Scene Reconstruction from Multiple Views Using a Locally Planar Model

Andrew Mullins, Adam Bowen, Roland Wilson, Nasir Rajpoot

University of Warwick
Coventry, CV4 7AL, UK
Email: {andy,fade,rgw,nasir}@dcs.warwick.ac.uk

Abstract

Image based rendering techniques allow reconstructions of a scene from an arbitrary viewpoint, given views of the scene. In this paper a locally planar representation of the scene structure is presented, where each block of pixels in the input images is assumed to correspond to some patch of an object's surface. A multi-resolution particle filter method of estimating the position and orientation of these patches is described. This method offers fast computation without sacrificing accuracy. The estimated patches often include a significant amount of redundant information. We present an algorithm for identifying this redundancy and producing a compressed data set. Finally we introduce a simple, fast, algorithm for generating reconstructions of the scene. Results on both synthetic and real data sets are presented to demonstrate the effectiveness of our techniques.

1 Introduction

Image based rendering is an increasingly important field, and the focus of our work. The goal is to synthesise novel views from images of a scene or object, typically without any geometry information. It is the lack of additional information, coupled with the desire for artefact free reconstructions, that makes image based rendering a challenging problem. The ease of data capture is what makes the field so appealing. The most common approaches derive from modelling the plenoptic function [1] via some form of image warping [3, 12, 11] and from geometry estimation approaches such as space carving [7, 4] and Patchlets [9].

The simplest image based rendering approaches, such as the light field [6], often fail when the scene geometry is fairly complex. This is the case for most scenes of interest, so they are frequently let down by their lack of information on the scene geometry. Geometry estimation approaches often rely on a number of assumptions about the structure of a scene and thus cause issues during the reconstruction phase. Space carving approaches, for example, rely on the scene being Lambertian and objects convex whilst image warping from depth maps suffers from all of the common problems associated with forward warping such as poor interpolation and holes. We wish to estimate scene structure in a context that will help alleviate many of these issues.

Estimating the structure of a scene from two or more views, separated either in space or time, is a fundamental problem in vision. It is the estimation of structure from many spatially separated views which we are concerned with in image based rendering, although temporal information may provide useful information too: see section 8, *Conclusions and Future Work*.

Algorithms for estimating the structure of a scene from multiple views are invariably based on an underlying model of the scene geometry and imaging geometry. For example, block based disparity estimation assumes that each $P \times P$ pixel block of one image is transformed by a simple translation into other images of the same scene. Other techniques assume that each $P \times P$ pixel block undergoes some more general transformation into other images. However, because these local methods do not explicitly reflect the geometry of scene structure and projection, they do not scale well to multiple cameras.

In [13], optical flow constraints in each view are related to the parameters of an icosahedron mesh. By solving for the parameters of this mesh using least squares, a polygon model of the object is constructed. This algorithm explicitly models the shape of an object, and the projective imaging geometry of the capturing cameras, but it makes assumptions

about the quantity and topology of the objects in the scene under inspection.

The algorithm presented in this paper also models perspective imaging geometry, and in addition provides a very general model for surfaces present within a scene. It is assumed that each $P \times P$ block of pixels in an image corresponds to some quadrilateral patch of a surface in the scene, which is parameterised by its position and orientation. There are conditions under which this assumption may not hold (for example, if the block contains a curved surface, or two separate surfaces), but at some scale of image, the assumption seems reasonable. This general surface model allows for the representation of multiple objects of arbitrary topology.

For the estimation of model parameters, multiresolution techniques have been widely applied in image processing in areas such as motion estimation [14] and image segmentation [16]. Using a multiresolution representation of an image, coarse estimates may be computed at relatively little cost from the lowest resolution image. The results for a given image region are then propagated to the next highest resolution and refined.

Such multiresolution algorithms have two beneficial properties for the estimation of planar patch parameters. They are computationally efficient, which is important when searching for model parameters in a high dimensional space, as here. In addition, lower resolution results act as a smoothness constraint on higher resolution results. Thus, even though for the sake of generality neighbouring planar patches are assumed to be independent (i.e. they are not assumed to be part of a continous surface), any multiresolution algorithm will favour smooth, continous surfaces over discontinous ones.

Often, probabilistic methods are applied in order to determine the relative weighting of lower resolution estimates to higher resolution results, in the final output of a multiresolution algorithm. For example, multiresolution Markov random fields [16] have been applied to image segmentation and Kalman filters used across scale to estimate motion and disparity [14].

In recent years, with increasing computing power and memory size, the particle filter has become a popular alternative to the Kalman filter. By approximating the distributions of parameters with a set of samples, and respective weights, the range of problems to which recursive parameter estimation can be applied has been extended to include those systems where the relationship between the model parameters and measurements is highly non-linear.

In this paper the model and its parameterisation are outlined in section 3, followed by the description of a particle filter used to estimate the parameters to this model in section 4. We introduce a greedy algorithm for eliminating redundancy in the resulting patch estimates in section 5 and describe our algorithm for image based rendering from patches in section 6. Finally, the results of applying these algorithms to to two synthetic and one real data set are presented in section 7.

2 Input

A camera array is used to take a set of images of a single scene. Each of the N cameras is assigned an index n from the set of all indices $\mathcal{N} = \{1 \dots N\}$. Each camera has an image associated with it, with the set of all images being

$$\mathcal{I} = \{I_1 \dots I_N\} . \tag{1}$$

Each image is also decomposed as a Gaussian pyramid with K levels $G_n = \{I_{n1} \dots I_{nK}\}$ from lowest to highest resolution $1 \dots K$ (i.e. $I_{nK} = I_n$). The full set of Gaussian pyramids being

$$\mathcal{G} = \{G_1 \dots G_N\} . \tag{2}$$

Each camera n also has a position associated with it, the full set of positions being

$$\mathcal{C} = \{c_1 \dots c_N\} . \tag{3}$$

Finally, each camera has a projection matrix associated with it, the full set of matrices being

$$\mathcal{M} = \{M_1 \dots M_N\} . \tag{4}$$

In the case of synthetic test data sets, the positions and orientations of each camera are known a priori, and in the case of the real data sets, these matrices are estimated using standard camera calibration software [15].

In addition, as input to the algorithm, each camera n has a set of neighbouring cameras in the array, whose indices shall be denoted $\mathcal{N}_n \subseteq \mathcal{N}$.

3 Model Parameterisation

Every Gaussian pyramid level, for every image is partitioned into $P \times P$ blocks. Assuming all the input images are of equal size, each camera's pyramid level k, will have the same number of blocks, B_k. The blocks are numbered in row order within a given camera and pyramid level, with m denoting this index in the general case. In summary, each block may be indexed by its camera, level, and row order index, n, k, m. It will be convenient to define the function f, and its inverse f^{-1} which maps a block's row index m, to its centre pixel $(a, b)^T$ and vice versa, (i.e. $m = f((a, b)^T), f^{-1}(m) = (a, b)^T$. Also we define $g(c, d, m) = f(m) - (P/2, P/2)^T + (c, d)^T, 1 \leq c, d \leq P$ to be the c, d-th pixel coordinate of block m (e.g. for an 8×8 pixel block centred at $(32, 16)$, $g(1, 1, f^{-1}(32, 16)) = (29, 13)$).

For a given source camera n with a known projection matrix M_n, the centre of each block m at a given scale k, specifies a directional ray $r_{nkm} = M_n^{-1}.(a, b, 1)^T$ from the camera through the scene, where $(a, b)^T = f^{-1}(m)$. The quadrilateral patch corresponding to this block of pixels must be centred at some point along this ray. We specify the centroid p's position

$$p_{nkm} = c_n + z_{nkm} \frac{r_{nkm}}{|r_{nkm}|}, \tag{5}$$

by its distance z_{nkm} along this ray.

We specify the orientation of each patch, as two components of the normal vector $(v_{nkm}, w_{nkm}, 1)$ in an Euler frame which is aligned such that the patch's centroid is the origin, and $c_n = \alpha.(0, 0, 1), \alpha > 0$, i.e. the z-axis is aligned with the line from the centre of the patch to the source camera's position. This representation serves two purposes: first, no matter what v_{nkm}, w_{nkm} is chosen, the patch can never become oriented such that it is perpendicular to the ray from the camera, but may only asymptotically approach perpendicular; secondly, the relationship between v_{nkm}, w_{nkm}, and the region to which the patch reprojects in neighbouring cameras, is approximately linear. This contributes significantly to the stability of the proposed algorithm.

The complete state vector at resolution k, consists of all three parameters z, v, w, for each block, in row order, and for every camera:

$$\begin{aligned} x_k = (\quad & z_{1k1}, v_{1k1}, w_{1k1}, \ldots \\ & z_{1kB_k}, v_{1kB_k}, w_{1kB_k}, \ldots \\ & \ldots \\ & z_{NkB_k}, v_{NkB_k}, w_{NkB_k})^T. \end{aligned} \tag{6}$$

The estimation of such a large state vector would almost certainly prove intractable for any reasonable size array. Firstly, therefore, we assume that each camera n's blocks are independent. This allows us to run N separate particle filters, each estimating the state vector:

$$x_{nk} = (z_{nk1}, v_{nk1}, w_{nk1}, \ldots, z_{nkB_k}, v_{nkB_k}, w_{nkB_k})^T. \tag{7}$$

Secondly, when *updating* each block within an image using the particle filter, we assume that they are independent of each other, and so we can simply update the B_k separate state vectors, with parameters:

$$x_{nkm} = (z_{nkm}, v_{nkm}, w_{nkm})^T. \tag{8}$$

Unfortantely, whilst it may now appear possible to implement the estimation of x_k as $N \times B_k$ independent particle filters, the dependency introduced between blocks between resolutions (i.e. the *prediction* part of the algorithm), prevents us from implementing the estimation process for a given camera as B_k separate particle filters. However, it is still possible to take advantage of their independence within levels by performing the update step of the particle filter algorithm separately. Thus, the estimation of x_k may be implemented as N lots of B_k pseudo-independent particle filters.

4 The Particle Filter

The particle filter is a recursive parameter estimation algorithm. The prior distribution of parameters at each step is updated to the posterior distribution given some measurement which has been made. The prior distribution at the next step is then predicted from this posterior. The particle filter represents the prior and posterior distribution as a set of samples (particles), $\{s^{[i]}\}_{i=1}^S$, and weights for each particle $\{w^{[i]}\}_{i=1}^S$.

It has been noted that certain particle filters, such as the *sequential importance sampling* (SIS) filter, perform poorly when estimating static parameters

(and across scale we do expect the parameters to remain static), as in such cases the problem of particle degeneracy is acute [2]. So, the particle filter chosen is the *sequential importance resampling* (SIR) filter, which resamples the posterior distribution at each step. Resampling is a costly stage in this algorithm, and so the resampling is performed simply by estimating a Gaussian approximation to the posterior, and drawing new samples from the resulting normal distribution. This is known as a *Gaussian particle filter* [5], and dictates that we maintain not only the mean parameter estimate for each patch, x_{nkm}, but also a covariance matrix estimate P_{nkm}.

4.1 Update Stage

4.1.1 Measurement Model

We define a function $h(n, n', x_{nkm}, (a,b)^T)$ as follows: a given block parameter x_{nkm}, specifies a surface patch centroid p_{nkm} as per equation (5), and a normal vector of the surface patch. By intersecting the pixel ray defined by the pixel $(a,b)^T$ with the plane defined by this centroid and normal, we obtain a point in 3-d space. We can now project this point into any camera n' using the appropriate projection matrix $M_{n'}$, and sample the pixel intensity at that point. This is the value of $h(n, n', x_{nkm}, (a,b)^T)$.

For a particular camera n, and a particular block m, the measurement Y_{nkm} at each step k in the algorithm is a matrix containing pixel intensity values of that block, in the source image n, with $Y^{[a,b]}_{nkm}, 1 \leq a,b \leq P$ being the grey level intensity of the a,bth pixel of the block. Now it is possible to obtain a fitness measure for the parameter x_{nkm} by comparing the block texture Y_{nkm} with the pixel intensities of the reprojected patch intensities obtained through the function $h(.)$. For the reprojection into a camera n', the fitness measure e is

$$e(Y_{nkm}, x_{nkm}, n') = \quad (9)$$

$$\sum_{a,b} |Y^{[a,b]}_{nkm} - h(n, n', x_{nkm}, g(a,b,m)|. \quad (10)$$

Assuming this fitness measure to be normally distributed, zero mean, we define the measurement likelihood as the sum of Gaussians

$$p(Y_{km}|x_{km}) = \quad (11)$$

$$\frac{1}{|\mathcal{N}_n|} \sum_{n' \in \mathcal{N}_n} N(e(Y_{nkm}, x_{nkm}, n'); 0, r_{nkm}), \quad ()$$

over each camera n' which is adjacent to camera n. r_{km} is the expected measurement noise and is set empirically.

4.1.2 Update Algorithm

At the start of each step (resolution) in the Gaussian particle filter algorithm, we have an estimate of the state vector $\bar{x}_{nk}$ from the previous step, as well as a covariance matrix $\bar{P}_{nk}$. We present here the algorithm to estimate a single camera n's patches. Each patch is updated independently as follows:

```
Function UpdatePatches( x̄_nk, P̄_nk )
    X̄_nk = Partition( x̄_nk, P̄_nk )
    X_nk = ∅
    For (x̄_nkm, P̄_nkm) ∈ X̄_nk
        (x_nkm, P_nkm) = UpdatePatch(x̄_nkm, P̄_nkm)
        X_k = X_k ∪ (x_nkm, P_nkm)
    EndFor
    x_nk, P_nk = Recombine( X_nk )
EndFunction

Function UpdatePatch( x̄_nkm, P̄_nkm )
    Draw S samples {s_k^[i]}_{i=1}^S ~ π(x̄_nkm, P̄_nkm)
    For i ∈ 1...S
        w^[i] = p(y_nkm|s_k^[i]) N(s^[i]; x̄_nkm, P̄_nkm) / π(s_k^[i]; x̄_nkm, P̄_nkm)
    EndFor
    x_nkm = Σ_{i=1}^S ŵ^[i] s_k^[i]
    P_nkm = Σ_{i=1}^S ŵ^[i] (s_k^[i] − x_nkm)(s_k^[i] − x_nkm)^T
EndFunction
```

where $\hat{w}^{[i]} = w^{[i]} / \sum_{i=1}^{S} w^{[i]}$ and where $\pi(.)$ is the importance sampling distribution. In practice, this is commonly equal to the prior distribution. The function `Partition` simply splits the mean vector and covariance matrix for all blocks belonging to camera n, into a set containing all the individual patch mean vectors and covariance matrices

$$\texttt{Partition}(x_{nk}, P_{nk}) = \{(x_{nk1}, P_{nk1}, \ldots (x_{nkB_k}, P_{nkB_k})\} \quad (12)$$

with $\texttt{Recombine} = \texttt{Partition}^{-1}$ performing the inverse operation.

4.2 Prediction Stage

4.2.1 Process Model

In moving from a lower resolution to a higher resolution, each source block $k-1, m$, is subdivided

into four blocks. Assuming that there are four mapping functions $t^{[1]}, t^{[2]}, t^{[3]}, t^{[4]}$, which each take a block $k-1, m$ and map it to unique row index at resolution k, and that there is a function $\texttt{Split}(x_{n,k-1,m}, j)$ which calculates the correct parameters for each of the four split patches $j \in 1 \ldots 4$, then

$$x_{k,t^{[j]}(k-1,m)} = \texttt{Split}(x_{n,k-1,m}, j) \;\; 1 \leq j \leq 4, \tag{13}$$

Thus, the process model is simply

$$p(x_{k,t^{[j]}(n,k-1,m)} | x_{n,k-1,m}) = N(x_{k,t^{[j]}(k-1,m)}; \texttt{Split}(x_{n,k-1,m}, j), Q_{nkm}). \tag{14}$$

If a patch is found to be at a surface discontinuity, where neighbouring patches are part of two distinct surfaces or objects in the scene, it should be expected that once the patch is subdivided, a subset of the children will belong to one surface, and the remainder to the other surface. For such patches, the process noise (with covariance Q_{nkm}) is expected to be relatively high.

Unfortunately, in linking the process noise to the discontinuity of the surface, the patches are no longer independent and cannot be updated as such. The solution employed is to update each patch independently without the addition of process noise, followed by the addition of the global process noise $Q_{nk} = q(x_{nk})$, to the state covariance matrix $\bar{P}_{nk} = \bar{P}'_{nk} + Q_{nk}$, where $q(x_{nk})$ is some appropriate function linking the discontinuity of patches to the process noise.

4.2.2 Prediction Algorithm

```
Function PredictPatches( x_{n,k-1}, P_{n,k-1} )
    X_{n,k-1} = Partition(x_{n,k-1}, P_{n,k-1})
    X̄'_{nk} = ∅
    For (x_{n,k-1,m}, P_{n,k-1,m}) ∈ X_{n,k-1}
        X^4_{nkm} = PredictPatch( x_{n,k-1,m}, P_{n,k-1,m} )
        X̄'_{nk} = X̄'_{nk} ∪ X^4_{nkm}
    EndFor
    x̄_{nk}, P̄'_{nk} = Recombine( X̄'_{nk} )
    Q_{nk} = q(x̄_{nk})
    P̄_{nk} = P̄'_{nk} + Q_{nk}
EndFunction

Function PredictPatch( x_{n,k-1,m}, P_{n,k-1,m} )
    Draw samples {s^{[i]}_{k-1}}_{i=1}^{S} ~ N(x_{n,k-1,m}, P_{n,k-1,m})
    For j ∈ 1...4
        For i ∈ 1...S
            s^{[i,j]}_k = Split(s^{[i]}_{k-1}, j)
        EndFor
        x̄_{n,k,f[j](k-1,m)} = (1/S) Σ_{i=1}^{S} s^{[i,j]}_k
        P̄'_{n,k,f[j](k-1,m)} =
            (1/S) Σ_{i=1}^{S} (s^{[i,j]}_k − x̄_{n,k,f[j](k-1,m)}) .
                (s^{[i,j]}_k − x̄_{n,k,f[j](k-1,m)})^T
        X^4_{nkm} =
            X^4_{nkm} ∪ (x̄_{n,k,f[j](k-1,m)}, P̄_{n,k,f[j](k-1,m)})
    EndFor
EndFunction
```

5 Simplification

The inherent redundancy in the image based representation results in an oversampling of the patch data. To improve the performance of compression and reconstruction algorithms we wish to be able to identify and remove redundant patches. We formulate the problem as identifying a subset of patches such that the input images, $\mathcal{I} = \{I_1 \ldots I_N\}$, may be reconstructed to within some error bound, ϵ. Let $S_n = 1 \ldots B_K$ be the set of patches for all blocks at the finest resolution of image for camera n, and let

$$S = \bigcup_{n=1}^{N} S_n \tag{15}$$

We choose to reformulate our problem as finding the smallest subset S' of the patches S that can reconstruct the original images to within some error bound ϵ. That is to say we wish to find

$$\underset{S'}{\operatorname{argmin}} \left|S'\right| \tag{16}$$

subject to

$$\frac{1}{C} \sum_{n=1}^{N} \sum_{a,b \in I_n} \min_{m \in S'} \left\| h(n', n, x_{n\bar{K}m}, (a,b)) - I_i(a,b) \right\|_2^2 \leq \tag{17}$$

Where C is a normalising constant equal to the number of images multiplied by the number of pixels in each image. Unfortunately finding an optimal solution to this problem is NP-Hard [10]. Indeed, finding any solution is a challenging problem due to the time and memory constraints imposed by the volume of data. We introduce a greedy (but sub-optimal) algorithm for selecting patches to go into S'.

Consider each block of pixels independently and identify the patches that can reconstruct a single

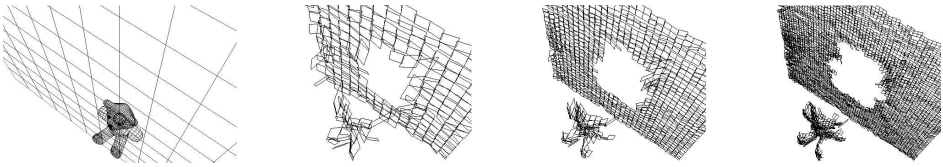

Figure 1: Output patches for teddy (far left), for three resolutions.

block to within the mean squared error threshold ϵ. Begin with $S' = \emptyset$. Then, at each iteration, we choose a patch in $S - S'$ to insert into S' that reconstructs the most blocks within the threshold ϵ that have not already been reconstructed by S'. Continue until all image blocks can be reconstructed by S'. Typically we use a block size of 2×2 and a threshold, ϵ, of 0.01 (20db) since the greedy algorithm will normally find a solution around an order of magnitude better than the requested quality.

6 Reconstruction

Each patch is modelled as a Gaussian centred on the patch centroid. The covariance is estimated using the spread of pixels used to compute the patch and the variance provided by the patch estimation process. In practice we treat the estimated variance in the depth parameter of the patch as the variance in the normal direction and use a symmetric variance parallel to the normal just large enough to cover the original image block. The reconstruction algorithm proceeds as follows

- For each pixel in the reconstruction find the set of patches whose Gaussian model is above some threshold parameter, τ, at the intersection between the reconstruction ray and the plane defined by the patch.
- Find the closest patch to the reconstruction camera, and take all those patches that are within some fraction, z_f, of the depth variation of the scene (typically between 1% and 10%).
- Perform a weighted blend between the closest patches, weighting each patch (i, j) proportionally to one over the angle between the vector to the reconstruction camera and to the source data camera. This strengthens the influence of patches observed from near the reconstruction viewpoint.

Our implementation makes extensive use of the graphics hardware to rapidly compute the reconstruction. For each patch we send a quadrilateral that bounds the patch to the graphics card together with a centre point, normal, and the perpendicular and parallel variance estimates for the patch. A combination of vertex and fragment shader programs then execute to threshold the patch at a pixel level and compute the necessary blending steps. The algorithm easily computes in real time on high end graphics hardware.

7 Results

To evaluate the performance of the structure estimation algorithm, a set of rendered images was created of two synthetic models, Teddy and Lucy which both have challenging aspects for any structure estimation algorithm. Each data set consists of an 8×8 array of images rendered from a square array. Ground truth patches for each, were generated from the models using a least squares fitting algorithm. Four levels of pyramid were used, with an 8×8 block size. The mean squared error in position and orientation compared to ground truth, for both data sets, is shown in the table below. The estimates for Teddy are compared with our best previous results from [8] which uses an optimization algorithm at a single resolution to estimate patch parameters.

	Particle Filter			*Simulated Annealing*		
Model	*z*	*v*	*w*	*z*	*v*	*w*
Teddy	0.08	0.04	0.06	16.2	0.34	0.37
Lucy	0.28	0.10	0.12	N/A	N/A	N/A

The resulting patches for a single source camera are shown for the Teddy data set in figure 1. In addition, the patch estimation algorithm was run on data from our real camera array, with the resulting patch centroids shown in figure 3 (c).

Figure 7 shows a plot of the reconstruction PSNR of the images verses the fraction of patches retained.

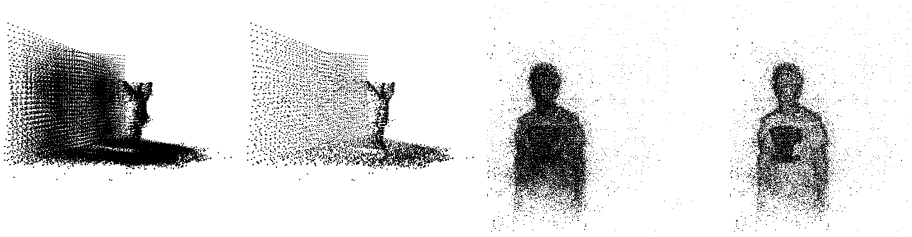

Figure 3: (a), (b) full and simplified patches from the 'Lucy' data set. (c), (d) full and simplified patches from the 'Andy' data set. ϵ was set to 0.01.

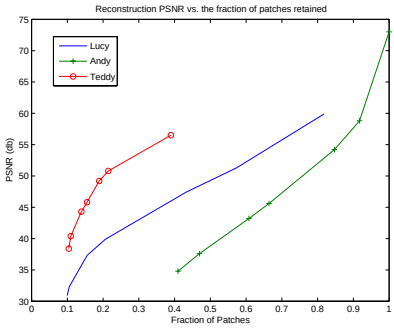

Figure 2: Input Image Reconstruction PSNR vs. Fraction of Patches Retained

As one would expect retaining more patches produces better reconstructions. It is also worth noting that the least planar data sets require more patches to achieve a similar reconstruction quality – simply because more patches are required to adequately represent the surface. This is most pronounced in the 'Andy' data set, where the near planar background was not estimated and so the object is significantly non-planar. The PSNR was computed as

$$\text{PSNR} = 10 \log_{10}\left(\frac{1}{\text{MSE}}\right) \tag{18}$$

where MSE is the mean square error of the optimum reconstruction

$$\text{MSE} = \frac{1}{C} \sum_{i=1}^{n} \sum_{x} \sum_{y} \min_{s_{j,k} \in S'} Q_i(x, y, s_{j,k}) \tag{19}$$

$$Q_i(x, y, s_{j,k}) = \| f_{j,k,i}(x, y) - I_i(x, y) \|_2^2 \tag{20}$$

Figure 3 shows the results of the simplification for a specific value of ϵ.

Figure 4 shows the results of reconstructing from a viewpoint not found in the original data set.

8 Conclusion

It has been shown that it is possible to estimate scene geometry, given an array of images, using a particle filter across scale. By employing a suitable parameterisation, measurement model, process model, and choice of particle filter, the resulting estimates are both accurate and quick to compute. The results are shown to be more accurate than our previous estimates, when compared to ground truth. By using the resulting estimates together with the original image data, we have shown it is possible to produce accurate reconstructions of the scene from any viewpoint. As part of future work, it is hoped to extend these techniques to video sequences, in which patch estimates for subsequent frames will be informed by previous frames, resulting in further increases in both accuracy and speed.

Acknowledgements

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

References

[1] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, Massachusetts, 1991.

[2] 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.

[3] W. Heidrich, H. Schirmacher, H. Kück, and H. P. Seidel. A warping-based refinement of lumigraphs. In Vaclav Skala, editor, *Proc. Winter School in Computer Graphics (WSCG) '99, Plzen, Czech Republic, February 1999*, pages 102–109, February 1999.

[4] John Isodoro and Stan Sclaroff. Contour generator points for threshold selection and a novel photo-consistency measure for space carving. Technical Report BUCS-TR-2003-025, Boston University Computer Science, December 2003.

[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] Wojciech Matusik. Image-based visual hulls. Master of science in computer science and engineering, Massachusetts Institute of Technology, February 2001.

[8] Andrew Mullins, Adam Bowen, Roland Wilson, and Nasir Rajpoot. Estimating planar patches for light field reconstruction. In *British Machine Vision Conference*, 2005.

[9] Don Murray and James J. Little. Patchlets: Representing stereo vision data with surface elements. In *Proc. Seventh IEEE Workshops on Application of Computer Vision*, volume 1, pages 192–199, January 2005.

[10] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227 – 234, April 1995.

[11] Voicu Popescu. *Forward Rasterization: A Reconstruction Algorithm for Images-Based Rendering*. PhD thesis, University of North Carolina, 2001.

[12] Hartmut Schirmacher. Warping techniques for light fields. In *Proc. Grafiktag 2000, Berlin, Germany, September 2000*, September 2000.

[13] I. O. Sebe, P. Ramanathan, and B. Girod. Multi-view geometry estimation for light field compression. In *Vision, Modeling, and Visualization*, 2002.

[14] E. P. Simoncelli. Coarse-to-fine estimation of visual motion. In *IEEE Eighth Workshop on Image and Multidimensional Signal Processing*, 1993.

[15] Tomas Svoboda, Daniel Martinec, and Tomas Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, pages 407–422, August 2005.

[16] Roland Wilson and Chang-Tsun Li. Multiresolution random fields and their application to image analysis. Technical Report CS-RR-361, Warwick University, Coventry, UK, 1999.

Figure 4: (a), (c) novel viewpoint reconstructions for the full data. (b), (d) novel viewpoint reconstructions for the simplified data.