Active Camera Calibration for a Head-Eye Platform using
the Variable State-Dimension Filter

Philip F. McLauchlan and David W. Murray

Abstract
This correspondence presents a new technique for calibrating a camera mounted on a controllable head-eye platform. It uses the trajectories of an arbitrary number of tracked corner features to improve the calibration parameter estimates over time, utilising a novel variable state dimension form of recursive filter. No special visual stimuli are required and no assumptions are made about the structure of the scene, other than that it is stationary relative to the head. The algorithm runs at 4 frames per second on a single Inmos T803 transputer, and is fully integrated into a real-time active vision system. Updated calibration parameters are regularly passed to the vision modules that require them. Although the algorithm requires an initial estimate of camera focal length, results are presented from real experiments demonstrating that convergence is achieved for initial errors up to 50%.

I. INTRODUCTION

Scene reconstruction and object recognition are areas of computer vision which have been plagued by the need for accurate camera calibration [19]. Calibration typically requires objects made to high precision to be placed in front of the cameras [13], [8] and requires considerable experimental care, making the methods impractical for an autonomous robot acting in an unstructured world.

For these reasons much emphasis has been placed recently in the fields of structure from motion [12], stereo [7], object recognition [21] and active vision [5] on algorithms that obviate camera calibration. However, when vision is to control robotic systems, there seems a limit to how far these ideas can be pushed. In order to make controlled motions, an active system must convert image quantities into angles and distances, requiring calibration of some form. For example, consider 1D fixation on a target point P which corresponds under perspective projection to a point x in the image. If the focal length is f, and ignoring any image distortions, the angle that the camera must turn through to fixate P is \( \theta = \tan^{-1}(x/f) \). At first sight, one of the merits of an active system exploiting visual feedback is that it can redirect gaze without knowing f. However, if one underestimates f, the value for \( \theta \) is overestimated, and vice versa, so that the choice of f affects the damping of the fixation control system.

In this correspondence we describe a calibration method which exploits our head/eye platform’s ability to make precisely known head movements while robustly tracking stationary points in an unstructured world. The approach is novel in several ways. First it utilises all measurements of image features tracked through multiple images in a computationally efficient and statistically principled manner. Second, the process is fully integrated into our real-time reactive vision system for gaze control [18], [17] and provides continual updates of calibration parameters to the other vision modules. Thirdly, the algorithm uses a novel form of recursive filter, which allows observations from an arbitrary number of tracked features to be incorporated. The variable state-dimension filter (VSDF) is a general solution for static estimation problems involving a global state and a variable number of local states, coupled to the global state but not to each other. In this work these are, respectively, the calibration parameters and the visual directions of the tracked features in the world. We show in Section IV that the filter can effectively deal with discarded states (corresponding to lost features) and has time complexity linear with the number of current local states (tracked features).

Our method has aspects in common with previously reported work. Like Thacker & Courtney [23], we aim to improve estimates over time by incorporating new image measurements as they arrive. Like Hartley [10] we calibrate using rotational camera motions only, and like Du & Brady [6] we use image features tracked over multiple frames. All these researchers together with Maybank & Faugeras [14] and Brooks et al. [4] emphasize the benefits of self-calibration, a paradigm — which well describes the present work — in which no special objects are used for visual stimuli, in which calibration parameters are updated over time and react to external disturbances, and in which calibration proceeds automatically in the background while other tasks are in progress.

In the following Section we discuss the calibration parameters for a single camera directed by two rotation axes. In Sections III through V we give a detailed description of the algorithm, which runs in parallel with the motion algorithms implemented on our real-time vision system [17]. Results are presented in Section VI for two versions of the algorithm, demonstrating the generalizable nature of our approach.

II. GEOMETRICAL PARAMETERS RECOVERED

In this section we discuss the calibration parameters for a single camera with two attached rotation axes. The camera is one of two mounted on a 4-axis stereo head/eye platform. In the full configuration, the two axes provide the elevation (up-down) and one of the two azimuth (left-right) degrees of freedom.

The camera and head geometry is sketched in Figure 2. The X, Y, Z frame is the stationary or resting head frame, defined by setting the X axis to coincide with the elevation axis, and the Y axis to coincide with the azimuth axis in its arbitrary initial position. After elevation by angle \( \theta_e \) and azimuth by angle \( \theta_a \), the gaze frame is transformed to the \((X', Y', Z')\) frame. A fixed but unknown transformation takes the head frame \((X', Y', Z')\) to the camera frame \((x, y, z)\). The z-axis is the optic axis and is by definition perpendicular to the x and y axes parallel to the image plane at \( z = -f \).

1This work was supported by grants from the UK EPSRC (grant GR/G30003), and from the EC’s Esprit Programme (IP 5390).

The authors are with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, UK; email pm or dwm@robots.ox.ac.uk
Ideally the transformation between \((X, Y, Z)\) and \((X', Y', Z')\) would be a pure rotation, and that between \((X', Y', Z')\) and \((x, y, z)\) would be an identity. Some head designs strive for this ideal using vernier adjustable camera mounts. Although important for applications in metrology, such effort seems misplaced for vision. Such devices cannot take account of internal misalignments in the camera which change as, for example, zoom is changed, which then must be determined by calibration. We prefer to use less precise mounts, and then calibrate for the entire misalignment.

The overall transformation from \((X, Y, Z)\) to \((x, y, z)\) can be written as a priori as the sum of a pure rotation and pure translation. The rotation comprises a part which can be changed very precisely using the rotation axes, and a fixed unknown part which arises from misalignment of the camera mount, camera body, and optic axis (or effectively the CCD chip) within the body. The translation, which will change as the rotation changes, arises from small offsets between the rotation axes, a relatively large displacement of the optic centre from the rotation centre, and a small displacement of the principal point from the nominal image centre, \((x, y) = (0, 0)\). Because the translation couples to the rotation it is theoretically possible to determine it. However, the effect on visual data of the translational component is also proportional to the inverse depth. In practice, on the present platform, the effect is negligible for depths \(Z > 2m\).

There are two further matters relating to the camera. First, and of minor significance as data for calibration is obtained close to the image centre, is that of radial distortion: this is routinely corrected for using the method described in [3]. Secondly, because the aspect ratio of the image frame is imperfectly known (because of lack of precise knowledge of hardware parameters such as camera A/D rates), two independent focal lengths are calculated for each of the \(x\) and \(y\) directions in the image.

To summarise, then, first we assume that when points at sufficient depth are viewed (in our case greater than \(2m\) distant) the transformation between \((X, Y, Z)\) and \((X', Y', Z')\) is a pure rotation which can be changed precisely. The mechanism of the camera platform used, Yorick [22], provides information about relative head angles to within \(0.01^\circ\), corresponding to 0.1 pixels in the image. Importantly, proper account is taken of timing delays in image capture and early processing so that head odometry is mapped properly to the corresponding image data [22].

Secondly, we assume the transformation between \((X', Y', Z')\) and \((x, y, z)\) frames is again a pure rotation. The three small angles associated with it (each typically less than \(3^\circ\)) are recovered by calibration, as are the two focal lengths of camera \(f_x\) and \(f_y\) in the \(x\) and \(y\) directions which take account of the aspect ratio.

A final assumption is that 3D points can be viewed which are stationary relative to frames’ origin. This simplifies the development of a calibration algorithm since both intrinsic and extrinsic estimated parameters will be constant. If the world moves, one has to recover the scene motion as well as the calibration, complicating matters considerably [14], [1], [11].

### III. Method

Take a stationary scene point \(P = (P_x, P_y, P_z)\) measured in the stationary gaze frame \((X, Y, Z)\) (thus \(P\) is constant). We shall first determine which point \((x, y)\) on the image plane it projects to, given the current elevation and azimuth angles \(\theta_e\) and \(\theta_a\).

Elevation is a rotation by angle \(\theta_e\) about the \(X\) axis and azimuth is a rotation by angle \(\theta_a\) about the elevated \(Y\) axis \(Y'\), thus they are represented by the matrices

\[
R_e = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_e & \sin \theta_a \\ 0 & -\sin \theta_e & \cos \theta_a \end{pmatrix}, \quad R_a = R_a \begin{pmatrix} \cos \theta_a & 0 & \sin \theta_a \\ 0 & 1 & 0 \\ -\sin \theta_a & 0 & \cos \theta_a \end{pmatrix} R_a^{-1}
\]

where we have used the abbreviations \(c_e = \cos \theta_e\), \(s_e = \sin \theta_e\) etc. Now define \(\hat{x}, \hat{y}\) and \(\hat{z}\) to be the unit vectors in the direction of the camera \((x, y, z)\) axes measured in the fixed \((X, Y, Z)\) frame. Let the small rotations from the \((x, y, z)\) axes to the \((X', Y', Z')\) axes be \(\phi_x, \phi_y\) and \(\phi_z\). Then the rotation from the camera frame to the head frame can be described by the matrix

\[
R_h = \begin{pmatrix} 1 & -\phi_z & \phi_y \\ \phi_z & 1 & -\phi_x \\ -\phi_y & \phi_x & 1 \end{pmatrix}
\]

Hence combining all the rotations together we have \((\hat{x} \, \hat{y} \, \hat{z}) = R_a R_e R_h\).

Now consider the fixed point \(P\). Its 3D coordinates relative to the camera are

\[
P = (p_x, p_y, p_z)^T = RP \text{ where } R = (\hat{x} \, \hat{y} \, \hat{z})^T. \tag{1}
\]

The image coordinates \((x, y)\) are related to the above by the standard projection equations

\[
x = -f x p_x / p_z, \quad y = -f y p_y / p_z. \tag{2}
\]

We now set \(P_z = 1\), expressing the fact that we can only measure the visual direction of the 3D point \(P\). This will be a valid simplification so long as the head angles \(\theta_e\) and \(\theta_a\) are kept well away from the ±90° point, which is where \(P_z\) approaches zero. Equations (2) provide the basis for an algorithm to estimate the head parameters from observations of stationary points in the world. This algorithm is described in Section V, and uses the variable state dimension filter described now.

### IV. The Variable State Dimension Filter (VSDF)

We wish to recover calibration parameters from the trajectory of tracked features. In essence this problem should not be very difficult: combine the measurements each feature provides about the state over time using (typically) a Kalman filter [2]. However there are two aspects of the problem at hand that complicate matters. Firstly, each feature, as well as providing information about the global state, i.e. the calibration parameters, also has its own local state, the 3D visual direction of the point in the world. Secondly, the visible features in the world are constantly changing as the head rotates. The combination of these implies that the full state vector, comprising the local and global state vectors, is constantly changing in size. A similar problem arose in the structure from motion system of Harris et al. [9], which estimates egomotion from tracked corner features. The solution chosen there was to have separate Kalman filters running on local state vectors attached to each corner, updating the global state vectors at the end of each iteration. However this approach is sub-optimal because it does not fully impose the rigidity constraint upon which the egomotion problem was formulated. The good reason for using Harris’s approach is that the time complexity of the Kalman filter is a cubic function of state vector size (because it involves matrix multiplications and inversions), so implementing a full Kalman filter would be prohibitive.

One of the contributions of this correspondence is to show that a certain class of estimation problems involving a single global state and multiple local states can be formulated in such a
way that the time-complexity is linear with full state vector size, thus making them computationally tractable. The observation required to formulate this new method is that each local state is coupled with its local measurement (feature position) and the global state, but not to the other local states.

Our approach is exactly equivalent to the Kalman filter, but uses a different formulation that brings out important aspects of the chosen problem set. We take advantage of the above observation that the equations relating parameters to features do not include relations between features. Mathematically speaking, this implies that the inverse covariance matrix of the full state vector will contain mostly zeros, as is shown below. It turns out that this property of the inverse covariance matrix means that it can be calculated in time linear with the number of features using an updating method as in the normal Kalman filter, and from this the state vector can be determined, also in linear time. Because we use inverse covariance the filter is related to the information filter [15]. The other benefit of our formulation is that it allows features to be removed from the state vector, by simply removing the corresponding row and column from the inverse covariance matrix. Subsequent updates of inverse covariance and state vector are exactly as if the features had been retained in the state vector. In other words we can fit the state vector dimension to the number of current features.

A. The filter in detail

We will first discuss the operation of the filter in general terms, and describe its use for calibration in Section V. Let us write the set of global parameters as a vector \( x \), corresponding to the local lengths and mechanical alignment errors for the camera calibration problem. Let the features being tracked have parameters to be estimated \( y_i, i = 1 \ldots n \) (the corner 3D world coordinates) and call these parameters the local parameters. The whole parameter set \( x, y_1, \ldots, y_n \) is termed the full state vector and is related to observations \( z_i(j) \) at time \( j \) by the measurement equation

\[
z_i(j) = h_i(j; x, y_i) + w_i(j)
\]

where \( w_i(j) \) is a Gaussian distributed vector with mean 0 and covariance \( P_0 \). Let the prior estimate of \( x \) be \( \hat{x}_0 \) with covariance \( P_0 \). We first consider the whole history of features up to time \( k \) and define \( d_i(j) \) as follows:

\[
d_i(j) = \begin{cases} 1 & \text{if feature } i \text{ is tracked at time step } j \\ 0 & \text{otherwise} \end{cases}
\]

for \( i = 1 \ldots n, j = 1 \ldots k \). The maximum likelihood solution for \( x, y_1, \ldots, y_n \) then minimises

\[
J(\hat{x}, \hat{y}_1, \ldots, \hat{y}_n) = (\hat{x} - \hat{x}_0)^T P_0^{-1}(\hat{x} - \hat{x}_0) + \sum_{i=1}^{n} \sum_{j=1}^{k} d_i(j) v_i(j)^T R_i(j)^{-1} v_i(j)
\]

where \( v_i(j) = z_i(j) - h_i(j; \hat{x}, \hat{y}_i) \) is the innovation. The first term measures the disagreement between the full state vector and the initial estimate \( \hat{x}_0 \), while the second term includes all measurements up to time \( k \). Note that no state dynamics are involved: \( x \) and \( y_i \) are assumed to be constant vectors.

We digress at this point to analyse the simplified problem of recursively estimating a single constant state vector \( x \) from measurements \( z(j) \), and subsequently apply the results to the complete system presented above. Given an estimate \( \hat{x}(k) \) of \( x \) at time step \( k \) with covariance \( P(k) \), and a new measurement \( z(k+1) = h(k+1; x) + w(k+1) \), the new estimate \( \hat{x}(k+1) \) should minimise

\[
J(\hat{x}(k+1)) = (\hat{x}(k+1) - \hat{x}(k))^T P(k)^{-1}(\hat{x}(k+1) - \hat{x}(k)) + \nu(k+1)^T R(k+1)^{-1} \nu(k+1)
\]

where \( R(k+1) \) is the covariance of the zero-mean measurement noise \( w(k+1) \). Linearizing around the previous estimate \( \hat{x}(k) \) and dropping the time step index for simplicity, we obtain the update rules

\[
P^{-1} = P^{-1} + H^T R^{-1} H, \quad \hat{x} \leftarrow \hat{x} + x_d
\]

which is a \( \chi^2 \) random variable and thus can be used to test agreement with the model. The number of \( \chi^2 \) degrees of freedom becomes \( \text{DOF} = \text{DOF} + \text{SIZE}(z) \), where \( \text{SIZE}(.) \) returns the size of its vector argument.

We can now apply these results to the case of estimating multiple state vectors \( x \) and \( y_i \). Consider first bundling all the \( y_i \)'s into a single vector \( y \). Thus the complete state vector is \( (x^T \ y^T)^T \). We accordingly partition the inverse covariance matrix \( P^{-1} \) into

\[
P^{-1} = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}
\]

The update rules for \( A, B \) and \( C \) given a new measurement \( z = h(x, y) + w \) are

\[
A \leftarrow A + D^T R^{-1} D, \quad B \leftarrow B + D^T R^{-1} E, \quad C \leftarrow C + E^T R^{-1} E
\]

where \( D \) and \( E \) are the individual Jacobians \( \partial h_\partial x \) and \( \partial h_\partial y \).

To use Equation (5) we require \( P \) itself, which can be written as:

\[
P = \begin{pmatrix} (A - BC^{-1} B^T)^{-1} - (A - BC^{-1} B^T)^{-1} B C^{-1} \\ -C^{-1} B^T (A - BC^{-1} B^T)^{-1} (C - B^T A^{-1} B)^{-1} \end{pmatrix}
\]

which means that we can write the changes in the state estimates as

\[
x_{d} = (A - BC^{-1} B^T)^{-1}(D^T - BC^{-1} E^T) R^{-1} \nu \quad y_{d} = C^{-1}(E^T R^{-1} \nu - B^T x_d)
\]

The reason for using this form of the state update formulae rather than directly using Eq. (5) is that when applied to the case of multiple \( y \)'s not coupled by the measurement process, the above formulae have lower computational complexity. This is the crux of the VSDF approach. The residual and degrees-of-freedom update rules are

\[
J \leftarrow J + \nu^T R^{-1}(\nu - Dx_d - Ey_d), \quad \text{DOF} \leftarrow \text{DOF} + \text{SIZE}(z).
\]
With multiple $y_i$'s, each with measurements $z_i = h_i(x, y_i) + w_i$, and with the noise vectors $w_i$ having covariances $R_i$, we can write $z$, $R$, $D$ and $E$ as "stacked" vectors and matrices:

$$
\begin{align*}
z &= \left( z_1 \ldots z_n \right), \\
R &= \left( \begin{array}{ccc} R_1 & \cdots & 0 \\
0 & \ddots & \vdots \\
0 & \cdots & R_n \end{array} \right), \\
D &= \left( \begin{array}{ccc} D_1 \\
\vdots \\
D_n \end{array} \right), \\
E &= \left( \begin{array}{ccc} E_1 \\
\vdots \\
E_n \end{array} \right).
\end{align*}
$$

Because $E$ and $R$ are now block-diagonal, we can also write the inverse covariance matrix as

$$
P^{-1} = \left( \begin{array}{ccc} A & B_1 & \cdots & B_n \\
B_1^T & C_1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & C_n \end{array} \right),
$$

The update rules for the terms of $P^{-1}$ are

$$
A \leftarrow A + \sum_{i=1}^{n} D_i^T R_i^{-1} D_i, \quad B_i \leftarrow B_i + D_i^T R_i^{-1} E_i,
$$

$$
C_i \leftarrow C_i + E_i^T R_i^{-1} E_i.
$$

(6)

Having updated $P^{-1}$, the state estimates are updated using the increments

$$
\dot{x} \leftarrow \dot{x} + M^{-1} \sum_{i=1}^{n} (D_i^T - B_i C_i^{-1} E_i^T) R_i^{-1} \nu_i
$$

where

$$
M = A - \sum_{i=1}^{n} B_i C_i^{-1} B_i^T.
$$

$$
\dot{y}_i \leftarrow \tilde{y}_i + C_i^{-1} (E_i^T R_i^{-1} \nu_i - B_i^T x_d), \quad i = 1 \ldots n.
$$

(7)

Note that if there is no measurement for a local state $i$, corresponding to $d_i$ in Eq. (3) being zero, $B_i$ and $C_i$ remain unchanged and the $i$th term in the update of $A$ is ignored. The increment applied to $\tilde{y}_i$ in this case is $-C_i^{-1} B_i^T x_d$. The $\chi^2$ residual and degrees-of-freedom update formulae are

$$
J \leftarrow J + \sum_{i=1}^{n} \nu_i^T R_i^{-1} \nu_i - D_i^T x_d - E_i y_i,
$$

$$
DOF \leftarrow DOF + \sum_{i=1}^{n} SIZE(z_i),
$$

where $x_d$ and $y_i$ are the increments applied to $\dot{x}$ and $\dot{y}_i$ respectively.

The above update formulae constitute the VSDF algorithm for achieving the minimization of $J$ in Eq. (4). By inspection it is clear that the computational complexity of a complete update for a single time-step is linear with $n$, as opposed to cubic if Eq. (5) were to be used directly. It remains to describe how the VSDF is initialized, and how the local state vectors $y_i$ are added and removed.

A.1 Initialization

The filter is given an initial estimated value $\tilde{x}_0$ of the global state vector $x$ and a covariance $P_0$. The following matrix and scalars are initialised: $A = P_0^{-1}$, $n = 0$, $J = 0$, $DOF = 0$.

A.2 Introducing a new local state

A local state vector $y_{n+1}$ is introduced, with initial estimate $\tilde{y}_{n+1}$ (determined from the initial measurement). New matrices $B_{n+1}$ and $C_{n+1}$ are initialized to zero. The $\chi^2$ degrees of freedom is decremented: $DOF \leftarrow DOF - SIZE(y_{n+1})$, because the number of estimated parameters is increasing by $SIZE(y_{n+1})$. Finally $n$ is incremented: $n \leftarrow n + 1$. This procedure effectively adds a new column and row block to the inverse covariance matrix $P^{-1}$.

A.3 Removing local states

Now we consider how to remove local states from the state vector, in the event of no more data becoming available for a certain feature. Let us say that state $y_i$ is to be removed. We shall therefore remove row and column block $l + 1$ in the inverse covariance matrix $P^{-1}$. However we must ensure that this does not affect subsequent calculations of the reduced state when new observations are made. This can be accomplished by subtracting from $A$ the following term: $A \leftarrow A - B_l C_l^{-1} B_l^T$. $B_l$ and $C_l$ are discarded and state $n$ is shifted down to position $l$ in the remaining states to fill the gap. Finally $n$ is decremented: $n \leftarrow n - 1$. Note that no extra storage is required to maintain the information about discarded states, and also that the complexity of every stage of the filter is linear in the number of features $n$.

V. IMPLEMENTING THE CALIBRATION ALGORITHM

The pair of equations (2) can be written as $z = h(x, y) + w$ with the definitions $x = (f_x, f_y, \phi_x, \phi_y, \phi_z)^T$, $y = P$ and $z = (x, y)^T$. The measurement function $h$ is defined as $h(x, y) = (-f_x (R^T P)/(R^T P) - f_y (R^T P)/(R^T P))^T$, where $R_i$ are the rows of the rotation matrix $R$, defined in terms of $\theta_x$, $\theta_y$, $\phi_x$ and $\phi_y$. $x$ constitutes the global set of parameters to be estimated and $z$ the measured position of the projection of $P = y$. We drop the subscripts $i$ in this section and consider a single tracked feature. To implement the VSDF recursive update we need to calculate the Jacobian matrices of $h$ with respect to $x$ and $y$, which are readily obtained by differentiation. We now fill in details concerning implementation of camera calibration in the VSDF framework.

A. Initialization

Initial values $f_x$ and $f_y$ of the focal lengths $f_x$ and $f_y$ in pixels are estimated (in our experiments we have observed an upper limit on the initial error of $\sim 50\%$). The angles $\phi_x$, $\phi_y$ and $\phi_z$ are assumed to be small so their initial values may be taken to be zero. We set the initial covariance $P_0$ to be diagonal, with entries $\sigma_f^2$ for the focal lengths and $\sigma_\theta^2$ for angles, where $\sigma_f$ and $\sigma_\theta$ are set to 30 pixels and 0.1 radians respectively, large to indicate lack of confidence in these initial estimates. The effect of the initial covariance declines over time.

B. Corner matching

Corners are detected [25] at frame-rate (25Hz) in a 64 x 32 pixel central image window. The assumption that tracked feature $i$ corresponds to a stationary scene point means that we can predict the position of a feature in a new image from the previous estimates $\hat{P}_i$ and $\hat{X}$ of the visual direction and calibration parameters, and the new head angles $\theta_x$, $\theta_y$, using Eqs. 2. We search for the feature in a small (3 x 3 pixel) window centred on the predicted position. If a corner is found we accept the match if the pixels around it are similar to those around the feature detected in the previous image. We test this by summing the absolute differences in pixel grey-level value over
a 7 × 7 window around each feature and dividing by the mean pixel value of both. If this ratio is less than 0.1 we accept the match. This provides a strong constraint on matching. We are more concerned with eliminating the possibility of false matches than achieving a match for every possible feature. If more than one possible match is found, all are rejected, again to help prevent false matches. If a unique match is found, the new feature position \(x_i, y_i\) is incorporated as a new observation (see next Section).

If a match cannot be found for a feature (or if there are ambiguous matches) it may still reappear in subsequent frames. We allow a feature to remain “invisible” for three consecutive frames before discarding it (i.e. removing its local state vector \(P_i\)). Fig. 2 shows several frames taken during a run of the calibration algorithm, with the tracked corners superimposed.

C. Incorporating new observations

The VSDF update procedure allows an arbitrary subset of the current local states \(i = 1, \ldots, n\) to have new observations incorporated. This subset corresponds to those features \(i\) with unique matches in the latest frame at image positions \(x_i, y_i\). We have \(z_i = (x_i, y_i)^T\), and take the covariance \(R_i\) of \(z_i\) to be diagonal with entries \(\sigma_{x_i}^2\), where \(\sigma_{xy} = 0.5\), corresponding to the estimated random error in feature positions.

D. Adding a new local state

A new corner feature \(n + 1\) appears, with initial image position \(x_{n+1, 0}, y_{n+1, 0}\). We determine the initial estimate of the 3D world direction \(P_{n+1}\) from the current parameters by rearranging Eqs. (2) into two linear equations in \(P_x\) and \(P_y\), which are then solved. The rotation matrix \(R\) is calculated using the latest estimate \(\hat{x}\) of \(x\) and for image position \(x_{n+1, 0}, y_{n+1, 0}\) with head angles \(\theta_x, \theta_y\), at the time the image was taken.

We add a new local state vector using the procedure in Section IV-A.2 and then pass the first observation into the VSDF.

E. General

We have implemented two versions of the algorithm on Yorick. The first calibrates only for focal length, the other is the full calibration algorithm described above. In both cases corner data from every sixth frame of the 25Hz image stream is used, so the algorithm runs at 4.25Hz. The focal length only algorithm can easily be constructed by redefining \(x\) as \((f_x, f_y)^T\) and removing all terms in \(\phi_\alpha, \phi_\beta\) and \(\phi_\gamma\) from the equations in Section V, effectively forcing them to zero.

In order to run the calibration algorithm (either version) the camera is first aligned on the head by hand. Initial values for the focal lengths are provided and the algorithm is started, simultaneously commencing head motion. At present we make the head follow a fixed trajectory, zig-zagging at constant angular speed of 6° s\(^{-1}\) for both elevation and azimuth. The range is limited to 20° for elevation, 30° for azimuth. When a limit is reached the head “bounces” back, reversing the relevant velocity. The trajectory does not affect the performance so long as it is chosen to cover a large enough range of the “space” of head angles. Thus if a purely horizontal trajectory were chosen, i.e. one that changed azimuth only, it would be impossible to measure the vertical focal length \(f_y\). We emphasize that the structure of the scene is unimportant provided it is greater than 2m distant.

VI. Results

A. Results for focal length algorithm

In Fig. 3 we show results for five runs of the focal length algorithm. The horizontal axis essentially measures time, but because the amount of data in the form of corner matches varies with time (there is not much to match on the ceiling, for example) we instead use the cumulative number of corner matches as our “time” axis. On average 3.5 matches are received every frame, so the total time for the above results to be obtained on each run was 3000/3.5/4.25 = 200 seconds. The five runs used different starting values of \((f_x, f_y)\): \((800,200), (700,600), (300,250), (350,650)\) and \((650,275)\). Thus for initial errors up to 50% the algorithm still converges. Of course the larger the error in the calibration parameters, the worse the feature prediction is, so less matches are found. The estimates must be good enough so that a sufficient number of matches are found to improve the calibration, thus improving the matching, etc. For larger initial errors one might increase the size of the matching search window, but we do not envisage this as being necessary.

In each case the algorithm converged to the same values. The mean value of \(f_x\), taken at the end of each run is 523.3 with a standard deviation of 1.7 pixels, i.e. 0.3% of the mean value. The mean and standard deviation of \(f_y\) are 391.8 and 1.46 (4.4% of the mean). Also shown plotted is the prediction error. This is the only absolute error measure available (equivalent to epipolar error in a stereo calibration algorithm [24]). It measures the root-mean-square (RMS) error between the predicted position of a feature and its actual position in the image. Since the tracking search window is limited to one pixel either side of the predicted position, and corner pixel positions are integers, the RMS error is in practice restricted to the range \((0.5,1)\), with occasional “lucky dips” below 0.5. As can be seen, the RMS error very quickly drops to near 0.5, showing that the tracking error is very small.

We should note that the lack of a “ground-truth” for the calibration parameters is not important for our current purposes. We wish to be able to track moving objects using the general motion tracker [20], and for this we must make predictions about image motion based on known head motion. The RMS prediction error in the calibration is exactly the measure that tells us
Fig. 3. Results for five runs of the focal length calibration algorithm, with different initial values of focal length. The horizontal axis measures the cumulative number of matches. The time taken for each run was approximately 200 seconds.

Fig. 4. Results for eight runs of the full calibration algorithm, including camera alignment offset angles. The horizontal axis measures the cumulative number of matches.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$f_x$</th>
<th>$f_y$</th>
<th>$\phi_x$</th>
<th>$\phi_y$</th>
<th>$\phi_z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit</td>
<td>pixels</td>
<td>pixels</td>
<td>radians</td>
<td>radians</td>
<td>radians</td>
</tr>
<tr>
<td>Mean</td>
<td>528.3</td>
<td>387.8</td>
<td>-0.028</td>
<td>0.054</td>
<td>-0.058</td>
</tr>
<tr>
<td>Std. dev.</td>
<td>0.9</td>
<td>1.0</td>
<td>0.007</td>
<td>0.011</td>
<td>0.0011</td>
</tr>
<tr>
<td>Ratio as %</td>
<td>0.2</td>
<td>0.3</td>
<td>25%</td>
<td>20%</td>
<td>20%</td>
</tr>
</tbody>
</table>

TABLE 1

how well we can do this.

B. Results for the full algorithm

Eight runs of the full calibration algorithm are shown in Fig. 4. As before the focal lengths quickly converge to the same value, slightly different from the values in the previous experiment because of the different calibration method and change in focussing adjustment between the runs. The angular alignment offsets take longer to converge, because of the smaller effect they have. The runs each took about 10 minutes, at which point the parameters were still converging. The results at the end of the runs are summarised in Table 1.

As expected the percentage errors in the alignment offsets are much larger than for the focal lengths, because they are small and have less effect. Perhaps slightly surprising is a measurable cyclotorsion $\phi_z$ of about $-0.3^\circ$, indicating either a misalignment of the camera mount or of the CCD within the camera.

Utilising the coupling between the image centre and the alignment offsets, we can see that the “effective” image centre is located at $(f_x\phi_y, -f_y\phi_x)$, i.e. at (28, 11) for the results in table I. At present our general motion detection and tracking algorithms [17], [20] use only the focal lengths. These are sufficient to perform predictions from image to image.

VII. DISCUSSION AND CONCLUSION

We have applied the variable state-dimension filter to the problem of calibrating a single camera mounted on a robot head by tracking corner features and utilising accurate knowledge of the camera rotation. The approach can be extended in several ways. Firstly we could use any image features, for instance line segments, as we have done for 3D structure and motion recovery. We would like to explore the issue of robustness and outlier detection more thoroughly than we have thus far. The feature matching algorithm we employ is very restrictive and does not allow many false matches, but there are situations when the model can break down. If the calibration changes, due for instance to manual refocussing, the VSDF will have to be reset. The breakdown can theoretically be detected by testing the residual $J$ using a $\chi^2$ test on DOF degrees of freedom. Mismatches for a given feature can also be tested for by measuring the change in $J$ that would occur if the measurements for that feature were removed. The residual change is also a $\chi^2$ random variable. We have implemented this test, but space restrictions prevent us giving full details.

We believe that the variable state dimension filter is the first algorithm that possesses all the essential properties for real-time vision of recursiveness, low computational complexity with respect to image data size, and statistically rigorous derivation. It has already been applied to structure-from-motion recovery.
where the similar problems of varying state size are encountered [16].

VIII. ACKNOWLEDGEMENTS

The authors are grateful for discussion with Paul Sharkey, Ian Reid, Mike Brady, Paul Beardsley, Andrew Zisserman and Larry Shapiro. The VSDF software is available as part of the Horatio vision libraries from the first author.

REFERENCES


