Fast Unsupervised Greedy Learning of Multiple Objects and Parts 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

Abstract

Williams and Titsias (2004) have shown how to carry out unsupervised greedy learning of multiple objects from images (GLOMO), building on the work of Jojic and Frey (2001). In this paper we show that the earlier work on GLOMO can be greatly speeded up for video sequence data by carrying out approximate tracking of the multiple objects in the scene. Our method is applied to raw image sequence data and extracts the objects one at a time. First, the moving background is learned, and moving objects are found at later stages. The algorithm recursively updates an appearance model of the tracked object so that possible occlusion of the object is taken into account which makes tracking stable. We apply this method to learn multiple objects in image sequences as well as articulated parts of the human body.

1. Introduction

We are given as input a set of images containing views of multiple objects, and wish to learn appearance-based models of each of the objects. Over the last decade or so a layer-based approach to this problem has become popular, where each object is modelled in terms of its appearance and region of support, see e.g. Wang and Adelson (1994), Irani et al. (1994). Jojic and Frey (2001) provided a principled generative probabilistic framework for this task, 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 this formulation is 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 that there are J possible values that the instantiation parameters of any one object can take on, then we will need to consider $O(J^L)$ combinations to explain any image. Jojic and Frey (2001) tackled this problem by using a variational inference scheme, searching over all instantiation parameters simultaneously. In contrast, Williams and Titsias (2004) developed a *sequential* approach to object discovery whereby each model is extracted in turn from the whole dataset using a robust statistical method. We denote this latter method greedy learning of multiple objects from images, or GLOMO. A fuller description of the GLOMO framework is given in section 2.

Both the method of Jojic and Frey (2001) and the GLOMO algorithm work on unordered sets of images. In this case training can be very slow as it is necessary to search over all possible instantiation parameters of at least a single object (in the GLOMO case) on every image. However, for video sequences we could considerably speed up the training by first tracking the objects before knowing their full structure. Tracking 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 neighborhood of these transformations or without search at all when the approximation is accurate. We have developed an algorithm that works in conjunction with the GLOMO method so that the stage where the GLOMO algorithm learns an object (by assuming unordered images) is now speeded up by applying first tracking and then learning based on a focused search. First, the moving background is tracked and learned, and moving foreground objects are found at later stages. The tracking algorithm itself recursively updates an appearance model of the tracked object so that occlusion is taken into account and approximates the transformations by matching this model to the frames through the sequence. This provides accurate transformations, e.g. in the experiments in section 5, we learn the objects without search by using only the transformations found by the tracking algorithm and obtain good results. Updating a model of the tracked object in a robust way enables also the algorithm to recover when it loses the track e.g. due to occlusion of the object.

A second contribution of this paper is that we learn articulated parts of the human body from video data using unsupervised learning. We apply the same algorithm for learning multiple objects in order to learn articulated parts, so that the parts are first tracked and then have their full structure learned. In section 5 we assume that parts can undergo translations and rotations and we extract three parts (the two arms and the head/torso) from a video sequence of the upper body.

The structure of the remainder of the paper is as follows:

In section 2 we describe the method of greedy learning of multiple objects. In section 3 we discuss the training algorithm based on tracking. Section 4 discusses unsupervised learning of parts and section 5 gives experimental results. We conclude with a discussion in section 6.

2. Greedy Learning of Objects

We consider a $P_x \times P_y$ image as a vector $\mathbf{x}$ of length P, where $P = P_x P_y$. Let there be L foreground objects in front of a background. Our goal is to learn an appearance model $\mathbf{f}_\ell$ and a mask $\boldsymbol{\pi}_\ell$ for each object ℓ, and an appearance model $\mathbf{b}$ for the background. Note that $\boldsymbol{\pi}_\ell$ and $\mathbf{f}_\ell$ are P-length vectors while $\mathbf{b}$ can be larger than the data image size if there is camera motion during the sequence. The mask defines the extent or support of each object: each entry in the mask $\boldsymbol{\pi}_\ell$ takes on a value in $[0, 1]$ which indicates the probability that this pixel is part of the ℓth object. (The background does not need a mask as it is taken to be the vector of ones.). Each object can undergo some transformation operations, such as translation, rotation, scaling etc., so we introduce a transformation variable j_ℓ for each object with a transformation matrix T_{j_ℓ} so that e.g. $T_{j_\ell}\mathbf{f}_\ell$ is the transformed appearance of the ℓth object. We consider also the case where the background can move, for example when a moving camera captures a sequence of frames, so there is also a transformation j_b. A static background is a special case of the moving background.

Below we first describe the generative model for multiple objects and then discuss how we can tractably train this model by using a greedy learning algorithm that extracts the objects one after the other. The description in this section largely follows Williams and Titsias (2004).

2.1. Generative model for multiple objects

We first consider the case when there is only one foreground object with appearance $\mathbf{f}$ and mask $\boldsymbol{\pi}$. Ignoring for a moment the effect of transformations we would have

$$p(x_i) = \pi_i p_f(x_i; f_i) + (1 - \pi_i) p_b(x_i; b_i) \qquad (1)$$

where $p_f(x_i; f_i) = N(x_i; f_i, \sigma_f^2)$ and $p_b(x_i; b_i) = N(x_i; b_i, \sigma_b^2)$ where $N(x; \mu, \sigma^2)$ denotes a Gaussian distribution over x with mean μ and variance σ^2. Now assume that the transformation space for the foreground object has been discretized to J_f values, indexed by j_f. Application of this transformation to the mask gives a transformed mask $T_{j_f}\boldsymbol{\pi}$, and similarly for $\mathbf{f}$. Similarly the background transformation can take on J_b values indexed by j_b. Thus

$$p(\mathbf{x}|j_f, j_b) = \prod_{i=1}^{P} [(T_{j_f}\boldsymbol{\pi})_i p_f(x_i; (T_{j_f}\mathbf{f})_i) + (\mathbf{1} - T_{j_f}\boldsymbol{\pi})_i p_b(x_i; (T_{j_b}\mathbf{b})_i)], \qquad (2)$$

where $\mathbf{1}$ denotes the vector with ones. The likelihood of an image $\mathbf{x}$ is $p(\mathbf{x}) = \sum_{j_f=1}^{J_f} \sum_{j_b=1}^{J_b} P_{j_f} P_{j_b} p(\mathbf{x}|j_f, j_b)$ where P_{j_b} and P_{j_f} are uniform prior probabilities.

The above generative model assumes one foreground and one background layer, however it can be easily extended to include L foreground objects assigned to L depth layers, see Williams and Titsias (2004) for more details.

Given a data set $\{\mathbf{x}^n\}$, $n = 1, \ldots, N$ we can adapt the parameters $\theta = (\mathbf{f}_1, \boldsymbol{\pi}_1, \sigma_1^2, \ldots, \mathbf{f}_L, \boldsymbol{\pi}_L, \sigma_L^2, \mathbf{b}, \sigma_b^2)$ 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 requires a search over $J_f^L J_b$ possibilities which is infeasible even in case of a single foreground object and a moving background, and gets exponentially worse as more objects are added. To deal with this problem Jojic and Frey (2001) used a variational inference scheme, searching over all instantiation parameters simultaneously. In contrast the greedy training algorithm of Williams and Titsias (2004) (described below) deals separately with each transformation by learning first the background and then the foreground objects.

2.2. Learning the background

At this stage we consider images containing a background and many foreground objects. However, we concentrate on learning only the background and regarding the foreground objects as outliers (at this stage). This goal can be achieved by *robustifying* the background model described above so that occlusion can be tolerated.

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 object 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 (Black and Jepson, 1996).

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

$$p(\mathbf{x}|j_b) = \prod_{i=1}^{P} p_b(x_i; (T_{j_b}\mathbf{b})_i) \qquad (3)$$

and $p_b(x_i; (T_{j_b}\mathbf{b})_i)$ has been robustified as explained above. The maximization of the likelihood over $(\mathbf{b}, \sigma_b^2)$ can be achieved by using the EM algorithm and searching over J_b transformations of the background. For example, the update

equation of the background $\mathbf{b}$ is

$$\mathbf{b} \leftarrow \sum_{n=1}^{N}\sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)[T_{j_b}^T(\mathbf{r}^n(j_b) * \mathbf{x}^n)]./ \sum_{n=1}^{N}\sum_{j_b=1}^{J_b} P(j_b|\mathbf{x}^n)[T_{j_b}^T\mathbf{r}^n(j_b)], \quad (4)$$

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 (4) the quantity $P(j_b|\mathbf{x})$ is the posterior probability of the transformation given the image $\mathbf{x}$ and the vector $\mathbf{r}(j_b)$ stores the value

$$r_i(j_b) = \frac{\alpha_b N(x_i; (T_{j_b}\mathbf{b})_i, \sigma_b^2)}{\alpha_b N(x_i; (T_{j_b}\mathbf{b})_i, \sigma_b^2) + (1-\alpha_b)U(x_i)} \quad (5)$$

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 example consider the case when j_b represents translations and for the training image $\mathbf{x}^n$ $P(j_b|\mathbf{x}^n) = 1$ for $j_b = j_b^n$ and 0 otherwise. For pixels which are ascribed to non-occluded background (i.e. $r_i^n(j_b^n) \simeq 1$) the values of $\mathbf{x}^n$ are transformed by $T_{j_b^n}^T$ which maps the $P_x \times P_y$ image $\mathbf{x}^n$ into a larger image of the size of $\mathbf{b}$ so that $\mathbf{x}^n$ is located in the position specified by j_b^n 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 $\mathbf{b}$.

2.3. Learning the foreground objects

Imagine that the background $\mathbf{b}$ and the most probable transformations $\{j_b^n\}$ in all training set images are known. What we wish to do next is to learn the first foreground object and ignore the rest objects (at this stage).

Since multiple objects can exist in our images, a different object from the one being modelled may be interposed between the foreground object we model and the camera, so that we again have a mixture model $p_f(x_i; f_i) = \alpha_f N(x_i; f_i, \sigma_f^2) + (1-\alpha_f)U(x_i)$ where α_f is the fraction of times a foreground pixel is not occluded. Having made this robustification, the model described with one foreground object plus a background (which is already known) can be trained using EM and maximizing (7) with respect to only this object.

A second foreground object is learned by removing those pixels explained by the first foreground object. On image $\mathbf{x}^n$ we infer transformation j_1^n, and at pixel i we obtain the posterior probability (or responsibility)

$$r_i^n(j_1^n) = \frac{\alpha_f N(x_i^n; (T_{j_1^n}\mathbf{f}_1)_i, \sigma_1^2)}{\alpha_f N(x_i^n; (T_{j_1^n}\mathbf{f}_1)_i, \sigma_1^2) + (1-\alpha_f)U(x_i^n)}, \quad (6)$$

and compute $\boldsymbol{\rho}_1^n = (T_{j_1^n}\boldsymbol{\pi}_1) * \mathbf{r}^n(j_1^n)$. $\boldsymbol{\rho}_1^n$ 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. Thus we construct a re-weighted objective function involving $\boldsymbol{\rho}_1^n$ and run the robust learning again. This procedure is then repeated, removing more pixels as more objects are learned.

Formally the greedy algorithm maximizes a lower bound on the log likelihood of the full layer-based probabilistic generative model for images with multiple objects. The algorithm is described in detail in Williams and Titsias (2004); below we provide a summary.

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
   (a) Learn the ℓ^{th} object parameters $\{\mathbf{f}_\ell, \boldsymbol{\pi}_\ell, \sigma_\ell^2\}$ by maximizing F_ℓ (see equation (7)) using the EM algorithm.
   (b) Infer the most probable transformation $\{j_\ell^n\}$ and update the weights $\mathbf{z}_\ell^n = \mathbf{z}_{\ell-1}^n * (\mathbf{1} - \boldsymbol{\rho}_\ell^n)$ where $\boldsymbol{\rho}_\ell^n$ is computed as described above.

where

$$F_\ell = \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell) \times \Big\{\sum_{i=1}^{P}(\mathbf{z}_{\ell-1}^n)_i \log[(T_{j_\ell}\boldsymbol{\pi}_\ell)_i p_{f_\ell}(x_i; (T_{j_\ell}\mathbf{f}_\ell)_i) + (\mathbf{1} - T_{j_\ell}\boldsymbol{\pi}_\ell)_i p_b(x_i; (T_{j_b^n}\mathbf{b})_i)] - \log Q^n(j_\ell)\Big\}. \quad (7)$$

During the optimization of F_ℓ, $Q^n(j_\ell)$ is proportional to exp of the quantity in braces $\{\}$ in equation (7) (not including the $-\log Q^n(j_\ell)$ term), normalized to sum to one. This optimization is carried out by EM where in the E-step the quantities, $Q^n(j_\ell)$, $\mathbf{r}^n(j_\ell)$ and $\mathbf{s}^n(j_\ell)$ are computed, and in the M-step we update the parameters $(\boldsymbol{\pi}_\ell, \mathbf{f}_\ell, \sigma_\ell^2)$. $\mathbf{s}^n(j_\ell)$ stores the values $s_i^n(j_\ell)$ given by

$$\frac{(T_{j_\ell}\pi_\ell)_i p_{f_\ell}(x_i^n; (T_{j_\ell}\mathbf{f}_\ell)_i)}{(T_{j_\ell}\boldsymbol{\pi}_\ell)_i p_{f_\ell}(x_i^n; (T_{j_\ell}\mathbf{f}_\ell)_i) + (\mathbf{1} - T_{j_\ell}\boldsymbol{\pi}_\ell)_i p_b(x_i^n; (T_{j_b^n}\mathbf{b})_i)}.$$

For example the updates of $\boldsymbol{\pi}_\ell$ and $\mathbf{f}_\ell$ are

$$\boldsymbol{\pi}_\ell \leftarrow \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)T_{j_\ell}^T[\mathbf{z}_{\ell-1}^n * \mathbf{s}^n(j_\ell)]./ \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell)[T_{j_\ell}^T\mathbf{z}_{\ell-1}^n], \quad (8)$$

$$\mathbf{f}_\ell \leftarrow \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell) T^T_{j_\ell}[\mathbf{z}^n_{\ell-1} * \mathbf{s}^n(j_\ell) * \mathbf{r}^n(j_\ell) * \mathbf{x}^n] ./ \sum_{n=1}^{N}\sum_{j_\ell=1}^{J_f} Q^n(j_\ell) T^T_{j_\ell}[\mathbf{z}^n_{\ell-1} * \mathbf{s}^n(j_\ell) * \mathbf{r}^n(j_\ell)]. \quad (9)$$

The above updates are very intuitive. Consider, for example, the ℓ^{th} appearance model $\mathbf{f}_\ell$ when $Q^n(j_\ell) = 1$ for $j_\ell = j^n_\ell$ and 0 otherwise. For pixels which are ascribed to the ℓ^{th} foreground and are not occluded (i.e. $(\mathbf{z}^n_{\ell-1} * \mathbf{s}^n(j^n_\ell) * \mathbf{r}^n(j^n_\ell))_i \simeq 1$), the values in $\mathbf{x}^n$ are transformed by $T^T_{j^n_\ell}$. This removes the effect of the transformation and thus allows the foreground pixels found in each training image to be averaged to produce $\mathbf{f}_\ell$.

3. Speed-up using tracking

In this section we present a robust tracking algorithm that applies to a sequence of frames $(\mathbf{x}^1, \ldots, \mathbf{x}^N)$ and approximates the corresponding set of transformations $(j^1, \ldots, j^N)$ that describe the motion of a single object. The algorithm is combined with the GLOMO method so that we track and learn all the differently moving objects sequentially. We start by tracking the background, while the foreground objects are ignored (using robust statistics) at this stage. Once the transformation of the background in all frames has been approximated we learn its full structure through a focused search. Note that this procedure simply replaces the step 1 of the greedy algorithm. Given that we know the background enables us to track and learn foreground objects, so that step 3(a) of the greedy algorithm is modified suitably.

Section 3.1 discusses how we can track the background while section 3.2 describes tracking of the foreground objects. In Section 3.3 we discuss related work.

3.1. Tracking the background

We wish first to track the background and ignore all the other motions related to the foreground objects. 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^1_b 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^2_b for the second frame by searching over a small discrete set of neighboring transformations centered at j^1_b and inferring the most probable one (i.e. the one giving the highest likelihood (3), 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 rest 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 value 1 for initialized pixels and 0 otherwise. The transformation of the first frame j^1_b is the identity, which means that the first frame is untransformed. The transformation of the second frame and in general any frame $t+1$, $t \geq 1$, is determined by evaluating the posterior

$$R(j^{t+1}_b) \propto \exp\left\{\sum_{i=1}^{P} (T_{j^{t+1}_b}\mathbf{m}^t)_i \log p_b(x^{t+1}_i; (T_{j^{t+1}_b}\mathbf{b}^t)_i)\right\} \quad (10)$$

over the set of possible j^{t+1}_b values around the neighborhood of j^t_b. Note that the quantity in the $\exp\{\}$ in (10) is similar to the log of the likelihood (3) with the only difference that pixels of the background that are not initialized yet are removed from consideration. Once we know j^{t+1}_b we use all the frames up to the frame $\mathbf{x}^{t+1}$ to update $\mathbf{b}$ according to

$$\mathbf{b}^{t+1} \leftarrow \sum_{n=1}^{t+1}[T^T_{j^n_b}(\mathbf{r}^n(j^n_b) * \mathbf{x}^n)] ./ \sum_{n=1}^{t+1}[T^T_{j^n_b}\mathbf{r}^n(j^n_b)]. \quad (11)$$

where the $\mathbf{r}^n(j^n_b)$ vectors have been updated according to (5) for the old value $\mathbf{b}^t$ of the background. Notice that the above update is given by equation (4) in the GLOMO method, by considering only the first $t+1$ frames as the training data and assuming that the found transformation values $\{j^1_b, \ldots, j^{t+1}_b\}$ take all the probability ($P(j^i_b|\mathbf{x}^i) = 1, i = 1, \ldots, t+1$). The mask $\mathbf{m}$ is also updated so that it always indicates the pixels of $\mathbf{b}$ that are explored so far.

As we process the frames the background $\mathbf{b}$ is adjusted so that any occluding foreground object is blurred revealing the background behind it. Having tracked the background we can then learn its full structure assuming a focused search over the neighborhood of the approximated transformations.

3.2. Tracking the foreground objects

Assume that the background is now known. The pixels which are explained by the background in each image $\mathbf{x}^t$ are flagged by the background responsibilities $\mathbf{r}^t(j^t_b)$ (computed by equation (5)). Clearly the mask $\overline{\mathbf{r}}^t(j^t_b) = \mathbf{1} - \mathbf{r}^t(j^t_b)$ roughly indicates all the pixels of frame $\mathbf{x}^t$ 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 simultaneously updating its mask $\boldsymbol{\pi}_1$ and appearance $\mathbf{f}_1$. The mask and the appearance are initialized so that $\boldsymbol{\pi}_1 = \mathbf{0.5} * \overline{\mathbf{r}}^t(j^t_b)$ and

$\mathbf{f}_1 = \mathbf{x}^1$, 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^{t+1}, with $t \geq 1$, of the frame $\mathbf{x}^{t+1}$ we find the most probable value of j_1^{t+1} according to the posterior

$$R(j_1^{t+1}) \propto \exp\{\sum_{i=1}^{P} (\mathbf{w}_1^{t+1})_i \log[(T_{j_1^{t+1}} \boldsymbol{\pi}_1)_i \times p_f(x_i^{t+1}; (T_{j_1^{t+1}} \mathbf{f}_1)_i) + (\mathbf{1} - T_{j_1^{t+1}} \boldsymbol{\pi}_1)_i (1-\alpha_b) U(x_i^{t+1})]\} \quad (12)$$

where $\mathbf{w}_1^{t+1} = \bar{\mathbf{r}}^{t+1}(j_b^{t+1})$. $R(j_1^{t+1})$ measures the goodness of the match at those pixels of frame $\mathbf{x}^{t+1}$ which are not explained by the background. Note that as the objects will, in general, be of different sizes, the probability $R(j_1^{t+1})$ over the transformation variable will have greater mass on transformations relating to the largest object. Recall that $p_f(x_i^{t+1}; (T_{j_1^{t+1}} \mathbf{f}_1)_i)$ includes an outlier component so that some badly misfit pixels can be tolerated.

Once we determine j_1^{t+1} we update mask and appearance $\boldsymbol{\pi}_1$ and $\mathbf{f}_1$. These updates are similar to those given by (8) and (9), with the only difference that we use only the first $t+1$ frames and the approximate transformations to obtain posterior probabilities $P(j_1^t | \mathbf{x}^t) = 1$. Note that the updates are robust to occlusion (because of the semantics of $\mathbf{r}^n(j_1^n)$) so that occlusion of the tracked object can be tolerated. This makes tracking reliable even for the video frames the object is occluded. Note also that as the frames are processed tracking becomes more stable since $\boldsymbol{\pi}_1$ approximates the mask of a single object and the $\mathbf{f}_1$ will contain a sharp and clear view for only the one object being tracked while the rest of the objects will be blurred. See Figure 1 for an illustrative example.

Once this first object model has been learned we can go through the images to find which pixels are explained by this model, and update the $\mathbf{z}$ vector accordingly as explained in section 2. We can now run the same tracking algorithm again by updating $\mathbf{w}_{\ell+1}^t$ ($\ell \geq 1$) as by $\mathbf{w}_{\ell+1}^t = \mathbf{z}_\ell^t * \mathbf{w}_\ell^t$ 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 $\mathbf{0.5} * \mathbf{w}_{\ell+1}^t$ while the appearance $\mathbf{f}_{\ell+1}$ is always initialized to the first frame $\mathbf{x}^1$.

3.3. Related Work

There is a huge literature on motion analysis and tracking in computer vision, and there is indeed much relevant prior work. Note that our goals are to extract appearance models from a whole sequence of images, rather than, say, to carry out motion segmentation from just a pair of images.

In terms of layer-based models, Wang and Adelson (1994) showed how to cluster optical flow vectors into layers using affine motion models, and k-means clustering. They then built this information up into layer models using inverse warping and median filtering. The method of Irani et al. (1994) also uses optical flow information, but subtracts out motions sequentially. The representation 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 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 work of Tao et al. (2000) is also relevant in that it deals with a background model and object models defined in terms of masks and appearances. However, note that the mask is assumed to be of elliptical shape (parameterized 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. (2002) uses a *polybone* model for the support 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 (2002) 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. Torr (1998), Wills et al. (2003). These attempt to track features through a sequence and cluster these tracks using different motion models. We believe that these methods are of considerable interest, particularly when combined with new ideas on features that are stable to various transformations such as scaling and rotation (Lowe, 2003). However, preliminary work on our example sequences (see below) suggests that it is difficult to obtain the long tracks of features through a sequence that are really needed to make this method work well; it is hard to "tie together" short, broken tracks into objects. This may

[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.

well be different for other image sequences, e.g. of moving man-made objects rather than people.

4. Learning about Parts

For the recognition of complex objects that contain several parts that can take on different configurations relative to each other, it has long been recognized that a strategy based on the recognition of the individual parts and their relationships is likely to be advantageous; see e.g. Biederman (1987) on recognition-by-components. Much work in computer vision along this track uses manually identified parts; recent examples of such work are Felzenszwalb and Huttenlocher (2000) and Schneiderman and Kanade (2004) which consider people and cars, and faces and cars respectively. In contrast we are interested in *learning* parts from data. Much less work has been done on this topic, but some specific contributions are discussed below in section 4.1.

Although it is possible to use clues from single images to break up objects into parts, the view we take in this paper is that if two putative parts do not vary their relative relationship throughout the dataset we regard them as a single part. Here we focus on articulated objects (e.g. a human body) where the decomposition into parts is intuitive and clear cut.

4.1. Related Work

Perona and co-workers have developed important work on unsupervised learning of objects and parts through a series of papers. Their general approach (as described in Weber et al., 2000) is as follows: First an interest operator is used to locate keypoints, then graylevel descriptors of these keypoints are extracted. These features are then clustered to give a number of "part" types; note that there can be several detections of a given part type in a given image. A model is then built that puts some number of part types in particular spatial relationships. The learning of this model is slow as one has to deal with the combinatorics of the assignment of all part type detections in an image to the part types in the model. Note that in the end the model only prescribes the feature appearance at certain locations in the image, and does not provide a full appearance model of the object. However, an important aspect of this work is that object classes (e.g. cars) can be learned, not just specific objects.

Shams and von der Malsburg (1999) have also developed a method for learning 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 the following observations on their work: (i) in their method the background must be removed otherwise it would give rise to large match regions; (ii) 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.

Lee and Seung (1999) described a non-negative matrix factorization method to tackle part decompositions. However, this interesting work does not deal with the problem of parts undergoing transformations, and so could not extract the kinds of parts found by our method.

5. Experiments

We demonstrate our method on two video sequences: the Frey-Jojic (FJ) sequence available from `http://www.psi.toronto.edu/layers.html` and a human upper body sequence.

In terms of computational time, the GLOMO method requires $O(JNM)$ operations per learning an object, where J is the number of transformations, N the number of frames and M the number of EM iterations needed for convergence. Here one operation consists of computing $p_b(x_i; b_i)$ or $p_f(x_i; f_i)$ for all image pixels. The algorithm using tracking needs roughly $O(I_1 N + N^2 + I_2 N M_2)$ operations per pixel, where I_1 is the number of transformations considered for searching to find the transformation of the next frame as we track the object through the sequence, I_2 is the number of transformations of the focused search in the learning stage, and M_2 is the corresponding number of EM iterations. The N^2 term is due to updates of the mask and appearance during tracking, since at each time we consider all the so far processed frames. Notice that $J_f \gg I_1,\ I_2$ so the tracking algorithm should enjoy considerable speedups.

The FJ sequence consists of 44 118×248 images (excluding the black border); it was also used by Williams and Titsias (2004) in their experiments. The results in Figure 1(bottom) were obtained using a 15×15 window of translations in units of one pixel during the tracking stage ($I_1 = 225$) and a 1×1 window ($I_2 = 1$), i.e. without search, around the approximated location during the learning stage. This learning stage requires EM which converged in about $M_2 = 30$ iterations. Figure 1 also shows the evolution of the initial mask and appearance ($t = 1$) through frames 10 and 20 as we track the first object (Frey). Note 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 algorithm with tracking needs a total of $13,156$ operations to find an object. The original GLOMO algorithm, considers all 118×248 possible translations i.e. $J_f = 29264$ and also requires $M_1 = 70$ iterations to reach convergence. Thus the total number of operations required by the GLOMO is $90,133,120$, which implies that we gain a speed up over 6850 per learning a single object. The real running

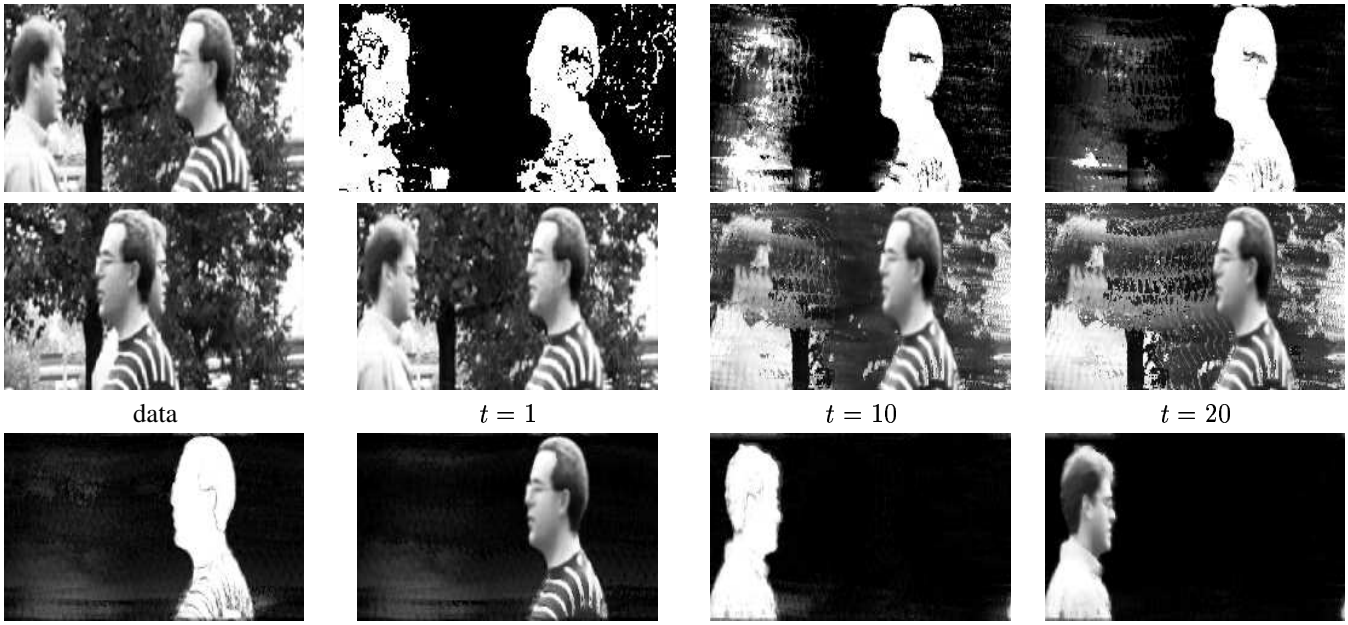

Figure 1: **Top**: The left hand column shows two data frames from the FJ sequence. The evolution of the mask and the appearance at frames 1, 10 and 20 is also shown. Notice how the mask becomes focused on one of the objects (Frey) and how the appearance remains clear and sharp only for Frey. **Bottom**: The final masks and the element-wise product of the mask and appearance model ($\boldsymbol{\pi} * \mathbf{f}$) learned for Frey and Jojic.

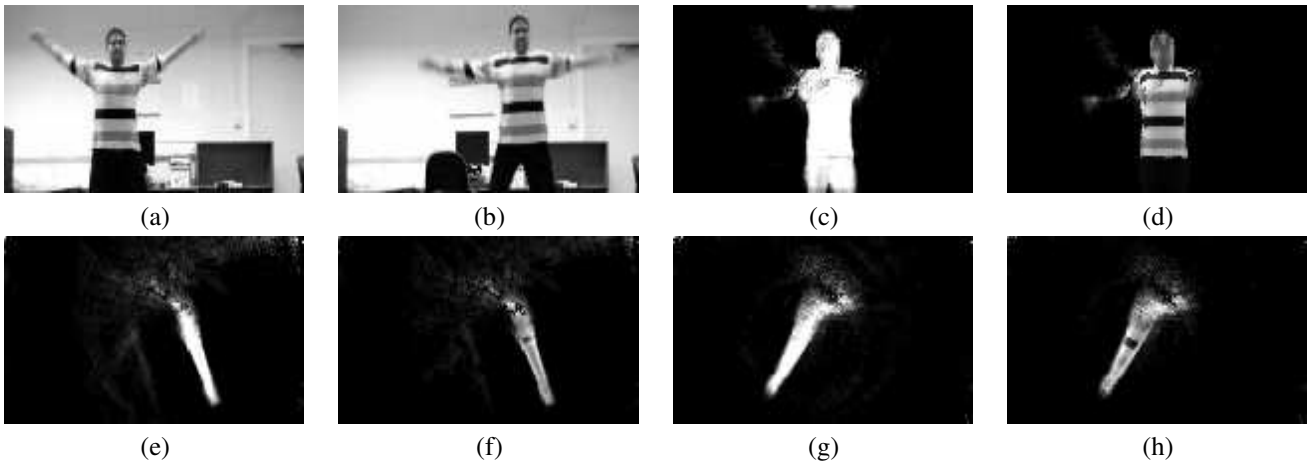

Figure 2: The first two panels (a) and (b) show two frames of the video sequence. Panels (c) and (d) show the learned mask and the element-wise product of the mask and appearance model for the head/torso, and the pair of panels (e)-(f) and (g)-(h) provide the same information for the two arms.

times of our MATLAB implementations roughly confirm the above, since the GLOMO method learns the whole sequence in 80 hours, while the algorithm using tracking runs in 3 minutes.

We demonstrate our method for learning parts of human body using a sequence of 76×151 images. Two frames of this sequence are shown in Figure 2. To learn the articulated parts we consider both translations and rotations so that the transformation matrix T_{j_ℓ} that applies to $\boldsymbol{\pi}_\ell$ and $\mathbf{f}_\ell$ implements a combination of a translation and a rotation. Particularly, T_{j_ℓ} is written as $T_{j_\ell} = T^{tr}_{j_\ell} T^{rot}_{j_\ell}$ so that we first rotate $\mathbf{f}$ expressing the vector $\mathbf{y} = T^{rot}_{j_\ell} \mathbf{f}_\ell$ and then translate $\mathbf{y}$ to obtain the transformed foreground. Note that the translations are in units of one pixel (as in the FJ sequence) while the rotations are implemented using the MATLAB nearest neighbor algorithm.

The tracking method uses a 15×15 window of translations and 23 rotations (at 2^o spacing) so that totally it searches over $I_1 = 5175$ transformations. After tracking a part we use a focused search around a $1 \times 1 \times 1$ window (i.e. no search) to learn its full structure. Figures 2(c)-(h) 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 3 hours. Notice also that running the original GLOMO method on this sequence is infeasible in practice. In particular the GLOMO method requires consideration of all possible 76×151 translations combined with, say, 200 rotations which immediately gives a very large number of transformations ($J_f = 2,295,200$) that should be considered.

6. Discussion

Above we have shown how to use a tracking method to greatly speed up GLOMO. We have also demonstrated that this method can identify articulated parts of objects, as in the human body example.

Some issues for further work include the identifying when a detected model is a part or an independent object (using mutual information), dealing with parts/objects that have internal variability and finding non-articulated parts.

References

Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147.

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.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2000). Efficient Matching of Pictorial Structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2000*, pages II:66–73.

Fitzgibbon, A. and Zisserman, A. (2002). On Affine Invariant Clustering and Automatic Cast Listing in Movies. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Proceedings of the Seventh European Conference on Computer Vision, ECCV 2002*, pages III 304–320. Springer. Lecture Notes in Computer Science 2353.

Irani, M., Rousso, B., and Peleg, S. (1994). Computing Occluding and Transparent Motions. *International Journal of Computer Vision*, 12(1):5–16.

Jepson, A. D., Fleet, D. J., and Black, M. J. (2002). A Layered Motion Representation with Occlusion and Compact Spatial Support. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P., editors, *Proceedings of the Seventh European Conference on Computer Vision, ECCV 2002*, pages I 692–706. Springer. Lecture Notes in Computer Science 2353.

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*, pages I:199–206. 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.

Lowe, D. G. (2003). Distinctive Image Features from Scale-Invariant Keypoints. Submitted to *International Journal of Computer Vision*, available from `http://www.cs.ubc.ca/spider/lowe/pubs.html`.

Schneiderman, H. and Kanade, T. (2004). Object Detection Using the Statistics of Parts. *International Journal of Computer Vision*, 56(3):151–177.

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*, pages II:134–141.

Torr, P. H. S. (1998). Geometric motion segmentation and model selection. *Phil. Trans. Roy. Soc. Lond. A*, 356:1321–1340.

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

Weber, M., Welling, M., and Perona, P. (2000). Unsupervised Learning of Models for Recognition. In *Proceedings of the Fifth European Conference on Computer Vision, ECCV 2000*, pages 18–32.

Williams, C. K. I. and Titsias, M. K. (2004). Greedy Learning of Multiple Objects in Images using Robust Statistics and Factorial Learning. *Neural Computation*, 16(5):1039–1062.

Wills, J., Agarwal, S., and Belongie, S. (2003). What Went Where. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2003*, pages I:37–44.