Multiresolution Particle Filters in Image Processing

By Andrew Mullins, Adam Bowen, Roland Wilson, Nasir Rajpoot

Abstract

Recursively estimating the likelihood of a set of parameters, given a series of observations, is a common problem in signal processing. The Kalman filter is a The particle filter is now a well-known alternative to the Kalman filter. It represents the likelihood as a set of samples with associated weights and so can approximate any distribution. It can be applied to problems where the process model and/or measurement model is non-linear. We apply the particle filter to the problem of estimating the structure of a scene from n views of that scene, by applying the particle filter across image resolutions.

1 Introduction

Estimating the structure of a scene from one, two, or n images is a fundamental problem in vision. Its range of applications include robot navigation, video coding, scene analysis, and image based rendering techniques such as light field rendering [(3)]. The latter provides the primary motivation for this work, which is concerned with estimating structure from an array of cameras which all image the same scene.

Algorithms for estimating structure from multiple cameras are invariably based on an underlying model of the scene and imaging geometry. Approaches which explicitly model the projective geometry can provide accurate estimates of structure. In [(5)], optical flow constraints in each view are related to the parameters of an icosahedron mesh allowing the use of least sqaures to estimate a mesh. However, this method assumes the existence of only one object and that silhouette information is available.

The model presented in this paper also attempts to reflect the scene and imaging geometry, whilst providing a general framework which allows for the representation of multiple and abitrary objects. It is assumed each $P \times P$ block of pixels in an image corresponds to some patch of a surface in the scene, with a particular position and orientation.

In this paper the model and its parameterisation is outlined, followed by the description of a particle filter used to estimate the parameters to this model, which works across successive resolutions of image.

2 Model Parameterisation

The estimation algorithm is based on a Gaussian pyramid with K levels, each resolution k having $N_1/2^{K-k} \times N_2/2^{K-k}$ pixels. Each time the algorithm is run, parameters are estimated with respect to a single camera, denoted the 'source' camera located at c_s, with each of its image pyramid levels being partitioned into $P \times P$ pixel blocks. In the discussion that follows, for notational convenience each block m is numbered in row-order.

For a given source camera with known parameters, the centre of each block m at a given scale k, specifies a directional ray r_{km} from the camera through the scene. The quadrilateral patch corresponding to this block of pixels must be centred at some point along this ray. We specify the centroid position

$$c_s + z_{km}\frac{r_{km}}{|r_{km}|}, \tag{1}$$

by its distance z_{km} along this ray.

We specify the orientation of each patch, as two components of the normal vector $(v_{km}, w_{km}, 1)$ in an Euler frame, such that the patch's centroid is the origin, and $c_s = \alpha.(0,0,1), \alpha > 0$, i.e. the 'forward' vector looks directly at the source camera.

The complete state vector at resolution k, consists of all three parameters, for each block, in row order:

$$x_k = (z_{k1}, v_{k1}, w_{k1}, \ldots, z_{kM}, v_{kM}, w_{kM})^T. \tag{2}$$

It will be convenient to refer to the subset of the state vector and covariance matrix which corresponds to just one block m. This will be denoted

$$x_{km} \quad = \quad (x_k^{(3m)}, x_k^{(3m+1)}, x_k^{(3m+2)})^T, \tag{3}$$

with a corresponding 3×3 covariance matrix P_{km}. In other words, it is assumed that patches are independent. Finally, the complete set of such sub-vectors and sub-matrices for every patch will be denoted $X_k = \{(x_{k1}, P_{k1}), \ldots, (x_{km}, P_{km})\} = \texttt{Partition}(x_k, P_k)$.

3 The Particle Filter

The particle filter represents the prior and posterior distribution of a given parameter r.v. 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 [(1)]. So, the particle filter chosen is the *sequential importance resampling* (SIR) filter, which resamples the posterior distribution at each step. In this case we resample all distributions as Gaussians giving a *Gaussian particle filter* [(2)].

3.1 Update Stage

3.1.1 Measurement Model

For a block m at a given scale k, the measurement y_{km} at each step is the pixel values of that block, in the source image. If this block images a patch with parameters x_{km}, then $h(x_{km}) = h_1(x_{km}) \ldots h_8(x_{km})$ is the expected measurement and is the pixel values of the regions to which the patch projects in the 8 neighbouring cameras. Assuming that the mean absolute error e, between each reprojected patch and the source block y_{km} is normally distributed, the measurement model is defined as the sum of gaussians

$$p(y_{km}|x_{km}) = \frac{1}{8}\sum_{n=1}^{8} N(e(y_{km}, h_n(x_{km})); 0, R_{km}). \tag{4}$$

where the measurement noise variance R_{km} is determined empirically.

3.1.2 Update Algorithm

At the start of each step (resolution) in the Gaussian particle filter algorithm, we have an estimate of the state vector $\bar{x}_k$ from the previous step, as well as a covariance matrix $\bar{P}_k$. Each patch is updated independently as follows:

```
Function UpdatePatches( x̄_k, P̄_k )
    X̄_k = Partition( x̄_k, P̄_k )
    X_k = ∅
    For (x̄_km, P̄_km) ∈ X̄_k
        (x_km, P_km) = UpdatePatch(x̄_km, P̄_km)
        X_k = X_k ∪ (x_km, P_km)
    EndFor
    x_k, P_k = Recombine( X_k )
EndFunction
```

```
Function UpdatePatch( x̄_km, P̄_km )
    Draw S samples {s_k^[i]}_{i=1}^S ~ π(x̄_km, P̄_km)
    For i ∈ 1...S
        w^[i] = p(y_km|s_k^[i]) N(s^[i]; x̄_km, P̄_km) / π(s_k^[i]; x̄_km, P̄_km)
    EndFor
    x_km = Σ_{i=1}^S ŵ^[i] s_k^[i],  P_km = Σ_{i=1}^S ŵ^[i] (s_k^[i] − x_km)(s_k^[i] − x_km)^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.

3.2 Prediction Stage

3.2.1 Process Model

In moving from a lower resolution to a higher resolution, each source block $k-1, m$, is subdivided into four blocks, with indices and which have parameters $x_{k,m,j} = \texttt{Split}(x_{k-1,m}, j) \;\; 1 \leq j \leq 4$, (the details of function `Split` being omitted). Thus, the process model is simply

$$p(x_{k,m,j}|x_{k-1,m}) = N(x_{k,m,j}; \texttt{Split}(x_{k-1,m}, j), Q_{km}). \tag{5}$$

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_{km}) is expected to be relativly high.

3.2.2 Prediction Algorithm

```
Function PredictPatches( x_{k−1}, P_{k−1} )
    X_{k−1} = Partition(x_{k−1}, P_{k−1})
    X̄'_k = ∅
    For (x_{k−1,m}, P_{k−1,m}) ∈ X_{k−1}
        X^4_km = PredictPatch( x_{k−1,m}, P_{k−1,m} )
        X̄'_k = X̄'_k ∪ X^4_km
    EndFor
    x̄_k, P̄'_k = Recombine( X̄'_k )
    Q_k = q(x̄_k)
    P̄_k = P̄'_k + Q_k
EndFunction
```

```
Function PredictPatch( x_km, P_km )
    Draw samples {s_k^[i]}_{i=1}^S ~ N(x_km, P_km)
    For j ∈ 1...4
        For i ∈ 1...S
            s_{k+1}^[i,j] = Split(s_k^[i], j)
        EndFor
        x̄_{k,m,j} = (1/S) Σ_{i=1}^S s_{k+1}^[i,j],  P̄'_{k,m,j} = (1/S) Σ_{i=1}^S (s_{k+1}^[i,j] − x̄_{k,m,j})(s_{k+1}^[i,j] − x̄_{k,m,j})^T
        X^4_km = X^4_km ∪ (x̄_{k,m,j}, P̄_{k,m,j})
    EndFor
EndFunction
```

4 Results

To evaluate the performance of the 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. Ground truth patches for each, were generated from the models using

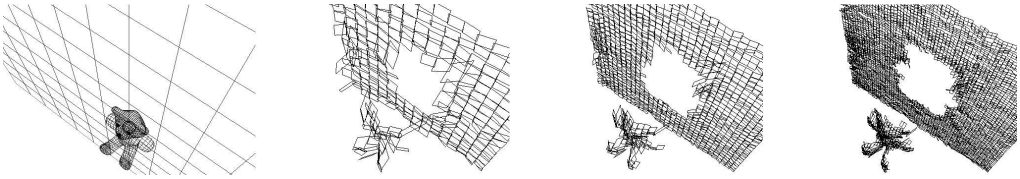

Figure 1: Output patches for teddy (far left), for three resolutions.

a least squares fitting algorithm. 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 [(4)].

	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.

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

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