Sequential Learning of Layered Models from Video

Michalis K. Titsias and Christopher K. I. Williams

School of Informatics, University of Edinburgh,
Edinburgh EH1 2QL, UK
M.Titsias@sms.ed.ac.uk, c.k.i.williams@ed.ac.uk
http://www.anc.ed.ac.uk/

Abstract. A popular framework for the interpretation of image sequences is the layers or sprite model, see e.g. [1], [2]. Jojic and Frey [3] provide a generative probabilistic model framework for this task, but their algorithm is slow as it needs to search over discretized transformations (e.g. translations, or affines) for each layer simultaneously. Exact computation with this model scales exponentially with the number of objects, so Jojic and Frey used an approximate variational algorithm to speed up inference. Williams and Titsias [4] proposed an alternative sequential algorithm for the extraction of objects one at a time using a robust statistical method, thus avoiding the combinatorial explosion.
In this chapter we elaborate on our sequential algorithm in the following ways: Firstly, we describe a method to speed up the computation of the transformations based on approximate tracking of the multiple objects in the scene. Secondly, for sequences where the motion of an object is large so that different views (or aspects) of the object are visible at different times in the sequence, we learn appearance models of the different aspects. We demonstrate our method on four video sequences, including a sequence where we learn articulated parts of a human body.

1 Introduction

A powerful framework for modelling video sequences is the layer-based approach which models an image as a composite of 2D layers, each one representing an object in terms of its appearance and region of support or mask, see e.g. Wang and Adelson [1] and Irani et al. [2]. A layered representation explicitly accounts for occlusion between the objects, permits motion segmentation in a general multiple-frame setting rather than in pairs of frames, and provides appearance models for the underlying objects. These properties can allow layered models to be useful for a number of different purposes such as video compression, video summarization, background substitution (e.g. alpha matting applications), object recognition (e.g. learning an object recognition system from video clips without needing human annotation) and others.

Jojic and Frey [3] provided a principled generative probabilistic framework for learning a layered model allowing transparency between the layers. Williams and Titsias [4] developed a similar model where layers strictly combine by occlusion. Learning these models using an exact EM algorithm faces the problem that as the number of objects increases, there is a combinatorial explosion of the number of configurations that need

to be considered. If there are L possible objects, and there are J transformations that any one object can undergo, then we will need to consider $O(J^L)$ combinations to explain any image. Jojic and Frey [3] tackled this problem by using a variational inference scheme searching over all transformations simultaneously, while Williams and Titsias [4] developed a *sequential* approach using robust statistics which searches over the transformations of one object at each time. Both these methods do not require a video sequence and can work on unordered sets of images. In this case, training can be very slow as it is necessary to search over all possible transformations of at least a single object on every image. However, for video sequences we could considerably speed up the training by first localizing each object based on a recursive processing of the consecutive frames. Recursive localization can approximate the underlying sequence of transformations of an object in the frames and thus learning can be carried out with a very focused search over the neighbourhood of these transformations or without search at all when the approximation is accurate. We refer to the recursive localization procedure as object tracking.

In this chapter we describe two developments of the above model. Firstly, assuming video data and based on tracking we speed up the method of Williams and Titsias [4]. First, the moving background is tracked and then its appearance is learned, while moving foreground objects are found at later stages. The tracking algorithm itself recursively updates an appearance model of the tracked object and approximates the transformations by matching this model to the frames through the sequence.

Secondly, in order to account for variation in object appearance due to changes in the 3D pose of the object and self occlusion, we model different visual aspects or views of each foreground object. This is achieved by introducing a set of mask and appearance pairs, each one associated with a different view of the object. To learn different viewpoint object models we use approximate tracking, so that we first estimate the 2D or planar transformations of each foreground object in all frames and then given these transformations we stabilize the video and learn the viewpoint models for that object using a mixture modelling approach.

The structure of the remainder of the chapter is as follows: In section 2 we describe the layered generative model which assumes multiple views for each foreground object. Section 3 describes learning the model from a video sequence, while section 4 discusses related work. Section 5 gives experimental results and we conclude with a discussion in section 6.

2 Generative layered model

For simplicity we will present the generative model assuming that there are two layers, i.e. a foreground object and a static background. Later in this section we will discuss the case of arbitrary number of foreground layers and a moving background.

Let $\mathbf{b}$ denote the appearance image of the background arranged as a vector. Assuming that the background is static, $\mathbf{b}$ will have the same size as the data image size (although note that for moving backgrounds, $\mathbf{b}$ will need to be larger than the image size). Each entry b_i stores the ith pixel value which can either be a grayscale intensity value or a colour value. In our implementation we allow coloured images where b_i is a three-dimensional vector in the RGB space. However, for notational convenience below we assume that b_i is a scalar representing a grayscale value.

In contrast to the background, the foreground object occupies some region of the image and thus to describe this layer we need both an appearance $\mathbf{f}$ and mask $\boldsymbol{\pi}$. The

foreground is allowed to move so there is an underlying transformation with index j that e.g. corresponds to translational or affine motion and a corresponding transformation matrix so that $T_j\mathbf{f}$ and $T_j\boldsymbol{\pi}$ is the transformed foreground and mask, respectively. A pixel in an observed image is either foreground or background. This is expressed by a vector of binary latent variables $\mathbf{s}$, one for each pixel drawn from the distribution

$$P(\mathbf{s}|j) = \prod_{i=1}^{P} (T_j\boldsymbol{\pi})_i^{s_i} (\mathbf{1} - T_j\boldsymbol{\pi})_i^{1-s_i}, \tag{1}$$

where $\mathbf{1}$ denotes the vector of ones. Each variable s_i is drawn independently so that for pixel i, if $(T_j\boldsymbol{\pi})_i \simeq 0$, then the pixel will be ascribed to the background with high probability, and if $(T_j\boldsymbol{\pi})_i \simeq 1$, it will be ascribed to the foreground with high probability. Note that $\mathbf{s}$ is the binary mask of the foreground object in an example image, while $\boldsymbol{\pi}$ is the prior untransformed mask that captures roughly the shape of the object stored in $\mathbf{f}$.

Selecting a transformation index j, using prior P_j over J possible values with $\sum_{j=1}^{J} P_j = 1$, and a binary mask $\mathbf{s}$, an image $\mathbf{x}$ is drawn from the Gaussian

$$p(\mathbf{x}|j, \mathbf{s}) = \prod_{i=1}^{P} N(x_i; (T_j\mathbf{f})_i, \sigma_f^2)^{s_i} N(x_i; b_i, \sigma_b^2)^{1-s_i}, \tag{2}$$

where each pixel is drawn independently from the above conditional density. To express the likelihood of an observed image $p(\mathbf{x})$ we marginalise out the latent variables, which are the transformation j and the binary mask $\mathbf{s}$. Particularly, we first sum out $\mathbf{s}$ using (1) and (2) and obtain

$$p(\mathbf{x}|j) = \prod_{i=1}^{P} (T_j\boldsymbol{\pi})_i N(x_i; (T_j\mathbf{f})_i, \sigma_f^2) + (\mathbf{1} - T_j\boldsymbol{\pi})_i N(x_i; b_i, \sigma_b^2). \tag{3}$$

Using now the prior P_j over the transformation j, the probability of an observed image $\mathbf{x}$ is $p(\mathbf{x}) = \sum_{j=1}^{J} P_j p(\mathbf{x}|j)$. Given a set of images $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ we can adapt the parameters $\theta = \{\mathbf{b}, \mathbf{f}, \boldsymbol{\pi}, \sigma_f^2, \sigma_b^2\}$ to maximize the log likelihood using the EM algorithm.

The above model can be extended so as to have a moving background and L foreground objects [4]. For example, for two foreground layers with parameters $(\mathbf{f}_1, \boldsymbol{\pi}_1, \sigma_1^2)$ and $(\mathbf{f}_2, \boldsymbol{\pi}_2, \sigma_2^2)$ and also a moving background, the analogue of equation (3) is

$$\begin{aligned} p(\mathbf{x}|j_1, j_2, j_b) = \prod_{i=1}^{P} & (T_{j_1}\boldsymbol{\pi}_1)_i N(x_i; (T_{j_1}\mathbf{f}_1)_i, \sigma_1^2) + (\mathbf{1} - T_{j_1}\boldsymbol{\pi}_1)_i \times \\ & \left[(T_{j_2}\boldsymbol{\pi}_2)_i N(x_i; (T_{j_2}\mathbf{f}_2)_i, \sigma_2^2) + (\mathbf{1} - T_{j_2}\boldsymbol{\pi}_2)_i N(x_i; (T_b\mathbf{b})_i, \sigma_b^2) \right], \end{aligned} \tag{4}$$

where j_1, j_2 and j_b denote the transformation of the first foreground object, the second foreground object and the background, respectively.

Furthermore, we can allow for an arbitrary occlusion ordering between the foreground objects, so that it can vary in different images, by introducing an additional hidden variable that takes as values all $L!$ possible permutations of the foreground layers.

2.1 Incorporating multiple viewpoints

The layered model presented above assumes that the foreground object varies due to a 2D (or planar) transformation. However, in many video sequences this assumption will not be true e.g. a foreground object can undergo 3D rotation so that at different times we may see different views (or aspects) of the object. For example, Figure 3 shows three frames of a sequence capturing a man walking; clearly the man's pose changes substantially during time. Next we generalize the layered model so that the appearance of a foreground object can be chosen from a set of possible appearances associated with different viewpoints.

Assume again that there are two layers: one static background and one moving foreground object. We introduce a discrete latent variable v, that can obtain V possible values indexed by integers from 1 to V. For each value v we introduce a separate pair of appearance $\mathbf{f}^v$ and mask $\boldsymbol{\pi}^v$ defined as in section 2. Each pair $(\mathbf{f}^v, \boldsymbol{\pi}^v)$ models a particular view of the object.

To generate an image $\mathbf{x}$ we first select a transformation j and a view v using prior probabilities P_j and P_v, respectively. Then we select a binary mask $\mathbf{s}$ from the distribution $P(\mathbf{s}|j,v) = \prod_{i=1}^{P} (T_j \boldsymbol{\pi}^v)_i^{s_i} (\mathbf{1} - T_j \boldsymbol{\pi}^v)_i^{1-s_i}$, and draw an image $\mathbf{x}$ from the Gausssian $p(\mathbf{x}|j,v,s) = \prod_{i=1}^{P} N(x_i; (T_j \mathbf{f}^v)_i, \sigma_f^2)^{s_i} N(x_i; b_i, \sigma_b^2)^{1-s_i}$. Note the similarity of the above expressions with equations (1) and (2). The only difference is that now the appearance $\mathbf{f}$ and mask $\boldsymbol{\pi}$ are indexed by v to reflect the fact that we have also chosen a view for the foreground object.

To express the probability distribution according to which an image is generated given the transformation, we sum out the binary mask and the view variable and obtain

$$p(\mathbf{x}|j) = \sum_{v=1}^{V} P_v p(\mathbf{x}|j,v), \tag{5}$$

where $p(\mathbf{x}|j,v)$ is given as in (3) with $\mathbf{f}$ and $\boldsymbol{\pi}$ indexed by v. Notice how the equation (5) relates to equation (3). Clearly now $p(\mathbf{x}|j)$ is a mixture model of the type of model given in (3) so that each mixture component is associated with a visual aspect. For example, if we choose to have a single view the latter expression reduces to the former one.

It is straightforward to extend the above model to the case of L foreground layers with varying viewpoints. In this case we need a separate view variable v_ℓ for each foreground object and a set of appearance and mask pairs: $(\mathbf{f}_\ell^{v_\ell}, \boldsymbol{\pi}_\ell^{v_\ell})$, $v_\ell = 1, \ldots, V_\ell$. For example, when we have two foreground objects and a moving background the conditional $p(\mathbf{x}|j_1, j_2, j_b, v_1, v_2)$ is given exactly as in (4) by introducing suitable indexes to the foreground appearances and masks that indicate the choices made for the viewpoint variables.

3 Learning

Given the set of possibly unordered images $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ a principled way to learn the parameters $\theta = (\{\mathbf{f}_1^{v_1}, \boldsymbol{\pi}_1^{v_1}, \sigma_{1,v_1}^2\}_{v_1=1}^{V_1}, \ldots, \{\mathbf{f}_L^{v_L}, \boldsymbol{\pi}_L^{v_l}, \sigma_{L,v_L}^2\}_{v_L=1}^{V_L}, \mathbf{b}, \sigma_b^2)$ is by maximizing the log likelihood $L(\theta) = \sum_{n=1}^{N} \log p(\mathbf{x}^n|\theta)$ using the EM algorithm. However, an exact EM algorithm is intractable. If the foreground objects and the background can undergo J transformations and assuming V views for each foreground object, the time

needed to carry out the E-step for a training image is $O(J^{L+1}V^L L!)$. Clearly, this computation is infeasible as it grows exponentially with L. A variational EM algorithm can be used to simultaneously infer the parameters, the transformations and the viewpoint variables in all images in time linear with L. However such algorithm can face two problems: (i) it will be very slow as the number of all transformations J can be very large (e.g. for translations and rotations can be of order of hundred of thousands) and (ii) simultaneous search over all the unknown quantities can be prone to severe local maxima, e.g. there is a clear danger of confusion the aspects of one object and the corresponding aspects of a different object.

Our learning algorithm works for video data and proceeds in stages so that, roughly speaking, each stage deals with a different set of unknown variables. Table 1 illustrates all the different steps of the algorithm. At `Stage 1`, we ignore the search needed to compute the occlusion ordering of the foreground objects and we focus on approximating the transformations $\{j_1^n, j_2^n, \ldots, j_L^n, j_b^n\}_{n=1}^N$ and inferring the viewpoint variables $\{v_1^n, \ldots, v_L^n\}_{n=1}^N$ for all training images. In this stage also we obtain good initial estimates for the parameters of all objects. At `Stage 2` we compute the occlusion orderings and we jointly refine all the parameters by optimizing the complete likelihood of the model where all the transformations, view variables and occlusion orderings have been "filled in" using their approximate values

`Stage 1` is the intensive part of the learning process and is divided in sub-stages. Particularly, the object parameters and their associated transformations are estimated in a greedy fashion so as to deal with one object at a time. Particularly, we first track the background in order to approximate the transformations $(j_b^1, \ldots, j_b^N)$ and then given these transformations we learn the background appearance. Then for each foreground object sequentially we track it and learn all of its different views.

- `Stage 1:`
  1. Track the background to compute the transformations $(j_b^1, \ldots, j_b^N)$. Then learn the background parameters $(\mathbf{b}, \sigma_b^2)$.
  2. Suppress all the pixels that have been classified as part of the background in each training image, so that $\mathbf{w}_1^n$ indicates the remaining non-background pixels in the image $\mathbf{x}^n$.
  3. for $\ell = 1$ to L
     (a) Using the $\mathbf{w}_\ell^n$ vectors track the ℓth object to compute the transformations $(j_\ell^1, \ldots, j_\ell^N)$. Then learn the parameters $\{\mathbf{f}_\ell^{v_\ell}, \boldsymbol{\pi}_\ell^{v_\ell}, \sigma_{\ell,v_\ell}^2\}_{v_\ell=1}^{V_\ell}$.
     (b) If $\ell = L$ go to `Stage 2`. Otherwise, construct the vectors $\mathbf{w}_{\ell+1}^n$ from $\mathbf{w}_\ell^n$ so that all the pixels classified as part of the ℓth object in all images are additionally suppressed.
- `Stage 2:` Using the inferred values of the parameters θ from `Stage 1`, the transformations, and the view variables, compute the occlusion ordering of the foreground layers in each image. Then using these occlusion orderings refine the parameters of the objects.

Table 1. The steps of the learning algorithm.

The next three sections explain in detail all the steps of the learning algorithm. Particularly, section 3.1 discusses learning the background (step 1 in `Stage 1`), section 3.2 describes learning the foreground objects (steps 2 and 3 in `Stage 1`) and section 3.3 discusses computation of the occlusion ordering of the foreground objects

and refinement of the parameters (`Stage 2`). A preliminary version of this algorithm was presented in [5].

3.1 Learning the background

Assume that we have approximated the transformations $\{j_b^1, \ldots, j_b^N\}$ of the background in each frame of the video. We will discuss shortly how to obtain such approximation using tracking. Using these transformations we wish to learn the background appearance.

At this stage we consider images that contain a background and many foreground objects. However, we concentrate on learning only the background. This goal can be achieved by introducing a likelihood model for the images that only accounts for the background while the presence of the foreground objects will be explained by an outlier process. For a background pixel, the foreground objects are interposed between the camera and the background, thus perturbing the pixel value. This can be modelled with a mixture distribution as $p_b(x_i; b_i) = \alpha_b N(x_i; b_i, \sigma_b^2) + (1 - \alpha_b)U(x_i)$, where α_b is the fraction of times a background pixel is not occluded, and the robustifying component $U(x_i)$ is a uniform distribution common for all image pixels. When the background pixel is occluded it 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 [6].

The background can be learned by maximizing the log likelihood $L_b = \sum_{n=1}^{N} \log p(\mathbf{x}^n | j_b^n)$ where

$$p(\mathbf{x}|j_b) = \prod_{i=1}^{P} \alpha_b N(x_i; (T_{j_b}\mathbf{b})_i, \sigma_b^2) + (1 - \alpha_b)U(x_i). \tag{6}$$

The maximization of the likelihood over $(\mathbf{b}, \sigma_b^2)$ can be achieved by using the EM algorithm to deal with the pixel outlier process. For example, the update equation of the background $\mathbf{b}$ is

$$\mathbf{b} \leftarrow \sum_{n=1}^{N} [T_{j_b^n}^T (\mathbf{r}^n(j_b^n) * \mathbf{x}^n)] ./ \sum_{n=1}^{N} [T_{j_b^n}^T \mathbf{r}^n(j_b^n)], \tag{7}$$

where $\mathbf{y} * \mathbf{z}$ and $\mathbf{y} ./ \mathbf{z}$ denote the element-wise product and element-wise division between two vectors $\mathbf{y}$ and $\mathbf{z}$, respectively. In (7) the vector $\mathbf{r}(j_b)$ stores the value

$$r_i(j_b) = \frac{\alpha_b N(x_i; (T_{j_b}\mathbf{b})_i, \sigma_b^2)}{\alpha_b N(x_i; (T_{j_b}\mathbf{b})_i, \sigma_b^2) + (1 - \alpha_b)U(x_i)} \tag{8}$$

for each image pixel i, which is the probability that the ith image pixel is part of the background (and not some outlier due to occlusion) given j_b.

The update for the background appearance $\mathbf{b}$ is very intuitive. For each image $\mathbf{x}^n$, the pixels which are ascribed to non-occluded background (i.e. $r_i^n(j_b^n) \simeq 1$) are transformed by $T_{j_b^n}^T$, which reverses the effect of the transformation by mapping the image $\mathbf{x}^n$ into the larger and stabilized background image $\mathbf{b}$ so that $\mathbf{x}^n$ is located within $\mathbf{b}$ in the position specified by j_b^n. Thus, the non-occluded pixels found in each training image are located properly into the big panorama image and averaged to produce $\mathbf{b}$.

Tracking the background. We now discuss how we can quickly approximate the transformations $\{j_b^1, \ldots, j_b^N\}$ using tracking. To introduce the idea of our tracking algorithm assume that we know the appearance of the background $\mathbf{b}$ as well as the transformation j_b^1 that associates $\mathbf{b}$ with the first frame. Since motion between successive frames is expected to be relatively small we can determine the transformation j_b^2 for the second frame by searching over a small discrete set of neighbouring transformations centered at j_b^1 and inferring the most probable one (i.e. the one giving the highest likelihood given by equation (6), assuming a uniform prior). This procedure can be applied recursively to determine the sequence of transformations in the entire video.

However, the background $\mathbf{b}$ is not known in advance, but we can still apply roughly the same tracking algorithm by suitably initializing and updating the background $\mathbf{b}$ as we process the frames. More specifically, we initialize $\mathbf{b}$ so that the centered part of it will be the first frame $\mathbf{x}^1$ in the sequence. The remaining values of $\mathbf{b}$ take zero values and are considered as yet not-initialized which is indicated by a mask $\mathbf{m}$ of the same size as $\mathbf{b}$ that takes the value 1 for initialized pixels and 0 otherwise. The transformation of the first frame j_b^1 is the identity, which means that the first frame is untransformed. The transformation of the second frame and in general any frame $n+1$, $n \geq 1$, is determined by evaluating the posterior probability

$$R(j_b) \propto \exp\left\{ \frac{\sum_{i=1}^{P} (T_{j_b}\mathbf{m}^n)_i \log p_b(x_i^{n+1}; (T_{j_b}\mathbf{b}^n)_i)}{\sum_{i=1}^{P} (T_{j_b}\mathbf{m}^n)_i} \right\}, \tag{9}$$

over the set of possible j_b values around the neighbourhood of j_b^n. The approximate transformation j_b^{n+1} for the frame is chosen to be $j_b^{n+1} = j_b^*$, where j_b^* maximizes the above posterior probability. Note that (9) is similar to the likelihood (6), with the only difference being that pixels of the background that are not initialized yet are removed from consideration and the score is normalized (by $\sum_{i=1}^{P} (T_{j_b}\mathbf{m}^n)_i$) so that the number of not-yet-initialized pixels (which can vary with j_b) does not affect the total score. Once we know j_b^{n+1}, we use all the frames up to the frame $\mathbf{x}^{n+1}$ (i.e. $\{\mathbf{x}^1, \ldots, \mathbf{x}^{n+1}\}$) to update $\mathbf{b}$ according to equation (7) where the vectors $\mathbf{r}^t(j_b^t)$ with $t = 1, \ldots, n+1$ have been have been updated according to equation (8) for the old value $\mathbf{b}^n$ of the background. The mask $\mathbf{m}$ is also updated so that it always indicates the pixels of $\mathbf{b}$ that are explored so far.

The effect of these updates is that as we process the frames the background model $\mathbf{b}$ is adjusted so that any occluding foreground object is blurred out, revealing the background behind. Having tracked the background, we can then learn its full structure as described earlier in this section.

3.2 Learning the foreground objects

Imagine that the background $\mathbf{b}$ and its most probable transformations in all training images have been approximated. What we wish to do next is to learn the foreground objects. We are going to learn the foreground objects one at each time. Particularly, we assume again that we have approximated the transformations $\{j_\ell^1, \ldots, j_\ell^N\}$ of the ℓth foreground object in all frames. This approximation can be obtained quickly using a tracking algorithm (see later in this section), that is repeatedly applied to the video sequence and each time outputs the transformations associated with a different object.

Learning of the ℓth foreground object will be based on a likelihood model for the images that only accounts for that foreground object and the background, while the presence of the other foreground objects is explained by an outlier process. Particularly,

the other foreground objects can occlude both the ℓth foreground object and the background. Thus, we robustify the foreground and background pixel densities so that the Gaussians in equation (2) are replaced by $p_f(x_i; f_i) = \alpha_f N(x_i; f_i, \sigma_f^2) + (1-\alpha_f)U(x_i)$ and $p_b(x_i; b_i) = \alpha_b N(x_i; b_i, \sigma_b^2) + (1-\alpha_b)U(x_i)$ respectively, where $U(x_i)$ is an uniform distribution in the range of all possible pixel values and α_f and α_b express prior probabilities that a foreground (resp. background) pixel is not occluded. Any time a foreground or background pixel is occluded this can be explained by the uniform component $U(x_i)$.

Based on this robustification, we can learn the parameters associated with all different aspects of the object by maximizing the log likelihood

$$L_\ell = \sum_{n=1}^{N} \log \sum_{v_\ell=1}^{V_\ell} P_{v_\ell} \prod_{i=1}^{P} \left\{ (T_{j_\ell^n} \boldsymbol{\pi}_\ell^{v_\ell})_i p_f(x_i^n; (T_{j_\ell^n} \mathbf{f}_\ell^{v_\ell})_i) + (\mathbf{1} - T_{j_\ell^n} \boldsymbol{\pi}_\ell^{v_\ell})_i p_b(x_i^n; (T_{j_b^n} \mathbf{b})_i) \right\}, \tag{10}$$

where $p_f(x_i^n; (T_{j_\ell^n} \mathbf{f}_\ell^{v_\ell})_i)$ and $p_b(x_i^n; (T_{j_b^n} \mathbf{b})_i)$ have been robustified as explained above. This maximization is carried out by EM where in the E-step the quantities, $Q^n(v_\ell)$, $\mathbf{r}^n(j_\ell)$ and $\mathbf{s}^n(j_\ell)$ are computed as follows. $Q^n(v_\ell)$ denotes the probability $p(v_\ell | x^n, j_b^n, j_\ell^n)$ and is obtained by

$$Q^n(v_\ell) = \frac{P_{v_\ell} p(\mathbf{x}^n | j_b^n, j_\ell^n, v_\ell)}{\sum_{v_\ell=1}^{V_\ell} P_{v_\ell} p(\mathbf{x}^n | j_b^n, j_\ell^n, v_\ell)}, \tag{11}$$

while the vectors $\overline{\mathbf{s}}^n(j_\ell)$ and $\mathbf{r}_i^n(j_1^n)$ store the values

$$\overline{s}_i^n(v_\ell) = \frac{(T_{j_\ell^n} \boldsymbol{\pi}_\ell^{v_\ell})_i p_{f_\ell}(x_i^n; (T_{j_\ell^n} \mathbf{f}_\ell^{v_\ell})_i)}{(T_{j_\ell^n} \boldsymbol{\pi}_\ell^{v_\ell})_i p_{f_\ell}(x_i^n; (T_{j_\ell^n} \mathbf{f}_\ell^{v_\ell})_i) + (\mathbf{1} - T_{j_\ell^n} \boldsymbol{\pi}_\ell^{v_\ell})_i p_b(x_i^n; (T_{j_b^n} \mathbf{b})_i)}, \tag{12}$$

and

$$r_i^n(v_\ell) = \frac{\alpha_f N(x_i^n; (T_{j_1^n} \mathbf{f}_\ell^{v_\ell})_i, \sigma_1^2)}{\alpha_f N(x_i^n; (T_{j_1^n} \mathbf{f}_\ell^{v_\ell})_i, \sigma_1^2) + (1-\alpha_f) U(x_i^n)}, \tag{13}$$

for each image pixel i. In the M-step we update the parameters $\{\mathbf{f}_\ell^{v_\ell}, \boldsymbol{\pi}_\ell^{v_\ell}, \sigma_{\ell,v_\ell}^2\}_{v_\ell=1}^{V_\ell}$. For example the updates of $\boldsymbol{\pi}_\ell^{v_\ell}$ and $\mathbf{f}_\ell^{v_\ell}$ are

$$\boldsymbol{\pi}_\ell^{v_\ell} \leftarrow \sum_{n=1}^{N} Q^n(v_\ell) T_{j_\ell^n}^T [\overline{\mathbf{s}}^n(v_\ell)] . / \sum_{n=1}^{N} Q^n(v_\ell) [T_{j_\ell^n}^T \mathbf{1}], \tag{14}$$

$$\mathbf{f}_\ell^{v_\ell} \leftarrow \sum_{n=1}^{N} Q^n(v_\ell) T_{j_\ell^n}^T [\overline{\mathbf{s}}^n(v_\ell) * \mathbf{r}^n(v_\ell) * \mathbf{x}^n] . / \sum_{n=1}^{N} Q^n(v_\ell) T_{j_\ell^n}^T [\overline{\mathbf{s}}^n(v_\ell) * \mathbf{r}^n(v_\ell)]. \tag{15}$$

The above updates are very intuitive. Consider, for example, the appearance $\mathbf{f}_\ell^{v_\ell}$. For pixels which are ascribed to the ℓth foreground and are not occluded (i.e. $(\mathbf{s}^n(v_\ell) * \mathbf{r}^n(v_\ell))_i \simeq 1$), the values in $\mathbf{x}^n$ are transformed by $T_{j_\ell^n}^T$ which reverses the effect of the transformation. This allows the foreground pixels found in each training image to be mapped in a stabilized frame and then be averaged (weighted by the viewpoint posterior probabilities $Q^n(v_\ell)$) to produce $\mathbf{f}_\ell^{v_\ell}$.

Tracking the foreground objects. The appearance of each foreground object can vary significantly through the video due to large pose changes. Thus, our algorithm should be able to cope with such variation. Below we describe a tracking algorithm that each time matches a mask $\boldsymbol{\pi}_\ell$ and appearance $\mathbf{f}_\ell$ to the current frame. Large viewpoint

variation is handled by on-line updating $\boldsymbol{\pi}_\ell$ and $\mathbf{f}_\ell$ so that each time they will fit the shape and appearance of the object in the current frame.

We first discuss how to track the first foreground object, so we assume that $\ell = 1$. The pixels which are explained by the background in each image $\mathbf{x}^n$ are flagged by the background responsibilities $\mathbf{r}^n(j_b^n)$ computed according to equation (8). Clearly, the mask $\overline{\mathbf{r}}^n(j_b^n) = \mathbf{1} - \mathbf{r}^n(j_b^n)$ roughly indicates all the pixels of frame $\mathbf{x}^n$ that belong to the foreground objects. By focusing only on these pixels, we wish to start tracking one of the foreground objects through the entire video sequence and ignore for the moment the rest foreground objects.

Our algorithm tracks the first object by matching to the current frame and then updating in an on-line fashion a mask $\boldsymbol{\pi}_1$ and appearance $\mathbf{f}_1$ of that object. The mask and the appearance are initialized so that $\boldsymbol{\pi}_1^1 = \mathbf{0.5} * \overline{\mathbf{r}}^1(j_b^1)$ and $\mathbf{f}_1^1 = \mathbf{x}^1$, where $\mathbf{0.5}$ denotes the vector with 0.5 values[1]. Due to this initialization we know that the first frame is untransformed, i.e. j_1^1 is the identity transformation. To determine the transformation of the second frame and in general the transformation j_1^{n+1}, with $n \geq 1$, of the frame $\mathbf{x}^{n+1}$ we evaluate the posterior probability

$$R(j_1) \propto \exp\left\{ \sum_{i=1}^{P} (\mathbf{w}_1^{n+1})_i \log\Big((T_{j_1}\boldsymbol{\pi}_1^n)_i \times \right.$$
$$\left. p_f(x_i^{n+1}; (T_{j_1}\mathbf{f}_1^n)_i) + (\mathbf{1} - T_{j_1}\boldsymbol{\pi}_1^n)_i (1-\alpha_b) U(x_i^{n+1}) \Big) \right\}, \quad (16)$$

where $p_f(x_i^{n+1}; (T_{j_1^{n+1}}\mathbf{f}_1^n)_i)$ is robustified as explained earlier, j_1 takes values around the neighbourhood of j_1^n and $\mathbf{w}_1^{n+1} = \overline{\mathbf{r}}^{n+1}(j_b^{n+1})$. $R(j_1)$ measures the goodness of the match at those pixels of frame $\mathbf{x}^{n+1}$ which are not explained by the background. Note that as the objects will, in general, be of different sizes, the probability $R(j_1)$ over the transformation variable will have greater mass on transformations relating to the largest object. The transformation j_1^{n+1} is set to be equal to j_1^*, where j_1^* maximizes the above posterior probability. Once we determine j_1^{n+1} we update both the mask $\boldsymbol{\pi}_1$ and appearance $\mathbf{f}_1$. The mask is updated according to

$$\boldsymbol{\pi}_1^{n+1} = \left(\beta \boldsymbol{\pi}_1^n + T_{j_1^{n+1}}^T [\overline{\mathbf{s}}^{n+1}] \right) ./ \left(\beta \mathbf{1} + T_{j_1^{n+1}}^T [\mathbf{1}] \right), \quad (17)$$

where β is a positive number. The vector $\overline{\mathbf{s}}^{n+1}$ expresses a soft segmentation of the object in the frame $\mathbf{x}^{n+1}$ and is computed similarly to equation (12). The update (17) defines the new mask as a weighted average of the stabilized segmentation in the current frame (i.e. $T_{j_1^{n+1}}^T[\overline{\mathbf{s}}^{n+1}(j_1^{n+1})]$) and the current value of the mask. β determines the relative weight between these two terms. In all our experiments we have set $\beta = 0.5$. Similarly, the update for the foreground appearance $\mathbf{f}_1$ is given by

$$\mathbf{f}_1^{n+1} = \left(\beta \mathbf{f}_1^n + T_{j_1^{n+1}}^T [\overline{\mathbf{s}}^{n+1} * \mathbf{r}^{n+1} * \mathbf{x}^{n+1}] \right) ./ \left(\beta \mathbf{1} + T_{j_1^{n+1}}^T [\overline{\mathbf{s}}^{n+1} * \mathbf{r}^{n+1}] \right). \quad (18)$$

The vector $\mathbf{r}^{n+1}$ is defined similarly to equation (13). Again the above update is very intuitive. For pixels which are ascribed to the foreground (i.e. $\overline{\mathbf{s}}^{n+1} * \mathbf{r}^{n+1} \simeq 1$), the values in $\mathbf{x}^{n+1}$ are transformed by $T_{j_1^{n+1}}^T$ into the stabilized frame which allows the

[1] The value of 0.5 is chosen to express our uncertainty about whether these pixels will ultimately be in the foreground mask or not.

foreground pixels found in the current frame to be averaged with the old value $\mathbf{f}^n$ in order to produce $\mathbf{f}^{n+1}$. Note that the updates given by (17) and (18) are on-line versions of the respective batch updates of the EM algorithm for maximizing the log likelihood (6) assuming that $V_1 = 1$.

The above tracking algorithm is a modification of the method presented in [5] with the difference that the batch updates of $(\mathbf{f}_1, \boldsymbol{\pi}_1)$ used there have been replaced by on-line counterparts that allow tracking the object when the appearance can significantly change from frame to frame.

Once the first object has been tracked we learn the different viewpoint models for that object by maximizing (10). When these models has been learned we can go through the images to find which pixels are explained by this object. Then we can remove these pixels from consideration by properly updating each $\mathbf{w}^n$ vector which allows tracking a different object on the next stage. Particularly, we compute the vector $\boldsymbol{\rho}_1^n = \sum_{v_1=1}^{V_1} Q^n(v_1)(T_{j_1^n}\boldsymbol{\pi}_1^{v_1}) * \mathbf{r}^n(v_1^n)$. $\boldsymbol{\rho}_1^n$ will 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. We can now run the same tracking algorithm again by updating $\mathbf{w}_{\ell+1}^n$ $(\ell \geq 1)$ as by $\mathbf{w}_{\ell+1}^n = (\mathbf{1} - \boldsymbol{\rho}_\ell^n) * \mathbf{w}_\ell^n$ which allows tracking a different object on the $\ell + 1$th iteration. Note also that the new mask $\boldsymbol{\pi}_{\ell+1}$ is initialized to $\mathbf{0.5} * \mathbf{w}_{\ell+1}^n$ while the appearance $\mathbf{f}_{\ell+1}$ is always initialized to the first frame $\mathbf{x}^1$.

3.3 Specification of the occlusion ordering and refinement of the object models

Once we run the greedy algorithm (`Stage 1` in Table 1), we obtain an estimate of all model parameters, an approximation of the object transformation in each training image as well as the probabilities $Q^n(v_\ell)$ which express our posterior belief that image $\mathbf{x}^n$ was generated by the view v_ℓ of model ℓ. Using now these quantities we wish to compute the occlusion ordering of the foreground objects in each training image. This is necessary since even when the occlusion ordering remains fixed across all video frames, the algorithm might not extract the objects in accordance with this ordering, i.e. discovering first the nearest object to the camera, then the second nearest object etc. The order the objects are found is determined by the tracking algorithm and typically the largest objects that occupy more pixels than others are more likely to be tracked first.

A way to infer the occlusion ordering of the foreground objects or layers in an image $\mathbf{x}^n$ is to consider all possible permutations of these layers and choose the permutation that gives the maximum likelihood. The simplest case is to have two foreground objects with parameters $\{\boldsymbol{\pi}_1^{v_1}, \mathbf{f}_1^{v_1}, \sigma_{1,v_1}^2\}_{v_1=1}^{V_1}$ and $\{\boldsymbol{\pi}_2^{v_2}, \mathbf{f}_2^{v_2}, \sigma_{2,v_2}^2\}_{v_2=1}^{V_2}$, respectively. From the posterior probabilities $Q^n(v_1)$ and $Q^n(v_2)$ corresponding to image $\mathbf{x}^n$ we choose the most probable views v_1^n and v_2^n. Conditioned on these estimated views as well as the transformations, the log likelihood values of the two possible orderings are

$$L_{kl}^n = \sum_{i=1}^{P} \log\Big\{ (T_{j_k^n}\boldsymbol{\pi}_k^{v_k^n})_i p_{f_k}(x_i^n; (T_{j_k^n}\mathbf{f}_k^{v_k^n})_i) + (\mathbf{1} - T_{j_k^n}\boldsymbol{\pi}_k^{v_k^n})_i \times$$
$$[(T_{j_l^n}\boldsymbol{\pi}_l^{v_l^n})_i p_{f_l}(x_i^n; (T_{j_l^n}\mathbf{f}_l^{v_l^n})_i) + (\mathbf{1} - T_{j_l^n}\boldsymbol{\pi}_l^{v_l^n})_i p_b(x_i^n; (T_{j_b^n}\mathbf{b})_i)]\Big\}, \qquad (19)$$

where $k = 1,\ l = 2$ or $k = 2,\ l = 1$. The selected occlusion ordering for the image $\mathbf{x}^n$ is the one with the largest log likelihood. When we have L foreground objects we work exactly analogously as above by expressing all $L!$ permutations of the foreground layers and selecting the one with the largest likelihood.

The above computation of the occlusion ordering takes $L!$ time and it can be used only when we have few foreground objects. However, in most of the cases we can further speed up this computation and estimate the occlusion ordering for large number of objects. The idea is that an object ℓ usually does not overlap (either occludes or is occluded) with all the other $L-1$ objects, but only with some of them. Thus, if for each object we identify the overlapping objects, the complexity in the worse case will be $O(G!)$ where G is the largest number of objects that simultaneously overlap with each other. Details of this algorithm together with illustrative examples are given in section B.3 in [7].

Once the occlusion ordering has been specified for each training image, we can maximize the complete log likelihood for the model described in section 2 (using the approximated transformations, viewpoints and the occlusion orderings) and refine the appearances and masks of the objects. Note that for this maximization we need the EM algorithm in order to deal with the fact that each pixel follows a $L+1$-component mixture distribution (for $L=2$ see equation (4)). However, this EM runs quickly since all the transformations, viewpoints and occlusion orderings have been "filled in" with the approximated values provided at previous stages of the learning process.

4 Related work

There is a huge literature on motion analysis and tracking in computer vision, and there is indeed much relevant prior work. Particularly, Wang and Adelson [1] estimate object motions in successive frames and track them through the sequence by computing optical flow vectors, fit affine motion models to these vectors, and then cluster the motion parameters into a number of objects using k-means. Darrell and Pentland [8], and Sawhney and Ayer [9] used similar approaches based on optical flow estimation between successive frames and apply the MDL principle for selecting the number of objects. Note that a major limitation of optical-flow based methods concerns regions of low texture where flow information can be sparse, and when there is large inter-frame motion. The method of Irani et al. [2] is much more relevant to ours. They do motion estimation using optical flow by matching the current frame against an accumulative appearance image of the tracked object. The appearance of a tracked object develops though time, although they do not take into account issues of occlusion, so that if a tracked object becomes occluded for some frames, it may be lost.

The work of Tao et al. [10] is also relevant in that it deals with a background model and object models defined in terms of masks and appearances. However, note that in their work the mask is assumed to be of elliptical shape (parameterised as a Gaussian) rather than a general mask. The mask and appearance models are dynamically updated. However, the initialization of each model is handled by a "separate module", and is not obtained automatically. For the aerial surveillance example given in the paper initialization of the objects can be obtained by simple background subtraction, but that is not sufficient for the examples we consider. Later work by Jepson et al. [11] uses a *polybone* model for the mask instead of the Gaussian, but this still has limited representational capacity in comparison to our general mask. Jepson et al. also use more complex tracking methods which include the birth and death of polybones in time, as well as temporal tracking proposals.

The idea of focusing search when carrying our transformation-invariant clustering has also been used before, e.g. by Fitzgibbon and Zisserman [12] in their work on automatic cast listing of movies. However, in that case prior knowledge that faces were

being searched for meant that a face detector could be run on the images to produce candidate locations, while this is not possible in our case as we do not know what objects we are looking for apriori.

As well as methods based on masks and appearances, there are also feature-based methods for tracking objects in image sequences, see e.g. [13], [14]. These attempt to track features through a sequence and cluster these tracks using different motion models. Allan et al. [15] describe a feature-based algorithm for computing the transformations of multiple objects in a video by simultaneously clustering features of all frames into objects. The obtained transformations are then used to learn a layered generative model for the video.

Currently in our method we ignore the spatial continuity in the segmentation labels of the pixels. This might result in noisy segmentations at some cases. A method for learning layers that incorporates spatial continuity has been recently considered by [14] and [16]. They use a layered model with a MRF prior for the pixel labels and make use of the graph cuts algorithm for efficient updating of the masks. Note that within our framework we could also incorporate a MRF for the pixel labels at the cost of increased computation.

Finally, the mechanism for dealing with multiple viewpoints using mixture models has been considered before in [17]. However, in this work they consider one object present in the images against a cluttered background, and only the appearance images of different poses of the object are learned (not masks). Also they do not apply tracking and they consider a global search over transformations. In contrast, our method can be applied to images with multiple objects and learns the background as well as different poses for the foreground objects. An important aspect of our method is the use of tracking (applied prior to learning) which stabilizes an object and then efficiently learns its views.

5 Experiments

We consider four video sequences: the Frey-Jojic (FJ) sequence (Figure 1) available from `http://www.psi.toronto.edu/layers.html`, the arms-torso video sequence showing a moving human upper body (Figure 2), the man-walking sequence (Figure 3) and the Groundhog day van sequence[2] (Figure 4). We will also assume that the number of different views that we wish to learn for each foreground object is known.

The FJ sequence consists of 44 118×248 images (excluding the black border). This sequence can be well modelled by assuming a single view for each of the foreground objects, thus we set $V = 1$ for both objects. During tracking we used a 15×15 window of translations in units of one pixel during the tracking stage. The learning stage requires EM which converged in about 30 iterations. Figure 1a shows the evolution of the initial mask and appearance ($t = 1$) through frames 10 and 20 as we track the first object (Frey). Notice that as we process the frames the mask focuses on only one of the two objects and the appearance remains sharp only for this object. The real running time of our MATLAB implementation for processing the whole sequence was 3 minutes. The computer used for all the experiments reported here was a 3GHz Pentium. Figure 1b shows the results after `Stage 1` of the algorithm is completed. Figure 1c shows the final appearances of the foreground objects after the computation of the occlusion ordering and the joint refinement of all the parameters. Comparing Figure 1b with Figure 1c,

[2] We thank the Visual Geometry Group at Oxford for providing the Groundhog day van data.

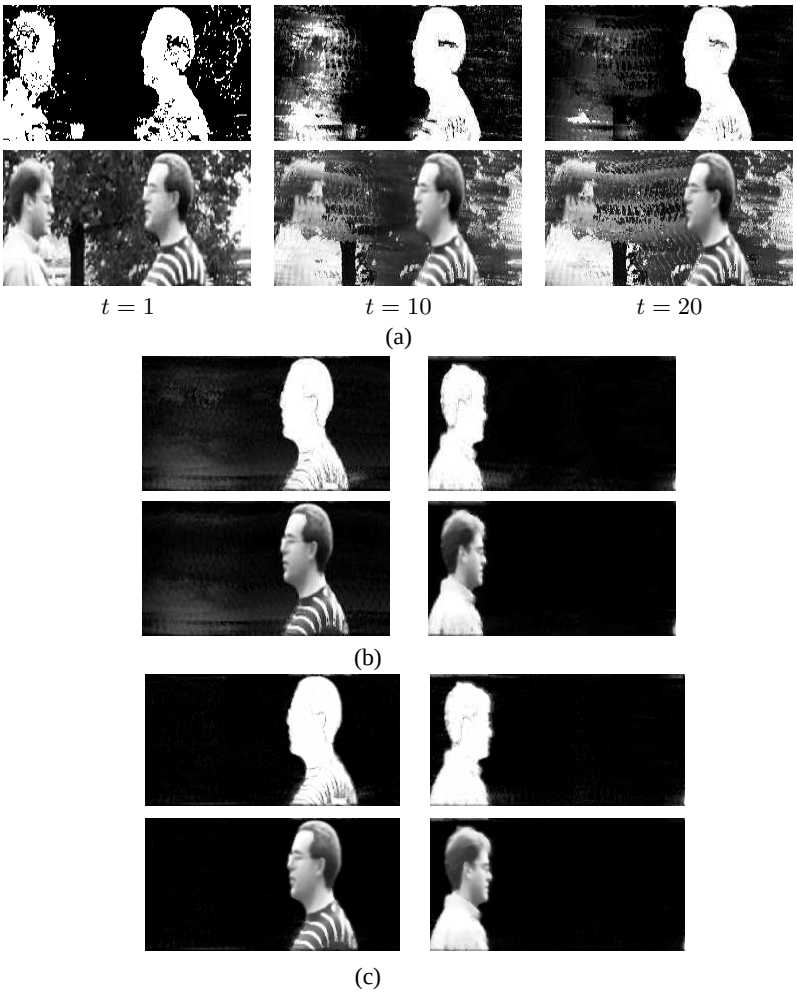
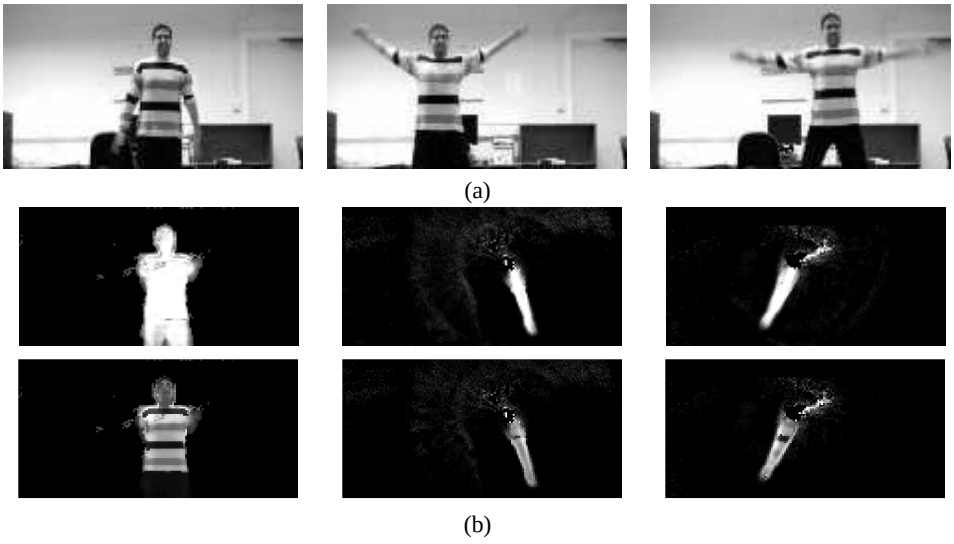


Fig. 1. Panel (a) shows the evolution of the mask $\boldsymbol{\pi}_1$ (top row) and the appearance $\mathbf{f}_1$ (bottom row) at times 1, 10 and 20 as we track the first object (Frey). Again notice how the mask becomes focused on one of the objects (Frey) and how the appearance remains clear and sharp only for Frey. Panel (b) shows the mask and the element-wise product of the mask and appearance model ($\boldsymbol{\pi} * \mathbf{f}$) learned for Frey (first column from the left) and Jojic (second column) using the greedy algorithm (after `Stage 1`; see Table 1). Panel (c) displays the corresponding masks and appearances of the objects after the refinement step.



(a)

(b)

Fig. 2. Panel (a) shows three frames of the arms-torso video sequence. Panel (b) displays the masks and appearances of the parts of the arms-torso video sequence. Particularly, the plots in the first column show the learn mask (top row) and the element-wise product of the mask and appearance (bottom row) for the head/torso. Any pair of panels in the other two columns provides the same information for the two arms.

Fig. 3. The panels in the first row show three frames of the man-walking sequence. The panels in the last two rows show the element-wise product of the mask (thresholded to 0.5) and appearance (showing against a grey background) for all six viewpoint models.

we can visually inspect the improvement over the appearances of the objects, e.g. the ghosts in the masks of Figure 1b have disappeared in Figure 1c.

When we carry out the refinement step, we always initialize the background and the foreground appearances and masks using the values provided by the greedy algorithm. The variances are reinitialized to a large value to help escape from local maxima. Note also that for this maximization we maintain the robustification of the background and foreground pixel densities (α_b and α_f are set to 0.9) in order to deal with possible variability of the objects, e.g. local clothing deformation, or changes of the lighting conditions. The EM algorithm used for the above maximization converges in few iterations (e.g. less than 20). This is because the objects' appearances obtained from the greedy algorithm are already close to their final values, and all the transformations of the objects have been "filled in" with the approximated values provided by the greedy algorithm.

We demonstrate our method for learning parts of human body using the arms-torso sequence that consists of 79 76×151 images. Three frames of this sequence are shown in Figure 2a. To learn the articulated parts we use translations and rotations so that the transformation matrix T_{j_ℓ} that applies to $\boldsymbol{\pi}_\ell$ and $\mathbf{f}_\ell$ implements an combination of translation and rotation. We implemented this using the MATLAB function *tformarray.m* and nearest neighbour interpolation. Note that we set the number of views $V_\ell = 1$ for all foreground objects. The tracking method searches over a window of 10×10 translations and 15 rotations (at 2^o spacing) so that it searches over 1500 transformations in total. Figures 2b shows the three parts discovered by the algorithm i.e. the head/torso and the two arms. Note that the ambiguity of the masks and appearances around the joints of the two arms with the torso which is due to the deformability of the clothing in these areas. The total real running time for learning this sequence was roughly one hour. Note that when we learn object parts we should also learn a joint distribution over the parts; a method for computing such a distribution is described in [7].

The man-walking sequence consists of 85 72×176 colour images. Figure 3 displays three frames of that sequence and also the learned visual aspects (the element-wise product of each appearance and mask pair). We assumed that the number of different views is six, i.e. $V_1 = 6$. When we applied the tracking algorithm we used a window of 15×15 translations in units of one pixel. Processing the whole video took 5 minutes.

In our fourth experiment, we used 46 144×176 frames of the Groundhog day van sequence. Figure 4a displays three frames of that sequence. During tracking we assumed a window of 15×15 translations in units of one pixel, plus 5 scalings for each of the two axes spaced at a 1% change in the image size, and 5 rotations at 2^o spacing. We assumed that the number of different views of the foreground object that we wish to learn is three, i.e. $V_1 = 3$. Figure 4b shows the learned prior mask ($\boldsymbol{\pi}_1^{v_1}$) and the element-wise product of the appearance ($\mathbf{f}_1^{v_1}$) and the mask for each view. Figure 4c shows the background that is also learned. Clearly, each appearance has modelled a different view of the van. However, the masks are a bit noisy; we believe this could be improved by using spatial continuity constraints. Processing the whole video took 6 hours, where the most of the time was spent during tracking.

6 Discussion

Above we have presented a general framework for learning a layered model from a video sequence. The important feature of this method is tracking the background and the

Fig. 4. Panel (a) shows three frames of the van sequence. Panel (b) shows the pairs of mask and the element-wise product of the mask and appearance (showing against a grey background) for all different viewpoints. Note that the element-wise products are produced by making the masks binary (thresholded to 0.5). Panel (c) displays the background.

foreground objects sequentially so as to deal with one object and the associated transformations at each time. Additionally, we have combined this tracking method with allowing multiple views for each foreground object so as to deal with large viewpoint variation. These models are learned using a mixture modelling approach. Tracking the object before knowing its full structure allows for efficient learning of the object viewpoint models.

Some issues for the future are to automatically identify how many views are needed to efficiently model the appearance of each object, to determine the number of objects, and to deal with objects/parts that have internal variability. Another issue is to automatically identify when a detected model is a part or an independent object. This might be achieved by using a mutual information measure, since we expect parts of the same object to have significant statistical dependence.

Acknowledgements

This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

1. J. Y. A. Wang and E. H. Adelson. Representing Moving Images with Layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.
2. M. Irani, B. Rousso, and S. Peleg. Computing Occluding and Transparent Motions. *International Journal of Computer Vision*, 12(1):5–16, 1994.
3. N. Jojic and B. J. Frey. Learning Flexible Sprites in Video Layers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2001*. IEEE Computer Society Press, 2001. Kauai, Hawaii.
4. C. K. I. Williams and M. K. Titsias. Greedy Learning of Multiple Objects in Images using Robust Statistics and Factorial Learning. *Neural Computation*, 16(5):1039–1062, 2004.
5. M. K. Titsias and C. K. I. Williams. Fast unsupervised greedy learning of multiple objects and parts from video. In *Proc. Generative-Model Based Vision Workshop*, 2004.
6. M.J. Black and A.D. Jepson. EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation. *Proc. ECCV*, pages 329–342, 1996.
7. M. K. Titsias. Unsupervised Learning of Multiple Objects in Images. PhD thesis, School of Informatics, University of Edinburgh, 2005.
8. T. Darrell and A. P. Pentland. Cooperative Robust Estimation Using Layers of Support. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):474–487, 1995.
9. H. S. Sawhney and S Ayer. Compact Representations of Videos Through Dominant and Multiple Motion Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):814–830, 1996.
10. H. Tao, H. S. Sawhney, and R. Kumar. Dynamic Layer Representation with Applications to Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages II:134–141, 2000.

11. A. D. Jepson, D. J. Fleet, and M. J. Black. A Layered Motion Representation with Occlusion and Compact Spatial Support. In *Proceedings of the Seventh European Conference on Computer Vision, ECCV 2002*, pages I 692–706. Springer, 2002. LNCS 2353.
12. A. Fitzgibbon and A. Zisserman. On Affine Invariant Clustering and Automatic Cast Listing in Movies. In *Proceedings of the Seventh European Conference on Computer Vision, ECCV 2002*, pages III 304–320. Springer, 2002. LNCS2353.
13. P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Roy. Soc. Lond. A*, 356:1321–1340, 1998.
14. J. Wills, S. Agarwal, and S. Belongie. What Went Where. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2003*, pages I:37–44, 2003.
15. M. Allan, M. K. Titsias, and C. K. I. Williams. Fast Learning of Sprites using Invariant Features. In *Proceedings of the British Machine Vision Conference 2005*, pages 40–49. 2005.
16. M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered pictorial structures from video. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 158–163, 2004.
17. B. J. Frey and N. Jojic. Transformation Invariant Clustering Using the EM Algorithm. *IEEE Trans Pattern Analysis and Machine Intelligence*, 25(1):1–17, 2003.