Estimation of a 3D motion field from a multi-camera array using a multiresolution Gaussian mixture model.

R Wilson, A Bowen, A Mullins and N M Rajpoot

University of Warwick, Computer Science Department, Coventry CV4 7AL, UK

Abstract. The problem of modelling geometry for video based rendering has been much studied in recent years, due to the growing interest in 'free viewpoint' video and similar applications. Common approaches fall into two categories: those which approximate surfaces from dense depth maps obtained by generalisations of stereopsis and those which employ an explicit geometric representation such as a mesh. While the former have generality with respect to geometry, they are limited in terms of viewpoint; the latter, on the other hand, sacrifice generality of geometry for freedom to pick an arbitary viewpoint. The purpose of the work reported here is to bridge this gap in object representation, by employing a stochastic model of object structure: a multiresolution Gaussian mixture. Estimation of the model and tracking it through time from multiple cameras is achieved by a multiresolution stochastic simulation. After a brief outline of the method, its use in modelling human motion using data from local and other sources is presented to illustrate its effectiveness compared to the current state of the art.

1 Introduction

The problem of modelling and tracking 3D objects from one or more video sequences has received considerable attention over recent years, as computational and capture costs have fallen. Techniques range from adaptation of stereo vision algorithms, eg. [2] to meshes and visual hulls [1]. Although the former requires no explicit model of the object to be tracked, it is restricted in viewing point and the positioning of cameras. On the other hand, techniques based on meshes inevitably make assumptions about the shapes of the objects and motions. The problem is that, without such constraints, a mesh or voxel representation has too many degrees of freedom to represent any plausible motion. What is needed is a way of constraining the motion to conform to the sorts of movements that, for example, dancers or similarly complex figures might undergo, while retaining the flexibility to represent a variety of objects and their movements. This naturally suggests a hierarchical model; such ideas have been explored, notably [1] and [3], in which the motion of the trunk is used to constrain movements of limbs.

To tackle this, it is necessary to construct a model which is flexible, yet stable enough to track motion over time. The model uses a stochastic description which allows the modelling of arbitrary data densities undergoing motion:

multiresolution Gaussian mixtures (MGM) [7]. We use the MGM model to represent objects in terms of a cloud of point measurements of their visual properties, such as colour, texture and motion. The model provides structure in terms of a hierarchy of mixture models. This hierarchical structure serves to reduce the complexity of motion estimation and to constrain the degrees of freedom, without being tied to a particular geometry, such as a mesh or voxel representation. The approach has been tested on a number of sequences, both publicly available ones, such as the Microsoft Research data [9], and some captured using a local camera array, a studio with 48 fire-wire digital video cameras.

After a brief introduction to the MGMM approach and an outline of the algorithms for constructing it and tracking it, we present a series of results to show its use in modelling a number of sequences for video based rendering applications. The model is capable of being used with arbitrary input data, from point clouds to voxels, with appropriate attributes. Moreover, the resulting motion estimates define a smooth, dense motion field, which may be used to animate arbitrary objects. These are features shared by no other representation of which we are aware. The paper is concluded with a discussion of the potential and limitations of the model.

2 MGM for Dynamic Scene Modelling

In [7], it was shown that a Gaussian mixture can be constructed in a top-down fashion, starting with a single component and splitting components when their fit to the data was poor, using a model-fit criterion, such as AIC or BIC. The key features of the MGM representation were shown to be

- *Universality*: the model is capable of approximating and arbitrary density to a specified level of accuracy.
- *Invariance to affine motions*: the model representing a density transformed by a smooth motion can be computed directly by an appropriate transformation of the original model. This follows because the Gaussian density is completely determined by its mean, $\boldsymbol{\mu}$, and covariance, $\boldsymbol{\Sigma}$, which transform under an affine motion defined by a linear transform, $\boldsymbol{A}$, and translation, $\boldsymbol{b}$, according to
$$\boldsymbol{\mu}' = \boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{b} \tag{1}$$
and
$$\boldsymbol{\Sigma}' = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^T \tag{2}$$
where ^{T} denotes transpose and $'$ denotes the transformed entity. In other words, the model structure is preserved under affine motions and hence does not need to be recomputed for each successive frame of a sequence.

These properties are central to its use in describing dynamic scenes, such as those found in video based rendering, where motions can be both complex and compound and objects are often articulated and of varying number in any given scene. A further difference between the present work and earlier uses of the model

is that the hierarchical structure plays a crucial role in motion estimation. To understand how this comes about, suppose that at some level, l, of the tree, we have a component (l, i), with parameters $\boldsymbol{\mu}_{l,i}, \boldsymbol{\Sigma}_{l,i}$, whose (affine) motion we estimate as $\boldsymbol{A}_{l,i}, \boldsymbol{b}_{l,i}$. Now we wish to estimate the motion of its 'children', ie. the components $\{l-1, j, p(j) = i\}$, where $p(i)$ is the *parent* of node i, that is, the affine parameters, $\boldsymbol{A}_{l-1,j}, \boldsymbol{b}_{l-1,j}$. We can use the estimates from (l, i) to define a more or less tight prior on the motion at $l-1, j, p(j) = i$, which has the effect of a soft constraint on the motion at level $l-1$ and speeds up computation, since the mean motion of the parent is a good initial guess for any component wholly contained within that parent, as the components $\{l-1, j, p(j) = i\}$ are. This is quite a different approach to that described in [7], which did not use the tree structure in the motion estimation. Moreover, the data in that work were 2D, representing a single sequence.

3 Construction of the Model

A bottom-up process was used in constructing the model to provide greater control over both the number of levels in the tree and in the number of components at each level, at the expense of a more complex method of choosing the priors for the components at a given level.

Whatever the source of the object description, we assume that it can be represented by a set of feature vectors, $\{\boldsymbol{f}_i \in R^n, 1 \leq i \leq N\}$, which are points in R^n. The dimension, n, depends on the model, but might include any or all of the position, orientation, colour and possibly motion of a 3D surface or volume element. The data density is then approximated by a normal mixture with, say, M_l components at level l, where $0 \leq l < L$ and $M_l < M_{l-1},\ M_0 = N$. At the top level of the hierarchy, the number of components, M_L, would usually be set to one more than the number of objects in the scene, with one component for the background. Below that level, each distinct object is represented by a tree, in which the components at a given level, l, may be regarded as *children* of the components at level $l+1$. Correspondingly, the components at level l constitute the input data for the clustering algorithm at level $l+1$ and the covariance of the data assigned to any component is included in the clustering at the next level of the tree.

A Bayesian clustering algorithm is used, based on the conjugate priors for the Gaussian mixture problem, namely a Dirichlet prior on the mixture weights, a normal on the means and a Wishart density on the covariances [8]. Although conjugate priors can be restrictive in some settings, their role here is purely a matter of computational convenience, allowing a Gibbs sampler to be used. The complete algorithm is as follows [8]:
Initialise

1. $l = 0$, $\mathbf{x_{0,i}} = \boldsymbol{f_i}$, $\mathbf{S_{0,i}} = \boldsymbol{\Delta}, \mathbf{1} \leq \mathbf{i} \leq \mathbf{N}$,
2. Select L, $M_l, 1 \leq l < L$.

where $\boldsymbol{\Delta}$ is the measurement error covariance.
Loop over Scale:
For $l = 1, L-1$

1. Simulate from the full conditionals:
   For $iter = 1, imax$

$$\pi \sim D(\alpha_l^*) \tag{3}$$
$$\boldsymbol{\Sigma}_{l,i}^{-1} \sim W(\boldsymbol{W}_{l,i}^*, r_{l,i}^*) \tag{4}$$
$$\boldsymbol{\mu}_{l,i} \sim N(\mathbf{y}_{l,i}^*, \tau_{l,i}^* \boldsymbol{\Sigma}_{l,i}) \tag{5}$$

   where

$$\alpha_l^* = \alpha_l + n_l \tag{6}$$
$$r_{l,i}^* = r + n_{l,i} \tag{7}$$
$$\tau_{l,i}^* = \frac{\tau_l}{\tau n_{l,i} + 1} \tag{8}$$
$$\mathbf{y}_{l,i}^* = \frac{\tau_l n_{l,i} \bar{\mathbf{x}}_{l,i} + \mathbf{y}_{l,i}}{\tau_l n_{l,i} + 1} \tag{9}$$
$$\boldsymbol{W}_{l,i}^* = \left(\boldsymbol{W}_{l,i}^{-1} + n_{l,i} \boldsymbol{S}_{l,i} + \frac{n_{l,i}}{\tau n_{l,i} + 1} \left(\bar{\mathbf{x}}_{l,i} - \mathbf{y}_{l,i} \right) \left(\bar{\mathbf{x}}_{l,i} - \mathbf{y}_{l,i} \right)^T \right)^{-1} \tag{10}$$

   $\bar{\mathbf{x}}_{l,i}$ is the sample mean of the l, i^{th} component, $\boldsymbol{S}_{l,i}$ the sample covariance and $n_{l,i}$ is the number of data points assigned to it. $\boldsymbol{W}_{l,i}$, α_l, r_l, $\tau_{l,i}$ and $\mathbf{y}_{l,i}$ are parameters on the prior distributions for each component. $D(\alpha)$ is a Dirichlet distribution with parameter α and $W(V, n)$ is a Wishart distribution with parameters V and n.
2. Derive the cluster assignments:
   Assign the class label of the l, i^{th} point, $\boldsymbol{z}_{l,j}$, randomly to one of the components $k \in 1..M_l$ proportionally to $N(\mathbf{x}_{l,j} - \boldsymbol{\mu}_{l,k}, \Sigma_{l,k})$ and update the sample means $\bar{\boldsymbol{x}}_{l,k}$ and covariances $\boldsymbol{S}_{l,k}$ according to

$$\bar{\boldsymbol{x}}_{l,k} = \frac{1}{n_{l,k}} \sum_{i | z_{l,i} = k} \bar{\boldsymbol{x}}_{l-1,i} \tag{11}$$
$$\boldsymbol{S}_{l,k} = \frac{1}{n_{l,k}} \sum_{i | z_{l,i} = k} (\boldsymbol{x}_{l-1,i} - \bar{\boldsymbol{x}}_{l,k})(\boldsymbol{x}_{l-1,i} - \bar{\boldsymbol{x}}_{l,k})^T + \sum_{i | z_{l,i} = k} n_{l-1,i} \boldsymbol{S}_{l-1,i} \tag{12}$$

3. When sufficient iterations have been run to achieve stationarity, assign data to clusters to maximise the posterior

$$\boldsymbol{x}_{l,k} \in \Lambda_{l,j} \text{ if } N(\mathbf{x}_{l,k} - \boldsymbol{\mu}_{l,j}, \Sigma_{l,j}) > N(\mathbf{x}_{l,k} - \boldsymbol{\mu}_{l,m}, \Sigma_{l,m}),\ m \neq j \tag{13}$$

   where $\Lambda_{l,i}$ is the index set of the l, ith component in the model. Then the data at level l are just the sample means

$$\boldsymbol{x}_{l,k} = \frac{1}{|\Lambda_{l,k}|} \sum_{i \in \Lambda_{l,k}} \boldsymbol{x}_{l-1,k} \tag{14}$$

and the sample covariances for the clusters, $\boldsymbol{S}_{l,k}$, are similarly used at the next level.

The number of iterations, $imax$, is chosen to ensure stationarity in the Markov Chain. Note that by using cluster means and covariances in place of the raw data at levels above $l = 0$, a huge saving in computation can be achieved with minimal impact on the quality of the approximation.

4 Tracking the Model

The model is tracked over time using an estimator whose form depends on the relationship between the data and the model state. Since the data in this case are surface elements projected onto a set of cameras, the relationship is non-linear; correspondingly, a particle filter was chosen for the problem. This is a sequential, Bayesian approach, both in time and in terms of the 'scale' hierarchy in the MGM model (cf. eqns (1)-(5)), where the estimate for level l is used in defining the prior for level $l-1$. Because of the hierarchical nature of the representation, it is sufficient to employ a rigid motion model for each component, requiring estimation of a 6-parameter vector $\boldsymbol{\theta}$ per component. As noted above, this both simplifies computations and constrains the motions. Estimation is done by maximising the posterior over the unknown position and orientation,

$$P(\boldsymbol{\theta}_{l,i}|\boldsymbol{Y}_j, j \in \Lambda_{l,i}) \propto \int_{\boldsymbol{\theta}_{l+1,p(i)}} d\boldsymbol{\theta} \prod_{j \in \Lambda_{l,i}} P(\boldsymbol{Y}_j|\boldsymbol{\theta}_{l,i}\boldsymbol{\theta}) P(\boldsymbol{\theta}_{l,i}|\boldsymbol{\theta}) \qquad (15)$$

where $\boldsymbol{Y}_k$ are the data, which in the experiments consisted of the average colour in each of a set of S patches selected at random from the set associated with component (l, i). Direct elimination of the 'nuisance' variable $\boldsymbol{\theta}_{l+1,p(i)}$ is impractical, but can be avoided by using the MAP estimate $\hat{\boldsymbol{\theta}}_{l+1,p(i)}$. At the top level of the MGM tree, the prior is derived instead from the estimate at the previous time. The particle filter employs a Gaussian approximation to the posterior and used the prior as the importance density, simplifying the computations [10]. A summary of the main steps in the algorithm is:

For levels $L > l \geq 0$,

1. Sample from importance density, $P(\boldsymbol{\theta}_{l,i}|\hat{\boldsymbol{\theta}}_{l-1,p(i)})$, to get weights $w_{l,i}, 1 \leq i \leq M_l$.
2. Compute likelihoods at current estimate $\theta_{l,i}$, $P(Y_j|\boldsymbol{\theta}_{l,i})$.
3. Use weighted likelihoods to estimate posterior mean and covariance.

Using the prior derived from the level above as the importance density both simplifies the computation and provides a good initialisation of the particle filter. This simply amounts to a rotation and translation of the mean and a rotation applied to the covariance in the obvious way, eqns (1)-(5). To understand how the motion field is constructed from the Gaussian components, note that at the finest level, we have a motion for each component $(0, i)$, consisting of two

elements: a translation vector $\boldsymbol{t}_{0,i}$ and a rotation matrix $\boldsymbol{R}_{0,i}$. Now, given an arbitary point in 3D, $\boldsymbol{x}$, say, there are corresponding probabilities $P(i|\boldsymbol{x})$ that it belongs to the ith component,

$$P(i|\boldsymbol{x}) = w_i N(\boldsymbol{x} - \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) / \sum_j w_j N(\boldsymbol{x} - \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{16}$$

where $N(.,.)$ is the 3D normal density with parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ for the ith component and w_i is a weight representing the population of the ith component. Where the MGM model uses additional data, such as colour or velocity, the projection onto the three spatial dimensions remains Gaussian. Now the translation at $\boldsymbol{x}$ is simply the conditional mean

$$\boldsymbol{t}(\boldsymbol{x}) = \sum_i P(i|\boldsymbol{x}) \boldsymbol{t}_i \tag{17}$$

and the rotation may be similarly interpolated, in common axes, using the exponential form

$$\boldsymbol{R}(\boldsymbol{x}) = \prod_i \boldsymbol{R}_i^{P(i|\boldsymbol{x})} \tag{18}$$

These are used to update the positions of the individual data elements from the estimated motion at the highest resolution in the MGM tree.

5 Experiments

Experiments were carried out on a number of sequences captured in the Department's studio, as well as two from the Microsoft Research site [9]. All were captured at 1024×768 pixel resolution; the Microsoft sequences are at 15 fps, the others at 30 fps. Unlike the Microsoft camera arrangement, which has eight cameras arrayed in a line, with a maximum disparity between adjacent cameras of 150 pixels, the Department cameras are arranged to provide full coverage of the room in a non-uniform way: 32 cameras are arranged in a recatngular array on one wall and the remainder in the corners and mid-points of each of the remaining three walls, with one in the ceiling space. The overlap between cameras is at most 50%, resulting in disparities of up to 500 pixels. Consequently, accurate modelling and rendering is a far more challenging problem for these data.

The MGM models were built with either two or three levels, with respectively $M_1 = 50,\ M_2 = 5$ and $M_1 = 250,\ M_2 = 50,\ M_3 = 5$ components, an arrangement that was chosen empirically, given the roughly 100000 points sampled from the 48 camera array. It was found that the Gibbs sampler for the clustering converged after 200 iterations and this was chosen for the estimation. The prior weights were set to $\alpha_l = M_{l-1}/M_l, r_l = 8, \tau_l = 1$ and $W_{l,i}$ was diagonal, with variances based on the spatial and colour variances, after extensive experimentation. The MGM approach was compared to publicly available K-means and hierarchical clustering algorithms, both in terms of model fit,

measured by squared error and computation times on different data sets. The results are summarised in Table 1, showing that MGM significantly outperforms the other methods in terms of both error and computation. The appearance of the induced clusters is illustrated in figure 3, which shows the cluster centres in colour for two sequences.

Motion estimation was only practicable for the locally captured sequences. The Gaussian particle filter used 10000 particles and the likelihoods were calculated using a set of at most 2000 patches for each component. The MGM model and motion estimator were reinitialised every 25-30 frames, to prevent error accumulation, using the visual hull. These choices were arrived at after experimentation and were found to give good results across the whole set of data. To validate the algorithms, a 'ground truth' sequence was obtained by manually tracking a finger in the 'Kate' sequence. The average error over 30 frames was of the order of 15mm, comparing favourably with the best results reported to date (figure 1). To demonstrate the estimator's ability to produce a dense motion field, it was used to animate a manually initialised skeleton, as shown in figure 3. Despite having no constraints on joint angles, the results were quite compelling. Finally, views synthesised for the 'Kate' and 'Breakdance' sequences are shown in figures 4 and 5. The reconstructions contain a few rendering artefacts, which result in signal-noise ratios of the order of 25-30dB, somehat worse than the figures obtained by disparity based warping [2], as might be expected. Figure 2 shows the reconstruction peak-rms SNR for both sequences, based on a 'leave-one-out' test, in which a known view is reconstructed from the remaining data. Of course, this is hardly the aim of free viewpoint video, but it does provide a quantitative means of comparison. Overall, the results demonstrate the effectiveness of a Bayesian approach within the MGM model framework.

6 Conclusions

This paper has provided an overview of a general model of dynamic scenes, using an MGM model of the data and motion estimators derived directly from that model, for use in multi-camera video. It is capable of modelling complex motions, while providing a set of constraints through the use of a hierarchical Bayesian motion estimator, speeding up computation and preventing the problem from becoming ill-posed. The motion estimates can be interpolated using the mixture model to animate arbitrary objects with motions derived from a given sequence. Although not real time in terms of computation, it has been observed that both the model construction and the particle filtering converge relatively quickly because of the strong priors, so that typically a few hundred iterations are sufficient to obtain satisfactory results. Moreover, the methods provide real time rendering of novel views. While more restrictive models will yield better results for particular applications, the MGM approach is powerful and general.

	K-Means	Hierarchical Clustering	MGMM
'Kate' sequence	327(789)	394(7839)	131(106)
'Hamutal' sequence	179(177)	200(802)	119(56)
'Breakdancers' sequence	132(121)	124(340)	73(24)
'Ballet' sequence	117(30)	113(14)	66(20)

Table 1. Energy (computation time in sec) of the final clustering in thousands; lower values indicate a bet ter fit.

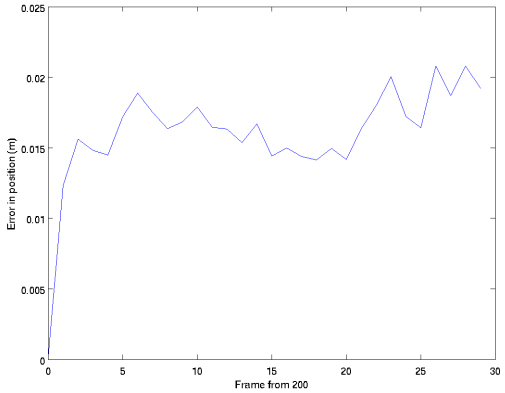


Fig. 1. Plot of the error in position vs frame number for the tracked index finger in 'Kate'

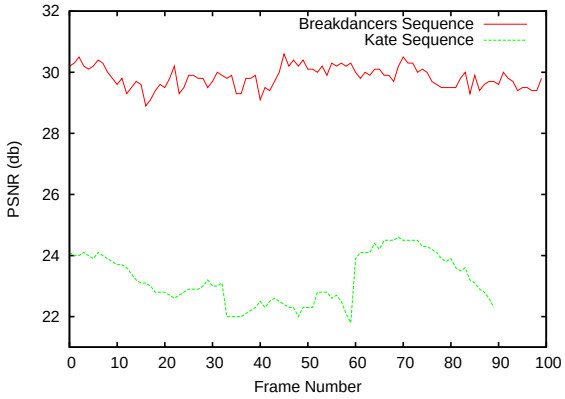
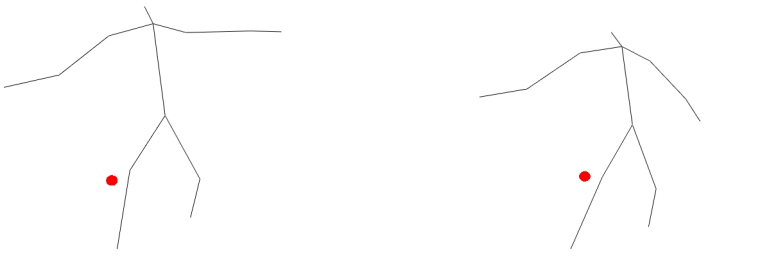


Fig. 2. Plot of the PSNR vs. frame number for the synthesised views.

References

1. K. M. Cheung, S. Baker and T. Kanade:
Shape from silhouette across time part I: theory and algorithms.
Int. J. Comput. Vision, 62:221–247 (2005).
2. C. L. Zitnick, S. B. Kang and M. Uyttendaele:
High-quality video view interpolation using a layered representation.
ACM Trans. Graphics, 23:600–608 (2004).
3. J. Mitchelson and A. Hilton:
Hierarchical tracking of multiple people.
Proc. BMVC-03, Norwich, (2003).
4. A. Bowen, A. Mullins, R. Wilson and N. Rajpoot:
Video based Rendering Using Surface Patches.
Proc. IEEE 3DTV Conf., Kos, (2007).
5. A. Mullins:
Stochastic Geometry Estimation for Video based Rendering.
PhD Thesis, University of Warwick, (2008).
6. A. Bowen:
Video based Rendering and Coding using a Planar Patch Model.
PhD Thesis, University of Warwick, (2008).
7. X. Mo and R. Wilson:
Video modelling and segmentation using Gaussian mixture models.
Proc. ICPR-04, Cambridge, (2004).
8. G. McLachlan and D. Peel:
Finite Mixture Models.
Wiley, New York (2000).
9. http://research.microsoft.com/vision/InteractiveVisualMediaGroup/
10. J. M. Kotecha and P. M. Djuric:
Gaussian particle filtering.
IEEE Trans. Sig. Proc., 51:2592–2601 (2003).



(a) Skeleton at time 0 (b) Skeleton at time 80

Fig. 3. Skeleton, animated using motion field from 'Kate'.

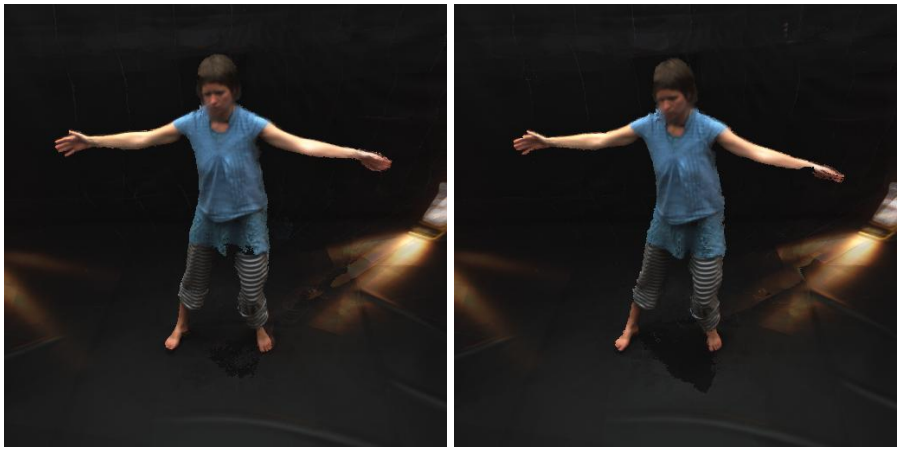
(a) Coloured centroids for the breakdancers sequence

(b) Coloured centroids for the 'Kate' sequence

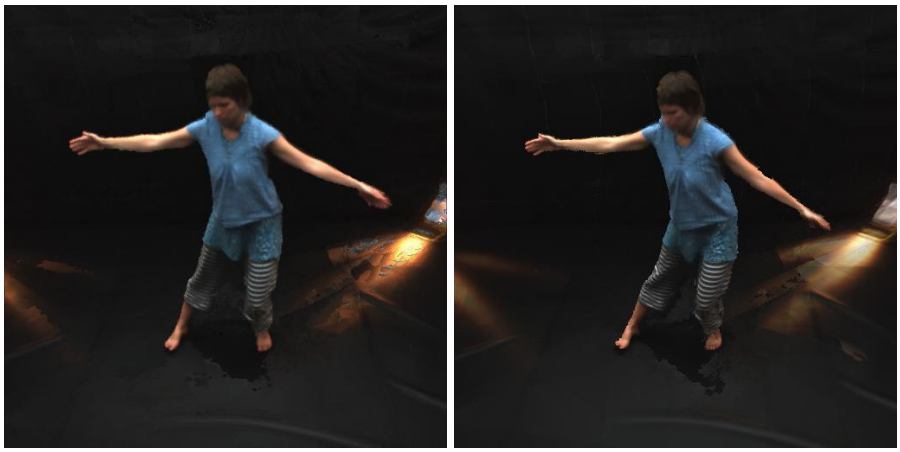
(c) The clustering of 'breakdance' at t=0.

(d) The clustering of 'Kate' at t=0.

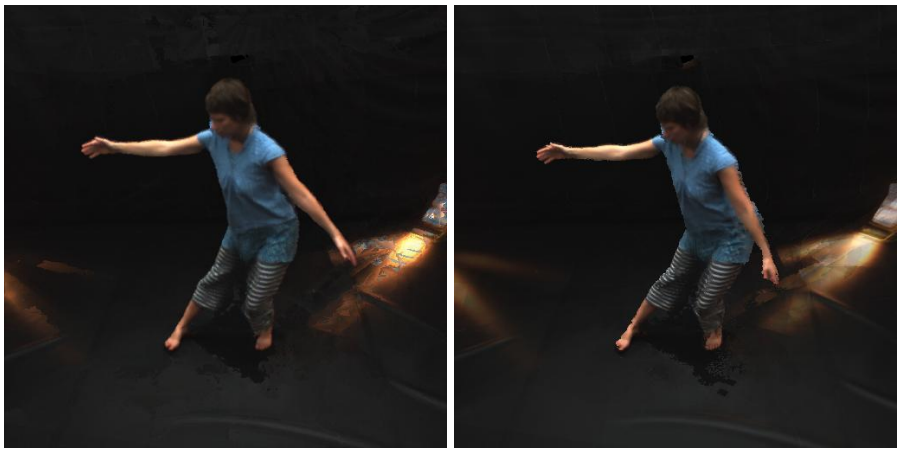
Fig. 4. Clustering the 'Breakdancers' and 'Kate' data sets using uniform and seeded priors.



(a) Frame 0 (b) Frame 15

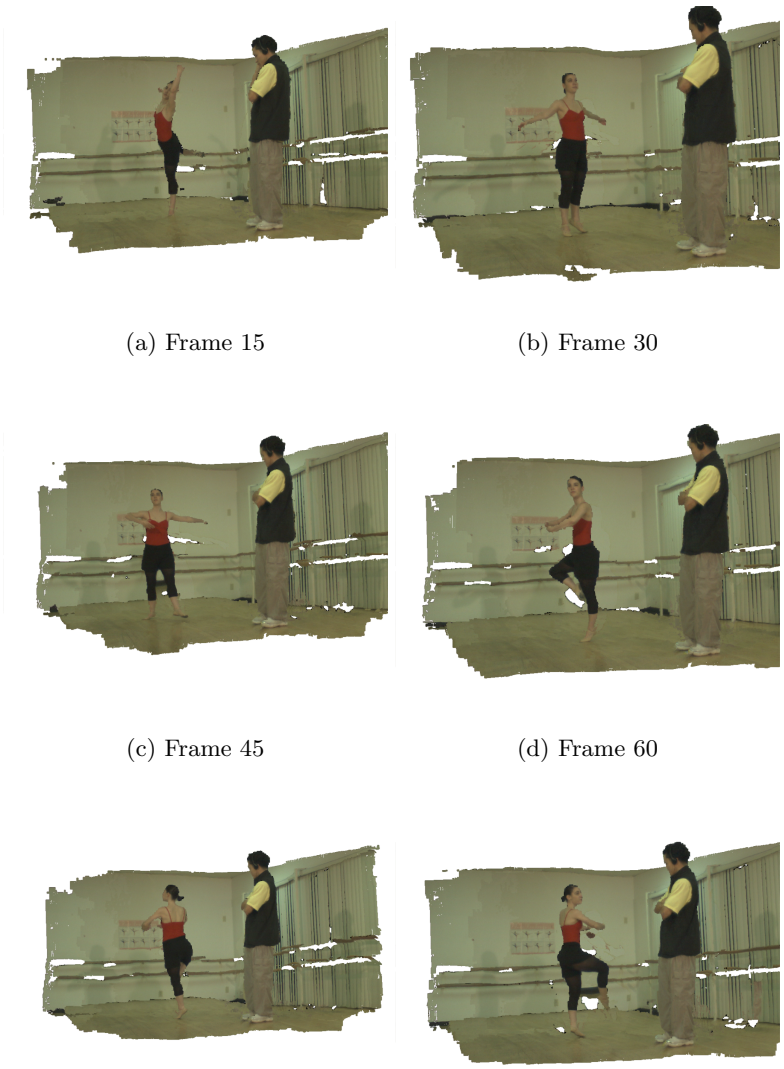


(c) Frame 30 (d) Frame 45



(e) Frame 60 (f) Frame 75

Fig. 5. Synthesised views of the 'Kate' sequence at half second intervals.



(a) Frame 15 (b) Frame 30

(c) Frame 45 (d) Frame 60

(e) Frame 75 (f) Frame 90

Fig. 6. Synthesised views of the 'Ballet' sequence at one second intervals.