VIDEO BASED RENDERING USING SURFACES PATCHES

Andrew Mullins, Adam Bowen, Roland Wilson, Nasir Rajpoot

University of Warwick
Coventry, CV4 7AL, UK

ABSTRACT

This paper describes a novel method to perform video based rendering. By capturing a set of real video sequences of a scene, the aim is to render a video sequence, in real time, from any viewpoint. By modelling the surfaces of a scene as a set of disjoint planar patches, we are able to efficiently estimate the parameters of the scene geometry. The patches can then be tracked over time using a multiresolution hierarchy. This time-varying surface model, and the images, are the input for the rendering algorithm, which uses a fuzzy z-buffer and projective texturing to generate reconstructions.

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 to produce realistic renderings, unless the number of input images is very large [3]. A logical extension of image based rendering is video based rendering. Image and video based algorithms typically require as input the original image textures, and some estimate of the scene geometry. In the case of video based rendering, the estimates of scene geometry will change over time.

Existing attempts to represent scene geometry typically use voxels [8], or a wireframe mesh [6]. 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 [5] 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, we capture the scene geometry in a general, easily estimated way.

In this paper we present an brief introduction to the surface representation, followed by a description of how this representation is updated 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.

2. 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 surfaces present. 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 quarilaterals is then estimated, using a multiresolution particle filter [5]. The output of this algorithm, is a vector for each patch m, composed of its centroid, p_m, and normal n_m,

$$x_m = \begin{bmatrix} p_m \\ n_m \end{bmatrix}. \tag{1}$$

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

3. HIERARCHICAL MODEL

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.

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 choice of the number of components K is problematic since there is little a priori knowledge of how many are required to adequately represent the data.

K is set to some fixed fraction of the number of data points, N, although this introduces the potential for over fitting. Since the observations, x_j, are 6 dimensional 27 parameters must be estimated for each component and so K is set to offer just enough data to fit the components:

$$K \approx N/30 \tag{5}$$

Although selecting such a large number of components is liable to over fit the data the effects can be overcome by the multiresolution structure introduced in this section which progressively fits coarser and coarser models. The components are estimated using a Gibbs Sampling process [4] 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 [4]. 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 around which a symmetric Wishart distribution $W_k = wI$ where w is a scale parameter is assumed. The Dirichlet parameter is set to a constant value indicating that the components are expected to have an equal number of sample points each.

The initial assignments 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.

We extend the Gaussian Mixture Model to a multiresolution framework to construct a Multiresolution Gaussian Mixture Model (MGMM). The top down approach described in [7] is not applicable here, because we split into more than two components at each scale and choosing the number and method of splits is non-trivial. Instead we apply 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:

$$\mathcal{M}_0 = \{\pi_{0,j}, \mu_{0,j}, \Sigma_{0,j}\} \tag{6}$$

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

$$\mathcal{X}_{i+1} = \{\mu_j | j \in 1..K_i\} \tag{7}$$

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.

4. 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:

$$\mathcal{X}'_i = \{x_m | \arg\min_{x_m} \|x_m - \mu_j\|, \mu_j \in \mathcal{X}_i\}. \tag{8}$$

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 $\mathcal{X}'_0 \dots \mathcal{X}'_L$, a multiresolution tree structure is formed, which will be described using the function $f(m)$, which will be 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,

$$y_{mt} = (t_x, t_y, t_z, \theta, \phi, \psi)^T. \tag{9}$$

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

$$\begin{aligned} Y_{mt} &= \{y^0_{mt} \cdots y^S_{mt}\}, &(10)\\ W_{mt} &= \{w^0_{mt} \dots w^S_{mt}\}. &(11) \end{aligned}$$

As a new set of image frames z_t is recorded, the weights are updated as

$$w^s_{mt} = w^s_{m,t-1} p(z_t | y^s_{mt}) \tag{12}$$

where $p(z|y)$ is the measurement likelihood pdf, and is based upon a comparison of the texture of the patch m at time $t-1$, with the new images z, given that they undergo motion of sample y. The more similar the pixel values are, the more likely we deem the sample to be.

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 [2]. 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

$$y^s_{mt} \sim N(Y_{m,t-1}, W_{m,t-1}), \tag{13}$$

and the weights must be adjusted accordingly, before being updated with (12).

$$w'^s_{m,t-1} = N(y^s_{mt}; Y_{m,t-1}, W_{m,t-1}). \tag{14}$$

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 $f(m)$'s distribution for the current frame. The samples for evaluation are drawn as

$$y^s_{mt} \sim \alpha N(Y_{m,t-1}, W_{m,t-1}) + \beta N(Y_{f(m),t}, W_{f(m),t}) \tag{15}$$

where $\alpha + \beta = 1$.

5. 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 also used to smooth out inaccuracies in estimation, however 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 hint as to which parts of the scene contain real surfaces. Since the Gaussian Mixture Model is implicitly estimating this density we refine our 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, thus we threshold out pixels with low likelihood.

$$N\left([x; n] \,|\mu_{k_m}, \Sigma_{k_m}\right) < \tau \tag{16}$$

where τ is the threshold parameter.

6. RESULTS

Figure 6 shows a number of results for our natural 'andy' dataset. The sequence is approximatly 60 frames long from 40 viewpoints. Each camera captures frames at 1024×768 and 30 frames per second. Whilst still demonstrating some noticable artefacts the reconstructions are of a reasonable quality and the model fits the data well.

7. 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.

8. ACKNOWLEDGEMENTS

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

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

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

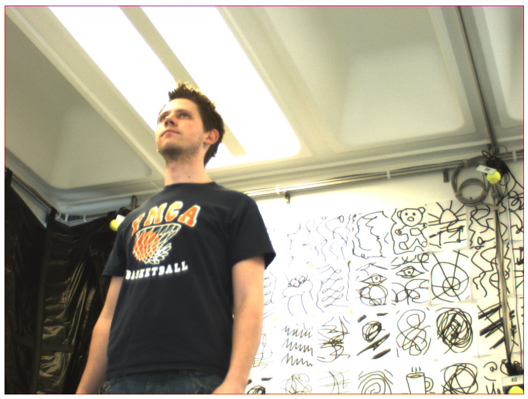
[4] Geoffrey McLachlan and David Peel. *Finite Mixture Models*, chapter 4, page 123. Wiley Series in Probability and Statistics. Wiley-Interscience, 2000. ISBN 0-471-00626-2.

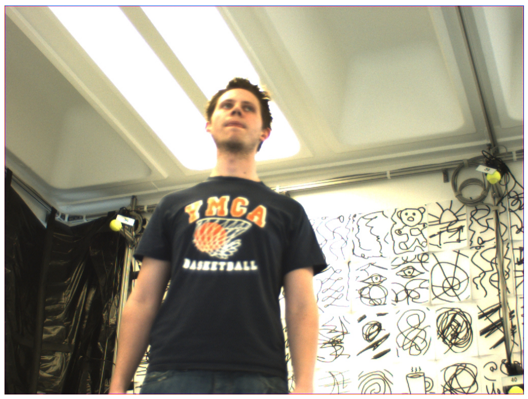
[5] Andrew Mullins, Adam Bowen, Nasir Rajpoot, and Roland Wilson. Multiresolution particle filters in image processing. In *Proceedings of the Mathematics in Signal Processing Conference*, 2006.

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

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

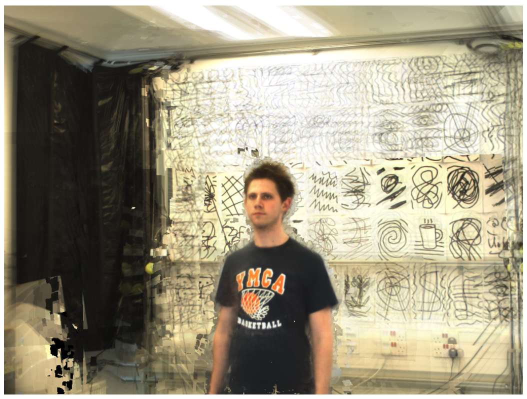
[8] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation tech. rep. cmu-ri-tr-01-35. September 2001.

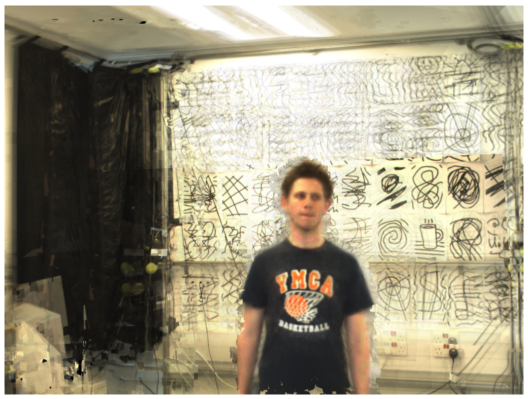

(a) Original view of frame 250


(b) Original view of frame 290

(c) Model at frame 250

(d) Model at frame 290


(e) Synthesised novel view at frame 250


(f) Synthesised novel view at frame 290