To be presented at IEEE Conference on Computer Vision and Pattern Recognition, Kauai, HA, Dec 9-14, 2001.

Learning Flexible Sprites in Video Layers

Nebojsa Jojic
Microsoft Research
http://www.ifp.uiuc.edu/∼jojic

Brendan J. Frey
University of Toronto
http://www.psi.toronto.edu

Abstract

We propose a technique for automatically learning layers of "flexible sprites" – probabilistic 2-dimensional appearance maps and masks of moving, occluding objects. The model explains each input image as a layered composition of flexible sprites. A variational expectation maximization algorithm is used to learn a mixture of sprites from a video sequence. For each input image, probabilistic inference is used to infer the sprite class, translation, mask values and pixel intensities (including obstructed pixels) in each layer. Exact inference is intractable, but we show how a variational inference technique can be used to process 320×240 *images at 1 frame/second. The only inputs to the learning algorithm are the video sequence, the number of layers and the number of flexible sprites. We give results on several tasks, including summarizing a video sequence with sprites, point-and-click video stabilization, and point-and-click object removal.*

1 Introduction

A simple and potentially very efficient way to model and automatically analyze video sequences is through the use of a layered representation [1]. A 3-dimensional scene is decomposed into a set of 2-dimensional objects in layers, hugely simplifying the geometry [1–4].

In this paper, we focus on learning the appearances of multiple objects in multiple layers, over the entire video sequence. In contrast to other work on learning the appearances of objects in videos (c.f., [5]), our algorithm is very general. The only input to our algorithm is the video sequence, the number of sprites and the number of layers. In fact, our algorithm can be applied to unordered collections of images.

Learning sprites from a video sequence is a difficult task, since the sprites have unknown shapes and sizes and must be disambiguated from the background, other sprites, sensor noise, lighting noise, and significant amounts of deformation.

We introduce "flexible sprites", which can deform from frame to frame, and we derive a variational expectation maximization algorithm for learning flexible sprites from video sequences. Once learned, the flexible sprite model can be used, *e.g.*, for video summarization, point-and-click video stabilization, and point-and-click object removal. Fig. 1 shows frames from a video sequence, plus a flexible sprite and background model that were learned in an unsupervised fashion.

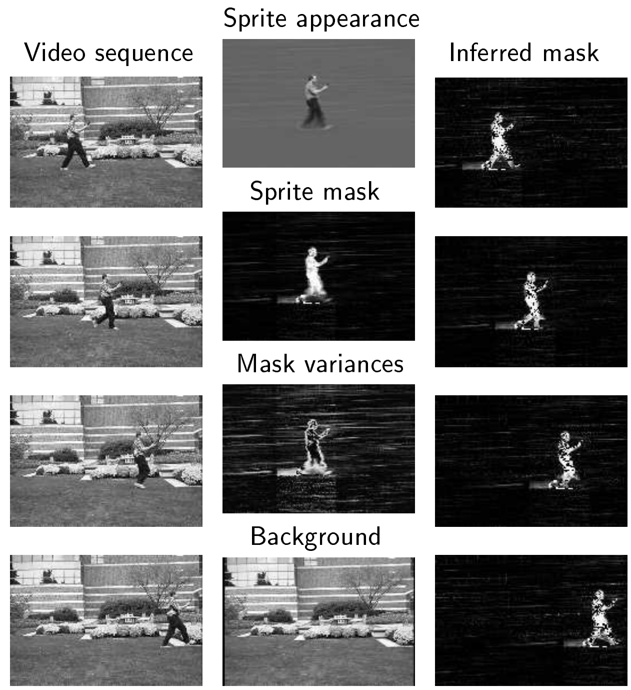

Figure 1: The generalized EM algorithm automatically decomposes a video sequence into a "flexible sprite" and a background image. The mask indicates which pixels in the sprite appearance map belong to the sprite. The *mask variance* allows the sprite to deform, *e.g.*, in the leg region. Once learned, the model can be used for many tasks. Here, the sprite mask for each frame is inferred and used to automatically track the deformable object. Even the barely visible shadow is tracked properly.

2 Sprites in Motion

Before introducing our probabilistic representation of "flexible sprites" and the corresponding unsupervised learning algorithm, we describe our imaging model for composing instances of flexible sprites. We refer to a particular instance of a flexible sprite as a "rigid sprite".

A rigid sprite is described by a raster-scan vector of greyscale pixel intensities $\mathbf{s}$ and a raster-scan vector of mask values $\mathbf{m}$. (The extension to color sprites is straight-forward.) When the sprites are learned from a video sequence, the size of the object is not known. To account for arbitrary object size, the images corresponding to these vectors are the *same size* as the input image. A mask value of 1 indicates the pixel belongs to the sprite, whereas a mask value of 0 indicates the pixel does not belong to the sprite.

The vector of observed pixel intensities $\mathbf{x}$ is explained by combining a vector of background pixel intensities $\mathbf{b}$ (which may contain other sprites, as described in the next section) with the instantiated sprite:

$$\mathbf{x} = \mathbf{m} * \mathbf{s} + \bar{\mathbf{m}} * \mathbf{b} + \mathit{noise}. \quad (1)$$

"$*$" indicates element-wise product, and

$$\bar{\mathbf{m}} = \mathbf{1} - \mathbf{m}$$

is the inverse mask. The first term in (1) erases sprite pixels that are transparent and the second term erases background pixels that are occluded by the sprite. The third term accounts for sensor noise, *etc.*.

In fact, we allow the mask values to be real numbers in $[0, 1]$, to account for semi-transparent "objects", such as windows and shadows.

We now extend this model to allow the sprite to undergo simple geometric transformations (translation, rotation, *etc.*) so that it can appear with different geometry in different frames of the video sequence. Following previous work by Frey and Jojic [6], the transformations are represented by a discrete set of possible transformations. For example, single-pixel translations in an $M \times N$ image (with wrap-around) are represented using MN possible shifts.

You may be concerned that discretizing the transformations will lead to slow algorithms. We hope you will be somewhat reassured by the fact that our current MATLAB implementation can process 320×240 images at 1 frame/second.

$\mathbf{T}$ is a transformation operator from the discrete set of transformation operators. For translations in an $M \times N$ image, $\mathbf{T}$ can take on MN values. The transformed version of $\mathbf{s}$ is written

$$\mathbf{Ts}.$$

So, $\mathbf{T}$ can be thought of as a permutation matrix that

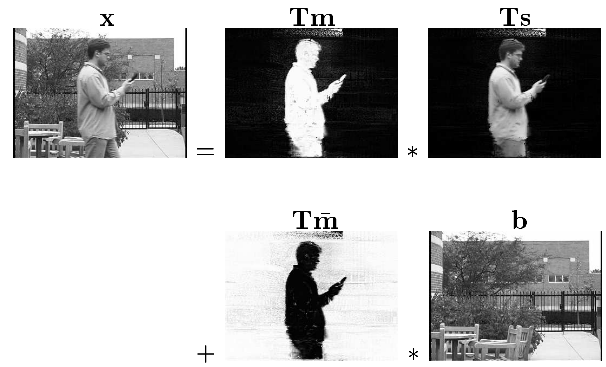

Figure 2: An image is formed by composing a translated sprite appearance map with a background appearance map, using a translated mask and its inverse, as given by (4). These maps were learned in an unsupervised fashion from a sequence a frames like $\mathbf{x}$, as described in Scn. 5.

rearranges the pixels in vector $\mathbf{s}$. Notice that $\mathbf{T}$ is also a random variable.

In contrast to (1), both the sprite mask and the sprite appearance map are transformed before being composed with the background:

$$\mathbf{x} = \mathbf{Tm} * \mathbf{Ts} + \mathbf{T}\bar{\mathbf{m}} * \mathbf{b} + \mathit{noise}. \quad (4)$$

Fig. 2 illustrates this composition, using sprites that were learned in an unsupervised fashion, as described in Scn. 5.

We assume the transformation preserves the number of pixels, *i.e.*, $\mathbf{Ts}$ is the same size as $\mathbf{s}$. To allow the sprite to overlap with the boundary of the input image without wrapping around to the other side, we introduce a "window layer", as described in Scn. 7.2.

3 Layers of Sprites

Multiple, occluding objects are modeled using layers of sprites. The layers are combined using the sprite masks, in a similar fashion as described above.

Layers are indexed by $\ell = 1, \ldots, L$, with layer L being the layer that is closest to the camera and layer 1 being the background layer. $\mathbf{s}_\ell$ and $\mathbf{m}_\ell$ are the sprite appearance map and sprite mask at layer ℓ. $\mathbf{T}_\ell$ is the transformation at layer ℓ.

Extending (4) by recursively replacing the background $\mathbf{b}$ with the output of the previous layer, we

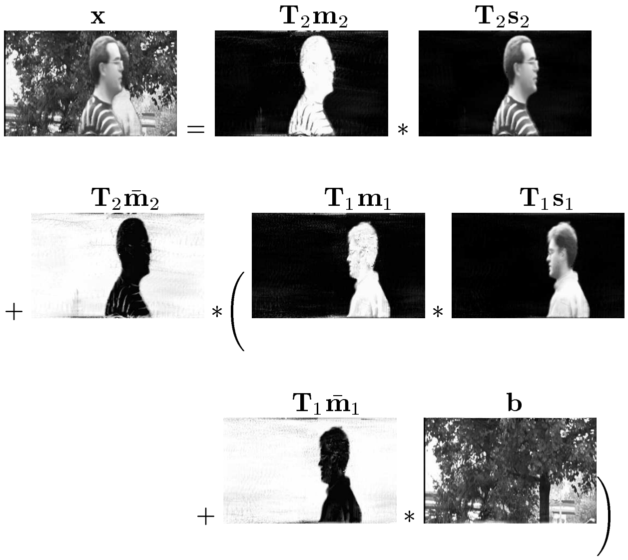

Figure 3: An image is formed by composing multiple layers of translated sprite appearance maps using translated sprite masks, as given by (5).

have

$$\begin{aligned}\mathbf{x} = \mathbf{T}_L\mathbf{m}_L * \mathbf{T}_L\mathbf{s}_L + \mathbf{T}_L\bar{\mathbf{m}}_L * \\ (\mathbf{T}_{L-1}\mathbf{m}_{L-1} * \mathbf{T}_{L-1}\mathbf{s}_{L-1} + \mathbf{T}_{L-1}\bar{\mathbf{m}}_{L-1} * \\ (\mathbf{T}_{L-2}\mathbf{m}_{L-2} * \mathbf{T}_{L-2}\mathbf{s}_{L-2} + \mathbf{T}_{L-2}\bar{\mathbf{m}}_{L-2} * \\ \cdots \\ (\mathbf{T}_1\mathbf{m}_1 * \mathbf{T}_1\mathbf{s}_1 + \mathbf{T}_1\bar{\mathbf{m}}_1 * \mathbf{b}))) \\ + \mathit{noise}. \end{aligned} \tag{5}$$

Fig. 3 illustrates this expression for two moving layers of sprites. These sprites were learned in an unsupervised fashion, as described in Scn. 5.

The recursive expression in (5) can be written more concisely by multiplying out the terms (*e.g.*, expressing the composition in Fig. 3 as a sum of 3 groups of products):

$$\mathbf{x} = \sum_{\ell=1}^{L}\Big(\Big(\prod_{i=\ell+1}^{L}\mathbf{T}_i\bar{\mathbf{m}}_i\Big) * \mathbf{T}_\ell\mathbf{m}_\ell * \mathbf{T}_\ell\mathbf{s}_\ell\Big) + \mathit{noise}.$$

The observed image can be constructed by adding together the sprites in all layers, masking each sprite by the product of the inverse masks for the sprites in later layers. For notational simplicity, we assume the background is modeled in layer 1.

Assuming the sensor noise is zero-mean Gaussian, the observation likelihood for the sprite appearances, sprite masks, and sprite transformations, $\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell\}$ is

$$\begin{aligned}&p(\mathbf{x}|\{\mathbf{s}_\ell, \mathbf{m}_\ell, \mathbf{T}_\ell\}) \\ &= \mathcal{N}\Big(\mathbf{x}; \sum_{\ell=1}^{L}\Big(\Big(\prod_{i=\ell+1}^{L}\mathbf{T}_i\bar{\mathbf{m}}_i\Big) * \mathbf{T}_\ell\mathbf{m}_\ell * \mathbf{T}_\ell\mathbf{s}_\ell\Big), \boldsymbol{\beta}\Big).\end{aligned} \tag{7}$$

$\boldsymbol{\beta}$ is a vector containing the pixel noise variances, and $\mathcal{N}(\cdot)$ is the probability density function for the normal distribution.

4 Flexible Sprites

So that the appearance of each sprite may vary from frame to frame, we use a probabilistic representation called a "flexible sprite". A flexible sprite may have different instantiations from frame to frame, but all of these instantiations are typical under the probability model that defines the flexible sprite.

To allow for multiple objects, we use a mixture of Gaussians for the prior. The prior distribution over the appearance and mask for sprite class c is

$$p(\mathbf{s}, \mathbf{m}|c) = \mathcal{N}(\mathbf{s}; \boldsymbol{\mu}_c, \boldsymbol{\phi}_c)\mathcal{N}(\mathbf{m}; \boldsymbol{\eta}_c, \boldsymbol{\psi}_c),$$

where $\boldsymbol{\mu}_c$ and $\boldsymbol{\eta}_c$ are vectors containing the mean appearance map and mean mask for object c, and $\boldsymbol{\phi}_c$ and $\boldsymbol{\psi}_c$ are vectors containing the pixel variances in the appearance map and mask for object c.

Assuming the sprite class, sprite transformation, sprite appearance map and sprite mask at layer ℓ are chosen independently of the sprites in other layers, the joint distribution is

$$\begin{aligned}&p(\mathbf{x}, \{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\}) \\ &= \mathcal{N}\Big(\mathbf{x}; \sum_{\ell=1}^{L}\Big(\Big(\prod_{i=\ell+1}^{L}\mathbf{T}_i\bar{\mathbf{m}}_i\Big) * \mathbf{T}_\ell\mathbf{m}_\ell * \mathbf{T}_\ell\mathbf{s}_\ell\Big), \boldsymbol{\beta}\Big) \\ &\cdot \prod_{\ell=1}^{L}\Big(\mathcal{N}(\mathbf{s}_\ell; \boldsymbol{\mu}_{c_\ell}, \boldsymbol{\phi}_{c_\ell})\mathcal{N}(\mathbf{m}_\ell; \boldsymbol{\eta}_{c_\ell}, \boldsymbol{\psi}_{c_\ell})\pi_{c_\ell}\rho_{\mathbf{T}_\ell}\Big),\end{aligned} \tag{9}$$

where π_c is the prior probability $p(c)$ of sprite class c, and $\rho_\mathbf{T}$ is the prior probability of transformation $\mathbf{T}$. The $\rho_\mathbf{T}$'s may be learned, but we fix them to uniform values.

5 Inference and Learning

Given the number of sprites C, the number of layers L, and the video sequence $\mathbf{x}^{(1)} \ldots \mathbf{x}^{(N)}$, probabilistic inference and learning are used to compute a single set of model parameters that represent the entire video sequence. These parameters include the prior probabilities of the sprite classes, $\pi_1 \dots \pi_C$; the means and variances of the sprite appearance maps, $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_C$ and $\boldsymbol{\phi}_1 \dots \boldsymbol{\phi}_C$; the means and variances of the sprite masks, $\boldsymbol{\eta}_1 \dots \boldsymbol{\eta}_C$ and $\boldsymbol{\psi}_1 \dots \boldsymbol{\psi}_C$; and the observation noise variances $\boldsymbol{\beta}$.

These parameters are initialized using the means and variances of the pixels in the video sequence. Then, a generalized expectation maximization algorithm is used to learn the flexible sprite models, so that the video sequence has high marginal probability under the model.

In the E-Step, the model parameters are assumed to be correct, and for each input image, probabilistic inference is used to fill in the values of the unobserved variables – the sprite class, sprite transformation, sprite appearance and sprite mask at each layer. In the M-Step, the model parameters are adjusted to increase the joint probability of the observations and the filled in unobserved variables. These two steps are repeated until convergence.

In fact, for each input image, the E-step fills in the unobserved variables with a *distribution* over plausible configurations (the posterior distribution), not just individual configurations. This is an important aspect of the EM algorithm. Initially, the parameters are a very poor representation of the data. So, any single configuration of the unobserved variables (*e.g.*, the most probable configuration under the posterior) will very likely be the wrong configuration.

The exact EM algorithm [7] uses the exact posterior in the E-Step and maximizes the joint probability with respect to the model parameters in the M-Step. The exact EM algorithm consistently increases the marginal probability of the data, performing maximum likelihood estimation.

Sometimes, the joint probability cannot be directly maximized. The "GEM" algorithm [7] uses the exact posterior in the E-Step, but just *partially* maximizes the joint probability in the M-Step, *e.g.*, using a nonlinear optimizer. The GEM algorithm also consistently increases the marginal probability of the data.

More generally, not only is an exact M-Step not possible, but computing the exact posterior is intractable. The generalized EM algorithm [8] permits the use of an approximation to the exact posterior in the E-step, and a partial optimization in the M-Step. The generalized EM algorithm consistently increases a lower bound on the marginal probability of the data.

5.1 Probabilistic inference (the E-Step)

For a given input image, the exact posterior is given by Bayes rule:

$$p(\{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\}|\mathbf{x}) = \frac{p(\mathbf{x}, \{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\})}{\sum \int p(\mathbf{x}, \{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\})},$$

where the sum $\sum$ and integral $\int$ are over all unobserved discrete and continuous variables, $c_1, \mathbf{T}_1, \mathbf{s}_1, \mathbf{m}_1, \dots, c_L, \mathbf{T}_L, \mathbf{s}_L, \mathbf{m}_L$.

Computing and storing this distribution is intractable. For each layer, there are CJ discrete configurations corresponding to combinations of the C sprite classes and the J transformations. The total number of discrete configurations is $(CJ)^L$. Later in this paper, we report results for an experiment where $C = 3$, $J = 76,800$ and $L = 3$, giving a total of over $10,000,000,000,000,000$ discrete configurations.

Even if the number of discrete configurations were tractable, for each discrete configuration, the distribution over the continuous variables (appearances and masks) is intractable. Although the continuous variables are Gaussian *a priori*, the observations are given by products of these variables. So, the posterior over the appearances and masks is not Gaussian and does not have a tractable closed form.

Since the exact posterior is not tractable, we compute a factorized approximation:

$$p(\{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\}|\mathbf{x}) \approx \prod_{\ell=1}^{L} q(c_\ell, \mathbf{T}_\ell) q(\mathbf{s}_\ell) q(\mathbf{m}_\ell).$$

Each of these q-distributions is an approximation to the corresponding marginal of the exact posterior distribution. For example, $q(c_\ell, \mathbf{T}_\ell) \approx p(c_\ell, \mathbf{T}_\ell|\mathbf{x})$.

To compute a good factorized approximation to the posterior, we parameterize each q-distribution and then jointly optimize the parameters of the q-distributions to maximize the negative relative entropy (Kullback-Leibler divergence) between the two distributions:

$$D = \sum \int \Big(\prod_\ell q(c_\ell, \mathbf{T}_\ell) q(\mathbf{s}_\ell) q(\mathbf{m}_\ell)\Big) \ln \frac{p(\{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\}|\mathbf{x})}{\prod_\ell q(c_\ell, \mathbf{T}_\ell) q(\mathbf{s}_\ell) q(\mathbf{m}_\ell)}.$$

This type of inference is called "variational inference" [9] and the parameters of the q-distributions are called "variational parameters".

$q(c_\ell, \mathbf{T}_\ell)$ is a discrete distribution, with one variational parameter for each configuration of c_ℓ and $\mathbf{T}_\ell$. So, for C classes, J transformations, and L layers, the posterior is represented by LCJ probabilities, instead of $(CJ)^L$ probabilities.

We use Gaussian approximations for $q(\mathbf{s}_\ell)$ and $q(\mathbf{m}_\ell)$, with variational parameters for the means and variances.

In fact, optimizing D directly is intractable, since $\ln p(\{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\}|\mathbf{x})$ does not have a simple form. To simplify the expression, we add the marginal probability of the observation, $\ln p(\mathbf{x})$, to D, obtaining

$$F = D + \ln p(\mathbf{x}) =$$
$$\sum\!\int\! \left(\textstyle\prod_\ell q(c_\ell, \mathbf{T}_\ell) q(\mathbf{s}_\ell) q(\mathbf{m}_\ell)\right) \ln \frac{p(\mathbf{x}, \{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\})}{\prod_\ell q(c_\ell, \mathbf{T}_\ell) q(\mathbf{s}_\ell) q(\mathbf{m}_\ell)}. \tag{11}$$

Since $\ln p(\mathbf{x})$ does not depend on the variational parameters, maximizing F produces the same variational parameters as maximizing D. However, maximizing F is tractable. From (9), it is clear that $\ln p(\mathbf{x}, \{c_\ell, \mathbf{T}_\ell, \mathbf{s}_\ell, \mathbf{m}_\ell\})$ simplifies to a sum of Mahalanobis distances and F is given by expectations of these distances under Gaussian q-distributions. The part of F in the form $\sum \int q \ln(1/q)$ is the total entropy of the q-distributions, which is easy to compute for discrete distributions and Gaussian distributions. So, computing F reduces to simple algebra.

F is *not* quadratic in the variational parameters, but is instead L-linear, for L layers. For example, with a single foreground sprite and a single background sprite, F is bilinear in the variational parameters. A nonlinear optimizer is used to maximize F with respect to the variational parameters, and the resulting q-distributions are used for learning (see below), object classification, object tracking, video stabilization, filling in occluded objects, and object removal.

The optimization of F makes use of convolutions and we use fast Fourier transforms for computational efficiency. In fact, for 3 layers and 3 sprites, our algorithm can process one 320×240 frame per second.

Once inference is complete, various useful tasks are readily performed. For example, the object in layer k is automatically removed from an image by classifying and locating the objects in each layer using the modes of $q(c_\ell, \mathbf{T}_\ell)$, $\ell = 1, \ldots, L$. Then, we use the modes of $q(\mathbf{s}_\ell)$ and $q(\mathbf{m}_\ell)$ to recover the instantiated appearance maps and masks for each object in the current image. Finally, we reconstruct the image, leaving layer k out of the reconstruction.

5.2 Learning using the generalized EM algorithm

The exact posterior is intractable, so exact EM and GEM are intractable and we use generalized EM to learn the model parameters from a video sequence.

Since the negative relative entropy D from above is always negative, from (11) we have

$$F \leq \ln p(\mathbf{x}).$$

For each video frame $\mathbf{x}^{(n)}$, there is a bound, $F^{(n)} \leq \ln p(\mathbf{x}^{(n)})$. Summing these bounds over the video frames gives a lower bound on the log-probability of the entire video sequence. So, maximizing the bound for each frame with respect to the frame's variational parameters and maximizing the sum of bounds with respect to the model parameters maximizes a lower bound on the likelihood of the video sequence.

After the model parameters are initialized to the means and variances of the pixels in the video sequence, a generalized E-Step and a generalized M-Step are repeated until convergence:

- Generalized E-Step: Increase F with respect to one set of variational parameters for each video frame

- Generalized M-Step: Increase F with respect to the model parameters

The experimental results reported in Scn. 7 were obtained using a MATLAB implementation of the general algorithm, with multiple layers of multiple flexible sprites. While it is quite straightforward to implement, the expressions contain too much linear algebra to include in this paper. So, we give the details of the algorithm for a single flexible sprite.

6 Algorithm Details for One Flexible Sprite

To clarify the above explanation of inference and learning, we now give details for a model with a single sprite. To simplify the model, we assume the variance in the background appearance is zero, and that the variance in the sprite appearance is zero. However, variance in the sprite mask is modeled, which allows the sprite to change shape from frame to frame.

The joint distribution over the observation $\mathbf{x}$, sprite mask $\mathbf{m}$ and transformation $\mathbf{T}$ is

$$p(\mathbf{x}, \mathbf{m}, \mathbf{T}) = \mathcal{N}(\mathbf{x}; \mathbf{Tm} * \mathbf{T}\boldsymbol{\mu} + \mathbf{T}\bar{\mathbf{m}} * \mathbf{b}, \boldsymbol{\beta}) \mathcal{N}(\mathbf{m}; \boldsymbol{\eta}, \boldsymbol{\psi}) \rho_{\mathbf{T}},$$

where $\boldsymbol{\mu}$ is the appearance map of the sprite.

We approximate $p(\mathbf{m}, \mathbf{T}|\mathbf{x})$ with

$$p(\mathbf{m}, \mathbf{T}|\mathbf{x}) \approx q(\mathbf{m}) q(\mathbf{T}) = \mathcal{N}(\mathbf{m}; \boldsymbol{\gamma}, \boldsymbol{\nu}) \xi_{\mathbf{T}},$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\nu}$ are vectors containing the mean and variance of the mask for the *current frame*, and $\xi_{\mathbf{T}}$ is the posterior probability of transformation $\mathbf{T}$ for the *current frame*.

The variational bound is

$$F = \sum_{\mathbf{T}} \int_{\mathbf{m}} q(\mathbf{m}) q(\mathbf{T}) \ln \frac{p(\mathbf{x}, \mathbf{m}, \mathbf{T})}{q(\mathbf{m}) q(\mathbf{T})}$$

$$= \sum_{\mathbf{T}} \xi_{\mathbf{T}} \ln(\rho_{\mathbf{T}}/\xi_{\mathbf{T}}) + \ln|2\pi e \boldsymbol{\nu}| - \frac{1}{2} \ln|\boldsymbol{\beta}|$$

$$- \frac{1}{2} \sum_{\mathbf{T}} \xi_{\mathbf{T}} \boldsymbol{\beta}^{-\mathsf{T}} \Big((\mathbf{x} - \mathbf{T}\boldsymbol{\gamma} * \mathbf{T}\boldsymbol{\mu} - \mathbf{T}\bar{\boldsymbol{\gamma}} * \mathbf{b})^2 + \mathbf{T}\boldsymbol{\nu} * (\mathbf{T}\boldsymbol{\mu} - \mathbf{b})^2 \Big)$$

$$- \frac{1}{2} \ln|2\pi\boldsymbol{\psi}| - \frac{1}{2} \boldsymbol{\psi}^{-\mathsf{T}} \Big((\boldsymbol{\gamma} - \boldsymbol{\mu})^2 + \boldsymbol{\nu} \Big),$$

where "$|\cdot|$" is the product of the elements in a vector, "T" indicates vector transpose, "$-\mathsf{T}$" indicates element-wise inverse followed by vector transpose, and "2" indicates element-wise squaring of a vector.

6.1 E-Step

Setting the derivatives of F with respect to $\boldsymbol{\gamma}$ and $\boldsymbol{\nu}$ to zero, we obtain the E-Step updates for the mask for the current frame:

$$\boldsymbol{\nu} \leftarrow \Big(\boldsymbol{\psi}^{-1} + \sum_{\mathbf{T}} \xi_{\mathbf{T}} \mathbf{T}^{-1} \big(\boldsymbol{\beta}^{-1} * (\mathbf{T}\boldsymbol{\mu} - \mathbf{b})^2 \big) \Big)^{-1}$$

and

$$\boldsymbol{\gamma} \leftarrow \boldsymbol{\nu} * \Big(\boldsymbol{\psi}^{-1}\boldsymbol{\eta} + \sum_{\mathbf{T}} \xi_{\mathbf{T}} \mathbf{T}^{-1} \big(\boldsymbol{\beta}^{-1} * (\mathbf{T}\boldsymbol{\mu} - \mathbf{b}) * (\mathbf{x} - \mathbf{b}) \big) \Big),$$

where $\mathbf{T}^{-1}$ is the inverse transformation of $\mathbf{T}$. The update for $\boldsymbol{\gamma}$ sets the mean of the mask for the current frame to a weighted combination of $\boldsymbol{\eta}$ and the average value of $(\mathbf{T}\boldsymbol{\mu} - \mathbf{b}) * (\mathbf{x} - \mathbf{b})$ under the posterior. The first term keeps the current mask close to its mean over the training set, $\boldsymbol{\eta}$, while the second term forces the mask to be large in regions where the background does not match the observed image. Parts of the object that deform quite a bit over the training data (such as swinging arms and legs) will lead to large values of the model mask variance $\boldsymbol{\psi}$ in the corresponding regions (see the description of the M-Step below). In those regions, $\boldsymbol{\psi}^{-1}$ is close to zero, and the second term dominates, *i.e.*, the mask is allowed to adapt to the current frame more closely. Fig. 1 shows how the inferred sprite mask can be used for tracking.

The update for $\boldsymbol{\nu}$ sets the variance of the mask for the current frame to be high in regions where the transformed sprite appearance map looks similar to the background ($\mathbf{T}\boldsymbol{\mu} \approx \mathbf{b}$) under all plausible transformations. The variance of the mask is lowest in regions where the transformed sprite appearance map looks different from the background under plausible transformations.

Setting the derivative of F with respect to ξ to zero (and including a Lagrange multiplier to ensure that $\sum_{\mathbf{T}} \xi_{\mathbf{T}} = 1$), we obtain the E-Step updates for the posterior probability of transformation $\mathbf{T}$:

$$\xi_{\mathbf{T}} \leftarrow \delta \rho_{\mathbf{T}} \exp\Big[\frac{1}{2} \boldsymbol{\beta}^{-\mathsf{T}} \Big((\mathbf{x} - \mathbf{T}\boldsymbol{\gamma} * \mathbf{T}\boldsymbol{\mu} - \mathbf{T}\bar{\boldsymbol{\gamma}} * \mathbf{b})^2 + \mathbf{T}\boldsymbol{\nu} * (\mathbf{T}\boldsymbol{\mu} - \mathbf{b})^2 \Big) \Big],$$

where δ is a scalar computed to ensure that $\sum_{\mathbf{T}} \xi_{\mathbf{T}} = 1$. This update makes the posterior probability $\xi_{\mathbf{T}}$ high for those transformations where the composition of the transformed sprite and background match the observed image. Note, however, that the last term penalizes transformations under which the transformed sprite looks similar to the background.

For each frame $\mathbf{x}^{(n)}$, these updates are iterated to obtain the variational parameters $\boldsymbol{\eta}^{(n)}$, $\boldsymbol{\nu}^{(n)}$ and $\xi^{(n)}$.

6.2 M-Step

Setting the derivative of F with respect to the model parameters to zero, we obtain the model parameter updates:

$$\boldsymbol{\psi} \leftarrow \frac{1}{N} \sum_n \Big(\boldsymbol{\nu}^{(n)} + (\boldsymbol{\gamma}^{(n)} - \boldsymbol{\eta})^2 \Big),$$

$$\boldsymbol{\eta} \leftarrow \frac{1}{N} \sum_n \boldsymbol{\gamma}^{(n)},$$

$$\boldsymbol{\mu} \leftarrow \Big(\sum_n \sum_{\mathbf{T}} \xi_{\mathbf{T}}^{(n)} \big(\boldsymbol{\gamma}^{(n)2} + \boldsymbol{\nu}^{(n)} \big) \Big)^{-1} * \Big(\sum_n \sum_{\mathbf{T}} \xi_{\mathbf{T}}^{(n)} \big(\boldsymbol{\gamma}^{(n)} * \mathbf{T}^{-1}\mathbf{x}^{(n)} - (\boldsymbol{\gamma}^{(n)} * \bar{\boldsymbol{\gamma}}^{(n)} - \boldsymbol{\nu}^{(n)}) * \mathbf{T}^{-1}\mathbf{b} \big) \Big).$$

$$\mathbf{b} \leftarrow \Big(\sum_n \sum_{\mathbf{T}} \xi_{\mathbf{T}}^{(n)} \mathbf{T} (\bar{\boldsymbol{\gamma}}^{(n)} + \boldsymbol{\nu}^{(n)}) \Big)^{-1} * \Big(\sum_n \sum_{\mathbf{T}} \xi_{\mathbf{T}}^{(n)} \mathbf{T} \big(\bar{\boldsymbol{\gamma}}^{(n)} * \mathbf{T}^{-1}\mathbf{x}^{(n)} - (\boldsymbol{\gamma}^{(n)} * \bar{\boldsymbol{\gamma}}^{(n)} - \boldsymbol{\nu}^{(n)}) * \boldsymbol{\mu} \big) \Big).$$

The mask variance is set to the average of the inferred mask variance for each video frame, plus the deviation of the inferred mask from the mask parameter. The mask parameter is set to the average of the inferred masks over the training set. The update for the appearance map is made intuitive by considering how it accumulates input from each video frame. In regions of the frame where the inferred mask is close to 1 (so, $\bar{\boldsymbol{\gamma}}^{(n)} \approx 0$) and there is little uncertainty in the inferred mask ($\boldsymbol{\nu}^{(n)} \approx 0$), the update accumulates the observed image. For intermediate values of the mask, a fraction of the background is subtracted from the observed image before it is accumulated.

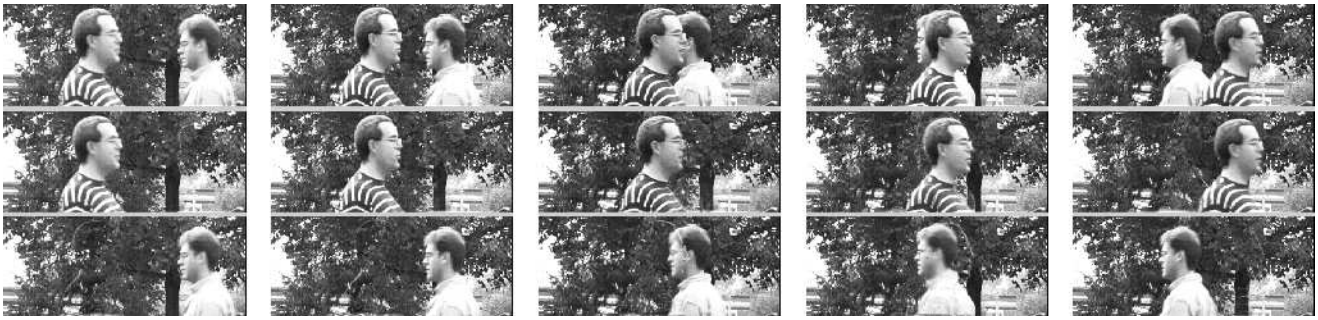

Figure 4: Three flexible sprites were learned from the input video (top row of frames), and then probabilistic inference was used to rerender the video frames, automatically removing various sprites.

7 Experimental Results

Using the technique described in the paper, it is possible to analyze a video sequence or even a collection of photographs and create a sprite decomposition of the video automatically. The applications that could benefit from such a representation are numerous - from video coding and water-marking to video surveillance. However, currently one of the most interesting applications is in low-end video/image manipulation tools. With the proliferation of digital media among home users, such tools are becoming highly desirable, but unfortunately visual data, especially video is currently very difficult to search, parse and edit. With the representation our algorithm creates, these and many other tasks are reduced to a few a mouse clicks. The flexible sprites can be easily used to create video panoramas on dynamic scenes; stabilize video with respect to specific objects, or even remove objects; capture quality stills from video, make video-textures, use sprites in presentations or on home pages, etc.

7.1 Point-and-click object removal and video stabilization

Each sprite has its own motion, appearance and mask, which can be changing over time. By inferring the appropriate values for a given frame, the model can recompose video without one or more sprites present in the original sequence. The top row of frames in Fig. 4 shows frames from an input video that was used to learn the flexible sprites shown in Fig. 3. Then, the model was used to rerender the video, with various sprites removed, as shown in the middle and bottom rows of frames in Fig. 4. The full video sequence is available in the supplemental file (`removal.avi`).

Similarly, after inferring the transformations $\mathbf{T}_\ell$ for all sprites in a given frame, it is straightforward to apply the inverse transformation and align with the sprite of interest, thus stabilizing a specific object instead of the background (`stabilization.avi`). Since learning is unsupervised, the user is only required to point and click on an object of interest.

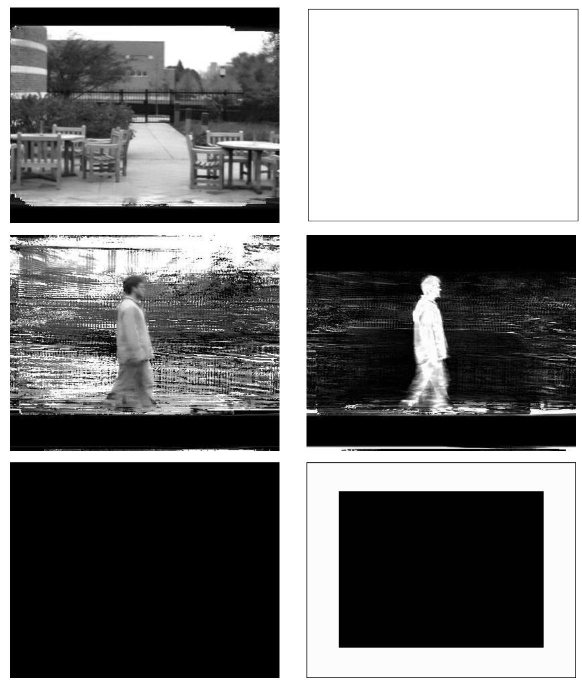

Figure 5: The appearance maps (left) and masks (right) for the moving background, moving object and stationary window for the `panorama.avi` video.

7.2 Panoramic sprites

Generating panoramas from stills or short video shots has attracted lots of attention in the computer vision community. While our model cannot currently produce the accurate disparity necessary for inferring geometry, it can properly infer flexible sprites from complex dynamics, like the one shown in `panorama.avi`. The sequence contains large foreground and background motion, interlacing, and motion blur artifacts that usually degrade optical flow approaches. Further, the foreground object is moving independently of the camera.

Figure 6: An input frame and the filled in panorama. See `panorama.avi`.

So that we could "fill in" the background model, we padded the borders of the original sequence with zeros and used a three-layer model where the front layer acts as a window that lets everything in the central region through. During learning, the algorithm uses the portion of the background that is inferred from each frame to improve the estimate of the background "panorama".

Fig. 5 shows the flexible sprites learned from the video sequence. The background sprite is a "panorama" that is larger than the input image. `panorama.avi` shows that the algorithm can produce a video that is stabilized with respect to the background or the foreground.

Fig. 6 shows the panorama that was filled in for one frame of the video sequence.

7.3 Object-specific editing

In the final example, we show a frame from a sequence of two flexible sprites passing each other in front of a cluttered background. The illumination conditions created bad contrast on the people in video, as the background had a large dynamic range. Since the sprite representation allows automatic segmentation of frames, it is possible to click on an object of interest and request an object-specific image adjustment as illustrated in Fig. 7.3.

8 Summary

Learning sprites from a realistic video sequence is a difficult task, since the sprite has an unknown shape and size and must be disambiguated from the background, other sprites, sensor noise, lighting noise, and significant amounts of deformation.

We introduced "flexible sprites", which can deform from frame to frame, and we derived a variational expectation maximization algorithm for learning flexible sprites from video sequences. Once learned, the flexible sprite model can be used, *e.g.*, for video summarization, point-and-click video stabilization, and point-and-click object removal. Through the use of fast Fourier transforms during variational inference, our algorithm is fast, able to process one 320×240 frame per second.

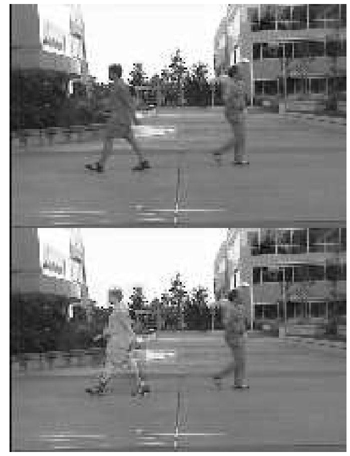

Figure 7: Object-specific illumination enhancement.

We believe that flexible sprites will prove to be an efficient, powerful tool for image and video processing.

References

[1] J. Y. A Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 625–638, 1994.

[2] P. H. S. Torr, R. Szeliski, and P. Anandan, "An integrated Bayesian approach to layer extraction from image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 297–303, 2001.

[3] A. Jepson and M. J. Black, "Mixture models for optical flow computation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1993, pp. 760–761.

[4] M. J. Black and D. J. Fleet, "Probabilistic detection and tracking of motion discontinuities," *International Journal on Computer Vision*, 2000.

[5] H. Tao R. Kumar and H. S. Sawhney, "Layer representation with applications to tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[6] B. J. Frey and N. Jojic, "Transformation invariant clustering and dimensionality reduction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001, To appear. Available at `http://www.cs.utoronto.ca/~frey`.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Proceedings of the Royal Statistical Society*, vol. B-39, pp. 1–38, 1977.

[8] R. M. Neal and G. E. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M. I. Jordan, Ed., pp. 355–368. Kluwer Academic Publishers, Norwell MA., 1998.

[9] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," in *Learning in Graphical Models*, M. I. Jordan, Ed. Kluwer Academic Publishers, Norwell MA., 1998.