighter shirt (Jojic) is found first. As explained in section 2.4, once an object is discovered by the GREEDY algorithm, roughly speaking its non-occluded pixels are removed from consideration. Figure 2 illustrates this point for two frames of the video sequence; one without occlusion and one with occlusion. Figure 2(a) shows the two video frames and Figure 2(b) the pixels (displayed in white) that are masked out from the next run. Note that when Frey occludes Jojic the white stripes of Frey's shirt are accounted for by the Jojic model. This is because the white colour of these stripes agrees with the learned white colour of Jojic's shirt. This does not cause problems for learning the second object (Frey) as there are many frames where the occlusion does not take place. In other experiments with two people wearing different coloured clothes no such effect takes place. Video sequences of the raw data and the extracted objects can be viewed at `http://www.dai.ed.ac.uk/homes/s0129556/lmo.html`.

[7]These data are used in Jojic and Frey (2001). We thank N. Jojic and B. Frey for making available these data via `http://www.psi.toronto.edu/layers.html`.

(a)



(b)

Figure 1: Learning two objects against a static background. Panel (a) displays some frames of the training images, and (b) shows the two objects and background found by the GREEDY algorithm. The plots in the upper row of (b) show the masks $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$. The first two plots in the lower row of (b) display the element-wise products $\boldsymbol{\pi}_1 * \mathbf{f}_1$ and $\boldsymbol{\pi}_2 * \mathbf{f}_2$ while the third plot displays the background $\mathbf{b}$.

(a) (b)

Figure 2: What the GREEDY algorithm removes from consideration once Jojic is found. Panel (a) displays two frames of the training images, and (b) plots the corresponding $\boldsymbol{\rho}_1$ vectors (see section 2.4), that indicate the pixels which are masked out from the second iteration.

Experiment 2. We also conducted an interesting variant on the above experiment. Rather than walking independently, two people now move together, keeping the same distance apart. This led to the extraction of a mask containing both people. Note that this is expected, since the pixels of the two people can be explained by the same transformation, so are considered as one object. Of course, it is open to debate whether we would wish to think of what is learned as one or two objects. In our opinion the ability to extract such regularities is very sensible, and quite widespread, e.g. in finding pairs of eyes. If it was desired, it would be a simple matter to run a connected components algorithm on the thresholded mask to pick out the two objects.

Experiment 3. In the data shown in Figure 3 two objects move against a moving background. Figure 3(a) shows some of the 36 70×140 images of

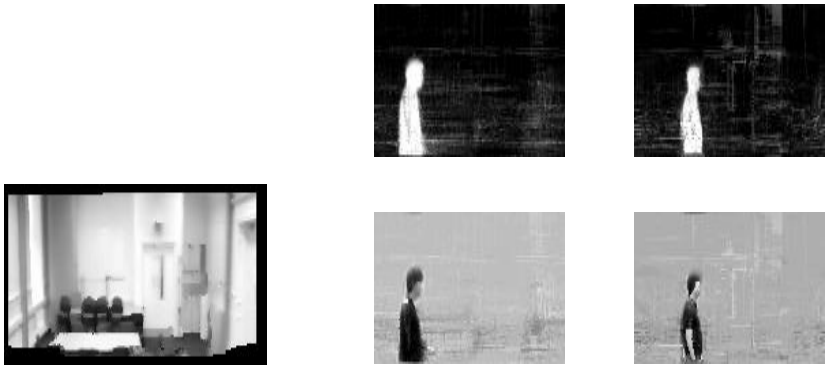
the video sequence. Note that the background changes from frame to frame because of the camera's movement. Notice also that there is motion blur in some of the frames and that one person is occluded by the other in many frames as they walk in the same direction. Figure 3(b) shows the results of the GREEDY algorithm where at the first stage we find the background and at the next two stages the moving objects are found.

Experiment 4. In Figure 4 five objects are learned against a static background, using a dataset of 80 images of size 66×88. Notice the large amount of occlusion in some of the training images shown in Figure 4(a). Results are shown in Figure 4(b) for the GREEDY algorithm.

Experiment 5. In Figure 5 we consider learning objects against random backgrounds. Actually three different backgrounds were used, as can be seen in the example images shown in Figure 5(a). Note that in this case we set $\alpha_b = 1$ since robustifying a random background does not make sense. There were 67 66×88 images in the training set. The results with the GREEDY algorithm are shown in Figure 5(b).

In some other experiments using a few random backgrounds our algorithm has not worked well. In these cases it seems that the foreground models tend to model structure that appears in some backgrounds rather than the foreground objects. These problems might be overcome by using more random backgrounds, as this would fit the random background assumptions better.

(a)



(b)

Figure 3: Learning two objects against a moving background. Panel (a) displays some frames of the training images, and (b) the panorama-background and the masks and rendered objects found by the GREEDY algorithm. (To show the rendered objects we reverse our usual convention and show the objects against a light background as the objects themselves are mainly dark.)

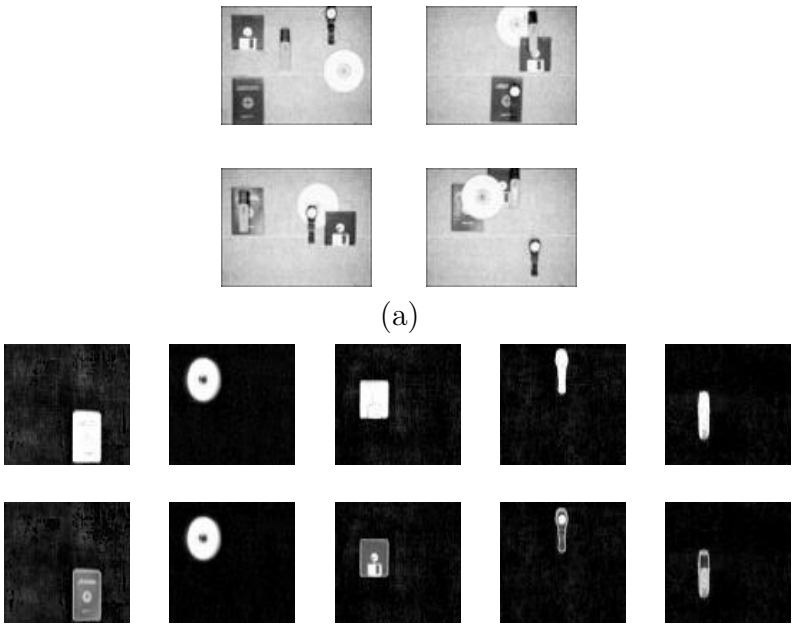


Figure 4: Learning five objects against a static background. Panel (a) displays some of the training images and (b) shows the masks and objects (displayed as described in the caption of Figure 1) learned by the GREEDY algorithm.

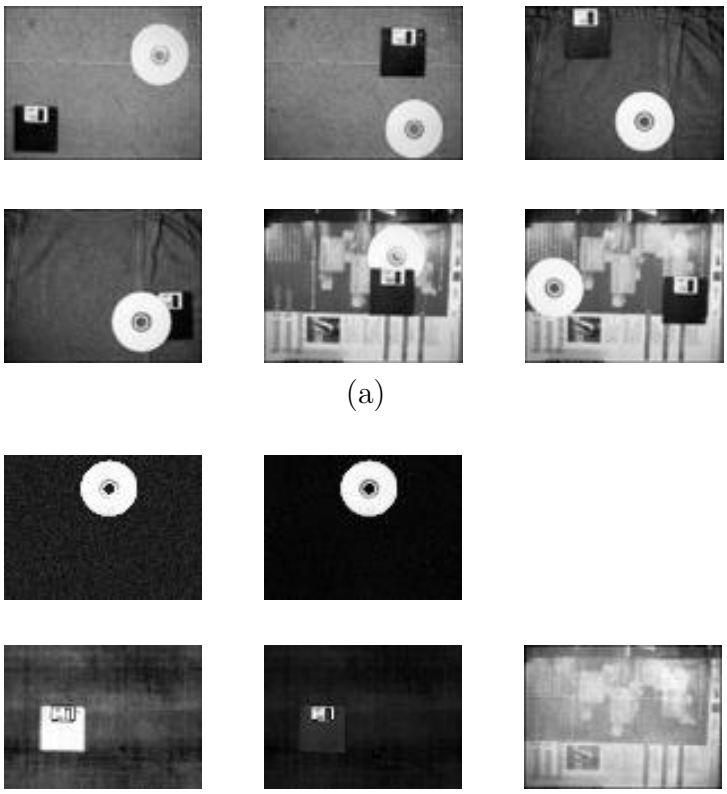


Figure 5: Two objects are learned from a set of images with three different backgrounds. Panel (a) displays some examples of the training images, and (b) shows the masks and objects found by the GREEDY algorithm, displayed as described in the caption of Figure 1.

4 Discussion

The starting point for this work is a full factorial model for the data instantiating multiple objects in their correct positions. However, as we have seen, a direct search over all $O(J_b J_f^L)$ values of the hidden variables is not feasible. Rather than use approximate simultaneous inference of the hidden variables we have developed a sequential method which extracts the background and foreground objects one-at-a-time from the input images. This is achieved by robustifying the generative model so that occlusions of either foreground or background can be tolerated. The results above show that this GREEDY algorithm is very effective at finding the background and foreground objects in the data.

It is interesting to compare our work with that of Shams and von der Malsburg (1999). They obtained candidate parts by matching images in a pairwise fashion, trying to identify corresponding regions in the two images. These candidate image patches were then clustered to compensate for the effect of occlusions. We make four observations: (i) instead of directly learning the models, they match each image against all others (with complexity $O(N^2)$), as compared to the linear scaling with N in our method; (ii) in their method the background must be removed otherwise it would give rise to large match regions; (iii) they do not define a probabilistic model for the images (with all its attendant benefits); (iv) their data (although based on realistic CAD-type models) is synthetic, and designed to focus learning on shape related features by eliminating complicating factors such as background, surface markings etc.

If video sequence data is available then it is possible to compute optical flow information, and this can be used as a cue to discover objects by clustering flow vectors into "layers". Some early work on this topic is by Wang and Adelson (1994), and an example of more recent work is that of Tao et al. (2000). Note that our method does not require a video sequence and can be applied to unordered collections of images, as illustrated in Experiments 4 and 5. Also, problems can arise for optical-flow based methods in regions of low texture where flow information can be sparse.

In our work the model for each pixel is a mixture of Gaussians. There is some previous work on pixelwise mixtures of Gaussians (see, e.g. Rowe and Blake 1995) which can, for example, be used to achieve background subtraction and highlight moving objects against a static background. Our work extends beyond this by gathering the foreground pixels into objects, and also allows us to learn objects in the more difficult non-static background case.

The GREEDYmethod has an analogue in neural network methods for Principal Components Analysis (PCA). To carry out PCA we can extract the principal component using Hebbian learning. If we then subtract of the projection of the input onto the principal direction, we can again use Hebbian learning to extract the second principal component, and so on (Sanger, 1989). This process parallels the successive discovery of objects in our method. However, we note that this sequential algorithm cannot be used if a full factor analysis model (with different noise variances on different visible dimensions) is to be learned.

The GREEDY algorithm has shown itself to be an effective factorial

learning algorithm for image data. We are currently investigating issues such as dealing with richer classes of transformations, detecting L automatically, and allowing objects not to appear in all images. Furthermore, although we have described this work in relation to image modelling, it can be applied to other domains. For example, one can apply the GREEDY approach to fitting mixture models, as we will describe in a forthcoming paper. Also, one can make a model for sequence data by having Hidden Markov models (HMMs) for a "foreground" pattern and the "background". Faced with sequences containing multiple foreground patterns, one could extract these patterns sequentially using a similar algorithm to that described above. It is true that for sequence data it would be possible to train a compound HMM consisting of $L+1$ HMM components simultaneously, but there may be severe local minima problems in the search space so that the sequential approach might be preferable.

Acknowledgements

CW thanks Geoff Hinton for helpful discussions concerning the idea of learning one object at a time, Zoubin Ghahramani for helpful discussions on the greedy algorithm and Andrew Fitzgibbon for pointers to the "layers" literature. We thank the anonymous referees for their comments which helped improve the paper.

References

Barlow, H. (1989). Unsupervised Learning. *Neural Computation*, 1:295–311.

Black, M. J. and Jepson, A. (1996). EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. In Buxton, B. and Cipolla, R., editors, *Proceedings of the Fourth European Conference on Computer Vision, ECCV'96*, pages 329–342. Springer-Verlag.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society B*, 39:1–38.

Frey, B. J. and Jojic, N. (1999). Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 1999*. IEEE Computer Society Press. Ft. Collins, CO.

Frey, B. J. and Jojic, N. (2002). Fast, large-scale transformation-invariant clustering. In Diettrich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 721–727. MIT Press.

Frey, B. J. and Jojic, N. (2003). Transformation Invariant Clustering Using the EM Algorithm. *IEEE Trans Pattern Analysis and Machine Intelligence*, 25(1):1–17.

Ghahramani, Z. (1995). Factorial Learning and the EM Algorithm. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neu-*

ral Information Processing Systems 7, pages 617–624. Morgan Kaufmann, San Mateo, CA.

Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*. Morgan Kaufmann.

Jojic, N. and Frey, B. J. (2001). Learning Flexible Sprites in Video Layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2001*. IEEE Computer Society Press. Kauai, Hawaii.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791.

McLachlan, G. J. and Krishnan, T. (1997). *The EM Algorithm*. Wiley, Chichester, UK.

Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic.

Rowe, S. and Blake, A. (1995). Statistical Background Modelling For Tracking With A Virtual Camera. In Pycock, D., editor, *Proceedings of the 6th British Machine Vision Conference*, volume volume 2, pages 423–432. BMVA Press.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473.

Shams, L. and von der Malsburg, C. (1999). Are object shape primitives learnable? *Neurocomputing*, 26-27:855–863.

Tao, H., Sawhney, H. S., and Kumar, R. (2000). Dynamic Layer Representation with Applications to Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Wang, J. Y. A. and Adelson, E. H. (1994). Representing Moving Images with Layers. *IEEE Transactions on Image Processing*, 3(5):625–638.

Williams, C. K. I. and Titsias, M. K. (2003). Learning About Multiple Objects in Images: Factorial Learning without Factorial Search. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

A Details of the GREEDY algorithm

We introduce first some notation. If $\mathbf{y}$ and $\mathbf{z}$ are two vectors of the same size, then $\mathbf{y} * \mathbf{z}$ defines the element-wise product between these vectors and $\mathbf{y} * \mathbf{y}$ is written as $\mathbf{y}^2$ for compactness. Similarly the element-wise division between two vectors is denoted by $\mathbf{y}./\mathbf{z}$. A vector containing ones is denoted by $\mathbf{1}$. Also summations of the form $\sum_{p=1}^{P} y_p z_p$ are written in vector notation $\mathbf{y}^T\mathbf{z}$, e.g. $\mathbf{y}^T\mathbf{1}$ denotes the sum of elements of $\mathbf{y}$.

In our implementation the transformation matrices of the foreground objects T_{j_ℓ} are permutation matrices. However, our derivations below regarding these matrices only require two constraints: (1) that the value of

each element of $T_{j_\ell}\boldsymbol{\pi}_\ell$ is a valid probability (i.e. lies in $[0,1]$) and (2) that $\log(T_{j_\ell}\boldsymbol{\pi}_\ell) = T_{j_\ell}\log\boldsymbol{\pi}_\ell$ and $\log(\mathbf{1} - T_{j_\ell}\boldsymbol{\pi}_\ell) = T_{j_\ell}\log(\mathbf{1} - \boldsymbol{\pi}_\ell)$, where $\log\mathbf{v}$ denotes the element-wise logarithm of a vector $\mathbf{v}$. These constraints certainly hold for matrices which have one 1 (and the other entries 0) in each row.

A.1 Learning the background

Here we derive the EM algorithm for learning a static or moving background. Learning the background consists of the first stage of the GREEDY algorithm and is carried out by maximizing the following log likelihood:

$$L_b = \sum_{n=1}^{N} \log \sum_{j_b=1}^{J_b} P_{j_b} \prod_{p=1}^{P} \{\alpha N(x_p^n; (T_{j_b}\mathbf{b})_p, \sigma_b^2) + (1-\alpha)U(x_p^n)\}. \tag{17}$$

Clearly this log likelihood corresponds to a mixture model (with J_b components) where the component densities are factorized over the pixels and each pixel density is a two-component mixture. Application of the EM is straightforward and we can easily show that the expected complete data log likelihood in the EM framework is:

$$Q_b = \sum_{n=1}^{N}\sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)\{(\mathbf{r}_{j_b}^n)^T[-\frac{1}{2\sigma_b^2}(\mathbf{x}^n - T_{j_b}\mathbf{b})^2 - \frac{1}{2}\log\sigma_b^2\mathbf{1}]\} + const, \tag{18}$$

where $P(j_b|\mathbf{x}^n) = \frac{p(\mathbf{x}^n|j_b)}{\sum_{i=1}^{J_b} p(\mathbf{x}^n|i)}$ is the posterior probability of the transformation hidden variable j_b given the image $\mathbf{x}^n$ and $\mathbf{r}_{j_b}^n$ is a P length vector with the p^{th} element storing the probability according to which the p^{th} pixel of image $\mathbf{x}^n$ is part of the non-occluded background given j_b: $(\mathbf{r}_{j_b}^n)_p = \frac{\alpha_b N(x_p^n;(T_{j_b}\mathbf{b})_p,\sigma_b^2)}{\alpha_b N(x_p^n;(T_{j_b}\mathbf{b})_p,\sigma_b^2)+(1-\alpha_b)U(x_p^n)}$.

In the E-step of the algorithm $P(j_b|\mathbf{x}^n)$ and $\mathbf{r}_{j_b}^n$ are obtained using the current parameter values. In the M-step the Q function is maximized with respect to the parameters $\{\mathbf{b}, \sigma_b^2\}$ giving the following update equations:

$$\mathbf{b} \leftarrow \sum_{n=1}^{N} \sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)[T_{j_b}^T(\mathbf{r}_{j_b}^n * \mathbf{x}^n)]./ \sum_{n=1}^{N} \sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)[T_{j_b}^T \mathbf{r}_{j_b}^n], \tag{19}$$

$$\sigma_b^2 \leftarrow \frac{\sum_{n=1}^{N} \sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)[(\mathbf{r}_{j_b}^n)^T(\mathbf{x}^n - T_{j_b}\mathbf{b})^2]}{\sum_{n=1}^{N} \sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)[(\mathbf{r}_{j_b}^n)^T \mathbf{1}]}. \tag{20}$$

The above equations provide an exact M-step. The update for the background appearance $\mathbf{b}$ is very intuitive. For example consider the case when $P(j_b|\mathbf{x}^n) = 1$ for $j_b = j^*$ and 0 otherwise. For pixels which are ascribed to non-occluded background (i.e. $(\mathbf{r}_{j_b}^n)_p \simeq 1$) the values of $\mathbf{x}^n$ are transformed by $T_{j^*}^T$ which maps the $P_x \times P_y$ image $\mathbf{x}^n$ into a larger image of size $M_x \times M_y$ so that $\mathbf{x}^n$ is located in the position specified by j_b and the rest of image pixels are filled with zero values. Thus, the non-occluded pixels found in each training image are located properly into the big panorama image and averaged to produce $\mathbf{b}$.

Note also that in the special case where the background is static the effect of transformation j_b is removed from all update equations and the parameters $\mathbf{b}$ and σ_b^2 are updated according to $\mathbf{b} \leftarrow \sum_{n=1}^{N}(\mathbf{r}^n * \mathbf{x}^n)./\sum_{n=1}^{N} \mathbf{r}^n$ and $\sigma_b^2 \leftarrow \frac{\sum_{n=1}^{N}(\mathbf{r}^n)^T(\mathbf{x}^n - \mathbf{b})^2}{\sum_{n=1}^{N}(\mathbf{r}^n)^T \mathbf{1}}$, respectively.

For random backgrounds, the above EM algorithm is not needed. In this case we simply set $\mathbf{b}$ to the mean of all training images and σ_b^2 to the mean variance of all different pixel variances. These background parameters are kept fixed for later stages.

A.2 Learning the foreground objects

Assume that we have already found the background as described previously. At each next stage the GREEDY algorithm searches for a foreground object. Below we describe how the ℓ^{th} foreground object is found, where $\ell = 1, \ldots, L$.

When we search for the ℓ^{th} object, the background as well as the $\ell - 1$ foreground objects have been found in previous stages[8]. As explained in section 2.4.3 we learn the ℓ^{th} object by maximizing the objective function

$$F_\ell = \sum_{n=1}^{N} \sum_{j_\ell=1}^{J_f} Q^n(j_\ell) \Big\{ \sum_{p=1}^{P} (\mathbf{z}_{\ell-1}^n)_p \log[(T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_{f_\ell}(x_p^n; (T_{j_\ell}\mathbf{f}_\ell)_p) + (\mathbf{1} - T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_b(x_p^n; (T_{j_b^n}\mathbf{b})_p)] - \log Q^n(j_\ell) \Big\}. \quad (21)$$

The above maximization can be done by a variational EM algorithm. In the E-step we maximize F_ℓ with respect to the $Q^n(j_\ell)$ which gives

$$Q^n(j_\ell) \propto \exp \left\{ \sum_{p=1}^{P} (\mathbf{z}_{\ell-1}^n)_p \log[(T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_{f_\ell}(x_p^n; (T_{j_\ell}\mathbf{f}_\ell)_p) + (\mathbf{1} - T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_b(x_p^n; (T_{j_b^n}\mathbf{b})_p)] \right\} \quad (22)$$

with $Q^n(j_\ell)$ normalized to sum to one. In the M-step we maximize F_ℓ with respect to the object parameters $\{\mathbf{f}_\ell, \boldsymbol{\pi}_\ell, \sigma_\ell^2\}$. For this maximization we need again the EM algorithm. The EM algorithm operates in the following Q

[8] Of course when we search for the first object there will be no previously learned foreground objects.

function

$$Q_\ell = \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)(\mathbf{z}^n_{\ell-1})^T[\overline{\mathbf{s}}^n_{j_\ell} * \log T_{j_\ell}\boldsymbol{\pi}_\ell + (\mathbf{1}-\overline{\mathbf{s}}^n_{j_\ell}) * \log(\mathbf{1}-T_{j_\ell}\boldsymbol{\pi}_\ell) + \overline{\mathbf{s}}^n_{j_\ell} * \mathbf{r}^n_{j_\ell} * (-\frac{1}{2\sigma^2_\ell}(\mathbf{x}^n - T_{j_\ell}\mathbf{f}_\ell)^2 - \frac{1}{2}\log\sigma^2_\ell\mathbf{1})] + const, \quad (23)$$

where each element of the vector $\overline{\mathbf{s}}^n_{j_\ell}$ stores the value $\frac{(T_{j_\ell}\pi_\ell)_p p_{f_\ell}(x^n_p;(T_{j_\ell}\mathbf{f}_\ell)_p)}{(T_{j_\ell}\boldsymbol{\pi}_\ell)_p p_{f_\ell}(x^n_p;(T_{j_\ell}\mathbf{f}_\ell)_p)+(\mathbf{1}-(T_{j_\ell}\boldsymbol{\pi}_\ell)_p)p_b(x^n_p;(T_{j^n_b}\mathbf{b})_p)}$ expressing the probability the pixel to be part of the object. Each element of $\mathbf{r}^n_{j_\ell}$ stores the probability the pixel to be non-occluded: $(\mathbf{r}^n_{j_\ell})_p = \frac{\alpha_f N(x^n_p;(T_{j_\ell}\mathbf{f}_\ell)_p,\sigma^2_\ell)}{\alpha_f N(x^n_p;(T_{j_\ell}\mathbf{f}_\ell)_p,\sigma^2_\ell)+(1-\alpha_f)U(x^n_p)}$.

The algorithm in the E-step computes the quantities, $Q^n(j_\ell)$, $\overline{\mathbf{s}}^n_{j_\ell}$ and $\mathbf{r}^n_{j_\ell}$ as described above and in the M-step we update the parameters as follows:

$$\boldsymbol{\pi}_\ell \leftarrow \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)T^T_{j_\ell}[\mathbf{z}^n_{\ell-1} * \overline{\mathbf{s}}^n_{j_\ell}]./\sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)T^T_{j_\ell}\mathbf{z}^n_{\ell-1}, \quad (24)$$

$$\mathbf{f}_\ell \leftarrow \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)T^T_{j_\ell}[\mathbf{z}^n_{\ell-1} * \overline{\mathbf{s}}^n_{j_\ell} * \mathbf{r}^n_{j_\ell} * \mathbf{x}^n]./\sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)T^T_{j_\ell}[\mathbf{z}^n_{\ell-1} * \overline{\mathbf{s}}^n_{j_\ell} * \mathbf{r}^n_{j_\ell}], \quad (25)$$

$$\sigma^2_\ell \leftarrow \frac{\sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)(\mathbf{z}^n_{\ell-1})^T[\overline{\mathbf{s}}^n_{j_\ell} * \mathbf{r}^n_{j_\ell} * (\mathbf{x}^n - T_{j_\ell}\mathbf{f}_\ell)^2]}{\sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)(\mathbf{z}^n_{\ell-1})^T[\overline{\mathbf{s}}^n_{j_\ell} * \mathbf{r}^n_{j_\ell}]}. \quad (26)$$

As with the updates for $\mathbf{b}$ and σ^2_b these updates make intuitive sense. Consider, for example, the ℓ^{th} appearance model $\mathbf{f}_\ell$ when $Q^n(j_\ell) = 1$ for $j_\ell = j^*$ and 0 otherwise. For pixels which are ascribed to the ℓ^{th} foreground and are not occluded (i.e. $(\mathbf{z}^n_{\ell-1} * \overline{\mathbf{s}}^n_{j^*} * \mathbf{r}^n_{j^*})_p \simeq 1$), the values in $\mathbf{x}^n$ are transformed by $T^T_{j^*}$ (which is $T^{-1}_{j^*}$ as the transformations are permutation matrices). This removes the effect of the transformation and thus allows the foreground pixels

found in each training image to be averaged to produce $\mathbf{f}_\ell$.