Fast Learning of Sprites using Invariant

Fast Learning of Sprites using Invariant Features

Moray Allan, Michalis K. Titsias and Christopher K. I. Williams
School of Informatics, University of Edinburgh, Edinburgh EH1 2QL
moray.allan@ed.ac.uk, M.Titsias@sms.ed.ac.uk,
c.k.i.williams@ed.ac.uk

Abstract

A popular framework for the interpretation of image sequences is the layers or sprite model of e.g. Wang and Adelson (1994), Irani et al. (1994). Jojic and Frey (2001) 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. In this paper we show that by using invariant features (e.g. Lowe's SIFT features) and clustering their motions we can reduce or eliminate the search and thus learn the sprites much faster. We demonstrate our algorithm on two image sequences.

1 Introduction

Given a set of images containing views of multiple specific objects, we wish to learn appearance/shape models of each of the objects. A popular framework for this problem is the layer-based approach which models an image as a composite of 2D layers, each one modelling an object in terms of its appearance and region of support, see e.g. [11], [3].

Jojic and Frey [4] provided a principled generative probabilistic framework for this task, where the background layer and the foreground layers are synthesized using an alpha-matting rule which allows transparency of the layers. Learning using an exact EM algorithm is intractable so these authors used a variational inference scheme considering translational motion of the objects. An alternative learning algorithm is to optimize the layer parameters sequentially by learning one object at each stage [12].

The main problem with the generative model framework described above is that it is computationally slow, as it is necessary to search over a large discretized set of possible transformations for each of the layers. For translational motions a large speedup can be obtained using FFT tricks, but for richer classes of transformations (e.g. image plane affine transformations with six degrees of freedom) the method is very computationally intensive. In this paper we show how this problem can be overcome by using invariant features (e.g. SIFT features, [6]). Using such features we can segregate the motions of different objects through the sequence. For each object identified we can then inverse-transform each image to a reference frame where a sprite for the object can be learned.

The structure of the remainder of the paper is as follows: In section 2 we describe the layered generative model, and how to train this model using the information provided by the invariant features on object motions. In section 3 we describe how the image features are extracted, and then clustered into objects using a sequence-based RANSAC algorithm.

Section 4 describes related work and section 5 gives details of experiments illustrating the algorithm in action. We conclude with a discussion in section 6.

2 Layered generative model

For simplicity we start by introducing the generative model for the case with only two layers: a foreground object and a static background. Let $\mathbf{b}$ denote the appearance model of the background arranged as a vector. If the background is static, $\mathbf{b}$ will have the same size as the input image size. (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 vector in RGB space. For notational convenience we assume below that b_i is a scalar representing a grayscale value.

In contrast to the background, the foreground object occupies a limited region of the image, so to describe this layer we need both an appearance $\mathbf{f}$ and mask π. Each element of the mask π is in $[0,1]$ and defines the shape of the sprite, as explained below. As the foreground object moves, there is an underlying transformation matrix T_j that corresponds e.g. to translational or affine motion, so that $T_j\mathbf{f}$ and $T_j\pi$ are the transformed foreground and mask, respectively. We assume that the foreground and background combine by occlusion, thus each pixel in an observed image is either from a foreground object or from the background. Given the transformation T_j, an image $\mathbf{x}$ is drawn as in [12]:

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

where $N(x; \mu, \sigma^2)$ denotes a univariate Gaussian distribution with mean μ and variance σ^2. Using prior probability P_j over transformation T_j, the probability of an observed image $\mathbf{x}$ is given by $p(\mathbf{x}) = \sum_{j=1}^{J} P_j p(\mathbf{x}|T_j)$. Usually we set P_j to be uniform over all allowable transformations. Given a set of images $\{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$ we can use the EM algorithm to find the parameters $\theta = \{\mathbf{b}, \mathbf{f}, \pi, \sigma_f^2, \sigma_b^2\}$ which maximise the log likelihood.

This model can be extended to have a moving background and L foreground objects. For example, consider a set of images where there are two foreground layers with parameters $(\mathbf{f}_1, \pi_1, \sigma_1^2)$ and $(\mathbf{f}_2, \pi_2, \sigma_2^2)$, and a moving background. Let T_{j_1}, T_{j_2} and T_{j_b} denote the transformations of the first foreground object, the second foreground object and the background, respectively. Then the analogue of equation (1) is

$$\begin{aligned} p(\mathbf{x}|T_{j_1}, T_{j_2}, T_{j_b}) = \prod_{i=1}^{P} & (T_{j_1}\pi_1)_i N(x_i; (T_{j_1}\mathbf{f}_1)_i, \sigma_1^2) + (\mathbf{1} - T_{j_1}\pi_1)_i (T_{j_2}\pi_2)_i N(x_i; (T_{j_2}\mathbf{f}_2)_i, \sigma_2^2) \\ & + (\mathbf{1} - T_{j_1}\pi_1)_i (\mathbf{1} - T_{j_2}\pi_2)_i N(x_i; (T_b\mathbf{b})_i, \sigma_b^2). \end{aligned} \tag{2}$$

Note that this model asserts an underlying occlusion order for the objects. The first foreground object exists closest to the camera and can never be occluded. The second object can be occluded by the first object and the background can be occluded by both foreground objects.

2.1 Learning the model

Training the above model using an exact EM algorithm is intractable. For L foreground objects and a moving background, where each undergoes J transformations, the time complexity is $O(J^{L+1})$. Approximate training methods such as the variational EM algorithm in [4] or the one-object-at-a-time method in [12] could be applied.

However, by tracking motion clusters of image features as described in section 3, we can compute transformations for each layer directly, avoiding the need for search. By tracking motion clusters of image features we can obtain estimates for the transformation of each object in each frame, compared to some reference frame for the object. We use these transformations to learn the parameters of each layer in two stages. In the first stage we estimate the layers by optimizing the parameters of each layer separately (in a greedy fashion), learning one layer at a time using robust statistics. The transformations provided by clustering the motions of local features are not perfect, so we may choose to allow a limited amount of local search around the transformations. In the second stage we use the layer estimates obtained in the first stage to compute the occlusion order of the foreground layers. The ordering with the highest likelihood is selected, and we then use this ordering to further refine the parameters of all layers by optimizing them jointly.

We now explain this procedure in more detail. Given training images $\{\mathbf{x}^1,\ldots,\mathbf{x}^N\}$, we first find the affine transformations j_b^n and $\{j_\ell^n\}_{\ell=1}^L$ in each image n using by clustering the motions of invariant features, as described in section 3. We identify the motion corresponding to the background by computing the inverse transformations for all the frames, stabilising them by cancelling out the effect of object motion. By assuming that the background is the largest object in the images and computing image differences with respect to a reference image we can then choose the background to be the object that on average occupies the largest static image area. We learn the background $\mathbf{b}$ by maximizing the likelihood $L_b = \sum_{i=1}^n \sum_{i=1}^P \log p_r(x_i^n|(T_{j_b^n}\mathbf{b})_i)$, where p_r is a robustified likelihood defined in equation 5. (We do not use local search on the background as the affines are typically good, being based on a large number of image features.)

Once we have an approximation for the background we estimate each foreground layer separately. For each object ℓ we have $N(j_\ell^n)$, the local neighbourhood of transformations around j_ℓ^n for frame n. We can learn the parameters of the ℓth object using local search by maximizing the variational lower bound of the log likelihood,

$$F_\ell = \sum_{n=1}^{N} \sum_{j_\ell \in N(j_\ell^n)} Q^n(j_\ell) \log \frac{P_{j_\ell} p(\mathbf{x}^n|T_{j_\ell}, T_{j_b^n})}{Q^n(j_\ell)} \leq \sum_{n=1}^{N} \log \sum_{j_\ell=1}^{J} P_{j_\ell} p(\mathbf{x}^n|T_{j_\ell}, T_{j_b^n}), \qquad (3)$$

where $Q^n(j_\ell)$ is a multinomial distribution over transformations of the object in the image n, so that $Q^n(j_\ell) = 0$ for any $j_\ell^n \notin N(j_\ell^n)$, and P_{j_ℓ} is uniform prior over transformations. In (3) the conditional density $p(\mathbf{x}^n|T_{j_\ell}, T_{j_b^n})$ is given by

$$p(\mathbf{x}^n|T_{j_\ell}, T_{j_b^n}) = \prod_{i=1}^{P} (T_{j_\ell}\pi_\ell)_i p_r(x_i; (T_{j_\ell}\mathbf{f}_\ell)_i) + (\mathbf{1} - T_{j_\ell}\pi_\ell)_i p_r(x_i; (T_{j_b^n}\mathbf{b})_i), \qquad (4)$$

where we have replaced the Gaussian foreground and background pixel densities by the following robustified counterparts:

$$p_r(x_i; f_i) = \alpha_f N(x_i; f_i, \sigma_f^2) + (1-\alpha_f)U(x_i); \;\; p_r(x_i; b_i) = \alpha_b N(x_i; b_i, \sigma_b^2) + (1-\alpha_b)U(x_i). \qquad (5)$$

Here $U(x_i)$ is a uniform distribution over the range of all possible pixel values, while α_f and α_b express prior probabilities that a foreground or background pixel is not occluded. This robustification allows us to deal with cases where foreground objects occlude each each other or the background: any time a foreground or background pixel is occluded it will be explained by the uniform component $U(x_i)$ [9, 12]. The also helps if the affine transformation produced by the feature tracking is inaccurate on a particular frame.

It is straightforward to maximize F_ℓ in (3) using a variational EM algorithm; in the E-step we maximize over the Q^ns and in the M-step over the object parameters $\{\pi_\ell, \mathbf{f}_\ell, \sigma_\ell^2\}$. For this EM learning the appearance $\mathbf{f}_\ell$ is initialized to random values within the range of image pixel values, σ_ℓ^2 to high equal values, and the elements of each mask π_ℓ to 0.5.

Once models for all objects have been learned in this fashion we can compute the occlusion ordering of the foreground layers. We do this by fixing the layer parameters to the values learned above, and the transformations to the MAP estimates computed from the Q^ns distributions, and then measuring the likelihood of each possible ordering. The ordering with the highest likelihood score is selected. For example, when we have only two foreground objects this computation involves considering the two possible permutations of the foreground layers in equation 2 for all images and then chosing the permutation with the highest likelihood. Once the ordering has been found we can refine the parameters of all layers by optimizing them jointly using an analogue of equation 2.

3 Feature matching

In this section we first describe which image features are extracted, and then discuss the algorithm used for clustering these features into coherent objects.

If we have multiple views of a scene, as in the frames of a video sequence, the objects in the scene can be learnt by looking for parts of the scene which move together. We can use image features to quickly perform this kind of motion analysis, by detecting the features present in different views and comparing the locations of features which match between these views.

We process video frames using the Scale Invariant Feature Transform (SIFT) feature detector [6] to find image features designed to be invariant to image translation, scaling, and rotation[1]. Each feature is described by a 128-dimensional feature descriptor whose computation is based on gradient orientation histograms for a number of regions local to the keypoint. (We emphasize that these features are only one example of invariant features and that other features as proposed in e.g. [7] could also be used and may possibly give better performance.) By running over the frames in a video sequence, and comparing the image features detected in each frame with the features already found, we build up a 'dictionary' of distinct features, and a 'concordance' which records all the frames in which each feature was seen, and its location in each of these frames. Features from each new frame are compared with those already in the dictionary using Lowe's nearest neighbour matching technique, matching a feature to its nearest neighbour in SIFT feature space if the distance to the nearest neighbour is less than 0.6 times the distance to the second nearest neighbour. We can reduce the computation needed in feature matching and subsequently in clustering by only adding to the feature dictionary features that have

[1]We thank David Lowe for making his invariant keypoint feature detector code publically available at `http://www.cs.ubc.ca/spider/lowe/keypoints/`.

appeared in at least two consecutive frames, which are likely to be the most informative. In the rest of this section we assume that this feature extraction and matching has already been performed, and use 'feature' to refer to all the detected image features across a sequence matched to the same entry in the feature dictionary.

3.1 Clustering features into objects

Given two views of a scene, we can use the RANSAC algorithm [1] to find clusters of image features with consistent motion between the views. Candidate clusters are initialised with randomly-chosen seed features, and expanded by adding other features whose motion is consistent, then the cluster containing the most features is chosen as best. By repeating this process from seed features which have not yet been assigned to a cluster, we can, for example, find all the independently-moving clusters in two frames of video.

If we had full information on the location of all image features in every frame of a video sequence, we could run RANSAC over the frames together to find the independently-moving objects. However, in practice our feature information is sparse, with individual features usually only detected in a few frames of a sequence. One solution to this problem would be to run RANSAC between all adjacent pairs of frames in the sequence, since adjacent frames will have many features in common, and then to cluster the clusters from each pair of frames to find the objects. Alternatively, we could run RANSAC between individual frames and a key frame of interest.

The algorithm described below deals with sparse feature location data, and avoids the problems of reconstructing objects from contradictory clusters in different frames by maintaining a 2D geometric model for each object separate from its view in any individual frame. This geometric model contains all the features which belong to the object, at positions learnt from all the frames where they appear.

3.2 Sequence RANSAC algorithm

The objects in a sequence are learnt one after the other by a sequence-based version of the RANSAC algorithm, summarised in Algorithm 1.

We start by selecting a seed frame from the sequence, and randomly choosing seed features from this frame. Since we want to find two-dimensional affine transformations between objects and frames, we need three seed features. An initial geometric model for the candidate object is created from the the three features' positions in the seed frame.

In each of the frames where these three features appear together, we can find an affine transformation between their positions in the frame and their positions in the geometric model. This sequence of affine transformations describes the motion of the object. The features we should aim to add to the object are those whose positions throughout the sequence are compatible with this motion.

In standard RANSAC all the compatible features can be found directly, but when we cluster features across an image sequence our seed features might only appear together in a small number of frames, and so only give us enough information to find compatible features in a few frames of the sequence. Therefore we need to iterate. At each iteration we look at the frames about which we have information so far, and add to the current object all the features which appear in them and are compatible with its motion there.

Any features which we add to the object can give us information about more frames to use at the next iteration.

To judge if a given feature is compatible with the current candidate object we look at all the frames where it appears along with at least three features from the object. We find transformations that map features currently assigned to the object from their locations in the frames to the object with the least squared error, and then project the feature onto the object using each of these these transformations. If the feature matched the current object's motion precisely, it would always be projected to the same position on it. By setting different threshold values for the variance of the projected positions, we can extract tighter or more relaxed motion groupings. If the variance is below our set threshold, and is lower than the variance in projected position which the feature had against the objects we have already extracted, then we assign it to the current object for the next iteration.

A lower variance threshold will give a larger number of smaller motion groupings, and will lead to affine transformations that more closely represent the motions of the features in each grouping. A higher threshold will allow a single motion grouping to approximate a more complex set of feature motions. The optimal threshold value is dependent not only on the video data being used, but on how we want to use the resulting affine transformations. However, for many video sequences a high threshold value ($\sigma = 20$ pixels) will still give good results, since features from one object will fit extremely badly to the motion of other objects. The experiments we report below were run with $\sigma = 4$ pixels. In general we have found that if there appear to be k coherent moving objects in the sequence, these will correspond to to the k motion groupings that contain the largest number of features. Any other motion groupings detected typically correspond to local deviations from the affine model.

We continue iterating until the list of features belonging to the current object stops changing or we reach some maximum number of iterations. Once we have created a number of candidate objects, we pick the one which was able to explain the motion of the largest number of features. We then mark all its features as ruled out from being seeds for any future objects, so that we avoid repeatedly extracting the same motion grouping.

We can continue extracting objects in this way as long as there are at least three potential seed features remaining, but since we will tend to find objects in order by how many features' motions they account for, we may decide to stop once we have extracted some predetermined maximum number of objects.

The algorithm described above is quite similar to that presented in [8], although note that since we are building 2D instead of 3D models the initialization of a model is much easier in our case.

4 Related Work

Early work on the layers model was carried out by Wang and Adelson [11] and Irani et al. [3] *inter alia* using optical flow information. For example, [11] used optical flow as the cue for segmenting the motions and to obtain the inverse transformations. Note that problems can arise for optical-flow based methods in regions of low texture where flow information will be sparse and unreliable, and due to the aperture problem. Although such regions will also yield few features, the fact that we are learning an appearance model and mask means that this problem can be overcome as regions of objects that do provide good

Algorithm 1 Object learning algorithm

repeat
  Find all the features not yet assigned to an object.
  for each of *n* candidate objects **do**
    Initialise a new object with *k* features from some key frame, which are not yet assigned to an object, creating a geometric model containing these features in the positions at which they appeared in the key frame.
    repeat
      Use least squares to find transformations which project the features in the object model from each frame to their positions in the object's geometric model.
      Project every feature from each frame where it appears to the object using these transformations, and calculate its variance in position.
      If the variance across all the frames was below a threshold, and lower than for any of the previous objects, add the feature to the object.
    until there is no change in the object, or the maximum number of iterations is reached.
  end for
  Choose the candidate object that accounts for most currently unassigned features.
until all features in the key frame are assigned to an object, or the maximum number of objects is found.

features will guide the learning for the whole object. On the other hand, for some low resolution or blurred image sequences it may be possible to obtain useful optical flow information even when very few features are detected. An example of recent work on the layers model is [5], who learn a parts decomposition from video data. They use cross-correlation of fragments to estimate motions, refined by a MRF which imposes rigidity constraints on neighbouring fragments.

There has also been recent work on object level grouping and model extraction from video using invariant features, e.g. [10], [8]. One important difference is that our work aims to extract 2D sprite models from the data rather than 3D sets of points. Also, note that the models created by e.g. Rothganger et al. [8] are described by local affine patches centered on the feature points, and do not give an overall appearance model of the object. These authors have also used more sophisticated feature-tracking techniques than us; for example [8] use a Kanade-Lucas-Tomasi feature tracker tuned to track affine-invariant patches, and Sivic et al. [10] use additional processing to carry out short-range and long-range track repair. It is likely that our results could be improved by such methods, although at the expense of extra computation. Wills et al [13] match interest points and cluster the motions using RANSAC. However, their work only deals with pairs of images, and so does not face all of the challenges of a video sequence.

5 Experiments

We use the sequence RANSAC algorithm to find motion clusters, then learn sprites corresponding to the motions found. The sprite extraction algorithm receives as input the affine transformations for each object and learns the mask and appearance for each object. Our

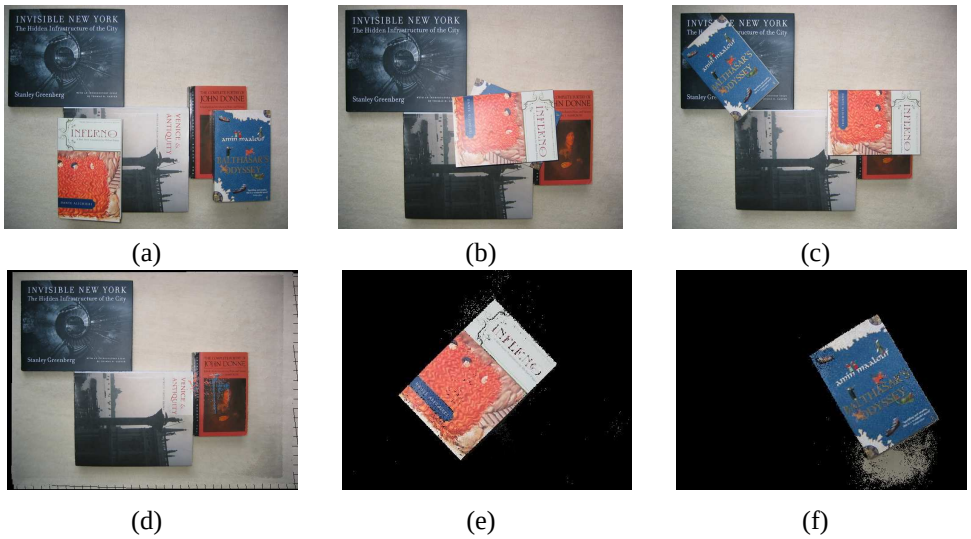

Figure 1: The first three panels (a), (b), and (c) show some frames from the book sequence. Panel (d) displays the learned background. Panels (e) and (f) show the element-wise product of the thresholded mask and appearance model for the two books.

MATLAB implementation, which is by no means has been optimised for speed, uses the MATLAB routine `imtransform` to carry out affine transformations of images.

Below we describe the results obtained on two different image sequences. The affines from sequence RANSAC were used directly, without local search. Illustrative video files can be found at `http://www.tardis.ed.ac.uk/~moray/features/`.

5.1 Books sequence

The 'books' sequence is a 19 frame stop-motion film of two books moving independently against a background of three further books (see Figure 1). Each frame is of size 640×480. The foreground books undergo translation and rotation, and one of the books occludes the other for several frames. The camera moves slightly between frames, so the overall transformations between frames involve further translation, rotation, and scaling.

Extracting features from the images and matching the extracted features across the sequence to compile a feature dictionary of distinct features took around 128 seconds. The resulting feature dictionary contained 2815 features.

Sequence RANSAC terminated after 4 seconds, when it finished assigning features to three objects. In all the experiments here we produced 20 candidates at each stage of sequence RANSAC. One of the objects corresponded to the background, a second to the right-hand moving book, and a third to the left-hand moving book.

The EM algorithm took 51 minutes to learn the background and the two books in the image sequence, using 25 iterations of EM and no local search. A method that needs to search explicitly over J transformations would take J times longer. For the books sequence a suitable discretization scheme would need to consider at least translations and

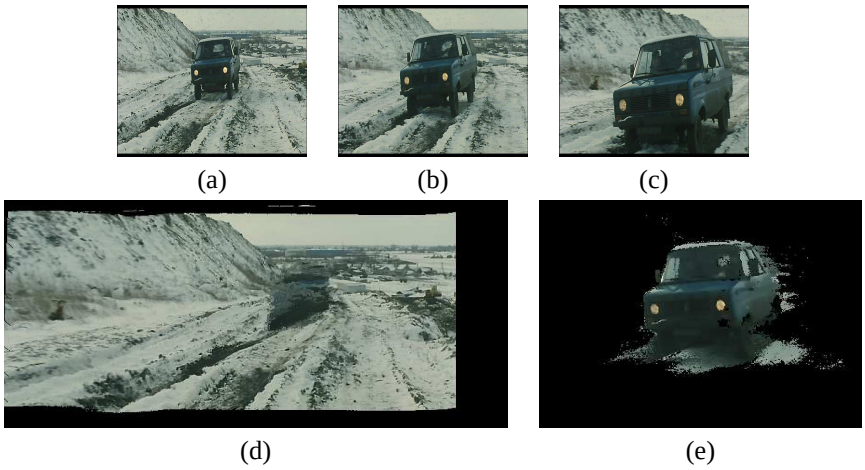

Figure 2: The first three panels (a), (b) and (c) show some frames from the *White* sequence. Panel (d) displays the learned background. Panel (e) shows the element-wise product of the thresholded mask and appearance model for the jeep.

rotations. Unless additional information can be brought to bear we would need to consider 640×480 translations and rotations at perhaps 2^o intervals (although FFT tricks could be used for the translations).

Figure 1 shows the element-wise products of the masks (thresholded at 0.5) with the books' appearance models, and the appearance model for the background. We can see that the objects in this sequence have been extracted well. The shadow of the second book, prominent at the start of the sequence, has been included in the mask for that book.

5.2 *White* sequence

Our second experiment uses a 35-frame, 720×576, sequence from the film *White* (see Figure 2). A jeep drives from the distance in the centre of the frame along a road passing to the camera's left. The camera pans to follow the jeep, viewing more of the background scene over the course of the shot than is visible in each frame.

Feature extraction and matching took 99 minutes, giving a feature dictionary of 17475 features. We compared each feature from an individual frames to entries in a flat list of features already in the dictionary; the time for matching could be greatly reduced by using a more complex data structure that would allow quicker access to the features most likely to match, as described in [6].

Sequence RANSAC found objects corresponding to the background and the jeep in 87 seconds. Sprite extraction took 31 minutes, using 30 iterations of EM.

The sprite models learned for the jeep (with the mask thresholded at 0.5) and for the background are shown in Figure 2. Since the camera pans to view a background scene larger than an individual frame, the background sprite we extract is a wide panoramic image. Part of the centre of the background sprite is blank, since this area is occluded by the jeep throughout the sequence, but the sprite is otherwise good. The jeep and its shadow have also been learned successfully.

6 Discussion

The results above show that the proposed algorithm is effective, and much faster than methods that need to use extensive search. The performance might well be improved by using a more sophisticated procedure (such as a smoothing model using a Kalman filter) on the sequences of affines.

Over a sequence, the appearance of a rigid object can change markedly due to changes in pose and self-occlusion. For example in the beginning of a sequence we may see a front view of a car, while at the end we may see a side view. The generative model can be modified so that the view of an object can vary significantly. We can achieve this by introducing a set of mask and appearance pairs, each one associated with a different viewpoint of the object, as proposed e.g. in [2]. Note also that this should not greatly increase the time taken as the number of viewpoints needed is typically small. We are currently experimenting with this enhancement to our method.

Finally, we note that the extracted sprites could be improved by applying additional constraints, such as an MRF prior over masks, or object-specific colour models.

References

[1] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981.

[2] B. J. Frey and N. Jojic. Transformation Invariant Clustering Using the EM Algorithm. *IEEE PAMI*, 25(1):1–17, 2003.

[3] M. Irani, B. Rousso, and S. Peleg. Computing Occluding and Transparent Motions. *IJCV*, 12(1):5–16, 1994.

[4] N. Jojic and B. J. Frey. Learning Flexible Sprites in Video Layers. In *Proc. CVPR*, pages I:199–206. IEEE Computer Society Press, 2001. Kauai, Hawaii.

[5] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Learning layered pictorial structures from video. Technical report, Oxford Brookes University, 2004.

[6] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[7] K. Mikolajczyk and C. Schmid. An affine invariant interest point operator. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Proc. ECCV*, pages I 128–142. Springer, 2002.

[8] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, Modeling and Matching Video Clips Containing Multiple Moving Objects. In *Proc. CVPR*, pages II:914–921, 2004.

[9] S. Rowe and A. Blake. Statistical Background Modelling For Tracking With A Virtual Camera. In D. Pycock, editor, *Proc. BMVC*, volume 2, pages 423–432. BMVA Press, 1995.

[10] J. Sivic, F. Schaffalitzky, and A. Zisserman. Object Level Grouping for Video Shots. In *Proc. ECCV*, 2004.

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

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

[13] J. Wills, S. Agarwal, and S. Belongie. What Went Where. In *Proc. CVPR*, pages I:37–44, 2003.