An $O(N^2)$ Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping

Steven A Holmes, Student Member, IEEE, Georg Klein, Member, IEEE, and David W Murray, Member, IEEE

Abstract—This paper develops a Square Root Unscented Kalman Filter (SRUKF) for performing video-rate visual simultaneous localization and mapping (SLAM) using a single camera. The conventional UKF has been proposed previously for SLAM, improving the handling of non-linearities compared with the more widely-used Extended Kalman Filter. However, no account was taken of the comparative complexity of the algorithms: in SLAM the UKF scales as $O(N^3)$ in the state length, compared to the EKF’s $O(N^2)$, making it unsuitable for video-rate applications with other than unrealistically few scene points. Here it is shown that the SRUKF provides the same results as the UKF to within machine accuracy, and that is can be reposed with complexity $O(N^2)$ for state estimation in visual SLAM. The paper presents results from video rate experiments on live imagery. Trials using synthesized data show that the consistency of the SRUKF is routinely better than that of the EKF, but that its overall cost settles at an order of magnitude greater than the EKF for large scenes.

Index Terms—Structure from motion; Simultaneous localization and mapping; Unscented Kalman filter

I. INTRODUCTION

A useful distinction to make between methods of recovering 3D structure from motion (SFM) is whether or not they are causal. Non-causal methods are typically used for model-generation, and involve a batch-mode recovery of all the camera poses and the 3D scene by minimizing image error in a bundle adjustment [1], [2], [3], [4]. Polished commercial packages (e.g. [5], [6]) are available, targeted at the market in adding spatially located special effects during the post-production phase of movies, without the need to track cameras during filming itself. The particular advantage of bundle adjustment is that it delivers an unequivocally optimal solution, though with two caveats. First is that the high-dimensional minimization involves a non-convex cost function, so that a good starting point is required. Second, and more fundamental, is that optimality requires all image data associated with a particular 3D feature to be included as such. If not, a new 3D feature is likely to be instantiated, effectively locking-in the error — something most likely to occur when part of the scene is revisited after a period of neglect.

Causal methods are de rigeur for “live” applications such as navigation and augmented reality. By contrast with batch methods, they usually retain only the current camera pose in the state, and update this and the scene structure from frame to frame. Apart from delivery of prompt information, their principal advantage is that data association is made easier using the available strong prior estimates of the values of the camera pose and scene motion and their uncertainties. Especially salient features can be actively sought in the image, and the camera, if under control, can be moved to view them better. In recursive methods that maintain a fully populated state covariance matrix, it is possible to adjust all the observed structure, camera state, and covariances when data association is restored after neglect (so-called loop closure). The disadvantage of recursive estimators is that the solution is unlikely to coincide with the optimal one, and indeed the filters are not guaranteed to remain consistent.

Maintenance of a full covariance matrix describing correlations between the state members is typical of the simultaneous localization and mapping (SLAM) approach to structure from motion. Introduced into the robotics field using sonar and laser sensors [7], [8], [9], [8], [10], it has been adapted more recently for stereovision and monocular SFM [11], [12], [13]. The approach differs from the pioneering work in recursive SFM such as that of Ayache and Faugeras [14] and Harris and coworkers [15], because at that time (mid to late 1980s) it was computationally expedient to assume complete independence between the camera state and structural elements. This prevents proper exploitation of loop-closure, both through an increased likelihood of failure to associate data, and through the inability to propagate information throughout the complete state.

Common to both traditional recursive SFM and the application of visual sensing to SLAM is the use of the Extended Kalman Filter (EKF). The EKF is almost always the algorithm of first resort when estimating the state evolution of non-linear dynamical systems, but it is often the algorithm of last resort too — despite the warnings of several authors that it can be prone to unacceptable
bias, inaccuracy, and indeed inconsistency, over the long term [16], [17], [18].

One aim in this paper is to investigate to what extent this is the case for the monocular recursive monoSLAM method of Davison et al. [13]. We compare how the EKF performs against its closest competitor, the Unscented Kalman Filter (UKF). The Unscented Transform was introduced by Julier, Uhlmann and Durrant-Whyte [19], [20], [16] as a method of reducing bias in non-linear transforms. In some sense, it is a poor man’s particle filter, with a small number of particles, or “sigma points”, being deterministically sampled from the underlying probability density function (pdf), propagated through the non-linear transformations, and then used to re-estimate a parametric description of the pdf. A number of authors have applied the UKF to SLAM problems in robotics [21], [22], [23]. Indeed Chekhlov et al. [24] used it for visual SLAM from a single camera. However, the principal purpose of their work lay elsewhere, and the UKF appears to have been used for convenience, removing the need to compute some unsavoury Jacobian matrices. The authors [24] remark that the UKF and EKF perform similarly, but no comparative results are presented.

Here we find that the UKF and EKF perform rather differently in terms of the quality of output. More critically, the EKF as applied to SLAM has lower computational complexity than the general UKF. While both the general UKF and EKF have $O(N^3)$ complexity in the number of states $N$, it has been shown that EKF-SLAM scales as $O(N^2)$ under the assumption that the 3D map does not evolve during state prediction [25]. In the state of length $N = (c + 3m)$, $c$ states describe the camera position and motion (often $c = 12$), and the other $3m$ states describe the 3D scene points. Complexity then is dominated by the $m$ scene points, making the EKF quite unviable in large-scale field robotics.

The second and principal aim of this paper is to show that the proper comparison to draw is not between the EKF and UKF, but between the EKF and the Square Root UKF (SRUKF) introduced by van der Merwe and Wan [17]. They showed that the Cholesky decomposition of the covariance matrix in the UKF could be avoided by updating and propagating the square root of the covariance rather than the covariance itself. The results are identical to within machine accuracy. While the general SRUKF remains $O(N^3)$ for state estimation, here we show that for SLAM the complexity can be reduced to $O(N^2)$.

Our approach differs then from that of Andrade-Cetto et al. [22] who partially finessed the issue of complexity by applying the UKF only to the state elements describing the sensor (attached rigidly to a vehicle), and using a KF on the structure update. Their rationale was that the scene evolution is linear (indeed it is idempotent). However, this negates the non-linearity in the measurement process. Here we exploit the idempotent evolution of the structural map, but only as a detail within the SRUKF. An idea which is somewhat similar to [22] is that of using dual Kalman Filters, proposed by Paul and Wan [29], where SLAM is posed as a state estimation problem for the camera location and a parameter estimation problem for the map. This removes the correlations between the map and camera states.

The third aim is to introduce all the implementation details needed to apply the SRUKF to the SLAM problem. This expands our previous work [30] to include (i) the use of the SRUKF to estimate the state of the system as time evolves (ii) the use of the Square Root Unscented Transform to insert inverse depth points and then convert them to 3D points and (iii) to show how to marginalize out a superfluous feature from the square root form of the map.

The rest of the paper is organized as follows. Section II outlines the physical models used for camera motion and scene structure in monoSLAM, and the models used for the state update and image measurement. We need only sketch the EKF solution to this problem as the detail can be found in the paper of Davison et al. [13]. Section III briefly describes the UKF. Again we will give only sufficient description so that the development of the Square Root UKF can be contrasted with it in Section IV. Details of the application of the SRUKF to the monoSLAM problem are given in Section V, and comparative results of the UKF, SRUKF and EKF obtained over many experiments on archetypal camera motions, are reported in Section VI. The paper concludes with clear guidance to machine vision practitioners as to which algorithm, the EKF, UKF or SRUKF, should best be deployed in different circumstances.

But first we consider whether, given the considerable attention given to taming complexity in the broader SLAM literature, there remains scope for even a quadratic process in visual SLAM.

A. Is there scope for any KF in visual SLAM?

A particularly successful strand of work in mitigating complexity has been the information filter formulation of SLAM: both the Sparse Extended Information Filter (SEIF) [26] and the Exactly Sparse Delayed State Filter (ESDSF) [27] can run in constant time under certain conditions. In both, the information form allows the stage of finding the posteriors of the state and information matrix to be performed in constant time, as the only parts of the information matrix that change are those corresponding to the camera, the feature observed and their blocks of mutual information. The SEIF allows the prediction of the prior to be performed in constant time instead of the cubic time of the Extended Information Filter. When normalized, the information matrix is typically dominated by weak links between features and the camera, links which can be removed in a consistent way to achieve a sparse matrix. This reduces the number of features being estimated to a small active set, allowing constant time filtering as long as the camera is exploring.
new areas. Whenever a loop is closed the SEIF is not con-
tant time. In contrast, the ESDSF maintains a list of
of all previous camera poses, instead of estimating the
locations of features in the environment. When a new
frame is received, the transformations between it and all
previous frames that it overlaps are calculated and used
for updating the state and, as noted earlier, this takes
constant time. If odometry is available, state prediction
can also be calculated in constant time, since only the
previous pose is involved in the calculation. The old pose
is marginalized only if it was estimated from odometry.
It then has mutual information with only one other pose
and can be marginalized in constant time.

Another successful approach to tackling complexity is
to estimate all of the previous poses and the positions of
landmarks. This is the approach taken in Incremental
Smoothing and Mapping (iSAM) [28], the robotics
approximation to bundle adjustment. The advantage of
estimating all the old poses is that the information
matrix remains sparse. iSAM incrementally adds new
poses, features and measurements to the system, but
does not slavishly re-estimate and re-linearize about
parts of the state that are only weakly correlated to the
new measurements. Updating with the measurements
and recovering the whole state trajectory can be done
in linear time. Occasionally the information matrix is
permuted to achieve a sparser factorization; when this
is done the whole system is re-linearized allowing an
optimal estimate.

For video rate vision, an immediate disadvantage of
iSAM is that the information vector grows linearly with
time. But a more general problem with all of the above
methods is that they work best with sensors which
provide range and bearing. The single camera used here
provides bearings only, requiring each part of the scene
to be viewed from a wide range of viewpoints to allow
the 3D locations of features to converge. By nature then,
monoSLAM is a loopy process. Repeated revisiting the
same area of scene tends to destroy the sparsity and
tidy structure of the information matrix that information
filters exploit.

Davison et al. [13] provide a more positive justifica-
tion for the EKF’s complexity being tolerable for visual
SLAM. They argue that, as a bearings only sensor,
monocular vision is never going to produce a dense map:
rather, using view-point robust features, output from
monoSLAM should properly, and at best, be regarded as
providing camera pose within a sparse framework of 3D
landmarks. However, while the EKF’s $O(N^2)$ compo-
ssity is tolerable for video-rate applications in modestly-
scaled visual environments, the UKF’s $O(N^3)$ compo-
sity can not be.

II. MONOCULAR VISUAL SLAM, AND ITS EKF
IMPLEMENTATION

The problem to be solved by filtering is the estimation
of the state $x$ and covariance $P$ of a discrete-time non-
linear dynamical process or plant, whose evolution is
modelled from time step $k$ to $k + 1$ as

$$x_{k+1} = F(x_k, u_k, a_k),$$

using observations which are related to the state through
a vector of non-linear functions

$$y_k = H(x_k, n_k).$$

Here, $u_k$ is any known control input, and $a_k$ and $n_k$ are
process and measurement noise, respectively.

In monocular visual SLAM the state vector $x_k$ has two
parts. One part is the map, the (initially) expanding set
of estimates $\{X^i\}^N_1 i = 1 \ldots m_k$ of the positions in the 3D
scene. The scene itself is modelled as static, so that the
evolution of the posterior to the prior at the next time
step is idempotent, $\{X^i\}^N_1 k = \{X^i\}^N_1 k_{k|k}$.

The other part $c_k$ describes the camera, here using
twelve entries, three each for the orientation (a vector
representation of the rotation matrix $R_k$) and position
$H_t$, and three each for the rotational and translational
components of motion, here represented by the coefficients
$\alpha$ of the generators of $SE(3)$. The $4 \times 4$ transformation
from scene to camera frame then evolves as

$$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}_{k+1|k} = \exp \left[ \sum_{n=1}^{6} c^a_k G^n \right] \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}_{k|k}$$

where the non-zero elements of the generators are

$$\begin{align*}
\text{in } G^1 : G_{32} &= -G_{23} = 1; \\
\text{in } G^2 : G_{13} &= -G_{31} = 1; \\
\text{in } G^3 : G_{21} &= -G_{12} = 1; \\
\text{in } G^4 : G_{14} &= 1; \\
\text{in } G^5 : G_{24} &= 1; \\
\text{in } G^6 : G_{34} &= 1.
\end{align*}$$

For small motions, the coefficients themselves are readily
identified with the rotational and translation members of
the velocity screw, and zero-mean Gaussian distributed
accelerations $a_k$ are applied to them at each step.

The measurements are assumed to arise from a perspec-
tive projection of the scene points into the camera

$$\lambda_j \begin{bmatrix} y_j^k \\ 1 \end{bmatrix} = c_k \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}_{k|k} \begin{bmatrix} X^k_j \\ 1 \end{bmatrix}$$

where $c_k$ is the camera’s intrinsic calibration which may
change but is always known, and $\lambda_j$ expresses the
unknown scale introduced by homogeneous coordinates.
For live experimentation a correction for radial distor-
tion is applied.

A. The EKF

To obtain a fast analytical recursive estimator for a
state requires its probability density function (pdf) to
be a simple function of a few parameters. The best-
known example is that of a system where the state
update and measurement processes are both linear, and
both corrupted by additive Gaussian noise. The pdf is
a multivariate Gaussian, described succinctly by a mean
and covariance, which can then be updated optimally
using the Kalman filter [31] [32].
The three Jacobian matrices have elements
\[
\nabla F_{ij} = \frac{\partial x_{i(k+1)}}{\partial x_{j(k)}}; \quad \nabla H_{ij} = \frac{\partial y_{i(k)}}{\partial x_{j(k)}}; \quad \nabla G_{ij} = \frac{\partial x_{i(k)} - y_{i(k)}}{\partial x_{j(k)}}.
\]

B. Example output

Fig. 1 shows a typical graphical output from a video-rate implementation of EKF monoSLAM. A calibration tile is used at the outset to make the system fully observable [33], to set the world frame, and to break the depth/speed scaling ambiguity, but needs not remain in view thereafter. The camera track is shown and, as the camera moves and rematches points, the depth uncertainties on the recovered structure points, initially greatly elongated along the direction from which the structure was first observed, are seen to shrink. The implementation is little changed from that of Davison et al. [13], and the interested reader should refer to that work for practical detail. One important change, however, is that we use the more recent method of feature initialization using an inverse depth or disparity representation proposed by Montiel et al. [34]. This will result in a particular problem for the UKF and SRUKF, as discussed later.

III. THE UKF ALGORITHM

The success or otherwise of the EKF broadly depends on how close to linear the update function and the measurement process are. Several authors, for example [35], [36], show examples of non-linear functions where the error introduced cause the filter to diverge. In two Dimensional SLAM systems the divergence of the EKF has been investigated [37], [38], [39], [40] and the divergence was found to arise due to linearization about the incorrect state.

The Unscented Kalman Filter (UKF) [16] avoids some of the pitfalls of linearization by selecting specific points, called sigma points, to represent the mean and covariance (and more generally to represent the underlying pdf), transforming these according to the non-linear transformations \( F(\cdot) \) for the state and \( H(\cdot) \) for the measurements, then using weighted values of the new sigma points to derive the new mean and covariance. Depending on the number and weight of the sigma points, the UKF can give accurate results to different orders of a Taylor expansion of the covariance matrix and account for different underlying pdfs. The common usage is to place \( 2N+1 \) sigma points for a state vector of dimension \( N \) — one for the mean, and \( 2N \) placed symmetrically about the means along on contours of the covariance matrix. This provides approximations accurate to 3rd order for Gaussian pdfs, regardless of the forms of \( F \) or \( H \). By contrast, the EKF is only 1st-order accurate [16].

A. Predicting the state and covariance

Given the update and measurement equations, (1) and (2), the first step in the UKF is to form an augmented state vector of dimension \( N_a \) and augmented covariance matrix by appending the mean and covariance of the process noise vector. Assuming the mean is zero, these are

\[
x_{k|k}^a = \begin{bmatrix} x_{k|k} \\ 0 \end{bmatrix}, \quad p_{k|k}^a = \begin{bmatrix} P_{k|k} & 0 \\ 0 & Q_k \end{bmatrix}.
\]

To predict over a time step \( k \) to \( k+1 \), the zeroth sigma point \( \chi_0 \) is defined equal to the current state, and a further \( 2N_a \) sigma points are determined by adding and subtracting in turn the transpose of the \( i \)-th row of the square root of \( P^a \) to the current mean. That is, defining \( S^a \) as the upper triangular matrix in the Cholesky factorisation

\[
S^a \top S^a = P_{k|k}^a,
\]

The most common approach when the system is non-linear is to linearize the plant and measurement processes about the current state estimate, and then to deploy the Kalman Filter mechanism to solve. The EKF predicts the mean and covariance will evolve over a time

\[
x_{k+1|k} = F(x_{k|k}) \quad \text{(5)}
\]

\[
P_{k+1|k} = \nabla F_{k|k} \nabla F^\top + \nabla G_k \nabla G^\top \nabla Q \nabla G^\top, \quad \text{(6)}
\]

where \( Q \) is the plant noise covariance \( \delta_{ij}E[a_i a_j^\top], \) and \( \nabla G \) is defined below.

The innovation covariance and Kalman gain matrices are derived as

\[
P_{yy} = \nabla H P_{k+1|k} \nabla H^\top + \nabla R \quad \text{(7)}
\]

\[
K = P_{k+1|k} \nabla H^\top P_{yy}^{-1}. \quad \text{(8)}
\]

The predicted measurements are \( y_{k+1|k} = H(x_{k+1|k}) \) and, once the actual measurements \( y_{k+1} \) at the timestep are available, the state and covariance updates are made using

\[
x_{k+1|k+1} = x_{k+1|k} + K(y_{k+1} - y_{k+1|k}) \quad \text{(9)}
\]

\[
P_{k+1|k+1} = P_{k+1|k} - K P_{yy} K^\top. \quad \text{(10)}
\]
Now the scale is $\gamma$ where $S$ and with these, the predicted state is augmented with the measurement and their covariance. For the fourth order errors for a Gaussian [16]. Note that each $\chi^j$ has the dimension $N$ of the original state, not the augmented state. The predicted mean and covariance are calculated as

$$
x_{k+1|k} = \sum_{j=0}^{2N_a} W_j \chi^j
$$

$$
P_{k+1|k} = \sum_{j=0}^{2N_a} W_j (\chi^j - x_{k+1|k})(\chi^j - x_{k+1|k})^\top
$$

where the $W_j$ are weights, defined above to minimize the fourth order errors for a Gaussian [16]. Note that in the SLAM problem, because the scene points are assumed stationary, not all entries in (17) and (18) need be computed explicitly.

### B. Predicting measurements

Because the measurement process is non-linear a further set of sigma points must be generated to determine the expected measurements and their covariance. For these, the predicted state is augmented with the measurement noise. Assuming mean zero,

$$
x_{k+1|k}^b = \begin{bmatrix} x_{k+1|k}^0 \\ 0 \end{bmatrix} \quad p_{k+1|k}^b = \begin{bmatrix} P_{k+1|k}^0 & 0 \\ 0 & R \end{bmatrix},
$$

and with $S^b_S^b = p_{k+1|k}^b$ the $2N_b + 1$ measurement sigma points are

$$
\begin{bmatrix}
\chi_0^+
\vdots
\chi_i^+
\vdots
\chi_{N_a+i}^+
\end{bmatrix} = \begin{bmatrix}
x_{k+1|k}^b
\vdots
x_{k+1|k}^b + \gamma_b [S^b_S^b]^\top
\vdots
x_{k+1|k}^b - \gamma_b [S^b_S^b]^\top
\end{bmatrix} i = 1, \ldots, N_b
$$

Now the scale is $\gamma_b = \sqrt{N_b/(1-W_0)}$. Each of these sigma points is used to generate a measurement $\zeta_j = H(\chi^j_j), j = 0, \ldots, 2N_b$, where again we concatenate the two parameters of (2) into one. The mean, covariance and cross-covariance are found as

$$
y_{k+1|k} = \sum_{j=0}^{2N_b} W_j \zeta_j
$$

$$
P_{yy} = \sum_{j=0}^{2N_b} W_j [\zeta_j - y_{k+1|k}][\zeta_j - y_{k+1|k}]^\top
$$

$$
P_{xy} = \sum_{j=0}^{2N_b} W_j [\chi^j_j - x_{k+1|k}][\zeta_j - y_{k+1|k}]^\top,
$$

where $\chi^j_j$ in the last expression has its augmented noise members lopped off.

### C. State and covariance update

Finally the Kalman gain matrix and state and covariance updates are derived in the fashion familiar from the EKF — c.f. (9)

$$
K = P_{xy}P_{yy}^{-1}
$$

$$
x_{k+1|k+1} = x_{k+1|k} + K[y_{k+1} - y_{k+1|k}]
$$

$$
P_{k+1|k+1} = P_{k+1|k} - KP_{xy}K^\top,
$$

where $y_{k+1}$ is the actual measurement taken at time step $k+1$.

### IV. The Square Root UKF

Whether or not the UKF gives better results than the EKF for monoSLAM — and this question is explored empirically later — the utility of the standard implementation is wholly compromised for video-rate applications by the Cholesky decomposition performed at each time step, the complexity of which scales as $N^3$. Recently, however, van der Merwe and Wan [17] introduced the Square Root UKF, a re-implementation of the general UKF which delivers exactly the same results (to within machine accuracy), but which cleverly avoids the decomposition by directly propagating the Cholesky factor $S$ rather than the covariance $P$. As mentioned above, the SRUKF is in general $O(N^2)$ for state estimation, but is $O(N^3)$ for parameter estimation.

### A. Predicting the state

At the beginning of the update cycle from time step $k$ to $k+1$, let $S_{k|k}$ be the upper Cholesky factor in the decomposition $S^b_{k|k} = S^b_{k|k} - P_{k|k}$, and let $T_k$ and $U_k$ be the upper Cholesky factors of the process and measurement noise covariances, $Q_k$ and $R_k$ respectively. Now the state and Cholesky factors are augmented as

$$
x_{k|k}^b = \begin{bmatrix} x_{k|k}^0 \\ 0 \end{bmatrix} \quad S_{k|k}^b = \begin{bmatrix} S_{k|k}^0 & 0 \\ 0 & T_k \end{bmatrix},
$$

and the sigma points are found using the transpose of the $i$-th row of $S_{k|k}^0$, just as in (14). Running these through the model update $\chi^+_i = F(\chi, u_k), i = 0, \ldots, 2N_a$ enables prediction of the mean just as before in (17), but finding
the predicted Cholesky factor is more complicated. It involves deriving a set of weighted deviations

\[ e_0 = \sqrt{|W_0|} (x_0 - \mathbf{x}_{k+1|k}) \]  
\[ e_i = \sqrt{W_i} (\mathbf{z}_i - \mathbf{x}_{k+1|k}^i) \quad i = 1, \ldots, 2N_a \]

and performing the \( O(N^3) \) QR decomposition on the \((2N_a \times N)\) matrix

\[ QR \leftarrow B = [ e_1 \quad e_2 \ldots \quad e_{2N_a}]^T. \]

The resulting \((2N_a \times N)\) matrix \(R\) is of the form

\[ R = \begin{bmatrix} S^- & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \]

where \(S^-\) has size \((N \times N)\). The predicted factor \(S_{k+1|k}\) is then found using Cholesky updating (or downdating if \(W_0 < 0\)),

\[ S_{k+1|k}^- = S^- - b e_0 e_0^T. \]

This process, note, is \(O(N^2)\). The special form of \(e_0\), its exclusion from the QR decomposition, and the subsequent correction by updating are necessary because, uniquely amongst the weights, \(W_0\) may be negative.

A neat method of Cholesky updating was developed by Golub in [41] and can be extended to Cholesky downdating [42] by defining a matrix

\[ M = \begin{bmatrix} i e_0 & S^- \end{bmatrix} \]

where \(i\) is the imaginary unit, \(b = 0\) for updating and \(b = 1\) for downdating. This representation is entirely equivalent to (32). The downdating then uses Givens matrices to zero the diagonal elements of \(S^-\). The downdated matrix is then simply all but the last row of the resulting matrix.

B. Predicting measurements

To ease further explanation it is convenient to regard \(S^-\) as the output of a function \(QRF\) that takes as its parameters; and also to regard \(S_{k+1|k}\) as the output of a function \(ChUpdate(S^-, \text{sign}(W_0), e_0)\).

The treatment of the measurements follows by analogy using as augmented state and factor

\[ x_{k+1|k} = \begin{bmatrix} x_{k+1|k}^m \\mathbf{0} \end{bmatrix} S^b_{k+1|k} = \begin{bmatrix} S_{k+1|k}^m & \mathbf{0} \\mathbf{0} & U_k \end{bmatrix}. \]

The \(2N_b + 1\) sigma points are found using (20) and, as earlier, predicted measurements are generated \(\mathbf{z}_j = H(\chi_j^+\mathbf{z})\) and their predicted mean \(y_{k+1|k}\) found as in (21). The weighted deviations

\[ e_0 = \sqrt{|W_0|} (\mathbf{y}_0 - y_{k+1|k}) \]  
\[ e_i = \sqrt{W_i} (\mathbf{z}_i - y_{k+1|k}) \quad i = 1, \ldots, 2N_b \]

are used to derive

\[ S_y^- = QRF(e_1, e_2, \ldots, e_{2N_b}) \]  
\[ S_y^+ = ChUpdate(S_y^-, \text{sign}(W_0), e_0). \]

The cross covariance is found just as in the UKF

\[ p_{xy} = \sum_{j=0}^{2N_b} W_j [\mathbf{y}_j^+ + \mathbf{x}_{k+1|k}] \mathbf{z}_j^+ \mathbf{z}_j^+ \]

where again the augmented noise terms are pruned from \(\mathbf{z}_j^+\). The final steps are

\[ K = p_{xy} [S_y^+ S_y^-]^{-1} - \sum_{j=0}^{2N_b} S_y^+ \mathbf{z}_j^+ \]

\[ x_{k+1|k+1} = x_{k+1|k} + K(y_{k+1|k} - y_{k+1|k}) \]

\[ S_{k+1|k+1} = SeqChUpdate(S_{k+1|k}, -1, K S_y^-) \]

(42) denotes repeated Cholesky updating using successive columns of \(K S_y\) as the updating vector.

V. THE SRUKF APPLIED TO VISUAL MONOLAM

The application of the Square Root UKF to monoSLAM has thrown out several challenges where particular care has been required during implementation to obtain an efficient and effective algorithm. The first is the crucial observation that a significant computational saving can be made at (30), the QR decomposition for the state prediction, which reduces the complexity of the SRUKF from \(O(N^3)\) to \(O(N^2)\). This saving is possible because the scene points are assumed to be stationary. During the prediction step the position and covariance of their map representations remain unchanged. Their estimates are evolved when new measurements add information about the scene. The second challenge concerns the initialization of features using the inverse depth method in both the SRUKF and UKF. The third is that deleting features in the SRUKF requires a subtler mechanism than either the EKF or UKF.

A. Exploiting the form of the state evolution

Because the map points do not move, the QR decomposition of (30) can be abbreviated. When written out in component form the sigma points for state prediction are

\[ \chi_{0,j} = (x_{k|k})_j \]  
\[ \chi_{i,j} = (x_{k|k}^a)_j + \gamma a S^i_{t,j} \quad 1 \leq i, j \leq N_a \]

\[ \chi_{N_a+i,j} = (x_{k|k}^a)_j - \gamma a S^i_{t,j} \]

where \(S^i_{t,j} = 0\) if \(i > j\). We order the state vector with the position of (30) can be abbreviated. When written out

\[ \chi_{0,j} = (x_{k|k})_j \]  
\[ \chi_{i,j} = (x_{k|k}^a)_j + \gamma a S^i_{t,j} \quad 1 \leq i, j \leq N_a \]

\[ \chi_{N_a+i,j} = (x_{k|k}^a)_j - \gamma a S^i_{t,j} \]

where \(S^i_{t,j} = 0\) if \(i > j\). We order the state vector with the position of (30) terms being \(N_b\) entries) and lastly the augmented noise terms (\(d\) entries), so that \(N = 3m + c + N_a = 3m + c + d\). Then each sigma point is put through the update function \(F\).

Because the map points do not move, the top \(3m\) elements remain unchanged. These are, for \(1 \leq i \leq N_a\) and \(1 \leq j \leq 3m\),

\[ \chi_{0,j} = (x_{k|k})_j \]  
\[ \chi_{i,j} = (x_{k|k}^a)_j + \gamma a S^i_{t,j} \]

\[ \chi_{N_a+i,j} = (x_{k|k}^a)_j - \gamma a S^i_{t,j} \]
Now consider the elements of $x_{k+1}^\top[k]$: 

$$(x_{k+1}^\top[k])_j = \sum_{i=0}^{2N_a} W_i \chi_{i,j}.$$  

(50)

For $1 \leq j \leq 3m$, the summation over index $i$ causes the $\pm \gamma a S_{i,j}$ terms to cancel, so that 

$$(x_{k+1}^\top[k])_j = (x_{k}^\top[k])_j \quad 1 \leq j \leq 3m$$  

(51)

but 

$$(x_{k+1}^\top[k])_j \neq (x_{k}^\top[k])_j \quad (3m+1) \leq j \leq N.$$  

(52)

The elements $1 \leq j \leq 3m$ of vectors $e_i$ ($1 \leq i \leq N_a$) are: 

$$(e_0)_j = \sqrt{W_0} (\gamma a_{ij} - (x_{k+1}^\top[k])_j) = 0$$  

(53)

$$(e_i)_j = \sqrt{W_i} (\chi_{i,j} - (x_{k+1}^\top[k])_j)$$  

$$(e_{N_a+i})_j = \sqrt{W_i} (\gamma a_{N_a+i,j} - (x_{k+1}^\top[k])_j)$$  

(54)

(55)

But as noted above $S^\top$ is upper triangular, so that for $1 \leq j \leq 3m$ 

$$B_{i,j} = (e_i)_j = \begin{cases} +\sqrt{W_i} \gamma a S_{i,j} & 1 \leq i \leq j \\ 0 & (j+1) \leq i \leq N_a \end{cases}$$  

(56)

The form of this sparse matrix $B$ is shown in Fig. 2(a). To recover $S^\top$ we could perform a QR decomposition on $B$ and use $R$. However the form of $B$ can be even further simplified to that of Fig. 2(b) by pre-multiplying with the orthonormal matrix

$$Q_1 = \begin{bmatrix} -I_1/\sqrt{2} & 0 & I_1/\sqrt{2} \\ 0 & I_2 & 0 \\ I_1/\sqrt{2} & 0 & I_1/\sqrt{2} \\ 0 & 0 & 0 \end{bmatrix}$$  

(57)

where $I_1$ is a $3m \times 3m$ identity, $I_2$ is a $(c+d) \times (c+d)$ identity. The $R$ from the QR decomposition is of course unchanged by this operation. Now $Q_1 B$ is already upper triangular, with only $c = 12$ columns to be handled by a specially-crafted QR decomposition.

### B. Initializing inverse depth points

In monoSLAM as described by Davison et al. [13], map management criteria aim to ensure that the number of features observable is compatible with both the desired localization accuracy and maintaining video frame-rate, and features are both deleted and added during the course of a run.

When adding a feature, because one view of a point does not provide depth information, the original version of monoSLAM “incubated” a newly observed feature using a particle filter until its pdf was sufficiently Gaussian to be entered into the EKF. More recently Montiel et al. [34] have proposed feature incubation within the filter using an inverse depth entry in the state vector, and a modified measurement model.

In the EKF implementation [34], a point feature is represented at first observation by six entries — three for the position $T_{FO}$ of the camera when the point was first observed, two for the direction $\theta$ of the ray from that initial camera position to the point, and one for the reciprocal depth $d^{-1}$. This is obviously an over-parameterisation of a point in 3D space, but the real problem is that when the point is first entered the camera’s actual position $T$ in the state and position $T_{FO}$ in the state are identical and fully correlated, making the covariance matrix rank-deficient. The EKF can cope with a rank-deficient matrix because it wraps the covariance matrix by the Jacobian $\nabla F$ and its transpose (see (6)). However, counter to the implication in [24], the UKF cannot handle features initialized in this way. It must fail during the Cholesky decomposition of the non positive-definite covariance matrix. However the failure is not catastrophic: most Cholesky routines continue to run, typically leaving a zero on the diagonal element when the algorithm has to calculate the square root of a negative number, and setting a flag to notify the user of the problem. The comparable failure in the SRUKF is at the Cholesky downdate step (32).

We develop now a method of avoiding failure in the SRUKF. To add a new feature a factored pdf

$$\begin{bmatrix} x_{k|k} \\ y_{new} \\ d^{-1} \end{bmatrix}, \begin{bmatrix} S_{k|k} \\ U_k \\ 0 \end{bmatrix}$$  

(58)

Fig. 2. (a) The matrix $B$ and (b) its simplification $Q_1 B$. Those that change are for $0 \leq i \leq 2N_a$ and $(3m + 1) \leq j \leq N$.

$$\chi_{i,j} = F_j(x_i).$$  

(49)

![Diagram](https://via.placeholder.com/150)
is defined, comprising the pdfs of the current state \((x_{k|k}, S_{k|k})\), the observation \((y_{new}, u_k)\), and the inverse depth \((d^{-1}, \sigma)\). From this a set of sigma points \(x_j\) are generated which are passed through a map “extending” function

\[
x_{k|k}^e = \begin{bmatrix} x_{k|k} \\ X^{m+1} \end{bmatrix} = M \begin{bmatrix} x_{k|k} \\ y_{new} \\ d^{-1} \end{bmatrix},
\]

(59)

where \(X^{m+1}\) is the new inverse depth feature in a rearranged form

\[
X^{m+1} = \begin{bmatrix} \theta \\ T_{FO} \end{bmatrix}.
\]

The inverse depth feature has been arranged in this order so that a modified version of Cholesky downdating in (32) can be used to calculate the “extended” factor \(S_{k|k}^e\). The reason for this will be explained shortly. After downdating, the ordering of the state vector must be reinstated, so the rows of \(x_{k|k}^e\) and \(S_{k|k}^e\) are re-arranged. These are now a valid vector and matrix square root for the SRUKF, but \(S_{k|k}^e\) is tidied into upper triangular form by QR decomposition.

If more than one feature is to be initialized, the pixel locations and inverse depth parameters can be stacked under the \(x_{k|k}\) in the pdf in (58). This larger pdf can then be passed through the map extending function, again leaving \(T_{FO}\) at the bottom of the state vector.

The reasons that the Cholesky downdate fails are twofold. The first is that a positive semidefinite matrix does not have a unique decomposition and the second is that truncation errors can cause a positive semidefinite matrix to have a negative eigenvalue. Higham [43] has shown that while a positive semi-definite square matrix does not have a unique Cholesky decomposition, it is possible to derive a permutation matrix \(\Pi\) such that the product

\[
\Pi \Pi^T = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}^T \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}
\]

(61)

does. The form of \(\Pi\) is found such that \(R_{11}\) is an upper triangular matrix of dimensions \(\text{rank}(A)\), \(R_{12}\) is a matrix with dimensions \(\text{rank}(A) \times \text{nullity}(A)\), and the 0 are zero matrices with dimensions to make the matrix square. With \(\Pi\) and the unique Cholesky decomposition of \(\Pi \Pi^T\) it is possible to find the quasi-unique decomposition of \(A\). The Cholesky downdating process required in the SRUKF can be similarly modified. The state has been arranged this way after the new feature has been inserted so that its covariance matrix has a unique decomposition. The matrix \(S^e\) that is produced from (31) has its last three rows filled with zeros and the vector \(e_0\) only has non-zero elements in the fourth, fifth and sixth elements from the bottom. This means that the downdating algorithm can simply be stopped before the last three rows of the factor. This is because they will always be entirely filled with zeros as in (61).

Once the depth estimate of a feature has converged it is converted to a 3D point feature. This is a straightforward application of the square root unscented transform to the state and covariance. In the QR decomposition the sigma point matrix can be pre-multiplied by an orthonormal matrix as was done in (57).

C. Deleting superfluous features

The final part of map management is to remove superfluous features. In the EKF and UKF this is simply a case of removing a row and column from the covariance matrix, equivalent to pre- and post-multiplying the covariance matrix by a dimension-reducing permutation matrix

\[
D = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & 0 & I_2 \end{bmatrix}
\]

and its transpose. If there are \(c\) rows and columns before the feature to be removed and \(d\) after then \(I_1\) is the identity matrix of dimension \(c\) and \(I_2\) is the identity matrix of dimension \(d\).

The equivalent action in the SRUKF is to post-multiply the upper triangle \(S\) of the Cholesky factorisation by \(D^\top\). If \(S_{12}\) and \(S_{22}\) are the matrix blocks that correspond to the superfluous feature, then

\[
S = \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ 0 & S_{22} & S_{23} \\ 0 & 0 & S_{33} \end{bmatrix}
\]

becomes

\[
SD^\top = \begin{bmatrix} S_{11} & S_{13} \\ 0 & S_{23} \\ 0 & S_{33} \end{bmatrix}.
\]

(64)

Although no longer triangular, this is a valid matrix square root which could be used to define the next set of sigma points. However, it is reformed into a triangular matrix so that when the prior is next calculated one can still simplify the sigma point matrix before computing the QR decomposition of Fig. 2. To achieve this re-triangulation, the middle rows are removed leaving

\[
S_{\text{old}} = \begin{bmatrix} S_{11} & S_{13} \\ 0 & S_{33} \end{bmatrix}
\]

(65)

and the matrix of the middle rows, \(V = [0 \; S_{23}]\), used in a sequential Cholesky update of \(S_{\text{old}}\) to obtain the new Cholesky factor

\[
S_{\text{new}} = \text{SeqChUpdate}(S_{\text{old}}, +1, V^\top).
\]

(66)

This involves three updates when deleting a regular 3D point, but six when deleting an inverse depth point.

D. Data Association

Active search [13] is used for data association in our implementation of the SRUKF. The search ellipses are calculated by reconstructing the diagonal blocks of the innovation covariance from \(S_q\) in (38). The innovation covariance is much smaller than the state covariance so the necessary blocks can be calculated quickly.
VI. EXPERIMENTAL COMPARISONS

The three filters, the EKF, UKF and SRUKF, have been implemented in C++ to run on a 2.66 GHz Intel Core 2 Duo Processor. While care has been taken with each to create efficient code and to use proven optimized libraries, no extraordinary effort has been made to reduce the execution time of one versus any other. All three filters run at frame rate but, due to their different complexities, break the real time constraint with different numbers of features.

Most important here is the real time implementation of the SRUKF, stills of which are shown in Fig. 3. The filter is running on video rate imagery provided by a Firewire camera and is initialized from the four corners of the black target. Features are then added until a map with about 30 features has been built.

In the following comparisons the filters run on synthetic data read in from file. Synthesized data have been used for the comparative tests, providing identical data and ground truth for comparison of compute time, complexity, accuracy, and consistency.

A. Algorithmic complexity

The first trials explore the comparative speed of the three algorithms, and determine their complexity as a function of increasing state size. To eliminate irrelevant overheads such as disk access, computing times were derived separately for the stages of (i) computing the prior, (ii) projecting the search ellipses into the synthetic image, which involves calculating the innovation covariance, and (iii) calculating the posterior, and then summed. The resulting computing times are shown as functions of state size in Fig. 4.

Below a state size of around 100, all three methods are dominated by a mix of fixed time and $O(N)$ operations, with a resulting complexity of about $O(N^{0.5})$. Beyond this size, equivalent to greater than 30 or so points in the map, both the EKF and SRUKF tend to $O(N^2)$ complexity. The EKF, however, is evidently a factor of around 10 to 16 times faster.

Above a state size of some 200 the UKF becomes significantly more costly than the SRUKF, and tends to $O(N^3)$ complexity. In our experiments at a state size of 2300 (770 3D points) the UKF has reached $O(N^{2.8})$.

In the general UKF algorithm, there are two operations with cubic complexity. The first is the Cholesky factorisation of (13) and the second is the outer product calculation in (18). It is the outer product calculation that is the more expensive. However, as noted earlier, UKF SLAM does not require explicit computation of all the covariance matrix entries, as the 3D scene points are static. So for this special case the outer product calculation is $O(N^2)$, delaying the onset of the ultimately dominant $O(N^3)$ behaviour. As a result, the SRUKF and UKF algorithms have a quite similar speed performance on small maps.

B. Timings

The SRUKF is faster than the UKF in its calculation of the first two stages mentioned earlier, those of (i) computing the prior and (ii) projecting the search ellipses into the image, owing to the absence of the Cholesky factorisation in both. However, the posterior update takes longer in the SRUKF, as it has to downdata the Cholesky factor multiple times. Table I and Fig. 4 show the EKF is much faster than either the UKF or the SRUKF. The advantage of propagating the Cholesky factor directly is evident in the difference between the SRUKF and the UKF in the first two stages. The price is paid in the posterior update when the Cholesky factorisation has to be modified using multiple sequential Cholesky downdates.

C. Comparability of results from SRUKF and UKF

Rather than compare output from all three filters, it is convenient first to demonstrate that SRUKF and UKF produce identical results. To avoid arbitrarily picking results for individual state members, we derive in Fig. 5 the normalized difference in the normalized estimation error squared (NEES), or squared Mahalanobis distance, generated by the SRUKF and UKF over a run of 500 frames. The normalized difference is used to take into account the large values of the NEES estimate. The mean of the difference is $-1.4 \times 10^{-6}$, and the standard deviation is $\sim 3 \times 10^{-6}$ indicating equality up to machine accuracy. For the consistency tests below we will therefore compare only the SRUKF and EKF.

D. Filter consistency of the SRUKF and EKF

A useful measure of filter performance is consistency. A filter is consistent if its state estimate $\hat{x}_{k|k}$ is unbiased, so that $E[\hat{x}_{k|k} - x_k] = 0$ where $x_k$ is ground truth and its NEES is bounded appropriately for a $\chi^2$ distributed variable with $N$ degrees of freedom ($N$ is the state’s dimension) at some chosen confidence level

$$
\epsilon_k^2 = [\hat{x}_{k|k} - x_k] \mathbf{P}_{k|k}^{-1} [\hat{x}_{k|k} - x_k] \in [b_l, b_U]. \quad (67)
$$

<table>
<thead>
<tr>
<th>Table I Absolute times for various parts of the algorithm for different numbers of scene points.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of points</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>200</td>
</tr>
<tr>
<td>500</td>
</tr>
</tbody>
</table>
Fig. 3. Example images and maps output from a video rate experiment using the SRUKF. a) Camera view of the first four features on the initialisation target and the corresponding map. b) After a few features have been added. c) New features have been added, the features in red could not be found. d) Mapping away from the initialisation target, the features in blue are not being measured. e) Returning to the initialisation target. Features which were initialized early have now converged to points.

Fig. 4. Log-log plots of the total times (in seconds) taken for the core “Kalman” stages of the UKF, SRUKF and EKF versus the size of the state. The occasional timing blips are due to systems programs which could not be switched off.

We compare the EKF and SRUKF by recording the NEES for the camera pose during three characteristic movements of a synthesized camera through synthesized scenes. Correct data association is assured. The three motions are (i) the camera translating laterally in front of a grid of features on a front-parallel plane where once a feature has left the field of view it is never observed again; (ii) a spiral motion about the optic axis, in a world with 3D features randomly distributed within a cube; and (iii) a random walk, in the same random scene. The synthetic camera is a model of that used for live experiments, with a field of view of $80^\circ \times 55^\circ$ and a notional $640 \times 480$ pixel image, though it omits radial distortion. To generate measurements, 3D points were projected into the image and their coordinates corrupted by Gauss random measurement noise with standard deviation $0.5 \text{ pixel}$. Plant noise of $1 \text{ m s}^{-2}$ and $1 \text{ rad s}^{-2}$ was assumed. For each experiment the results of 50 runs each 500 frames long were averaged.

The first motion is one of constant velocity and conforms to the motion model. Nevertheless, the plots of the NEES for EKF and SRUKF in Fig. 6 show that both filters became inconsistent after several tens of frames. The SRUKF is slower to do so, and has a lower normalized error estimate, but the degree of over-confidence (i.e. $\rho$ is too small) in both filters grows steadily over time. Fig. 7 shows the squared Euclidian distance between the estimate and the true position. The error in position for the estimate is very small so the inconsistancy is due to the covariances becoming too small. Fig. 8 shows the mean error in $x$-position and $z$-position for SRUKF on this dataset along with $\pm 1$ standard deviation. There is no systematic bias, indicating that the filters are correctly tuned.

During the spiral motion, the optic axis of the camera remained pointing in the same direction and the camera
Fig. 5. The normalized difference in NEES (or squared Mahalanobis distance) between the SRUKF and UKF. The mean is close to zero, and the standard deviation lies at the output precision of the estimates (6 significant figures).

Fig. 6. The normalized estimation error squared (NEES) plotted as a function of frame number for the EKF (upper line) and the SRUKF when the camera translates past a planar grid of points. Also shown are the upper and lower bounds on $\chi^2$ distributed varied for 300 degrees of freedom at the 95% confidence limit.

was moved in the plane paralleled to the image plane. The motion involves what has been found empirically and theoretically [33] to be a favourable motion for monoSLAM, where well-localized structure is revisited while building up neighbouring areas, in a characteristic “SLAM Wiggle”. This motion also produces inconsistent estimates from both filters, as seen in Fig. 9, but both NEES measures exhibit an oscillation with the same frequency as the camera motion. In contrast to the first motion, where the over-confidence grew monotonically, this simulation in which the same structures are revisited shows bounded inconsistency. As before, the EKF tends to under-estimate the uncertainty compared with the SRUKF.

In the random walk, the camera undergoes a random acceleration at each timestep which, as in the first example, conforms with constant velocity model. These runs produce the most overconfident estimates. (In Fig. 10 the upper and lower bounds for $\chi^2$ are indistinguishable from the $x$-axis.) As in the other experiments, the EKF is the more over-optimistic, but again the trends in the NEES plots over time copy one another. Bailey et al. [40] derived similar results comparing the EKF and UKF for mapping a 2D environment using a range bearing sensor. The reason is the same — local improvements in linearisation do not make a large difference to consistency.

VII. CONCLUSIONS

In this paper we have argued on the grounds of computational complexity that if an Unscented KF is to be used for visual SLAM it should not be the general UKF but the Square Root UKF introduced by van der Merwe and Wan, at $O(N^3)$ for parameter estimation. After briefly reviewing both, we have detailed an efficient $O(N^2)$ implementation of the SRUKF for visual monoSLAM. In particular, we have shown how the fact that the scene structure is stationary can be exploited, how structure can be deleted, and how inverse depth points can be initialized. We have noted that there is a degeneracy in the covariance matrix just after introducing such a feature using the representation of Montiel et al. [34]. The degeneracy does not damage the performance of the EKF, but affects both UKF and SRUKF, but has passed unnoticed in a previous implementation of the UKF.

The most immediate results from the experiments are that the SRUKF and UKF perform identically in terms of the quality of structure and motion output and that the complexity of the SRUKF visual SLAM has been successfully reduced to $O(N^2)$ in the state size. In this sense it becomes competitive with the EKF.
However, we find the execution time of the SRUKF is a factor of some $10^{1.2}$ higher. Although no extraordinary care has been taken to optimize either algorithm, the EKF code is simple and well-used, and it is unlikely that further performance can be squeezed from it. There may be further gains to be made in the SRUKF, but it seems unlikely that the performance deficit could be reduced much below an order of magnitude.

The SRUKF is more consistent than the EKF but on a number of motions both filters rather quickly move into a regime where the normalized error estimate squared lies well above the upper bound set for a $\chi^2$ random variable. Both over-estimate the confidence with which the estimates are known. This result agrees with that of [40] who also found that the EKF-SLAM algorithm is inconsistent.

For practitioners concerned with video-rate performance, our results suggest that the EKF remains properly the algorithm of choice for single camera visual SLAM. While the SRUKF does produce better results, we suggest this is outweighed by the speed handicap. Visual SLAM using the SRUKF may find its niche in problem domains where the map is known a priori to be of some fixed and small size.

Fig. 8. The mean and one sigma bounds for the error in x-position (upper) and z-position (lower).

Fig. 9. The NEES plotted as a function of frame number for the EKF (upper line) and the SRUKF when the camera spirals in a structure comprising randomly placed 3D points in a cube. Also shown are the upper and lower bounds on $\chi^2$ distributed varied for 300 degrees of freedom at the 95% confidence limit.

Fig. 10. Plots of the NEES for the EKF (upper) and SRUKF for the camera undergoing a random walk. The upper and lower $\chi^2$ bounds (300 degrees of freedom, 95% confidence limit) are both at the bottom of the graph, but are indistinguishable at this scale.

ACKNOWLEDGEMENTS

This work was supported by Grants GR/S97774 and EP/D037077 from the UK Engineering and Physical Science Research Council. SAH is supported by an EPSRC Research Studentship. The authors are grateful for conversations with Professor Gene H. Golub and Dr Simon Julier, and for the insightful comments of the reviewers.

REFERENCES
