SURFACE ESTIMATION AND TRACKING USING SEQUENTIAL MCMC METHODS FOR VIDEO BASED RENDERING

Adam Bowen, Andrew Mullins, Roland Wilson, Nasir Rajpoot

University of Warwick
Coventry, CV4 7AL, UK

ABSTRACT

Video based rendering algorithms attempt to render videos of a scene from an arbitrary viewpoint, given a set of input video sequences taken from several fixed viewpoints. These algorithms require either a dense camera array or some knowledge of scene structure. By applying sequential Markov Chain Monte Carlo (MCMC) methods, we show it is possible to estimate the surfaces visible within a scene, and track them over time, in an efficient manner. Initially, a particle filter is applied across image scale to estimate the surfaces present in a scene at a fixed point in time. Following this, surfaces are tracked over time using a particle filter which takes advantage of both frame-to-frame dependancies, and a hierarchical surface model derived from a multiresolution Gaussian mixture model analysis of the surface data. This time-varying surface model, and the images, are the input for a rendering algorithm which uses a fuzzy z-buffer and projective texturing to generate reconstructions.

Index Terms— Rendering, Monte Carlo methods, Tracking filters, Gaussian mixture models.

1. INTRODUCTION

Image based rendering algorithms synthesise images of a scene from an arbitrary viewpoint, given a finite number of real images of the scene as input. The scene geometry must be taken into account (either explicitly, e.g. [3] or implicitly, e.g. [2]) to produce realistic renderings, unless the number of input images is very large (e.g. [6]). A logical extension of image based rendering is video based rendering [12], [4]. Video based rendering algorithms also typically require some estimate of the scene geometry, but in the case of video based rendering, these estimates will change over time as objects move within the scene.

Existing attempts to represent scene geometry typically use voxels [11], or a wireframe mesh [9]. Voxels provide a low-level representation of the surfaces, and are constrained by their resolution. In addition it is difficult to estimate voxels for concave objects. Mesh representations of surfaces necessarily make assumptions about the number and topology of objects within a scene. The solution proposed in [8] is to use a 'patch' based representation which assumes that surfaces within a scene are locally smooth. By estimating the position and orientation of a set of surface elements, the scene geometry is captured in a general, easily estimated way.

In this paper extend our previous results to show how surfaces are tracked over time to allow for dynamic as well as static rendering. This is followed by a description of the rendering algorithm, and some example renderings using a real data set. The paper concludes with a discussion of the potential of our methods.

2. ALGORITHM OVERVIEW

The algorithm for estimating surfaces consists of four main components.

I Surface patches are estimated at time $t = 0$ using a multiresolution particle filter (Section 3, or for details see [8]).

II Surface patches are clustered to form a multiresolution Gaussian mixture hierarchy using MCMC (Section 4).

III The hierarchy of surfaces patches is tracked from frame to frame using sequential MCMC methods (Section 5).

IV The resulting surface estimates and the original images are used to render, in real-time, a sequence from an arbitrary viewpoint (Section 6).

3. STATIC SURFACE PATCH ESTIMATION

Given a set of input images of a scene, the initial aim is to create a set of 'patches' which describe the visible surfaces. This is performed by partitioning each input image into blocks, and assuming that each of these blocks corresponds to some planar quadrilateral in 3-D space. The position and orientation of these quadrilaterals is then estimated, using a multiresolution particle filter [8]. The output of this algorithm, is a vector for each patch m, composed of its centroid, p_m, and normal n_m,

$$x_m = \begin{bmatrix} p_m \\ n_m \end{bmatrix}. \quad (1)$$

Once the initial surface estimation has taken place, it is simply updated as new frames of the multiple video sequences arrive.

4. HIERARCHICAL CLUSTERING

Tracking the motion of every patch independently is a computationally expensive task. To simplify the tracking a hierarchical Gaussian Mixture Model (GMM) of the scene is constructed, so that related patches can be tracked as a group rather than individually. Each vertex in the tree represents the data in the corresponding subtree. Coarse motion estimates on high level models of the data may then be sequentially propagated down the hierarchy.

This greatly simplifies the motion estimation, whilst allowing the number of degrees of freedom of motion to be controlled by estimating for the appropriate scales.

The set of these vectors, $x_j \in \mathcal{X}$, is modelled as a mixture of K Gaussians

$$x_m \sim \sum_{i=1}^{K} \pi_i N(\mu_i, \Sigma_i) \tag{2}$$

where $N(\mu, \Sigma)$ is an unconstrained multivariate Normal distribution, π_i is the weight, or probability, of a draw from the i^{th} component and is constrained such that

$$\sum_{i=1}^{K} \pi_i = 1 \tag{3}$$

$$0 \leq \pi_i \leq 1 \quad i \in 1..K \tag{4}$$

μ_i is the mean and Σ_i the covariance of component i.

The number of components, K, is chosen to ensure that the data can be modelled in a flexible way, to cater for non-rigid, articulated motions whilst avoiding overfitting. For N data points, K is typically set to $K \approx N/30$.

The components are estimated using a Gibbs Sampling process [7] run for a small number of iterations (approximately 200) due to the complexity of the model fitting. Half the iterations are discarded as 'burn-in'. Despite this the models constructed form a reasonable approximation to the data.

For computational simplicity we use conjugate priors for the mixture [7]. Although these have their limitations we have found them adequate in this application. The prior for π is modelled using a Dirichlet distribution, the mean of each component a normal distribution, and the covariance a Wishart distribution.

The aim of the GMM process is to form a coarse model of the data, so it is desirable that the prior represents the approximate distribution of data points. To achieve this K data points are chosen at random to act as *seed points*. Each seed point, k, provides a prior on the mean of the k^{th} component, with the variance modelled by a symmetric Wishart distribution $W_k = wI$ where w is a scale parameter. The Dirichlet parameter is set to a constant indicating that the components are expected to have an equal number of sample points each.

The initial clusters are generated by assigning each data point x_i to the closest seed point. It is also possible to assign components randomly, but this results in numerical difficulties and requires more iterations for similar results.

The estimation of multiresolution Gaussian Mixtures has been previously discussed in [10]. However, the method of construction is top-down which is inappropriate due to difficulties with choosing the number and nature of splits at coarse scales. Instead we adopt the following bottom-up approach.

At the finest scale are the original data points $\mathcal{X}$. Let this be level 0 of the hierarchy $\mathcal{X}_0$ and generate a mixture model as described above:

$$\mathcal{M}_0 = \{\pi_{0,j}, \mu_{0,j}, \Sigma_{0,j}\} \tag{5}$$

where $j \in 1..K_i$. The data at scale $(i+1)$ are now described in terms of the components of the model at scale i, M_i

$$\mathcal{X}_{i+1} = \{\mu_j | j \in 1..K_i\} \tag{6}$$

to which it is now possible to fit a further mixture model $\mathcal{M}_{i+1}$ using the Gibbs Sampling process, except that the seed points are now drawn according to the component weights $\pi_{i,j}$ rather than uniformly from the set of data points. Successive levels of the multiresolution structure are constructed in this manner until the number of components has reduced to a small number, usually 1.

5. DYNAMIC SURFACE PATCH ESTIMATION

The first step is to take the set of mixture model means for each resolution of the GMM, and transform them into sets of patch parameters from the original data set. This is done by simply choosing the patch from the original data set, which is closest to each component's mean at that resolution:

$$\mathcal{X'}_i = \{x_m | \arg\min_{x_m} \|x_m - \mu_j\|, \mu_j \in \mathcal{X}_i\}. \tag{7}$$

As part of the Gibbs Sampling process, each component from a given resolution of the GMM is associated with a component from the next lowest resolution, which may be viewed as its 'parent'. By applying the same association to the newly created multiresolution data sets $\mathcal{X'}_0 \ldots \mathcal{X'}_L$, a multiresolution tree structure is formed. The tree structure will be described using the function $g(m)$, which will be taken to mean the parent patch of patch m.

Motion estimates are then calculated, starting with the coarsest patches, represented in the set $\mathcal{X'}_L$. The motion for each patch m in this set, at time t, is parameterised as a 6-D motion vector, with three translation components, and three Euler rotation angles, $y_{mt} = (t_x, t_y, t_z, \theta, \phi, \psi)^T$.

A particle filter [1] is used to estimate the motion for these patches, as more image frames become available. The particle

filter represents probability distributions as a set of samples, with associated weights. So, the motion distribution for the patch m at time t is represented as

$$Y_{mt} = \{y^0_{mt} \dots y^S_{mt}\}, \quad (8)$$
$$W_{mt} = \{w^0_{mt} \dots w^S_{mt}\}. \quad (9)$$

As a new set of image frames $\mathcal{Z}_t$ is recorded, the weights are updated as

$$w^s_{mt} = w^s_{m,t-1} p(\mathcal{Z}_t | y^s_{mt}) \quad (10)$$

where $p(\mathcal{Z}|y)$ is the measurement likelihood, where the measurement $\mathcal{Z} = \{z_1 \dots z_Z\}$ is the set of images taken at time t from the set of cameras $\mathcal{C} = \{c_1 \dots c_Z\}$. To calculate this measurement likelihood, we first consider that if a patch m originally corresponded to a $P \times P$ pixel block, its original position and orientation x_m define a $P \times P$ set of points in 3-D space, corresponding to each of the original pixels. These points shall be denoted:

$$\mathcal{P}_m = \{p_1 \dots p_{P \times P}\}, \quad (11)$$

and each has a corresponding colour associated with it from the original image:

$$\mathcal{L}_m = \{l_1 \dots l_{P \times P}\}. \quad (12)$$

We define the matrix T_y to be the transformation matrix implied by the motion parameter y. If $f(c,p)$ is the pixel to which point p projects in camera c, then the likelihood may be evaluated as

$$p(\mathcal{Z}|y,m) = \quad (13)$$
$$\frac{1}{Z \times P \times P} \sum_{n=1}^{Z} \sum_{m=1}^{P \times P} N(z_n(f(c_n, T_{ym}.p_m)) - l_m; 0, q)$$

for a given patch m. To improve the robustness of this likelihood measure, we may estimate not only the likelihood for the patch we are estimating, m, but also some subset of its children T assuming that they undergo the same rigid motion. Thus, the final likelihood is

$$p(\mathcal{Z}|y) = |T|^{-1} \sum_{m \in \mathcal{T}} p(\mathcal{Z}|y,m) \quad (14)$$

The samples and weights now form an estimate of the posterior distribution $p(y_m|z_t \dots z_0)$. To form the prior distribution $p(y_{m,t+1}|y_{mt})$ for the next image frames, we assume constant motion. Due to memory constraints, this is implemeted by resampling the distribution assuming it to be Gaussian, making this a Gaussian Particle Filter [5]. In other words, if $N(Y_{mt}, W_{mt})$ is the weighted normal distribution for a given set of samples and weights at time t, then the new samples are drawn as

$$y^s_{mt} \sim N(Y_{m,t-1}, W_{m,t-1}), \quad (15)$$

and the weights must be adjusted accordingly, before being updated with (10).

$$w'^s_{m,t-1} = N(y^s_{mt}; Y_{m,t-1}, W_{m,t-1}). \quad (16)$$

In summary, for each frame, the prior distribution of the motion vector is updated to become the posterior motion distribution. In this iterative process, the posterior distribution becomes the prior distribution for the next frame of video.

The motion for a patch m derived from a finer resolution of the GMM is estimated similarly, except that the prior at each time step is a weighted combination of the previous frame's posterior distribution for this patch, and the parent patch $g(m)$'s distribution for the current frame. The samples for evaluation are drawn as

$$y^s_{mt} \sim \alpha N(Y_{m,t-1}, W_{m,t-1}) + \beta N(Y_{g(m),t}, W_{g(m),t}) \quad (17)$$

where $\alpha + \beta = 1$.

6. RECONSTRUCTION

Reconstructions are generated by projecting a subset of the original images onto the patch quadrilaterals generated by blocks from those images. Fuzzy depth buffering is used to smooth out inaccuracies in estimation, but this still retains a sizeable portion of noise generated by poorly estimated geometry. Fortunately, the density of patches within a given volume provides a strong cue as to which parts of the scene contain real surfaces. Since the Gaussian Mixture Model is implicitly estimating this density, we refine the reconstruction algorithm by filtering out reconstructed pixels that do not fit the assigned mixture component well.

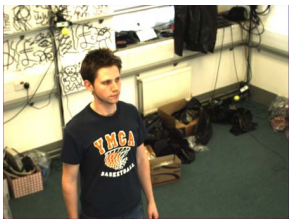
The quadrilateral corresponding to each patch m from camera c is rendered by projecting image I_c onto the quadrilateral. For each pixel written it is possible to find the position x of the intersection between the ray emitted through the pixel and the quadrilateral and the corresponding normal n (which is constant for planar patches). The patch m is assigned to component k_m, and pixels with low likelihood are thresholded out.

$$N([x;n] | \mu_{k_m}, \Sigma_{k_m}) < \tau \quad (18)$$

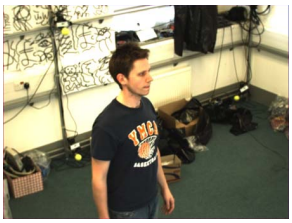
where τ is the threshold parameter.

7. RESULTS

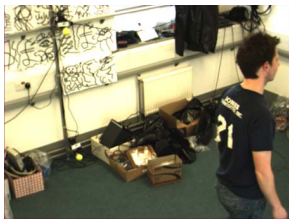
Figure 7 shows a number of results for our natural 'andy' dataset. The sequence is approximately 60 frames long from 40 viewpoints. Each camera captures frames at 1024×768 and 30 frames per second. Whilst still demonstrating some noticeable artefacts the reconstructions are of a reasonable quality and the model fits the data well.



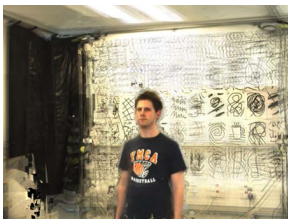
(a) Original view of frame 0



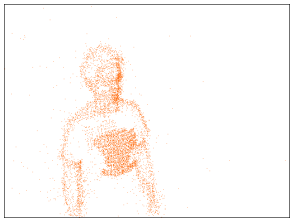
(b) Original view of frame 40



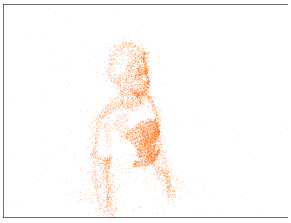
(c) Original view of frame 80



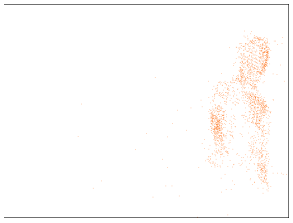
(d) Synthesised novel view at frame 0



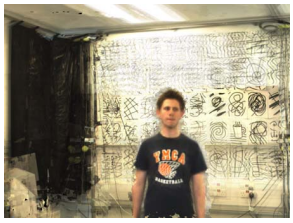
(e) Model at frame 0



(f) Model at frame 40



(g) Model at frame 80



(h) Synthesised novel view at frame 40

8. CONCLUSIONS

By choosing the right representation for the surfaces present within a scene, we have shown that it is possible to easily track the objects in a scene over time, and to render those objects from an arbitrary viewpoint. Further work needs to be carried out to compress the scene geometry estimates and image data to create data sets of a manageable size.

9. ACKNOWLEDGEMENTS

This research is funded by EPSRC project 'Virtual Eyes', grant number GR/S97934/01.

10. REFERENCES

[1] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[2] A.W. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *Ninth IEEE ICCV, Nice, France, October 2003*, volume 2, pages 1176–1183, October 2003.

[3] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of ACM Siggraph '96, New Orleans, LA, August 1996*, pages 43–54. ACM Press, New York, August 1996.

[4] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 4(1):34–47, 1997.

[5] Jayesh H. Kotecha and Petar M. Djuric. Gaussian particle filtering. *IEEE Transactions On Signal Processing*, 51(10), 2003.

[6] Marc Levoy and Pat Hanrahan. Light field rendering. *Computer Graphics*, 30(Annual Conference Series):31–42, 1996.

[7] Geoffrey McLachlan and David Peel. *Finite Mixture Models*, chapter 4, page 123. Wiley Series in Probability and Statistics. Wiley-Interscience, 2000. ISBN 0-471-00626-2.

[8] Andrew Mullins, Adam Bowen, Nasir Rajpoot, and Roland Wilson. Multiresolution particle filters in image processing. In *Proceedings of the Mathematics in Signal Processing Conference*, 2006.

[9] P. Ramanathan, E. Steinbach, P. Eisert, and B. Girod. Geometry refinement for light field compression.

[10] Roland Wilson. MGMM: Multiresolution Gaussian Mixture Models for Computer Vision. In *ICPR '00: Proceedings of the ICPR*, page 1212, Washington, DC, USA, 2000. IEEE Computer Society.

[11] S. Vedula, S. Baker, S. Seitz, and T. Kanade. Shape and motion carving in 6d. In *Proceedings. IEEE Conference on Computer Vision and Pattern Recognition, 2000.*, volume 2, pages 592–598, 2000.

[12] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004.