A Primer on XIVO

Stephanie Tsuei, Xiaohan Fei

February 24, 2023

XIVO, or "Xiaohan's Inertial-aided Visual Odometry" is a SLAM algorithm and software package based on the Extended Kalman Filter (EKF) that estimates state using measurements of angular velocity and linear acceleration from an IMU and feature tracks extracted from a stream of RGB images. XIVO was first introduced as "Corvis" in [5], which presented a basic observability analysis. In the original implementation of Corvis, the equations of motion contained no inputs – both the IMU measurements and feature tracks are modeled as system outputs. By 2015, Corvis had been modified so that IMU measurements would become model inputs and only the feature tracks were system outputs. This second version is the model analyzed in [4]. Additional features added at that time, some of which made Corvis no longer an EKF, were One-Point RANSAC outlier rejection [1], advanced feature covariance initialization, and the option to use a Huber Loss instead of the standard L2 loss in the EKF measurement update step.

When technical debt was deemed insurmountable in 2019, the Corvis software was scrapped and rewritten as XIVO by Xiaohan Fei. XIVO was then maintained, instrumented, and modified as needed for the experiments detailed in this thesis. The basic equations are still those in [4] – IMU measurements are system inputs. However, not all features of Corvis were kept and there are more (optional) calibration states than those mentioned in [4] and [5]. This Appendix aims to give the full description of the state, autocalibration state, and dynamics equations implemented in XIVO as of January 2023.

The steps implemented in XIVO are: EKF prediction, EKF measurement update, feature detection and tracking, 2D outlier rejection, 3D outlier rejection, and loop closure. A flowchart of the steps is given in Figure 1. The software modules are listed in Figure 2. This Appendix is a guide to understanding the components of XIVO, but is not a user's guide or software documentation.

1 Preliminaries

1.1 Notation

We use the following conventions for notation in this Appendix:

- Vectors are represented by lower-case symbols, such as x, y, and z. Individual scalar elements inside the vectors are denoted with subscripts, e.g. x_1, x_2, x_3 for $x \in \mathbb{R}^3$. Vector indices are 1-indexed (like in Matlab).
- Matrices are represented with upper-case symbols. All upper-case symbols are matrices, unless specifically specified to be something else below.
- Coordinated frames are designated as lower-case subscripts, e.g. x_a refers to a vector x resolved in coordinate frame a.
- An upper-case R, with or without subscripts, always refers to a rotation matrix; $R \in SO(3)$.
- An upper-case T, with or without subscripts, always refers to the translation component of a rigid body transformation; $T \in \mathbb{R}^3$
- A lower-case g pared with no subscripts always refers to the gravity vector $[0, 0, -9.8]$ m/s^2.

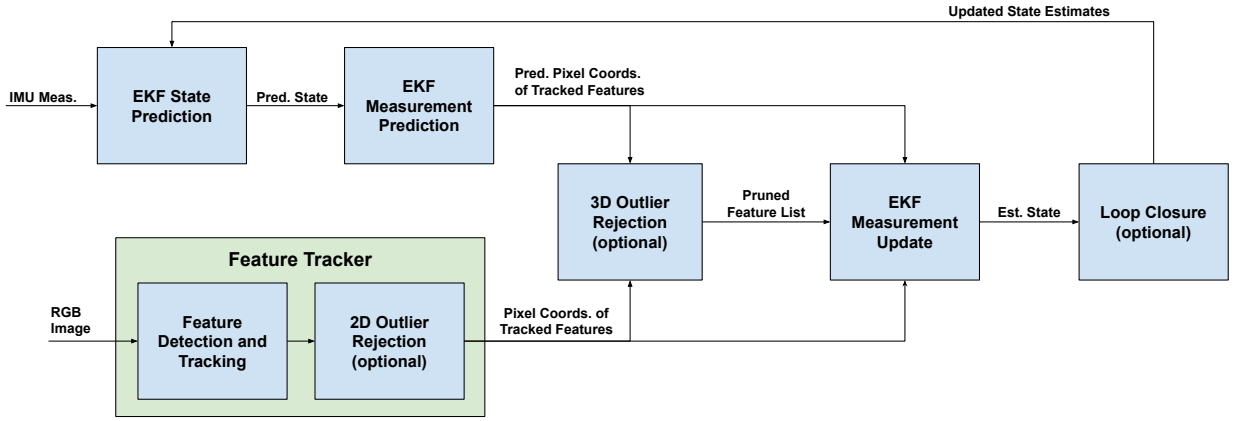


Figure 1: **XIVO Overview.** IMU Measurements are first used to propagate the estimated state χ forward in time. After propagation of χ, XIVO then calculates predicted image locations of all features currently tracked by the Feature Tracker. In the Feature Tracker, feature tracks are extracted from RGB images using either Lucas-Kanade Sparse Optical Flow or Correspondence Matching. Feature Tracks may then be pruned for outliers using planar outlier rejection (e.g. RANSAC, LMEDS, RHO), an optional step when using sparse optical flow, but effectively a required step when using Correspondence Matching. For further rejection of outlier feature tracks, XIVO also contains implementations of the 3D outlier rejection algorithms Mahalnobis Gating and One-Point RANSAC [1]. With all outliers removed, the last steps are the standard EKF measurement update and an optional loop closure.

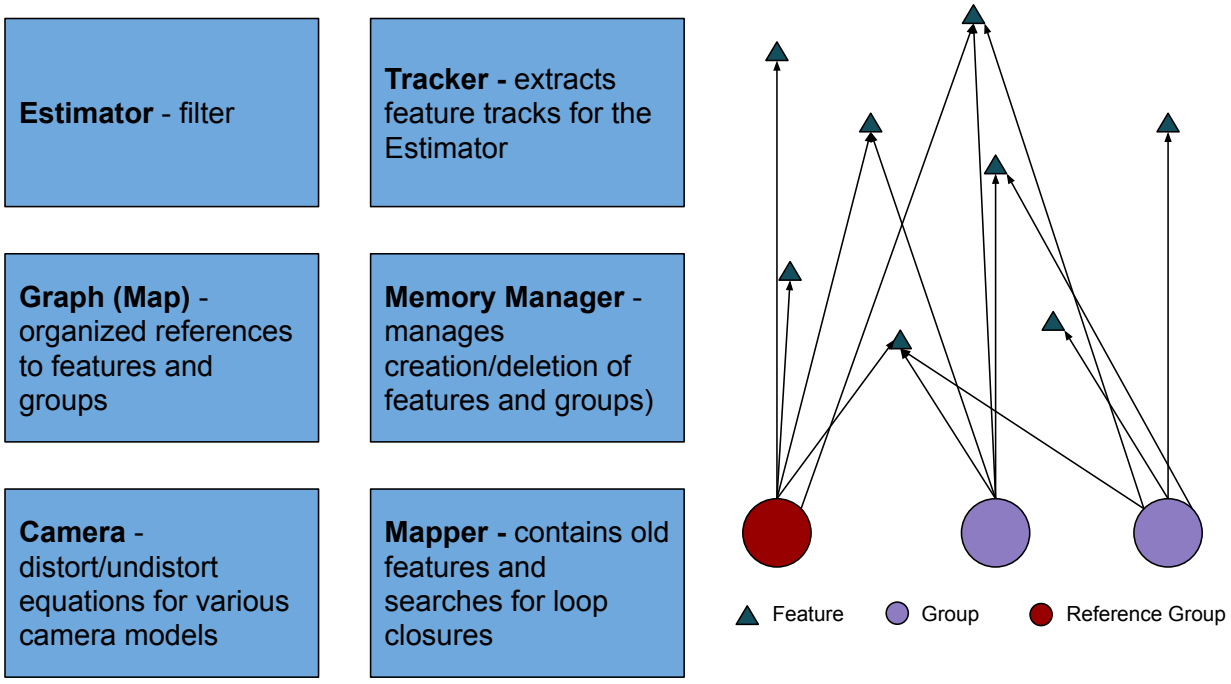


Figure 2: **Software Objects in XIVO.** The `Estimator`, `Tracker`, `Graph`, `Memory Manager`, `Camera`, and `Mapper` are implemented as C++ Singletons. `Features` and `Groups` exist outside the Singletons. The components that require access to `Features` and `Groups` contain organized pointers.

- A lower-case w, with or without subscripts, always refers to a rotation vector. $R(w)$ is the rotation matrix corresponding to the rotation vector.
- A lower-case ω always refers to angular velocity.
- An upper-case V always refers to linear velocity.
- A lower-case a, when not part of a subscript, always refers to linear acceleration.
- Linear velocity, angular velocity, and linear acceleration are written with three subscripts. For example ω_{sb}^{b} is the angular velocity of the body frame with respect to the spatial frame resolved in the body frame.
- The upper-case symbols X, Y, Z with no subscripts represent the x, y, and z-coordinates of a position of a feature in $\mathbb{R}^3$.
- The symbol X with a subscript q (e.g. $X_q = [X, Y, Z]$) is a vector in $\mathbb{R}^3$ containing the position of a feature or group in coordinate frame q.
- The symbol χ represents the entire EKF-state.
- For $x \in \mathbb{R}^3$, $[x]_\times$ is the skew-symmetric matrix constructed with elements of x.
- The projection operator $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is given by:

$$\pi(x) = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix}$$

- The camera intrinsics operator is denoted $\pi^c : \mathbb{R}^2 \to \mathbb{R}^2$ and depends on the camera model. π^c depends on both the camera intrinsics matrix K and any lens distortion parameters.
- A row of dots $\dots$ at the top, bottom, or side of a matrix indicates that the rest of the matrix is all zeros. For example:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \dots & \dots & \dots \end{bmatrix}$$

1.2 Representation of 3D Rotations and Coordinate Transformations

Our notation is consistent with Chapter 2 of [8]. Consider two coordinate systems a and b and a point $x_b \in \mathbb{R}^3$, resolved in b. Then, the coordinates of the same point resolved in coordinate system a is given by:

$$x_a = R_{ab} x_b + T_{ab}. \tag{1}$$

From (1), we have the following expressions for the inverse:

$$\begin{aligned} x_b &= R_{ab}^{-1}(x_a - T_{ab}) \\ R_{ba} &= R_{ab}^{-1} = R_{ab}^{T} \\ T_{ba} &= -R_{ab}^{-1} T_{ab}. \end{aligned} \tag{2}$$

XIVO uses rotation vectors to represent 3D rotation in its state equations, since rotation vectors are the tangent space $so(3)$ of group $SO(3)$. For a rotation vector $w = [w_1, w_2, w_3]^T$, the magnitude of w, $\theta = \|w\|$ is the magnitude, or angle in radians, and $\hat{w}$ is the axis. The rotation matrix representation of w is given by

$$R(w) = \exp\left([w]_\times\right)$$

where exp is the matrix exponential.

In order to avoid singularities and for ease of numerical integration, however, XIVO represents internal rotations using quaternions rather than rotation vectors. The Sophus library[1] is used for quaternion integration, and for conversion to rotation matrices and rotation vectors as needed.

1.3 Special Jacobians

Jacobian with Respect to a Rotation Vector A closed-form expression for The Jacobian of $R(w)$ is given in [2]. However, due to issues with numerical stability, we do not derive dynamics and measurement Jacobians using the equations in [2]. Instead, we substitute the first-order Taylor series approximation $R(w) \approx I + [w]_\times$ and then compute derivatives with respect to w. The identity $[w]_\times a = -[a]_\times w$ is useful in the derivatives.

Matrix Product Jacobian Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. Then,

- $\frac{\partial AB}{\partial B}$ has dimension $mp \times np$. If the size of B is known, it is a function of the matrix A. In the texts below, this is written as a function $\frac{\partial AB}{\partial B}_{n,p}(X)$.
- $\frac{\partial AB}{\partial A}$ has dimension $mp \times mn$. If the size of A is known, it is a function of the matrix B. In the texts below, this is written as a function $\frac{\partial AB}{\partial A}_{m,n}(X)$.
- $\frac{\partial A^T}{\partial A}$ has dimension $mn \times mn$. It is a function of the matrix A. In the texts below, this is written as a function $\frac{\partial A^T}{\partial A}(X)$.
- $\frac{\partial A}{\partial A^u}$, where A_u is the upper triangle of a square matrix $A \in \mathbb{R}^{m \times m}$, has dimension $m^2 \times \frac{m(m+1)}{2}$ and is a constant. In the texts below, this is written as $\frac{\partial A}{\partial A^u}(X)$.

2 Filter Equations

2.1 Coordinate Frames

XIVO uses the following coordinate frames:

- Spatial/world frame s – A static frame that is never moving. For issues related to observability [4, 5], it is fixed to be the position and orientation of the IMU at startup.
- Body/IMU frame b – A frame attached to the IMU. The exact direction of the axes are determined by the manufacturer and how the IMU is mounted.
- Camera frame c – A frame attached to the camera's pinhole. The Z axis points outward perpendicular to the image plane. The X axis points horizontally to the right side of the image plane. The Y axis completes the triad, i.e. the Y axis points down.
- Pixel frame p – A 2D frame whose origin is the top-left corner of the image. The projection operator and the camera intrinsics map points in the c frame with units of meters to points in the p frame with units of pixels.
- Gravity frame g – A frame that is aligned with gravity and co-located with the b frame. In this frame, gravitational acceleration has value $[0, 0, -9.8]^T$.
- Reference body frame b_r – A frame fixed at the past position and orientation of a body frame b. The EKF state χ will contain multiple values of g_{sb_r}, but no more than one instance of g_{sb_r} will appear in any equation.

[1] `https://github.com/strasdat/Sophus`

- Reference camera frame c_r – A frame fixed at the past position and orientation of a camera frame c. The transformation $g_{sc_r} = g_{sb_r} \circ g_{bc}$

2.2 States

Ego-state: These states describe the position and orientation of the IMU relative to its initial condition. The dimension of each state is shown in parentheses:

- w_{sb} (3), T_{sb} (3) - pose of the body frame with respect to the spatial frame, i.e. $x_s = R(w_{sb})x_b + T_{sb}$.
- V_{sb}^b (3) - velocity of the body frame with respect to the spatial frame.
- b_g^b (3) - bias of the IMU's gyroscope in the body frame, i.e.

$$\bar{\omega}_{sb}^b = \omega_{sb}^b - b_g^b \tag{3}$$

  where ω_{sb}^b is the measured angular velocity and $\bar{\omega}_{sb}^b$ is the true angular velocity.

- b_a^b (3) - bias of the IMU's accelerometer in the body frame, i.e.

$$\bar{a}_{sb}^b = a_{sb}^b - R_{bs}R_{sg}g - b_a^b \tag{4}$$

  where a_{sb}^b is the measured angular velocity and $\bar{a}_{sb}^b$ is the true angular velocity.

Calibration States: These states enable autocalibration. The dimension of each state is shown in parentheses. Optional states are those that may be excluded from the state vector.

- w_{bc} (3), T_{bc} (3) - pose of the camera frame with respect to the body frame.
- w_{sg} (2) - First two elements of the orientation of the gravity vector with respect to the spatial frame. Since the rotation around the z-axis of the gravity frame is unobservable (and does not matter), we take the third element of w_{sg} to be 0.
- (Optional) t_d (1) - An estimate of the time difference between when an image is acquired and when an image is timestamped.
- (Optional) C_g (9) and C_a (6) - IMU Calibration parameters. If enabled, equations (3) and (4) become

$$\begin{aligned} \bar{\omega}_{sb}^b &= C_g\omega_{sb}^b - b_g^b \\ \bar{a}_{sb}^b &= C_a(a_{sb}^b - R_{bs}R_g g) - b_a^b \end{aligned} \tag{5}$$

  where the matrices C_a and C_g take the form

$$\begin{aligned} C_a &= \begin{bmatrix} C_{a,1} & C_{a,2} & C_{a,3} \\ 0 & C_{a,3} & C_{a,4} \\ 0 & 0 & C_{a,6} \end{bmatrix} \\ C_g &= \begin{bmatrix} C_{g,1} & C_{g,2} & C_{g,3} \\ C_{g,4} & C_{g,5} & C_{g,6} \\ C_{g,7} & C_{g,8} & C_{g,9} \end{bmatrix}. \end{aligned} \tag{6}$$

  An IMU with no misalignment or scale error has $C_a = C_g = I_{3\times3}$. These parameters can be found using the procedure detailed in [11].

- (Optional) Intrinsic camera calibration parameters (up to 9). The exact number of camera calibration parameters depends on the chosen model and includes both projection and distortion parameters. XIVO currently supports the pinhole (4 parameters), equidistant (8 parameters), arctangent (5 parameters), and radial-tangential (9 parameters) models.

Map states:

- w_{sb_r} (3 per group), T_{sb_r} (3 per group) - pose of each group with respect to the spatial frame, i.e. the value of w_{sb} and T_{sb} at the time the group was created.
- x_{c_r} (3 per feature) - position of each feature resolved in the camera frame in a log-depth or inverse-depth representation *at the time the feature was first observed*. In other words, if the position of the feature in the camera frame at the time it was first observed is $[X, Y, Z]$, then $x_{c_r} = [X/Z, Y/Z, \log(Z)]$ or $x_{c_r} = [X/Z, Y/Z, 1/Z]$. Whether to use a log-depth or an inverse-depth representation is a compile-time parameter.

2.3 The Error State

Due to issues of numeric stability, we do not estimate the mean and covariance of the entire state directly. Instead, we estimate the state and the covariance of the error state. At the end of each measurement update, the error is "absorbed" into the state. More precisely, let

- n_x be the dimension of the state
- n_u be the dimension of the system input
- n_y be the dimension of the measurements
- $\chi \in \mathbb{R}^{n_x}$ be the state and $\hat{\chi} \in \mathbb{R}^{n_x}$ be the estimated state.
- $e \in \mathbb{R}^{n_x}$ be the error state
- $u \in \mathbb{R}^{n_u}$ be the system input, $\bar{u}$ be the nominal, noise-free input
- $y \in \mathbb{R}^{n_y}$ be the measurements
- P be the covariance of e
- $\mu \sim \mathcal{N}(0, Q)$ be additive noise to u
- $\eta \sim \mathcal{N}(0, R)$ be sensor noise

The dynamics and measurement model are given by:

$$\begin{aligned} \dot{\chi} &= f(\chi, e, u, \mu) \\ y &= h(\chi, e) + \eta. \end{aligned} \tag{7}$$

Note that even though the error state is additive to the state and noise terms are additive to u and y, they can be incorporated nonlinearly into $f(\chi, e, u, \mu)$ and $h(\chi, e)$. During the prediction step of the EKF, we integrate the continuous dynamics one timestep:

$$\begin{aligned} \dot{\hat{\chi}} &= f(\hat{\chi}, \bar{u}) \\ \dot{P} &= FP + PF^T + GQG^T \\ \dot{e} &= 0 \end{aligned} \tag{8}$$

where $\hat{\chi}$ is the current estimated value of χ, $\bar{u}$ is the nominal value of the input, $F = \frac{\partial f}{\partial e}|_{e=0,\chi=\hat{\chi}}$ and $G = \frac{\partial f}{\partial u}|_{u=\bar{u},\mu=0}$. The measurement step then becomes

$$\begin{aligned} K &= PH^T(HPH^T + R)^{-1} \\ e &= K(y - h(\hat{\chi}, e)) \\ P &= (I - KH)P(I - KH)^T + KRK^T \end{aligned} \tag{9}$$

where $H = \frac{\partial h}{\partial e}|_{e=0,\chi=\hat{\chi}}$, y is the measurement, and K is the Kalman gain.

Finally, after each measurement step, we reset the error state:

$$\begin{aligned} \hat{\chi} &\leftarrow \hat{\chi} + e \\ e &\leftarrow 0. \end{aligned} \tag{10}$$

2.4 Nominal Equations of Motion (EKF State Prediction)

Let t_0 and t_f be two timesteps at which IMU measurements are received. Since IMUs operate very fast, we will assume that the average value of the two measurements is the constant measured velocity and linear acceleration during this time. Denote these values as ω_{sb}^b and a_{sb}^b. These are the system inputs.

Adjusting for gravity, bias, and IMU calibration states,

$$\begin{aligned} \bar{\omega}_{sb}^b &= C_g \omega_{sb}^b - b_b^b \\ \bar{a}_{sb}^b &= C_a(a_{sb}^b - R_{bs}R_g g) - b_a^b \end{aligned} \tag{11}$$

Time derivatives of components of the state vector are then:

$$\begin{aligned} \dot{R}_{sb} &= R_{sb}[\bar{\omega}_{sb}^b]_\times \\ \dot{T}_{sb} &= R_{sb}V_{sb}^b \\ \dot{V}_{sb}^b &= \bar{a}_{sb}^b \end{aligned} \tag{12}$$

For issues related to observability, we enforce $R_{sb}(0) = I$ and $T_{sb}(0) = 0$ by holding one group in the map fixed (see 3.1). All other states (calibration and map states) are static and therefore have time derivatives equal to 0.

2.5 Nominal Measurement Model (EKF Measurement Prediction)

For each tracked feature, the low-level feature tracking measures the location of the feature in pixels $x_p = (x_{pix}, y_{pix})$. Let R_{sb_r} and T_{sb_r} be the pose of the feature's associated group. Then, the nonlinear measurement equation is:

$$x_p = \pi^c\left(\pi\left(X_c\right)\right) \tag{13}$$

where

$$\begin{aligned} X_c &= R_{bc}^T(R_{sb}^T(X_s - T_{sb}) - T_{bc}) \\ X_s &= R_{sb_r}(R_{bc}X_{c_r} + T_{bc}) + T_{sb_r} \end{aligned}$$

2.5.1 Camera Models

XIVO supports the following implementations of camera distortion models π^c. In the below text, let $x_c^{2d} = [X/Z, Y/Z]^T = [\bar{x}, \bar{y}]^T$

Pinhole (no distortion). Parameters: f_x, f_y, c_x, c_y

$$\begin{bmatrix} x_{pix} \\ y_{pix} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} \tag{14}$$

Equidistant. Parameters: $f_x, f_y, c_x, c_y, k_0, k_1, k_2, k_3$
Let $\theta = \arctan\left(\|x_c^{2d}\|\right)$ and $\phi = \arctan(\bar{y}, \bar{x})$. Then the pixel coordinates are:

$$\begin{aligned} x_{pix} &= f_x r \cos(\phi) + c_x \\ y_{pix} &= f_y r \sin(\phi) + c_y \end{aligned} \tag{15}$$

where

$$r = \theta + k_0\theta^3 + k_1\theta^5 + k_2\theta^7 + k_3\theta^9. \tag{16}$$

Radial-Tangential. Parameters: $f_x, f_y, c_x, c_y, p_1, p_2, k_1, k_2, k_3$ Let $r = \|x_c^{2d}\|$. Then,

$$\begin{aligned} x_{pix} &= c_x + f_x\left(2p_1\bar{x}\bar{y} + p_2(2\bar{x}^2 + r^2) + \bar{x}(1 + k_1r^2 + k_2r^4 + k_3r^6)\right) \\ y_{pix} &= c_y + f_y\left(2p_2\bar{x}\bar{y} + p_1(2\bar{y}^2 + r^2) + \bar{y}(1 + k_1r^2 + k_2r^4 + k_3r^6)\right). \end{aligned} \tag{17}$$

Arctangent. Parameters: f_x, f_y, c_x, c_y, w
Let

$$\begin{aligned} w_2 &= 2\tan\left(\frac{w}{2}\right) \\ f &= \frac{1}{w}\frac{\arctan w_2\|x_c^{2d}\|}{\|x_c^{2d}\|}. \end{aligned} \tag{18}$$

Then,

$$\begin{aligned} x_{pix} &= f_x f\bar{x} + c_x \\ y_{pix} &= f_y f\bar{y} + c_y. \end{aligned} \tag{19}$$

2.6 Incorporating the Error State and Noise into Nominal Equations

Let $\tilde{x}$ be a perturbation on variable x. Equations of motion that include the error state can be made by substituting the following perturbations into all equations in Sections 2.4 and 2.5.

- $w_{sb} \leftarrow w_{sb} + \tilde{w}_{sb}$ and $R_{sb} \leftarrow R_{sb}R(\tilde{w}_{sb})$
- $T_{bc} \leftarrow T_{bc} + \tilde{T}_{bc}$
- $V_{sb}^b \leftarrow V_{sb}^b + \tilde{V}_{sb}^b$
- $b_a \leftarrow b_a + \tilde{b}_a$
- $b_g \leftarrow b_g + \tilde{b}_g$
- $w_{bc} \leftarrow w_{bc} + \tilde{w}_{bc}$ and $R_{bc} \leftarrow R_{bc}R(\tilde{w}_{bc})$
- $T_{bc} \leftarrow T_{bc} + \tilde{T}_{bc}$
- $w_g \leftarrow w_g + \tilde{w}_g$ and $R_g \leftarrow R_gR(\tilde{w}_g)$
- **(Optional) IMU Calibration:** $C_g \leftarrow C_g + \tilde{C}_g$ and $C_a \leftarrow \tilde{C}_a$
- **(Optional) Temporal Calibration:** $R_{sb}R(\tilde{w}_{sb}) \leftarrow R_{sb}R(\tilde{w}_{sb})R(\delta_{rot})$ and $T_{sb} + \tilde{T}_{sb} \leftarrow T_{sb} + \tilde{T}_{sb} + \delta_T$. If using online IMU calibration, then

$$\begin{aligned} \delta_t &= V_{sb}^b(t_d + \tilde{t}_d) \\ \delta_{rot} &= \bar{\omega}\tilde{t}_d + (\tilde{C}_g\omega - \tilde{b}_g)(t_d + \tilde{t}_d). \end{aligned} \tag{20}$$

Otherwise, the perturbation is

$$\begin{aligned} \delta_t &= V_{sb}^b(t_d + \tilde{t}_d) \\ \delta_{rot} &= \bar{\omega}\tilde{t}_d \end{aligned} \tag{21}$$

- **(Optional) Camera Calibration:** all additive perturbations
- **Group States:** $w_{sb_r} \leftarrow \tilde{w}_{sb_r}$, $R_{sb_r} \leftarrow R_{sb_r} R(\tilde{w}_{sb_r})$, $T_{sb_r} \leftarrow \tilde{T}_{sb_r}$
- **Feature States:** $x_{c_r} \leftarrow x_{c_r} + \tilde{x}_{c_r}$
- **IMU Noise:** $\omega \leftarrow \omega + n_{imu}$

2.7 Covariance Matrix Prediction

The dynamics of the covariance matrix are given by the Lyapunov equation

$$\dot{P} = FP + PF^T + GQ_{imu}G^T \tag{22}$$

where $F = \frac{\partial \dot{\chi}}{\partial e}$ is the Jacobian of (12) with respect to the error state e and $G = \frac{\partial \dot{\chi}}{\partial n_{imu}}$ is the Jacobian of (12) with respect to modeled IMU noise n_{imu}.

2.8 Augmenting the State and Covariance Matrix with new Features and Groups

The number of map states (tracked features and reference groups) within the EKF state at any given time is variable, but XIVO's implementation does not vary the size of the state χ, error state e, or covariance matrix $\hat{P}$. All three are allocated enough space for the maximum number of features and groups at initialization to limit the number of system calls that XIVO will make. The maximum number of tracked features and reference groups within the EKF state are compile-time parameters.

Each tracked feature and reference group is assigned a "slot" in the error vector e and covariance P. Unused slots have zero error, zero covariance, and zero correlation with any other part of the EKF state. Therefore, they will not affect any other part of the state during EKF measurement update, but will just make the measurement update more expensive. Removing a feature or group from the state is achieved by zeroing out the relevant rows and columns of the covariance matrix and error vector.

When adding a new reference group to the EKF state, the group is initialized with the current value of R_{sb} and T_{sb}. Its 6×6 covariance block is also initialized with the current covariance of R_{sb} and T_{sb}. When adding a new feature to the EKF state, its value and covariance are initialized with the values and 3×3 covariance from the subfilter performing its depth initialization (see 5). Neither new features nor groups are initialized with any cross-correlation with other states.

3 Mapping: Management of Features and Groups

3.1 Killing Two Birds with One Stone: A Note on Observability and Feature Position Uncertainty.

Detected features, indexed i, are 3D points in space. The map could, theoretically, be represented as a collection of points $X_s^i \in \mathbb{R}^3$ rather than in the groups noted above. The representation we use, however, simultaneously addresses two problems:

1. Global gauge, the initial value of g_{sb}, is unobservable and must be fixed – fixing it requires removing six degrees of freedom from the state while ensuring that values of g_{sb} may be updated. An Extended Kalman Filter does not naturally provide a means of enforcing an initial condition; it only provides a means of fixing a portion of the state of a particular value.

2. The uncertainty of a feature's position is not well-represented by a ball or spheroid in $\mathbb{R}^3$. Because features, 3D points in space, are detected and tracked through 2D images, when their position is represented in the camera frame c, it is much easier to estimate their direction (X and Y values), than their depth (Z value). This observation is only true when its position is represented in a c frame where it is visible, such as frame c_r, or the frame at which each feature was first detected; if its location were estimated in the s frame the uncertainty would be distributed in all three cardinal directions.

The location of the camera at which a feature was first detected, however, is something that is not exactly known. Since past c frames are not exactly known, estimating the location of a feature requires estimating the location of the past c frame as well. This means that the location of a feature i is represented by the following quantities:

- $g_{sb_r} \in \mathrm{SE}(3)$ - the location of the body coordinate frame when the feature was first detected
- $g_{bc} \in \mathrm{SE}(3)$ - the camera-IMU extrinsic calibration, which is already estimated as part of the state
- $X^i_{c_r} \in \mathbb{R}^3$ - the location of the feature in the camera frame when it was first detected

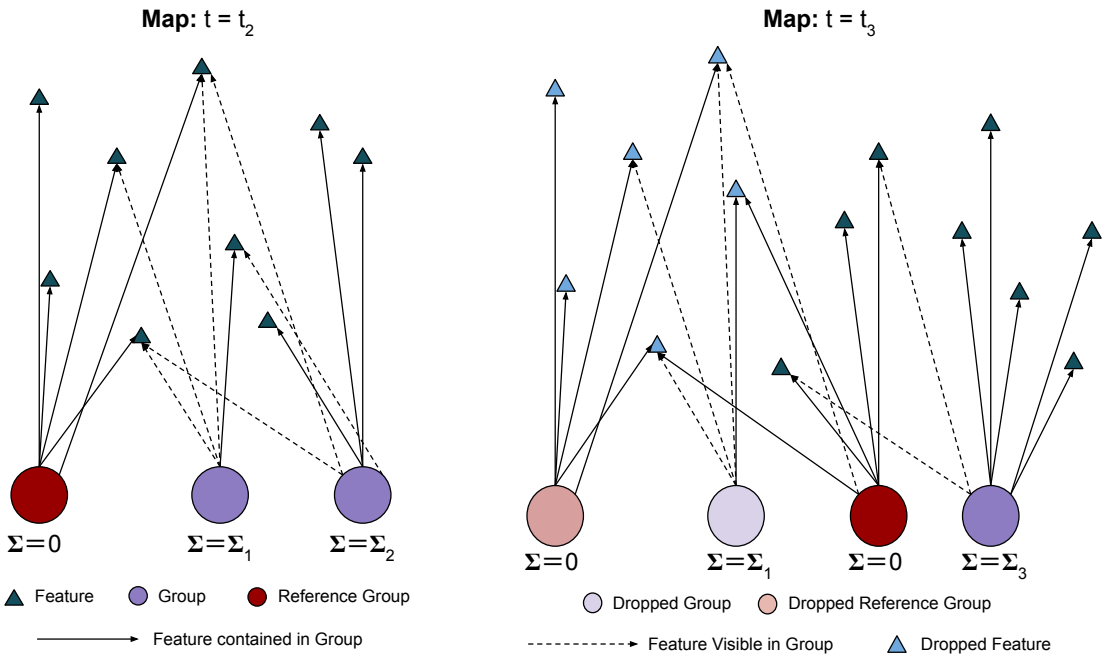
This representation requires 9 additional parameters per feature (g_{sb_r}, X_{c_r}) instead of 3, which is both computationally inefficient and an overparameterization. Since a feature is a point in 3 dimensions, there are an infinite number of ways to make 9 parameters satisfy the observed feature locations during state estimation. This computational inefficiency and overparameterization can be solved by having multiple features, first detected in the same frame, share the 6 parameters in g_{sb_r}. Then, the number of states used to represent N features is $6 + 3N$ instead of $9N$. As the total number of degrees of freedom is $3N$, this is still an overparameterization. However, if the X and Y values of three non-collinear $X^i_{c_r}$ that use the same g_{sb_r} are fixed, then the total number of degrees of freedom in the group is the correct value of $3N$. The three features that are held fixed in each group are called *gauge features*. Groups must therefore "own" at least three features.

With this representation of the map, the enforcement of a global gauge transformation may be accomplished by holding one of the g_{sb_r} fixed. At initialization, XIVO will always fix the g_{sb_r} corresponding to its initial position because it enforces that the initial condition $g_{sb}(0) = (I, 0)$ and because there will be many possible features to choose as the gauge feature. This is pictured in the left portion of Figure 3.

3.2 Feature and Group Management

Since the state of the EKF vector has size $O(N)$ and the measurement update step has time complexity $O(N^3)$ and still must run in real-time, it is impractical to keep all visible features in the state. On the other hand, the complexity of XIVO's feature tracker is only $O(N)$ for the Lucas-Kanade Tracker and $O(N^2)$ for the Correspondence Tracker. Furthermore, observing 3D features in 2D means that a depth value must be estimated (or guessed) before it is added to the EKF state vector; more details on feature depth initialization are given in Section 5. Enforcing observability also means that groups within the EKF state must contain at least three features. These facts lead to the following design choices:

- The number of features, and groups of features, in the EKF state is limited to the compile-time parameters `EKF_MAX_FEATURES` and `EKF_MAX_GROUPS`.
- The number of features in the feature tracker is limited to `tracker.max_features`, a run-time parameter that is larger than `EKF_MAX_FEATURES`.
- Features tracked by the feature tracker, but are not part of the EKF state, are in a depth initialization mode, in which a $O(1)$ complexity filter estimates its depth. Features that have been in a depth initialization mode for at least `subfilter.ready_steps`, a run-time parameter, are deemed `READY`.
- When features are added to the EKF state, they are added either to an existing group already in the EKF state, or in groups of at least three features. When a group is added, three gauge features are chosen. When adding groups to the EKF state, the selection process will add the group(s) with the most number of `READY` features.
- Individual features are removed from the EKF state when they fall out-of-view or when they are flagged by outlier rejection.
- When a gauge feature is removed from the EKF state and its group has more than three features remaining, a new gauge feature is chosen from the remaining features.



(a) Initial Map

(b) After Reference Group Update

Figure 3: **Map.** XIVO's map can be visualized as a graph with two types of nodes (Features and Groups) and two types of edges (Ownership and Visibility). A feature f_i is *owned* by a group $g_{sb_r} \in \mathrm{SE}(3)$, when its estimated position in the spatial frame X_s^i is calculated using the parameters of group g_{sb_r}. A feature f_i may also be *visible* in other groups, or past values of R_{sb} and T_{sb} in the map. Although features are initially owned by the group where it is first detected, its state may be parameterized by any group in which it is visible. In order to enforce a global gauge, the covariance of a single group containing at least three features must be fixed at all times – the first reference group is always the initial position (left figure). A group is *dropped* when fewer than three features remain visible. When a reference group is dropped, a new group is chosen as the reference and its covariance is fixed (right figure).

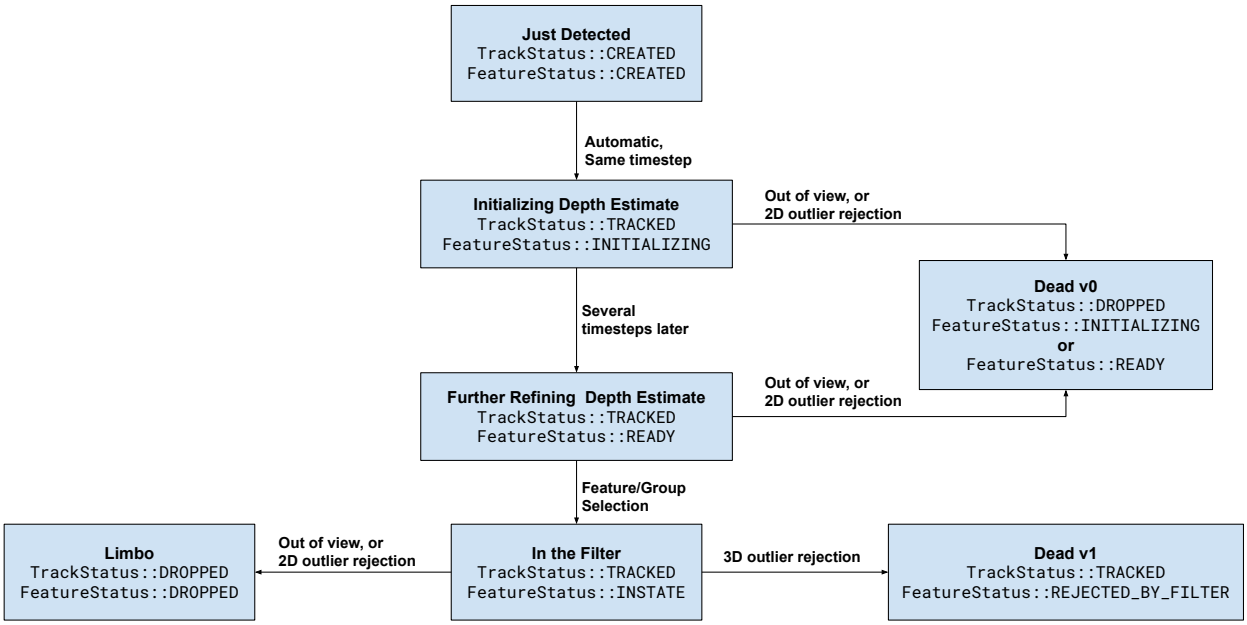


Figure 4: **Life of a XIVO Feature.** A feature has two state variables, one maintained by the feature tracker, `TrackStatus`, the other maintained by the EKF, `FeatureStatus`. Details about transitions are described in the main text of Section 3.2.

- When a group has fewer than three features remaining, the entire group is removed from the EKF state. XIVO will attempt to transfer the features to other groups within the EKF state by recalculating x_{c_r} for a separate group.

The possible lifecycles a feature may have is captured by in Figure 4. The logic for adding features into the EKF state is given in Figure 5.

4 Jacobians

The order of elements in the state vector χ are:

$$\chi = \begin{bmatrix} \chi_{\text{IMU}} & \chi_{\text{calib}} & \chi_{\text{opt}} & \chi_{\text{group}} & \chi_{\text{features}} \end{bmatrix} \tag{23}$$

where

$$\chi_{\text{IMU}} = \begin{bmatrix} w_{sb} & T_{sb} & V_{sb} & b_g^b & b_a^b \end{bmatrix} \tag{24}$$

$$\chi_{\text{calib}} = \begin{bmatrix} w_{bc} & T_{bc} & w_{sg} \end{bmatrix} \tag{25}$$

$$\chi_{\text{opt}} = \begin{bmatrix} t_d & C_g & C_a & \theta \end{bmatrix} \tag{26}$$

$$\chi_{\text{group}} = \begin{bmatrix} w_{sb_r,1} & T_{sb_r,1} & w_{sb_r,2} & T_{sb_r,2} & \dots & w_{sb_r,n_g} & T_{sb_r,n_g} \end{bmatrix} \tag{27}$$

$$\chi_{\text{features}} = \begin{bmatrix} x_{c_r}^1 & x_{c_r}^2 & \dots & x_{c_r}^{n_f} \end{bmatrix} \tag{28}$$

n_g is the number of feature groups (see Section 3.1) and n_f is the number of features. χ_{group} will have size $6n_g$ and χ_{features} will have size $3n_f$. Let n_{opt} be the dimension of χ_{opt}. The value of n_{opt} will depend on the number of optional features used. The dimension of χ is $n_\chi = 23 + 6n_g + 3n_f + n_{\text{opt}}$.

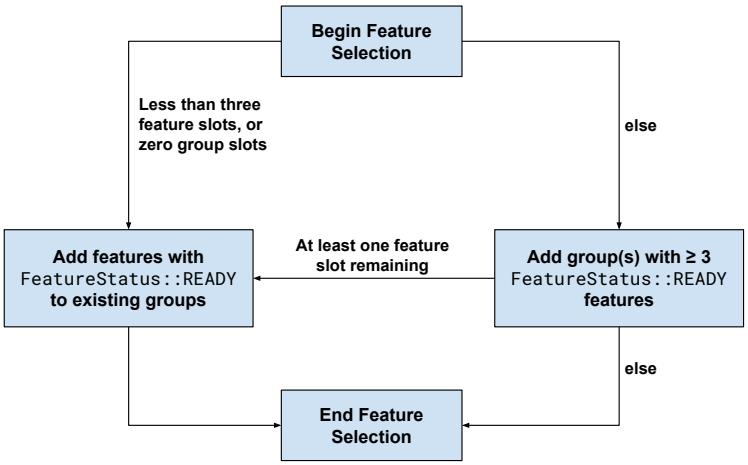


Figure 5: **Adding Features into the EKF.** At each timestep, XIVO will attempt to add features into the EKF state. It will always try to add at least one new group before adding features to existing groups.

4.1 Our approach to deriving approximate Jacobians.

For "block-Jacobians" (Jacobians of vectors with respect to vectors), we incorporate the error state and noise into nominal equations of motion and measurement update equations as in Section 2.6. Then we algebraically isolate the error state variables. The final results are documented in this section.

(Optional) Online Temporal and IMU Calibration If using online temporal and IMU calibration, we need additional Jacobians with respect to error states $\tilde{t}_d$, $\tilde{C}_g$, and $\tilde{C}_a$.

If we are tracking with online temporal calibration and online IMU calibration, then R_{sb} is instead[2]

$$R_{sb} = \bar{R}_{sb} R(\tilde{w}_{sb}) R(\delta_{rot}) \tag{29}$$

where

$$\delta_{rot} = (\bar{C}_g \omega_m - \bar{b}_g)\tilde{t}_d + (\tilde{C}_g \omega_m - \tilde{b}_g)(\bar{t}_d + \tilde{t}_d) \tag{30}$$

4.2 The Matrix F (eqs. (8), (22))

The matrix $F \in \mathbb{R}^{n_\chi \times n_\chi}$ has form:

$$\begin{bmatrix} \frac{\partial \dot{\chi}}{\partial \chi_{\text{IMU}}} & \frac{\partial \dot{\chi}}{\partial \chi_{\text{calib}}} & \frac{\partial \dot{\chi}}{\partial \chi_{\text{opt}}} & \mathbf{0}_{n_\chi \times 6n_g} & \mathbf{0}_{n_\chi \times 3n_f} \end{bmatrix} \tag{31}$$

[2]Without online IMU calibration, $\bar{C}_g = I$ and $\tilde{C}_g = 0$.

Values of individual matrices are:

$$\frac{\partial \dot{\chi}}{\partial \chi_{\text{IMU}}} = \begin{bmatrix} -[\bar{\omega}^b_{sb}]_\times & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -R_{sb}[\bar{a}^b_{sb}]_\times & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -R_{sb} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{32}$$

$$\frac{\partial \dot{\chi}}{\partial \chi_{\text{calib}}} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & (-R_{sb}[g]_\times)_{\text{first two cols}} \\ \cdots & \cdots & \cdots \end{bmatrix} \tag{33}$$

$$\frac{\partial \dot{\chi}}{\partial \chi_{\text{opt}}} = \begin{bmatrix} \mathbf{0}_{3\times1} & \frac{\partial \dot{R}_{sb}}{\partial C_g} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times n_\theta} \\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times9} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times n_\theta} \\ \mathbf{0}_{3\times1} & \mathbf{0}_{3\times9} & \frac{\partial \dot{V}_{sb}}{\partial C_a} & \mathbf{0}_{3\times n_\theta} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{34}$$

where

$$\frac{\partial \dot{R}_{sb}}{\partial C_g} = \begin{bmatrix} (\omega^b_{sb})^T & \mathbf{0}_{1\times3} & \mathbf{0}1 \times 3 \\ \mathbf{0}_{1\times3} & (\omega^b_{sb})^T & \mathbf{0}_{1\times3} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & (\omega^b_{sb})^T \end{bmatrix} \in \mathbb{R}^{3\times9} \tag{35}$$

$$\frac{\partial \dot{V}_{sb}}{\partial C_a} = \left(\frac{\partial AB}{\partial A}_{3\times3}(a^b_{sb})\right)\left(\frac{\partial AB}{\partial B}_{3\times3}(R_{sb})\right)\left(\frac{\partial A}{\partial A^u}\right) \in \mathbb{R}^{3\times6} \tag{36}$$

4.3 The Matrix G (eqs. (8), (22))

Let $u = \begin{bmatrix} n_g & n_a & n_{b_g} & n_{b_a} \end{bmatrix}^T$ be the input vector. Then, the matrix G has dimension $n_{imu} \times n_\chi$ and has form:

$$G = \begin{bmatrix} -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & R_{sb} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{37}$$

All rows of G not explicitly noted are completely zero.

4.4 The Matrix H (eq. (9))

The dimension of H is $2n_f \times n_\chi$, consisting of n_f blocks with dimension $2 \times n_\chi$. Each block is the Jacobian of predicted pixel coordinates x_p of a feature j with respect to the error states $\tilde{\chi}$ at $\tilde{\chi} = 0$. Assume that feature j is owned by group i. Then, each two-row block is given by:

$$\frac{\partial x_p}{\partial \tilde{\chi}}|_{\tilde{\chi}=0} = \frac{\partial \pi^c(x_c)}{\partial \theta} + \frac{\partial \pi^c(x_c)}{\partial x_c} \cdot \frac{\partial \pi(X_c)}{\partial X_c} \cdot \frac{\partial X_c}{\partial \tilde{\chi}}|_{\tilde{\chi}=0}. \tag{38}$$

The quantity $\frac{\partial \pi^c(x_c)}{\partial \theta}$ is equal to zero if online camera calibration is not enabled.

- $\frac{\partial \pi^c(x_c)}{\partial x_c} \in \mathbb{R}^{2\times3}$ depends on the camera model used. For the pinhole projection model,

$$\frac{\partial \pi^c(x_c)}{\partial x_c} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \end{bmatrix}. \tag{39}$$

Expressions for the other camera models are more complex. They were automatically differentiated and autocoded from equations given in Section 2.5.1 using the MATLAB Symbolic Toolbox.

- If using log-z projection,

$$\frac{\partial \pi(X_c)}{\partial X_c} = \begin{bmatrix} 1/Z & 0 & -X/Z^2 \\ 0 & 1/Z & -Y/Z^2 \\ 0 & 0 & 1/Z \end{bmatrix} \tag{40}$$

  for $X_c = [X, Y, Z]^T$.

- If using inverse-z projection,

$$\frac{\partial \pi(X_c)}{\partial X_c} = \begin{bmatrix} 1/Z & 0 & -X/Z^2 \\ 0 & 1/Z & -Y/Z^2 \\ 0 & 0 & -1/Z^2 \end{bmatrix} \tag{41}$$

- $\frac{\partial X_c}{\partial \tilde{\chi}}$ is a matrix with dimension $3 \times n_\chi$ with form (the three "rows" below are really one row):

$$\frac{\partial X_c}{\partial \tilde{\chi}} = \begin{bmatrix} \frac{\partial X_c}{\partial \tilde{w}_{sb}} & \frac{\partial X_c}{\partial \tilde{T}_{sb}} & \mathbf{0}_{3\times 3} & \frac{\partial X_c}{\partial b_g} & \mathbf{0}_{3\times 3} & \frac{\partial X_c}{\partial \tilde{w}_{bc}} & \frac{\partial X_c}{\partial \tilde{T}_{bc}} \\ \mathbf{0}_{2\times n_\chi} & \frac{\partial X_c}{\partial \tilde{t}_d} & \frac{\partial X_c}{\partial \tilde{C}_g} & \mathbf{0}_{6\times n_\chi} & \mathbf{0}_{n_\theta \times n_\chi} & \mathbf{0}_{6(i-1)\times n_\chi} & \frac{\partial X_c}{\partial \tilde{w}_{sb_r}} \\ \frac{\partial X_c}{\partial \tilde{T}_{sb_r}} & \mathbf{0}_{6(n_g-i)\times n_\chi} & \mathbf{0}_{3(j-1)\times n_\chi} & \frac{\partial X_c}{\partial x_{c_r}} & \mathbf{0}_{3(n_f-j)\times n_\chi} & & \end{bmatrix} \tag{42}$$

(Approximate) expressions within $\frac{\partial X_c}{\partial \tilde{\chi}}$ that are always present are given by:

$$\frac{\partial X_c}{\partial \tilde{T}_{sb_r}} = -R_{bc}^T R_{sb}^T \tag{43}$$

$$\frac{\partial X_c}{\partial \tilde{T}_{bc}} = -R_{bc}^T + R_{bc}^T R_{sb}^T R_{sb_r} \tag{44}$$

$$\frac{\partial X_c}{\partial \tilde{T}_{sb_r}} = R_{bc}^T R_{sb}^T \tag{45}$$

$$\frac{\partial X_c}{\partial \tilde{w}_{sb}} = R_{bc}^T [R_{sb}^T (X_s - T_{sb})]_\times \tag{46}$$

$$\frac{\partial X_c}{\partial \tilde{w}_{sb_r}} = -R_{bc}^T R_{sb}^T R_{sb_r} [R_{bc} X_c + T_{bc}]_\times \tag{47}$$

$$\frac{\partial X_c}{\partial \tilde{w}_{bc}} = [X_c]_\times - R_{bc}^T R_{sb}^T R_{sb_r} R_{bc} [X_{c_r}]_\times \tag{48}$$

$$\frac{\partial X_c}{\partial x_{c_r}} = R_{bc}^T R_{sb_r} R_{bc} \frac{\partial X_{c_r}}{\partial x_{c_r}} \tag{49}$$

If using the log-z projection, $\frac{\partial X_{c_r}}{\partial x_{c_r}} \in \mathbb{R}^{3\times 3}$ is given by

$$\frac{\partial X_{c_r}}{\partial x_{c_r}} = \begin{bmatrix} \exp(Z) & 0 & X\exp(Z) \\ 0 & \exp(Z) & Y\exp(Z) \\ 0 & 0 & \exp(Z) \end{bmatrix} \tag{50}$$

where $x_{c_r} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$. If using the inverse-z projection, $\frac{\partial X_{c_r}}{\partial x_{c_r}} \in \mathbb{R}^{3\times 3}$ is instead.

$$\frac{\partial X_{c_r}}{\partial x_{c_r}} = \begin{bmatrix} 1/Z & 0 & X/Z \\ 0 & 1/Z & Y/Z \\ 0 & 0 & -1/Z^2 \end{bmatrix}. \tag{51}$$

(Optional) Online Camera Calibration. If online camera calibration is enabled, the quantity $\frac{\partial \pi^c(x_c)}{\partial \theta} \in \mathbb{R}^{2 \times n_\theta}$ is nonzero. (The matrix is a sub-block of $\frac{\partial x_p}{\partial \tilde{\chi}}$.) For the pinhole camera model with $\theta = [f_x, f_y, c_x, c_y]^T$, this quantity is:

$$\frac{\partial \pi^c(x_c)}{\partial \theta} = \begin{bmatrix} X & 0 & 1 & 0 \\ 0 & Y & 0 & 1 \end{bmatrix} \tag{52}$$

where $x_c = [X, Y, Z]^T$.

For other camera models, the expressions are complex and were autocoded from the equations in Section 2.5.1 using MATLAB Symbolic Toolbox.

(Optional) Online Temporal Calibration If online temporal calibration is enabled, then the following Jacobians are nonzero:

$$\frac{\partial X_c}{\partial t_d} = -R_{bc}^T [\bar{\omega}_{sb}^b]_\times X_b + R_{sb}^T V_{sb} \tag{53}$$

$$\frac{\partial X_c}{\partial b_g} = -\frac{\partial AB}{\partial B}_{3,1} (R_{bc}^T [X_b]_\times t_d) \tag{54}$$

(Optional) Online IMU Calibration with Online Temporal Calibration. If online IMU calibration and online temporal calibration are both enabled, then the following Jacobian is nonzero and takes the form:

$$\frac{\partial X_c}{\partial C_g} = \frac{\partial AB}{\partial B}_{3,1} (R_{bc}^T [X_b]_\times t_d) \cdot \frac{\partial C_g \omega_{sb}^b}{\partial \omega_{sb}^b} \tag{55}$$

where

$$\frac{\partial C_g \omega_{sb}^b}{\partial \omega_{sb}^b} = \begin{bmatrix} (\omega_{sb}^b)^T & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & (\omega_{sb}^b)^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & (\omega_{sb}^b)^T \end{bmatrix} \in \mathbb{R}^{3\times9} \tag{56}$$

(Optional) Online IMU Calibration with no Online Temporal Calibration. If online IMU calibration, but not online temporal calibration, is enabled, then the measurement equation is not dependent on C_a or C_g.

5 Feature Depth Initialization

In Section 3.1, it was noted that since features are 3D points observed in only two dimensions, that when the position of a feature is estimated in the camera frame at the time it was first detected, there is a lot more uncertainty in its depth than the X and Y directions. In the camera frame at the time it was detected, the X and Y values can be observed instantly, but the depth can only be observed, and initialized, over time.

Before a feature can be added into the main EKF state, its depth value must be initialized to some value. An inaccurate initialized value will adversely affect the EKF's estimate of the current state g_{sb}. This section describes the algorithms used to initialize a depth estimate. A flowchart of all possibilities is pictured in Figure 6.

5.1 Subfilter Equations

The main component used to estimate feature depth is a subfilter, an incomplete Extended Kalman Filter. Unlike the main state vector, which contains multiple features, each feature has its own subfilter; subfiltering N features therefore has a computational complexity of $O(N)$.

The state of the subfilter is $x_{c_r} = [X/Z, Y/Z, \log(Z)]^T$ with constant dynamics $\dot{x}_{c_r} = 0$. The measurement equation is:

$$x_p = \pi^c \left(\pi \left(g_{bc}^{-1}(t) \circ g_{sb}^{-1}(t) \circ g_{sb_r} \circ g_{bc}(t) \circ \pi^{-1}(x_{c_r}) \right) \right) \tag{57}$$

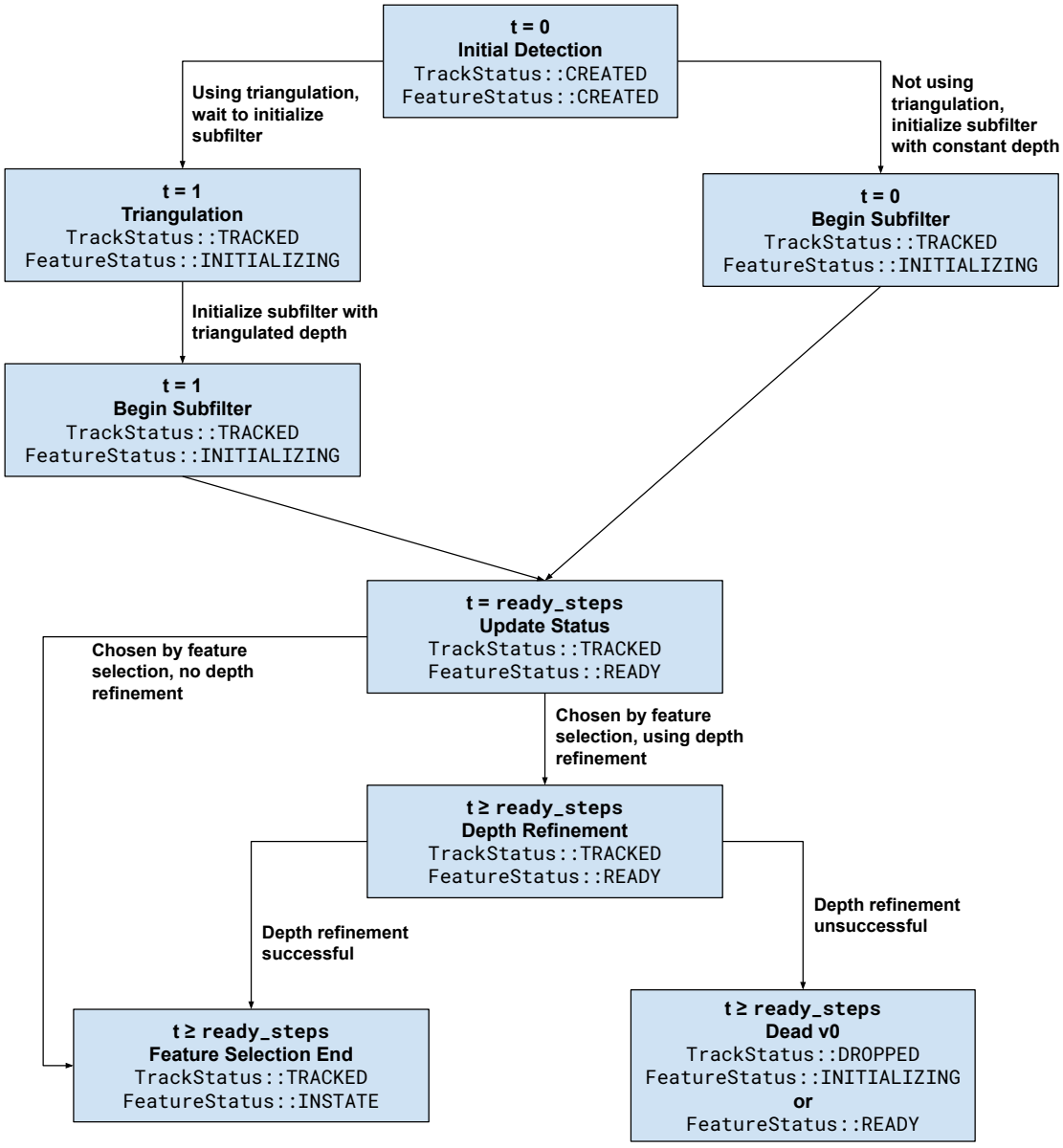


Figure 6: **XIVO's Feature Depth Initialization Process.** This flowchart illustrates the states of a feature during the depth initialization process. `TrackStatus` and `FeatureStatus` are the same as those used in Figure 4. The main variables affecting the process are whether or not two-view triangulation is performed, or whether or not depth refinement is performed.

where g_{bc} and g_{sb} are read from the main EKF state at each timestep and used as constants in the subfilter update. If other features sharing the same group parameters g_{sb_r} are currently in the main EKF state, then g_{sb_r} is also read from the EKF state. Otherwise, g_{sb_r} is a constant, initialized to the value of g_{sb} at time the feature was first detected. Equation (57) is identical to the main EKF measurement update equation, except that it only estimates the values of x_{c_r}. To compensate for the fact that g_{sb}, g_{sb_r}, and g_{bc} will change every timestep, the covariance values of feature tracks used for the measurement update of the subfilter should be larger than the covariance values of feature tracks used in the main EKF filter.

The initial state of the subfilter is $x_{c_r,0}$. The X and Y coordinates of $x_{c_r,0}$ can be computed by inverting the camera intrinsics $\pi^c(\cdot)$. The Z coordinate is initialized with a constant run-time parameter `z_init`. The initial covariance of the subfilter is a diagonal matrix. The parameters of the diagonal matrix are user-defined run-time parameters.

Before a feature is added to the EKF state, the subfilter must run for `ready_steps`, a user-defined number of parameters. Afterward, it becomes a candidate for addition to the state.

5.2 Two-View Triangulation (Optional)

Rather than initializing the subfilter of all features with a constant runtime parameter `z_init`, we may calculate a separate value of `z_init` for each feature using two-view triangulation. The two views are the first two frames in which the feature is visible. Two-view triangulation uses the estimates of g_{sb} at the latest two timesteps and the latest estimate of g_{bc} as if they were correct.

XIVO contains five implementations of two-view triangulation:

- Essential Matrix [7]
- Direct Linear Transformation [10]
- Minimization of L1, L2, and L-∞ angular reprojection error [6]

Since the first two views of a feature are used in triangulation, triangulation will often fail in practice due to a lack of parallax. When triangulation fails, XIVO will use the user-defined constant `z_init` instead of the triangulation output.

5.3 Depth Refinement (Optional)

Depth refinement occurs right before a feature is added to the EKF state. This may be after more timesteps than `ready_steps`. The depth refinement will directly edit the state of the subfilter, but not the covariance.

Depth refinement partially solves the following optimization problem:

$$\underset{x_{c_r}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{N_{obs}}(x_p(i) - x_p^{obs}(i))^T R^{-1}(x_p(i) - x_p^{obs}(i)) \tag{58}$$

where R is the measurement covariance (a runtime parameter), $x_p(i)$ is the predicted measurement at timestep i computed from x_{c_r} and $x_p^{obs}(i)$ is the actual feature measurement at timestep i. The partial solving is done using a fixed number (a runtime parameter) of Newton steps.

If the total L2 error

$$\sum_{i=1}^{N_{obs}}(x_p(i) - x_p^{obs}(i)) \tag{59}$$

exceeds the threshold `max_res_norm`, a runtime parameter, then depth refinement *failed*. Otherwise, depth refinement is *successful*.

6 Loop Closure (Optional)

Loop closure is an optional measurement update using features with state `Limbo` (see Fig. 4). If loop closure is enabled, XIVO's mapper module (enabled with the compile-time option `USE_MAPPER`) will look for possible loop closures after each measurement update. The loop closure process is:

1. Search for loop closures by matching descriptors of features in the current EKF state to features in `Limbo`. Descriptor matching is performed using the DBoW2 bag-of-words library [3] and a nearest neighbor search. For each matched feature, retrieve the *old* value of X_s from the map.

2. Use Perspective-and-Point RANSAC to remove outliers from the list of matches. We use the Lambda Twist [9] solver. Since Perspective-and-Point RANSAC requires at least four matches, we consider all points outliers if there are fewer than four matches.

3. For the old values of X_s that remain, perform an EKF measurement update with the following measurement equation:
$$x_p = \pi^c(\pi(R_{bc}^T(R_{sb}^T X_s - T_{sb}) - T_{bc})) \tag{60}$$
R_{bc}, T_{bc}, R_{sb}, and T_{sb} are read from the current state vector. X_s is treated like a known constant. The measurement covariance associated with equation (60) is a runtime parameter.

References

[1] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for EKF-based Structure from Motion. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3498–3504, October 2009.

[2] Guillermo Gallego and Anthony Yezzi. A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates. *Journal of Mathematical Imaging and Vision*, 51(3):378–384, March 2015.

[3] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[4] J. Hernandez, K. Tsotsos, and S. Soatto. Observability, identifiability and sensitivity of vision-aided inertial navigation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2319–2325, May 2015.

[5] E. Jones and S. Soatto. Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430, April 2011.

[6] Seong Hun Lee and Javier Civera. Closed-Form Optimal Two-View Triangulation Based on Angular Errors. pages 2681–2689, 2019.

[7] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, September 1981. Number: 5828 Publisher: Nature Publishing Group.

[8] Richard M. Murray and Zexiang Li. *A Mathematical Introduction to Robotic Manipulation*. Routledge, Boca Raton, 1 edition edition, March 1994.

[9] Mikael Persson and Klas Nordberg. Lambda Twist: An Accurate Fast Robust Perspective Three Point (P3P) Solver. pages 318–332, 2018.

[10] I.E. Sutherland. Three-dimensional data input by tablet. *Proceedings of the IEEE*, 62(4):453–461, April 1974. Conference Name: Proceedings of the IEEE.

[11] David Tedaldi, Alberto Pretto, and Emanuele Menegatti. A robust and easy to implement method for IMU calibration without external equipments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3042–3049, May 2014. ISSN: 1050-4729.