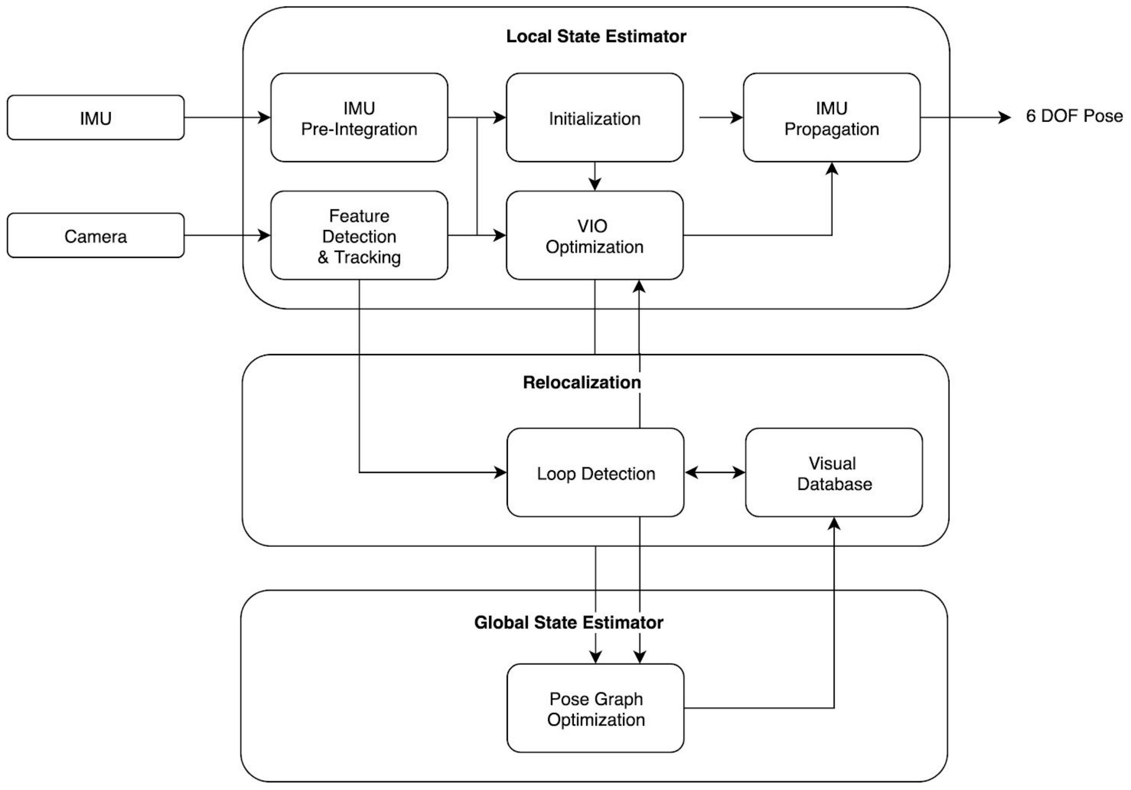
VINS Mono

Body-Of-Knowledge

Holger Lange @ www.aiwerkstatt.com

The goal of this document is to describe the **Body-Of-Knowledge** behind **VINS-Mono**, a state-of-the-art Monocular Visual-INertial System, developed by Tong Qin, Peiliang Li, and Shaojie Shen.

T. Qin, P. Li, and S. Shen; **A robust and versatile monocular visual-inertial state estimator**; IEEE Trans. Robot.; 2018



State Estimation and Localization

Localization provides **metric six degrees-of-freedom (DOF) state estimation**.

Local State Estimators suffer from long term drifting. To eliminate long-term drift a complete system that includes **local state estimation**, **global state estimation** and **relocalization** (incl. loop closures) is needed.

Local State Estimator

Monocular Visual-Inertial Odometry (VIO)

An **Inertial Measurement Unit (IMU)** and a **monocular camera** provide the **minimum sensor suite** for metric six degrees-of-freedom (DOF) state estimation. However, the **lack of direct distance** measurement poses significant challenges in terms of IMU processing, estimator initialization, extrinsic calibration, and nonlinear optimization.

Inertial Measurement Units (IMU)

Inertial Measurement Units (IMU) include **accelerometers** that measure **linear acceleration** and **gyroscopes** that measure **angular velocity**.

Measuring the local gravity vector and the linear acceleration provide information about the change in position and about the orientation. The IMU provides **observable metric scale** and **roll** and **pitch** angles.

The complexity of estimating the position and orientation using an IMU lies in the parametrization of the orientation and in the nonlinear nature of the orientation.

Dynamic Model

IMU measurements combine the force for countering **gravity** g and the **platform dynamics**, and are affected by **bias** b and additive (assumed) Gaussian $N(0, \Sigma)$ white **noise** n. If the sensors are properly calibrated, the measurements in the different axes are independent and the Gaussian noise covariance matrices Σ can be assumed diagonal. Bias are modeled as random walks, whose derivatives are Gaussian.

The world coordinate system is assumed to be stationary. The magnitude of the earth rotation is small compared to the actual angular velocity measurements and therefore can be ignored.

Linear acceleration

$$\hat{a}_t = a_t + R_t^W g^W + b_{a_t} + n_a$$
$$\dot{b_{a_t}} = n_{b_{a_t}} \sim N(0, \sigma^2_{b_{a_t}})$$
$$n_a \sim N(0, \sigma_a^2)$$

Hamilton quaternions are used to represent **rotations**
$R = R(q)$ rotation matrix from quaternion

Angular velocity

$$\hat{w}_t = w_t + b_{w_t} + n_w$$
$$\dot{b_{w_t}} = n_{b_{w_t}} \sim N(0, \sigma^2_{b_{w_t}})$$
$$n_w \sim N(0, \sigma_w^2)$$

The centrifugal acceleration is considered to be absorbed in the local gravity vector. The magnitude of the Coriolis acceleration is small compared to the linear acceleration measurements and therefore can be ignored.

Ordinary Differential Equations

Relationship between position, velocity and linear acceleration
$$\dot{p} = v$$
$$\dot{v} = a$$
Relationship between orientation (unit quaternion) and angular velocity
$$\dot{q} = \frac{1}{2} q \otimes w = \frac{1}{2}[w]_R q = \frac{1}{2}\Omega(w)q$$
$$\Omega(w) = [w]_R = \begin{bmatrix} 0 & -w_x & -w_y & -w_z \\ w_x & 0 & w_z & -w_y \\ w_y & -w_z & 0 & w_x \\ w_z & w_y & -w_x & 0 \end{bmatrix}$$

Integration

Integration of the angular velocity provides the **orientation**. After subtracting gravity using the orientation, double integration of the acceleration provides the **position**.

The integration of the IMU measurements is done between two consecutive frames.

$$p^W_{B_{k+1}} = p^W_{B_k} + v^W_{B_k}\triangle t_k + \int\int_{t\in[t_k,t_{k+1}]}(R^W_t(\hat{a}_t - b_{a_t} - n_a) - g^W)dt^2$$
$$v^W_{B_{k+1}} = v^W_{B_k} + \int_{t\in[t_k,t_{k+1}]}(R^W_t(\hat{a}_t - b_{a_t} - n_a) - g^W)dt$$
$$q^W_{B_{k+1}} = q^W_{B_k} \otimes \int_{t\in[t_k,t_{k+1}]}\frac{1}{2}\Omega(\hat{w}_t - b_{w_t} - n_w)q^{B_k}_t dt$$

$\hat{(.)}$ measured
$\triangle t = t_{k+1} - t_k$

Position, **velocity** and **orientation** states in the **world frame** can be **propagated by IMU measurements**, but when the starting states change (due to adjusted poses), they need to be re-propagated.

IMU Pre-Integration

Pre-integration of the parts that are related to **linear acceleration** and **angular velocity** in the **body frame** avoids computational expensive re-propagation.

The pre-integration is based on an **on-manifold rotation** formulation that derives the **covariance** propagation using **continuous time error state dynamics** and adding a **posterior bias correction**.

$$R^{B_k}_W p^W_{B_{k+1}} = R^{B_k}_W(p^W_{B_k} + v^W_{B_k}\triangle t_k - \frac{1}{2}g^W\triangle t^2_k) + \alpha^{B_k+1}_{B_k}$$
$$R^{B_k}_W v^W_{B_{k+1}} = R^{B_k}_W(v^W_{B_k} - g^W\triangle t_k) + \beta^{B_k+1}_{B_k}$$
$$q^{B_{k+1}}_W \otimes q^W_{B_{k+1}} = \gamma^{B_k}_{B_{k+1}}$$
$$\alpha^{B_k+1}_{B_k} = \int\int_{t\in[t_k,t_k+1]}(R^{B_k}_t(\hat{a}_t - b_{a_t} - n_a))dt^2$$
$$\beta^{B_k+1}_{B_k} = \int_{t\in[t_k,t_k+1]} R^{B_k}_t(\hat{a}_t - b_{a_t} - n_a)dt$$
$$\gamma^{B_k}_{B_{k+1}} = \int_{t\in[t_k,t_k+1]}\frac{1}{2}\Omega(\hat{w}_t - b_{w_t} - n_w)\gamma^{B_k}_t dt = \int_{t\in[t_k,t_k+1]}\frac{1}{2}\gamma^{B_k}_t \otimes (\hat{w}_t - b_{w_t} - n_w)dt$$

Additive **noise** n is unknown and **left at 0**.

The discrete-time implementation uses the **midpoint method** for numerical integration.

$\alpha^{B_k}_{B_k} = 0$, $\beta^{B_k}_{B_k} = 0$, $\gamma^{B_k}_{B_k} = q_1$

$$\hat{\gamma}_{i+1}^{B_k} = \hat{\gamma}_i^{B_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\hat{w}_i - b_{w_i} + \hat{w}_{i+1} - b_{w_i})\delta t \end{bmatrix}$$

$$\hat{\beta}_{i+1}^{B_k} = \hat{\beta}_i^{B_k} + \frac{1}{2}[R(\hat{\gamma}_i^{B_k})(\hat{a}_i - b_{a_i}) + R(\hat{\gamma}_{i+1}^{B_k})(\hat{a}_{i+1} - b_{a_i}))]\delta t$$

$$\hat{\alpha}_{i+1}^{B_k} = \hat{\alpha}_i^{B_k} + \hat{\beta}_i^{B_k}\delta t + \frac{1}{4}[R(\hat{\gamma}_i^{B_k})(\hat{a}_i - b_{a_i}) + R(\hat{\gamma}_{i+1}^{B_k})(\hat{a}_{i+1} - b_{a_i})]\delta t^2$$

i and δt relate to IMU measurements.

Error-State Kalman Filter

High-frequency IMU data is integrated into the **nominal-state** that does not take into account the noise terms and other possible model imperfections, thereby accumulating errors. These errors are collected in the **error-state** and the **Error-State Kalman Filter** provides a **posterior Gaussian estimate** of the error-state. It **only** performs the **prediction step** as no measurements (e.g. GPS, vision, etc.) are available. The **error-state's mean** is injected into the **nominal-state** and reset to zero. The **error-state's covariance** is **updated** to reflect the reset.

$\hat{X} = X + \delta X$, $\hat{X}$ true, Xnominal, δX error, $(\cdot)_m$ measured

$$\dot{\hat{X}} = \begin{bmatrix} \dot{\hat{\alpha}} \\ \dot{\hat{\beta}} \\ \dot{\hat{\gamma}} \\ \dot{\hat{b}}_a \\ \dot{\hat{b}}_w \end{bmatrix} = \begin{bmatrix} \hat{\beta} \\ \hat{R}(a_m - \hat{b}_a - n_a) \\ \frac{1}{2}\hat{\gamma} \otimes (w_m - \hat{b}_w - n_w) \\ n_{b_a} \\ n_{b_w} \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \\ \dot{b}_a \\ \dot{b}_w \end{bmatrix} = \begin{bmatrix} \beta \\ R(a_m - b_a) \\ \frac{1}{2}\gamma \otimes (w_m - b_w) \\ 0 \\ 0 \end{bmatrix}$$

Linearizing the **dynamics** of the **error-state** $\delta\dot{X}$ ignoring small second-order terms.

1) Linear velocity related error

Small-signal approximations

$\hat{R} = R\delta R = R(I + [\delta\Theta]_X)$ without second-order term

$\hat{b_a} = b_a + \delta b_a$

$$\dot{\beta} + \dot{\delta\beta} = \dot{\hat{\beta}} = \hat{R}(a_m - \hat{b_a} - n_a)$$

$$R(a_m - b_a) + \dot{\delta\beta} = \dot{\hat{\beta}} = R(I + [\delta\Theta]_X)(a_m - b_a - \delta b_a - n_a)$$

$$\begin{aligned}\dot{\delta\beta} = &-Ra_m + Rb_a\\ &+Ra_m - Rn_a - Rb_a - R\delta b_a\\ &+R[\delta\Theta]_X a_m - R[\delta\Theta]_X n_a - R[\delta\Theta]_X b_a - R[\delta\Theta]_X \delta b_a\end{aligned}$$

$\dot{\delta\beta} = -Rn_a - R\delta b_a - R[a_m]_X\delta\Theta + R[b_a]_X\delta\Theta$ eliminate second-order terms

$$\dot{\delta\beta} = -R[a_m - b_a]_X\delta\Theta - Rn_a - R\delta b_a$$

2) Linear orientation related error

Small-signal approximations

$\hat{\gamma} = \gamma \otimes \delta\gamma$

$\hat{b_w} = b_w + \delta b_w$

$$\gamma \otimes \dot{\delta\gamma} = \dot{\hat{\gamma}} = \frac{1}{2}\hat{\gamma} \otimes (w_m - n_w - \hat{b_w})$$

$$\dot{\gamma} \otimes \delta\gamma + \gamma \otimes \dot{\delta\gamma} = \dot{\hat{\gamma}} = \frac{1}{2}\gamma \otimes \delta\gamma \otimes (w_m - n_w - b_w - \delta b_w)$$

$$\frac{1}{2}\gamma \otimes (w_m - b_w) \otimes \delta\gamma + \gamma \otimes \dot{\delta\gamma} = \frac{1}{2}\gamma \otimes \delta\gamma \otimes (w_m - n_w - b_w - \delta b_w)$$

$$\frac{1}{2}\gamma \otimes w \otimes \delta\gamma + \gamma \otimes \dot{\delta\gamma} = \frac{1}{2}\gamma \otimes \delta\gamma \otimes \hat{w}$$

$$2\dot{\delta\gamma} = \delta\gamma \otimes \hat{w} - w \otimes \delta\gamma = [\hat{w}]_R\delta\gamma - [w]_L\delta\gamma = \begin{bmatrix} 0 & (w - \hat{w})^T \\ \hat{w} - w & -[w + \hat{w}]_X \end{bmatrix}\delta\gamma$$

The **orientation** is assumed to lie on **a manifold** by modeling the orientation and its uncertainty using a spherical distribution, which naturally restricts the orientation estimates and their uncertainties to be in SO(3).

The error term of the overparameterized four-dimensional rotation quaternions are defined as a perturbation around its mean. $\delta\Theta$ is a three-dimensional small perturbation.

$$\delta\gamma = \begin{bmatrix} 1 \\ \delta\Theta \end{bmatrix}, \quad \dot{\delta\gamma} = \begin{bmatrix} 0 \\ \dot{\delta\Theta} \end{bmatrix}$$

$\hat{w} = w_m - n_w - \hat{b_w}$

$w = w_m - b_w$

$$\begin{bmatrix} 0 \\ \dot{\delta\Theta} \end{bmatrix} = \begin{bmatrix} 0 & (\delta b_w + n_w)^T \\ -\delta b_w - n_w & -[2w_m - n_w - 2b_w - \delta b_w]_X \end{bmatrix}\begin{bmatrix} 1 \\ \frac{1}{2}\delta\Theta \end{bmatrix}$$

$$0 = (\delta b_w + n_w)^T \frac{1}{2}\delta\Theta$$

$$\delta\dot{\Theta} = -\delta b_w - n_w - [2w_m - n_w - 2b_w - \delta b_w]_X \frac{1}{2}\delta\Theta$$

$$\delta\dot{\Theta} = -[w_m - b_w]_X \delta\Theta - \delta b_w - n_w$$

$$\delta\dot{X} = \begin{bmatrix} \delta\dot{\alpha} \\ \delta\dot{\beta} \\ \delta\dot{\Theta} \\ \delta\dot{b_a} \\ \delta\dot{b_w} \end{bmatrix} = \begin{bmatrix} \delta\beta \\ -R[a_m - b_a]_X\delta\Theta - Rn_a - R\delta b_a \\ -[w_m - b_w]_X\delta\Theta - \delta b_w - n_w \\ n_{b_a} \\ n_{b_w} \end{bmatrix}$$

$$\begin{bmatrix} \delta\dot{\alpha}_t^{B_k} \\ \delta\dot{\beta}_t^{B_k} \\ \delta\dot{\theta}_t^{B_k} \\ \delta\dot{b}_{a_t} \\ \delta\dot{b}_{w_t} \end{bmatrix} = \begin{bmatrix} 0 & I & 0 & 0 & 0 \\ 0 & 0 & -R_t^{B_k}[\hat{a}_t - b_{a_t}]_X & -R_t^{B_k} & 0 \\ 0 & 0 & -[\hat{w}_t - b_{w_t}]_X & 0 & -I \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta\alpha_t^{B_k} \\ \delta\beta_t^{B_k} \\ \delta\theta_t^{B_k} \\ \delta b_{a_t} \\ \delta b_{w_t} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ -R_t^{B_k} & 0 & 0 & 0 \\ 0 & -I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} n_a \\ n_w \\ n_{b_a} \\ n_{b_w} \end{bmatrix}$$

$$\delta\dot{x}_t = F_t\delta x_t + G_t n_t$$

Prediction Step

$$\dot{x} = \lim_{\delta t \to \infty} \frac{x(t+\delta t) - x(t)}{\delta t}$$

$$x(t+\delta t) = x(t) + \dot{x}\delta t$$

$$\delta x_{t+\delta t} = (I + F_t\delta t)\delta x_t + (G_t\delta t)n_t$$

The **error covariance** $P_{B_{k+1}}^{B_k}$ is computed recursively with the initial covariance $P_{B_k}^{B_k} = 0$

$$P_{t+\delta t}^{B_k} = (I + F_t\delta t)P_t^{B_k}(I + F_t\delta t)^T + (G_t\delta t)Q(G_t\delta t)^T$$

$Q = (\sigma_a^2, \sigma_w^2, \sigma_{b_a}^2, \sigma_{b_w}^2)$ diagonal covariance matrix of noise

The **Jacobian** matrix $J^{B_k}_{B_{k+1}}$ of $\delta x^{B_k}_{B_{k+1}}$ w.r.t $\delta x^{B_k}_{B_k}$ is computed recursively with the initial Jacobian $J^{B_k}_{B_k} = I$

$$J_{t+\delta t} = (I + F_t \delta t) J_t$$
$$t \in [k, k+1]$$

When the estimation of the **biases changes slightly**, the pre-integration is adjusted by this **first-order approximation** with respect to the bias.

$$\alpha^{B_k}_{B_{k+1}} \approx \hat{\alpha}^{B_k}_{B_{k+1}} + J^{\alpha}_{b_a}\delta b_{a_k} + J^{\alpha}_{b_w}\delta b_{w_k}$$
$$\beta^{B_k}_{B_{k+1}} \approx \hat{\beta}^{B_k}_{B_{k+1}} + J^{\beta}_{b_a}\delta b_{a_k} + J^{\alpha}_{b_w}\delta b_{w_k}$$
$$\gamma^{B_k}_{B_{k+1}} \approx \hat{\gamma}^{B_k}_{B_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J^{\gamma}_{b_w}\delta b_{w_k} \end{bmatrix}$$

with $J^{\alpha}_{b_a}$ is the sub-block matrix in $J^{B_k}_{B_{k+1}}$corresponding to $\frac{\delta x^{B_k}_{Bk+1}}{\delta b_a}$

When the estimation of the **biases changes significantly**, the pre-integration is **re-propagated**.

The **bias** between two IMU measurements is assumed **constant**.

The discrete-time implementation uses the **midpoint method** for numerical integration.

$$\delta\dot{\Theta}^{B_k}_t = -[\frac{w_t + w_{t+1}}{2} - b_{w_t}]_X \delta\Theta^{B_k}_t - \delta b_{w_t} - \frac{n_{w_t} + n_{w_{t+1}}}{2}$$
$$\delta\Theta^{B_k}_{t+1} = [I - \delta t[\frac{w_t + w_{t+1}}{2} - b_{w_t}]_X]\delta\Theta^{B_k}_t - \delta b_{w_t}\delta t - \frac{n_{w_t} + n_{w_{t+1}}}{2}\delta t$$

$$\begin{aligned}\delta\dot{\beta}^{B_k}_t &= -R^{B_k}_t[a_t - b_{a_t}]_X \delta^{B_k}_t - R^{B_k}_t \delta b_{a_t} - R^{B_k}_t n_{a_t} \\ &= -\frac{1}{2}R^{B_k}_t[a_t - b_{a_t}]_X\delta\Theta^{B_k}_t - \frac{1}{2}R^{B_k}_{t+1}[a_{t+1} - b_{a_t}]_X\delta\Theta^{B_k}_{t+1} - \frac{1}{2}(R^{B_k}_t + R^{B_k}_{t+1})\delta b_{a_t} \\ &\quad -\frac{1}{2}(R^{B_k}_t + R^{B_k}_{t+1})n_{a_t}\end{aligned}$$

$$\begin{aligned}\delta\beta^{B_k}_{t+1} = \{&-\frac{\delta t}{2}R^{B_k}_t[a_t - b_{a_t}]_X - \frac{\delta t^2}{2}R^{B_k}_{t+1}[a_{t+1} - b_{a_t}]_X \\ &[I - \delta t[\frac{w_t + w_{t+1}}{2} - b_{w_t}]_X]\}\delta\Theta^{B_k}_t \\ &+\frac{\delta t^2}{2}R^{B_k}_{t+1}[a_{t+1} - b_{a_t}]_X\delta b_{w_t} - \frac{1}{2}(R^{B_k}_t + R^{B_k}_{t+1})\delta b_{a_t} + \frac{\delta t^2}{4}R^{B_k}_{t+1}[a_{t+1} - b_{a_t}]_X n_{w_t}\end{aligned}$$

$$+\frac{\delta t^2}{4}R_{t+1}^{B_k}[a_{t+1}-b_{a_t}]_X n_{w_{t+1}} - \frac{\delta t}{2}R_t^{B_k}n_{a_t} - \frac{\delta t}{2}R_{t+1}^{B_k}n_{a_{t+1}} + \delta\beta_t^{B_k}$$

$$\delta\dot{\alpha}_t^{B_k} = \frac{1}{2}(\delta\beta_t^{B_k} + \delta\beta_{t+1}^{B_k})$$

$$\delta\alpha_{t+1}^{B_k} = \delta\alpha_t^{B_k} + \frac{1}{2}(\delta\beta_t^{B_k} + \delta\beta_{t+1}^{B_k})\delta t$$

$$\begin{bmatrix}\delta\alpha_{k+1}\\ \delta\Theta_{k+1}\\ \delta\beta_{k+1}\\ \delta b_{a_{k+1}}\\ \delta b_{w_{k+1}}\end{bmatrix} = \begin{bmatrix} I & F_{01} & \delta t & F_{03} & F_{04}\\ 0 & F_{11} & 0 & 0 & -\delta t\\ 0 & F_{21} & I & F_{23} & F_{24}\\ 0 & 0 & 0 & I & 0\\ 0 & 0 & 0 & 0 & I\end{bmatrix}\begin{bmatrix}\delta\alpha_k\\ \delta\Theta_k\\ \delta\beta_k\\ \delta b_{a_k}\\ \delta b_{w_k}\end{bmatrix}$$

$$+\begin{bmatrix} G_{00} & G_{01} & G_{02} & G_{03} & 0 & 0\\ 0 & -\delta t/2 & 0 & -\delta t/2 & 0 & 0\\ \frac{-R_k\delta t}{2} & G_{21} & \frac{-R_{k+1}\delta t}{2} & G_{23} & 0 & 0\\ 0 & 0 & 0 & 0 & \delta t & 0\\ 0 & 0 & 0 & 0 & 0 & \delta t\end{bmatrix}\begin{bmatrix} n_{a_k}\\ n_{w_k}\\ n_{a_{k+1}}\\ n_{w_{k+1}}\\ n_{b_a}\\ n_{b_w}\end{bmatrix}$$

$$F_{01} = -\frac{\delta t^2}{4}R_k[\hat{a}_k - b_{a_k}]_X - \frac{\delta t^2}{4}R_{k+1}[\hat{a}_{k+1} - b_{a_k}]_X[I - [\frac{\hat{w}_k + \hat{w}_{k+1}}{2} - b_{w_k}]_X\delta t]$$

$$F_{03} = -\frac{\delta t^2}{4}(R_k + R_{k+1})$$

$$F_{04} = \frac{\delta t^3}{4}R_{k+1}[\hat{a}_{k+1} - b_{a_k}]_X$$

$$F_{11} = I - [\frac{\hat{w}_k + \hat{w}_{k+1}}{2} - b_{w_k}]_X\delta t$$

$$F_{21} = -\frac{\delta t}{2}R_k[\hat{a}_k - b_{a_k}]_X - \frac{\delta t}{2}R_{k+1}[\hat{a}_{k+1} - b_{a_k}]_X[I - [\frac{\hat{w}_k + \hat{w}_{k+1}}{2} - b_{w_k}]_X\delta t]$$

$$F_{23} = -\frac{\delta t}{2}(R_k + R_{k+1})$$

$$F_{24} = \frac{\delta t^2}{2}R_{k+1}[\hat{a}_{k+1} - b_{a_k}]_X$$

$$G_{00} = -\frac{\delta t^2}{4}R_k$$

$$G_{01} = G_{03} = \frac{\delta t^3}{8}R_{k+1}[\hat{a}_{k+1} - b_{a_k}]_X$$

$$G_{02} = -\frac{\delta t^2}{4}R_{k+1}$$

$$G_{21} = G_{23} = \frac{\delta t^2}{4} R_{k+1}[\hat{a}_{k+1} - b_{a_k}]_X$$

Monocular Camera

Visual processing for **SLAM** (Simultaneous Localization and Mapping) can be categorized into either **direct** or **indirect** methods according to the definition of visual residual models. Direct approaches minimize **photometric error** while indirect approaches **minimize geometric displacement**. Direct methods require a good initial guess due to their small region of attraction, while indirect approaches consume extra computational resources on extracting and matching features. Vins-Mono can be considered a direct method, as the feature tracking minimizes the photometric error.

Calibration

Vins-Mono assumes a **calibrated camera**. Camera calibration can be performed using the **CamOdoCal** library, which uses the **Ceres Solver** to solve optimization problems.

Feature Detection and Tracking

New features are **detected** in the **latest frame** and then **tracked** between **consecutive frames**. **Outlier rejection** is performed on the tracked features. New features are detected to maintain a **minimum number of 100 to 300 features in each image**. The **2D features** are transformed to **normalized undistorted projection vectors** in the camera frame and their **velocity** from the last frame is calculated.

Contrast Limited Adaptive Histogram Equalization (CLAHE)

The feature detection provides the option to apply **Contrast Limited Adaptive Histogram Equalization (CLAHE)** to the input images.

The **Contrast Limited Adaptive Histogram Equalization (CLAHE)** algorithm divides an image into small tiles and in each tile applies histogram equalization. To reduce the effect of noise amplification, if any histogram bin is above a specific contrast limit, those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied.

The **Contrast Limited Adaptive Histogram Equalization (CLAHE)** is implemented using the **OpenCV** function **createCLAHE().**

Shi-Tomasi Corner Detector

The feature detection uses **Shi-Tomasi corners** and enforces a **uniform feature distribution** by setting a **minimum separation of pixels** between two neighboring features.

The **Shi-Tomasi Corner Detector**, described in Shi and Tomasi 1994, improved on the **Harris**

Corner Detector by introducing a different formula for the score R.

$$R = \frac{trace(M)}{2} - \frac{1}{2}\sqrt{(traced(M))^2 - 4det(M)}$$

$R = min(\lambda_1, \lambda_2)$	Smallest eigenvalue
$T < R$	Pixel is a **corner**
T	Threshold

The **Shi-Tomasi corner detector**, including the **minimum separation of pixels between two neighboring features**, is implemented using the **OpenCV** function goodFeaturesToTrack().

The **Harris Corner Detector**, described in Harris and Stephens 1988, determines which windows in an image produce large variations when moved. The variations are expressed as a score. Pixels get classified as corners based on the score for the associated window.

Good features are those that provide a high score when the window is moved in any direction, which translates into **maximizing** the **sum of squared differences** between the **original window** and the **window moved** by d_x, d_y.

$$E(d_x, d_y) = \sum_{x,y} w(x,y)(I(x + d_x, y + d_y) - I(x,y))^2$$

$I(x,y)$	Intensity at location x, y in the image patch in the window
d_x, d_y	Displacement
$w(x,y)$	Window function
$w(x,y) = \exp - \frac{x^2 + y^2}{2\sigma^2}$	Gaussian window function

Approximation by first-order Taylor expansion

$$I(x + d_x, y + d_y) \approx I(x,y) + \frac{\delta I}{\delta x} d_x + \frac{\delta I}{\delta y} d_y$$

$$E(d_x, d_y) \approx \sum_{x,y} w(x,y)[I(x,y) + \frac{\delta I}{\delta x} d_x + \frac{\delta I}{\delta y} d_y - I(x,y)]^2$$

$I_x = \frac{\delta I}{\delta x} = I \otimes (-1, 0, 1)$	Convolution with Sharr operator (central difference)

$$I_y = \frac{\delta I}{\delta y} = I \otimes (-1, 0, 1)^T$$

$$E(d_x, d_y) \approx \sum_{x,y} w(x,y)(I_x^2 d_x^2 + 2 I_x I_y d_x d_y + I_y^2 d_y^2)$$

$$E(d_x, d_y) \approx [d_x d_y](\sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ \\ I_x I_y & I_y^2 \end{bmatrix}) \begin{bmatrix} d_x \\ \\ d_y \end{bmatrix}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$ Spatial gradient matrix = **structure matrix**

$$E(d_x, d_y) \approx [d_x d_y] M \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Score R based on eigenvalues λ_1 and λ_2 of M

$R = det(M) - k(trace(M))^2$	
$det(M) = \lambda_1 \lambda_2$	
$trace(M) = \lambda_1 + \lambda_2$	
k	Tunable sensitivity parameter

$\lvert R \rvert \approx 0$	$\lambda_1 \approx \lambda_2 \approx 0$ pixel is **flat**
$R < 0$	$\lambda_1 \ll \lambda_2$ or $\lambda_2 \ll \lambda_1$: pixel is an **edge**
$R \gg 0$	$0 \ll \lambda_1 \approx \lambda_2$: pixel is a **corner**

Lucas-Kanade sparse Optical Flow

Features are tracked using the **Lucas-Kanade sparse optical flow** algorithm, described in Lucas and Kanade 1981, extended with **iterative** calculations and a **coarse-to-fine pyramid** approach, as described in Bouguet 2000.

Lucas-Kanade sparse optical flow algorithm uses a differential local optimization method, for the optical flow estimation. It assumes that the **optical flow** is **constant within a neighbourhood of a point under consideration**, and it solves the basic optical flow equations for all the pixels in that neighbourhood by **least squares**.

$$E(d_x, d_y) = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} (I_k(x,y) - I_{k+1}(x+d_x, y+d_y))^2$$

I_k, I_{k+1}	Consecutive images

$p = [p_x p_y]^T$	Point in I_k
$d = [d_x d_y]^T$	Optical flow
w_x, w_y	Window size in x and y from point

Compute the partial derivative with respect to the parameters d, set to 0 and solve for extremum.

$$\frac{\delta E(d)}{\delta d}|_{d=\hat{d}} = 0$$

$$-2\sum_{x,y}(I_k(x,y) - I_{k+1}(x+d_x, y+d_y))[\frac{\delta I_{k+1}}{\delta x}\frac{\delta I_{k+1}}{\delta y}] = 0$$

Approximation by first-order Taylor expansion

$$I_{k+1}(x+d_x, y+d_y) = I_{k+1}(x,y) + \frac{\delta I_{k+1}}{\delta x}d_x + \frac{\delta I_{k+1}}{\delta y}d_y$$

The optical flow vector has to be **small** enough (less than 1 pixel) for the differential equation of the optical flow to hold. A **coarse-to-fine** approach using a **pyramid** allows to extend the reach and increases robustness.

$$-2\sum_{x,y}(I_k(x,y) - I_{k+1}(x,y) - [\frac{\delta I_{k+1}}{\delta x}\frac{\delta I_{k+1}}{\delta y}]\begin{bmatrix} d_x \\ d_y \end{bmatrix})[\frac{\delta I_{k+1}}{\delta x}\frac{\delta I_{k+1}}{\delta y}] = 0$$

The image derivatives are calculated from I_k instead of I_{k+1} so that the spatial gradient matrix M does not need to be re-calculated in the iterative operation of the optical flow.

$I_x = \frac{\delta I}{\delta x} = I \otimes (-1,0,1)$	Convolution with Sharr operator (central difference)
$I_y = \frac{\delta I}{\delta y} = I \otimes (-1,0,1)^T$	
$\delta I = \frac{\delta I}{\delta t} = I_k - I_{k+1}$	Temporal image derivative

$$\sum_{x,y}(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}\begin{bmatrix} d_x \\ d_y \end{bmatrix} - \begin{bmatrix} I_x \delta I \\ I_y \delta I \end{bmatrix}) = 0$$

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Spatial gradient matrix

$$b = \sum_{x,y} \begin{bmatrix} I_x \delta I \\ \\ I_y \delta I \end{bmatrix}$$

Image mismatch vector

Solving linear equations

$$M \begin{bmatrix} d_x \\ d_y \end{bmatrix} = b$$

M must be invertible, which is the case when the eigenvalues satisfy $\lambda_1 \geq= \lambda_2 > 0$. The approach works best when λ_1 and λ_2 are large enough and have similar magnitude, which is nicely matched by the corners detected by the Shi-Tomasi Corner Detector!

To get an accurate solution using the standard Lucas-Kanade optical flow equation it is necessary to **iterate multiple times** in a Newton fashion. At each iteration only the temporal image derivative δI needs to be re-calculated using the location provided by the last optical flow estimate d. Then the image mismatch vector gets updated and a new optical flow estimate d is calculated.

Computations are done with sub-pixel accuracy using bilinear interpolation.

An **image pyramid** is created by applying an anti-aliasing filter kernel and sub-sampling by 2 in x and y. Starting from the original image and then applying it n times recursively, creates n hierarchical layers. The typical anti-aliasing filter kernel is [1/4 1/2 1/4]×[1/4 1/2 1/4]T, here [1/16 1/4 3/8 1/4 1/16]×[1/16 1/4 3/8 1/4 1/16]T is used.

The **coarse-to-fine** approach starts with estimating the optical flow at the highest level (coarse) and then the estimates from a higher level are used to initialize the estimation at the next lower level. The last estimation is done at the lowest level (fine).

Features are lost when the point falls outside the image or when the image patches in the window around the point vary too much between the two images, determined by affine image matching. Note that drift can be a problem in tracking a feature over a long sequence of images.

The **Lucas-Kanade sparse optical flow** algorithm with pyramids is implemented using the **OpenCV** function calcOpticalFlowPyrLK().

Outlier rejection using 8-Point Algorithm with RANSAC

Outlier rejection is beneficial, as the tracking uses a least-square approach. Least-squares assume that errors are zero-mean Gaussians, which can be thrown off by outliers (grossly wrong values).

Outlier rejection is performed using **Random Sample Consensus (RANSAC)** with a **fundamental matrix** model as described in Hartley and Zisserman 2003.

RANSAC is used with the 8-point algorithm to calculate the fundamental matrix given a probability of 99% to be able to fit the model. RANSAC iterates through the following steps:

1. Select random sample points (hypothetical inliers): 8 points
2. Fit a model: calculate fundamental matrix F from $x_2^T F x_1 = 0$ with 8 point algorithm
3. Test all points against the model: calculate distance x_2 from epipolar line Fx_1
4. Count inliers: based on distance threshold
5. Determine the best set of points in terms of maximum number of inliers

RANSAC using the calculation of the fundamental matrix is implemented using the **OpenCV** function **findFundamentalMat()**.

All outliers are removed from the list of points tracked.

Undistorted Points

The feature detection and tracking module assumes a **calibrated camera.** The **intrinsic camera parameters** are read in by the CamOdoCal library and used for the camera model to perform the inverse projection from all tracked **distorted points in the image** to **normalized undistorted projection vectors** in the camera frame.

Vins-Mono supports the same **camera models** as the CamOdoCal library, which includes the pinhole camera model (model type: PINHOLE) , the unified projection model (model type: MEI), the equidistant fish-eye model (model type: KANNALA_BRANDT) and the omnidirectional camera model (model type: SCARAMUZZA).

The inverse projection from points in the image to projection vectors in the camera frame is implemented using the **CamOdoCal** library’s function **liftProjective()**.

Velocity

The velocity is calculated for all tracked points based on the normalized undistorted projection vectors in the camera frame.

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \frac{X^{I_k} - X^{I_{k-1}}}{dt} \\ \frac{Y^{I_k} - Y^{I_{k-1}}}{dt} \end{bmatrix}$$

dt	Time interval between when images I_{k-1} and I_k were taken

Initialization

Accurate initial values are crucial to bootstrap any monocular visual-inertial system.

Acceleration excitation is required to make the scale observable, which means that the initialization cannot be done from a stationary condition, but from an unknown moving state (on-the-fly).

The on-the-fly estimator initialization provides **pose**, **velocity**, **gravity vector**, **gyroscope bias**, and **3D feature location**, for bootstrapping the system from unknown initial states. The initialization is also used for **failure recovery**.

The accelerometer bias is ignored during the initialization. Acceleration bias is coupled with gravity which has a large magnitude compared to platform dynamics, and given the short time of the initialization is hard to observe.

Structure-from-Motion (SfM)

The initialization procedure starts with a Structure-from-Motion approach to estimate a graph of **up-to-scale 3D camera poses** and **3D feature positions**. A sliding window of frames is used to bound computational complexity.

If **feature tracking is stable** (more than a certain number of tracked features) and that there is **sufficient parallax** (more than a certain number of pixels) between the **latest frame** and **another frame** in the sliding window, then the **relative camera pose**, **rotation** and **up-to-scale transformation**, between those two frames are calculated from the **essential matrix** obtained by the **8-point algorithm with RANSAC** using the tracked features. Otherwise, the latest frame is just added to the sliding window and we continue with a new frame.

The **scale is set arbitrarily** and starting with the **point triangulation** of the tracked features between the **latest frame** and the **other frame**, the **camera poses of all frames** in the sliding window and the **3D points of all tracked features** are determined, frame by frame, using the **Perspective-n-Point (PnP)** algorithm followed by **point triangulation**.

A global **bundle adjustment** refines the **camera poses** and **3D points** to provide a statistical optimal solution.

As there is no knowledge about the world frame, the first frame in the sliding window is set to be the reference frame C_0. All camera poses and feature positions are represented with respect to that reference frame. Knowing the **extrinsic parameters** between the **camera** and the **IMU**'s body frame, we can transform the poses from camera frame to IMU's body frame.

$p_{C_k}^{C_0}, q_{C_k}^{C_0}$	Poses in first camera's frame
p_C^B, q_C^B	**Extrinsic parameters** between **camera** and **IMU**'s body frame
$q_{B_k}^{C_0} = q_{C_k}^{C_0} \otimes (q_C^B)^{-1}$ $sp_{B_k}^{C_0} = sp_{C_k}^{C_0} - R_{B_k}^{C_0} p_C^B$	Transform poses from camera frame to IMU's body frame
s	**Scaling factor**

The **scaling factor** is used to align the visual up-to-scale poses to the metric scale provided by the IMU.

Essential Matrix using 8-Point Algorithm with RANSAC

The tracked features are provided as normalized projection vectors in the camera frame. Using the x and y coordinates of the features' projection vectors as the pixel coordinates to calculate the fundamental matrix with the **8-point algorithm with RANSAC** implies that the camera calibration matrix is the identity matrix and that the calculated fundamental matrix is identical to the **essential matrix**.

The **8-point algorithm with RANSAC** is implemented using the **OpenCV** library's function **findFundamentalMat()**.

Relative Camera Pose from the Essential Matrix

The **camera pose calculation from the essential matrix** is implemented using the **OpenCV** library's function **recoverPose()**.

Point Triangulation

The **point triangulation from 2 images** determines a **3D point** based on its observed projections in 2 images.

Perspective-n-Point (PnP) Algorithm

The **Perspective-n-Point (PnP)** algorithm determines the **pose of a camera** by **minimizing the reprojection error** between the observed projections and the projected points in a single frame using the Levenberg-Marquardt optimization.

The **Perspective-n-Point (PnP) algorithm** is implemented using the **OpenCV** library's function **solvePnP()**.

Bundle Adjustment

The **bundle adjustment** determines a statistical optimum solution for all **camera poses** and **3D points** by **minimizing the reprojection error** between all observed projections and the projected points in all frames in the sliding window using the Levenberg-Marquardt optimization.

The **optimization for the bundle adjustment** is implemented using the **Ceres** library.

Visual-Inertial Alignment

Gyroscope Bias

$q_{B_k}^{C_0}, q_{B_k+1}^{C_0}$	Rotation of 2 consecutive frames from SfM
$\gamma_{B_{k+1}}^{B_k}$	Rotation in body frame from IMU preintegration

Solve least squares over all frames k in sliding window to determine initial gyroscope bias

$$\min_{\delta b_w} \sum_k ||q_{B_k+1}^{C_0} \otimes q_{B_k}^{C_0} \otimes \gamma_{B_k+1}^{B_k}||^2$$

$$\gamma_{B_{k+1}}^{B_k} \approx \hat{\gamma}_{B_{k+1}}^{B_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_{w_k} \end{bmatrix}$$

Linearized rotation in respect to gyroscope bias from IMU preintegration

$$\hat{\gamma}_{B_{k+1}}^{B_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_w}^{\gamma} \delta b_{w_k} \end{bmatrix} = q_{B_k}^{C_0} \otimes q_{B_k+1}^{C_0}$$

$$J_{b_w}^{\gamma} \delta b_{w_k} = 2 \hat{\gamma}_{B_{k+1}}^{B_k}{}^{-1} \otimes q_{B_k}^{C_0} \otimes q_{B_k+1}^{C_0}$$

$$H \delta b_{w_k} = z$$

Stacking up all frames k in sliding window

$$H = J_{b_w}^{\gamma}$$

$$z = 2 \hat{\gamma}_{B_{k+1}}^{B_k}{}^{-1} \otimes q_{B_k}^{C_0} \otimes q_{B_k+1}^{C_0}$$

Cholesky Factorization

$$(H^T H) \delta b_{w_k} = H^T z$$

$$A \delta b_{w_k} = b$$

$$A = LDL^T$$

$$LDL^T \delta b_{w_k} = b$$

$$DL^T \delta b_{w_k} = y$$

Forward substitution

$L^T \delta b_{w_k} = y$ Back substitution

The **optimization** is implemented using the **Eigen** library's A.ldlt().solve(b) function.

Re-propagate IMU preintegration with new gyroscope bias.

Velocity, Gravity Vector and Metric Scale

$$R_W^{B_k} p_{B_{k+1}}^W = R_W^{B_k}(p_{B_k}^W + v_{B_k}^W \triangle t_k - \frac{1}{2} g^W \triangle t_k^2) + \alpha_{B_k}^{B_k+1}$$

Position in body frame

$$R_W^{B_k} v_{B_{k+1}}^W = R_W^{B_k}(v_{B_k}^W - g^W \triangle t_k) + \beta_{B_k}^{B_k+1}$$

Velocity in body frame from IMU preintegration

Position and velocity between 2 consecutive frames in 1st camera's frame

$$\begin{aligned}\alpha_{B_{k+1}}^{B_k} &= R_{C_0}^{B_k}(s(\hat{p}_{B_{k+1}}^{C_0} - \hat{p}_{B_k}^{C_0}) + \frac{1}{2} g^{C_0} \triangle t_k^2 - R_{B_k}^{C_0} v_{B_k}^{B_k} \triangle t_k) \\ &= R_{C_0}^{B_k}(s\hat{p}_{C_{k+1}}^{C_0} - R_{B_{k+1}}^{C_0} p_C^B - (s\hat{p}_{C_k}^{C_0} - R_{B_k}^{C_0} p_C^B) + \frac{1}{2} g^{C_0} \triangle t_k^2 - R_{B_k}^{C_0} v_{B_k}^{B_k} \triangle t_k \\ &= -\triangle t_k v_{B_k}^{B_k} + \frac{1}{2} R_{C_0}^{B_k} \triangle t_k^2 g^{C_0} + R_{C_0}^{B_k}(\hat{p}_{C_{k+1}}^{C_0} - \hat{p}_{C_k}^{C_0}) s + p_C^B - R_{C_0}^{B_k} R_{B_{k+1}}^{C_0} p_C^B\end{aligned}$$

$$\begin{aligned}\beta_{B_{k+1}}^{B_k} &= R_{C_0}^{B_k}(R_{B_{k+1}}^{C_0} v_{B_{k+1}}^{B_{k+1}} + g^{C_0} \triangle t_k - R_{B_k}^{C_0} v_{B_k}^{B_k}) \\ &= -v_{B_k}^{B_k} + R_{C_0}^{B_k} R_{B_{k+1}}^{C_0} v_{B_{k+1}}^{B_{k+1}} + \triangle t_k R_{C_0}^{B_k} g^{C_0}\end{aligned}$$

$$\hat{z}_{B_{k+1}}^{B_k} = H_{B_{k+1}}^{B_k} X_I$$

$$\begin{bmatrix} \alpha_{B_{k+1}}^{B_k} - p_C^B + R_{C_0}^{B_k} R_{B_{k+1}}^{C_0} p_c^b \\ \beta_{B_{k+1}}^{B_k} \end{bmatrix} = \begin{bmatrix} -\triangle t_k I & 0 & \frac{1}{2} R_{C_0}^{B_k} \triangle t_k^2 & R_{C_0}^{B_k}(\hat{p}_{C_{k+1}}^{C_0} - \hat{p}_{C_k}^{C_0}) \\ -I & R_{C_0}^{B_k} R_{B_{k+1}}^{C_0} & R_{C_0}^{B_k} \triangle t_k & 0 \end{bmatrix} \begin{bmatrix} v_{B_k}^{B_k} \\ v_{B_{k+1}}^{B_{k+1}} \\ g^{C_0} \\ s \end{bmatrix}$$

Solve least squares over all frames k in sliding window to determine initial velocity, gravity vector and metric scale.

$$\min_{X_I} \sum_k ||\hat{z}_{B_{k+1}}^{B_k} - H_{B_{k+1}}^{B_k} X_I||^2$$

$$X_I = [v_{B_0}^{B_0}, v_{B_1}^{B_1}, \cdots v_{B_n}^{B_n}, g^{C_0}, s]^T$$

$HX_I = \hat{z}$ Stacking up all frames k in sliding window

Cholesky Factorization

$$(H^T H)X_I = H^T z$$
$$AX_I = b$$

$$A = LDL^T$$
$$LDL^T X_I = b$$
$$DL^T X_I = y$$ Forward substitution
$$L^T X_I = y$$ Back substitution

The **optimization** is implemented using the **Eigen** library's **A.ldlt().solve(b)** function.

Gravity Refinement

The gravity vector is refined by its known magnitude. The remaining 2 DOF are re-parameterized with 2 variables in tangent space.

$$g^{C_0} = ||g|| \frac{g^{C_0}}{||g^{C_0}||} + \begin{bmatrix} b_1 & b_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$
$$||g|| \approx 9.81 \frac{m}{s^2}$$
g^{C_0}, b_1, b_2 are orthogonal and b_1, b_2 are the basis spanning the tangent space

If $normalize(g^{C_0}) \neq [0,0,1]^T$ otherwise use $[1,0,0]^T$
$$b_1 = normalize(g^{C_0} \times [0,0,1]^T)$$
$$b_2 = normalize(g^{C_0} \times b_1)$$

Substitute g^{C_0} and solve recursively for w_1, w_2 together with the other state variables as before until converge (w_1, w_2 close to zero)

$$H^{B_k}_{B_{k+1}} X_I = \begin{bmatrix} -\triangle t_k I & 0 & \frac{1}{2} R^{B_k}_{C_0} \triangle t_k^2 [b_1 \quad b_2] & R^{B_k}_{C_0} (\hat{p}^{C_0}_{C_{k+1}} - \hat{p}^{C_0}_{C_k}) \\ -I & R^{B_k}_{C_0} R^{C_0}_{B_{k+1}} & R^{B_k}_{C_0} \triangle t_k [b_1 \quad b_2] & 0 \end{bmatrix} \begin{bmatrix} v^{B_k}_{B_k} \\ v^{B_{k+1}}_{B_{k+1}} \\ w_1 \\ w_2 \\ s \end{bmatrix}$$

World Frame

The world frame is defined at the origin of the 1st camera frame with the z-axis rotated to the gravity vector. Yaw can be set arbitrarily.

$R_{C_0}^{W}$	Rotation between 1st camera frame and world frame

Finding the **rotation between the z-axis of the 1st camera frame and the gravity vector** is implemented using the **Eigen** library's **FromTwoVectors()** function.

All variables, including the body frame velocities, are rotated from the 1st camera frame to the world frame. Translational components from the SfM are scaled to metric units.

VIO Optimization

After the estimator initialization, **state estimation** is performed as a tightly-coupled **nonlinear optimization** in form of a **visual-inertial bundle adjustment** in a **sliding window**.

Visual-Inertial Bundle Adjustment

The state estimation can be thought as a **Maximum A Posteriori (MAP)** problem, minimizing the sum of priors and the Mahalanobis norm of all measurement residuals.

$$\min_{X}\{||r_p - H_p X||^2 + \sum_{k} ||r_B(\hat{z}_{B_{k+1}}^{B_k}, X)||^2_{P_{B_{k+1}}^{B_k}} + \sum_{l,j} p(||r_C(\hat{z}_l^{C_j}, X)||^2_{P_l^{C_j}})\}$$

$X = [x_0, x_1, \cdots x_n, x_C^B, \lambda_0, \lambda_1, \cdots \lambda_m]$	**State vector**
$x_k = [p_{B_k}^W, q_{B_k}^W, v_{B_k}^W, b_a, b_w], k \in [0, n]$	IMU state in IMU body frame at time of image k
$x_C^B = [p_C^B, q_C^B]$	Extrinsic parameters
λ_l	Inverse depth of feature l from first observation
n	Number of keyframes
m	Number of features in sliding window
$p(s)$	Cauchy loss
$P_l^{C_j}$	Covariance of a fixed length in tangent space

Solve **nonlinear least squares** with iterative **Levenberg-Marquardt**

The **optimization for the bundle adjustment** is implemented using the **Ceres** library. "LocalParametrization" uses the Lie algebra SO(3) for incremental attitude updates.

Marginalization

To bound computational complexity, IMU states x_k and features λ_l from the sliding window are marginalized into a prior.

If the second last frame is a keyframe, it will stay and the oldest frame is marginalized with its corresponding measurements. If the second last frame is not a keyframe, the visual measurements are thrown out and the IMU measurements are kept. Not all measurements of non-keyframes are marginalized in order to maintain sparsity. The scheme aims to keep spatially separated keyframes in the sliding window to ensure sufficient parallax for feature triangulation and maximize the probability of maintaining accelerometer measurements with large excitation.

Marginalization is performed using the Schur complement. A new prior is constructed based on all marginalized measurements related to the removed state and added onto the existing prior.

Priors obtained from marginalization

$$r_p - H_p X$$

Nonlinear least squares

$$X = \min_x ||f(x)||_P$$
$$x_{k+1} = x_k + \triangle x$$
$$J^T P J \triangle x = J^T P f(x)$$
$$H \delta x = b$$

$$\begin{bmatrix} H_{mm} & H_{mr} \\ H_{rm} & H_{rr} \end{bmatrix} \begin{bmatrix} \delta X_m \\ \delta X_r \end{bmatrix} = \begin{bmatrix} b_m \\ b_r \end{bmatrix}$$

X_m	States to be marginalized
X_r	States remaining

Schur Complement

$$\begin{bmatrix} I & 0 \\ -H_{rm}H_{mm}^{-1} & I \end{bmatrix} \begin{bmatrix} H_{mm} & H_{mr} \\ H_{rm} & H_{rr} \end{bmatrix} \begin{bmatrix} \delta X_m \\ \delta X_r \end{bmatrix} = \begin{bmatrix} I & 0 \\ -H_{rm}H_{mm}^{-1} & I \end{bmatrix} \begin{bmatrix} b_m \\ b_r \end{bmatrix}$$
$$\begin{bmatrix} H_{mm} & H_{mr} \\ 0 & -H_{rm}H_{mm}^{-1}H_{mr} + H_{rr} \end{bmatrix} \begin{bmatrix} \delta X_m \\ \delta X_r \end{bmatrix} = \begin{bmatrix} b_m \\ -H_{rm}H_{mm}^{-1}b_m + b_r \end{bmatrix}$$

$$H_{mm}\delta X_m = b_m$$
$$H_p \delta X_r = r_p$$
$$H_p = (-H_{rm}H_{mm}^{-1}H_{mr} + H_{rr})\delta X_r$$
$$r_p = -H_{rm}H_{mm}^{-1}b_m + b_r$$

$$X_{m:n} = \min_{x_{m:n}} \sum_{t=m}^{n} \sum_{k} ((H_p \delta X_r - b_p) + ||z_t^k - h_t^k(X_{m:n})||^2_{\omega_t^k})$$

IMU Model

IMU Measurement Residual between 2 consecutive frames

$R_W^{B_k} p_{B_{k+1}}^W = R_W^{B_k}(p_{B_k}^W + v_{B_k}^W \triangle t_k - \frac{1}{2} g^W \triangle t_k^2) + \alpha_{B_k}^{B_k+1}$	Position in body frame
$R_W^{B_k} v_{B_{k+1}}^W = R_W^{B_k}(v_{B_k}^W - g^W \triangle t_k) + \beta_{B_k}^{B_k+1}$	Velocity in body frame
$q_W^{B_{k+1}} \otimes q_{B_{k+1}}^W = \gamma_{B_{k+1}}^{B_k}$	Rotation in body frame from IMU preintegration

$$r_B(\hat{z}_{B_{k+1}}^{B_k}, X) = \begin{bmatrix} \delta\alpha_{B_{k+1}}^{B_k} \\ \delta\theta_{B_{k+1}}^{B_k} \\ \delta\beta_{B_{k+1}}^{B_k} \\ \delta b_a \\ \delta b_w \end{bmatrix} = \begin{bmatrix} R_W^{B_k}(p_{B_{k+1}}^W - p_{B_k}^W - v_{B_k}^W \triangle t_k + \frac{1}{2} g^W \triangle t_k^2) - \hat{\alpha}_{B_{k+1}}^{B_k} \\ 2[(\hat{\gamma}_{B_{k+1}}^{B_k})^{-1} \otimes (q_{B_{k+1}}^W)^{-1} \otimes q_{B_{k+1}}^W]_{xyz} \\ R_W^{B_k}(v_{B_{k+1}}^W - v_{B_k}^W + g^W \triangle t_k) - \hat{\beta}_{B_{k+1}}^{B_k} \\ b_{a_{k+1}} - b_{a_k} \\ b_{w_{k+1}} - b_{w_k} \end{bmatrix}$$

B	Set of all IMU measurements
$\delta\theta_{B_{k+1}}^{B_k}$	3D error-state representation of a quaternion
$[\cdot]_{xyz}$	Vector part of a quaternion

$X = [p_{B_k}^W, q_{B_k}^W][v_{B_k}^W, b_{a_k}, b_{w_k}][p_{B_{k+1}}^W, q_{B_{k+1}}^W][v_{B_{k+1}}^W, b_{a_{k+1}}, b_{w_{k+1}}]$ State vector

Accelerometer and gyroscope biases are included for online correction.

Lie Algebra $SO(3)$ is used for attitude (vs. quaternions) to perform perturbation analysis and optimize iteratively. Update quaternion in state with attitude increment.

$q_{B'}^W = q_B^W \delta q_{B'}^B$ Attitude update

$R_B^W(\delta\theta) = R_B^W exp(\delta\theta_{B'}^B) = R_B^W(I + \delta\theta_{B'}^B, \times)$	Attitude with perturbation
$R_W^B(\delta\theta) = exp(\delta\theta_{B'}^B) R_W^B = (I - \delta\theta_{B'}^B, \times) R_W^B$	
$\delta\theta_{B'}^B$	Rotation vector corresponding to attitude perturbation

Jacobian

$$J[0]_{15x6}=\frac{r_B(\hat{z}^{B_k}_{B_{k+1}},X)}{\delta[p^W_{B_k},q^W_{B_k}]}=\begin{bmatrix}-R^{B_k}_W & [R^{B_k}_W(p^W_{B_{k+1}}-p^W_{B_k}-v^W_{B_k}\triangle t_k+\frac{1}{2}g^W\triangle t_k^2)]_X\\ 0 & [(q^W_{B_{k+1}})^{-1}\otimes q^W_{B_k}]_L[[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1}]_R]_{3x3}\\ 0 & [R^{B_k}_W(v^W_{B_{k+1}}-v^W_{B_k}+g^W\triangle t_k)]_X\\ 0 & 0\\ 0 & 0\end{bmatrix}$$

$$J[1]_{15x9}=\frac{r_B(\hat{z}^{B_k}_{B_{k+1}},X)}{\delta[v^W_{B_k},b_{a_k},b_{w_k}]}=\begin{bmatrix}-R^{B_k}_W\triangle t_k & -J^\alpha_{b_a} & -J^\alpha_{b_w}\\ 0 & 0 & -[[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1}\otimes(q^W_{B_{k+1}})^{-1}\otimes q^W_{B_{k+1}}]_R]_{3x3}J^\gamma_{b_w}\\ -R^{B_k}_W & -J^\beta_{b_a} & -J^\beta_{b_w}\\ 0 & -I & 0\\ 0 & 0 & -I\end{bmatrix}$$

$$J[2]_{15x6}=\frac{r_B(\hat{z}^{B_k}_{B_{k+1}},X)}{\delta[p^W_{B_{k+1}},q^W_{B_{k+1}}]}=\begin{bmatrix}R^{B_k}_W & 0\\ 0 & [(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1}\otimes(q^W_{B_k})^{-1}\otimes q^W_{B_{k+1}}]_L]_{3x3}\\ 0 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix}$$

$$J[3]_{15x9}=\frac{r_B(\hat{z}^{B_k}_{B_{k+1}},X)}{\delta[v^W_{B_{k+1}},b_{a_{k+1}},b_{w_{k+1}}]}=\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ R^{B_k}_W & 0 & 0\\ 0 & I & 0\\ 0 & 0 & I\end{bmatrix}$$

J[0]

$$\begin{aligned}\frac{\delta\delta\alpha^{B_k}_{B_{k+1}}}{\delta p^W_{B_k}}&=\lim_{\delta p^{B_k}_{B_k}\to 0}\frac{R^{B_k}_W(p^W_{B_{k+1}}-(p^W_{B_k}+\delta p^W_{B_k})-v^W_{B_k}\triangle t_k+\frac{1}{2}g^W\triangle t_k^2)-\hat{\alpha}^{B_k}_{B_{k+1}}-\delta\alpha^{B_k}_{B_{k+1}}}{\delta p^W_{B_k}}\\ &=-R^{B_k}_W\end{aligned}$$

$$\begin{aligned}\frac{\delta\delta\alpha^{B_k}_{B_{k+1}}}{\delta q^W_{B_k}}&=\lim_{\delta\theta^{B_k}_{B_k}\to 0}\frac{exp(\delta\theta^{B_k'}_{B_k})R^{B_k}_W(p^W_{B_{k+1}}-p^W_{B_k}-v^W_{B_k}\triangle t_k+\frac{1}{2}g^W\triangle t_k^2)-R^{B_k}_W(p^W_{B_{k+1}}-p^W_{B_k}-v^W_{B_k}\triangle t_k+\frac{1}{2}g^W\triangle t_k^2)}{\delta\theta^W_{B_k}}\\ &=\lim_{\delta\theta^{B_k}_{B_k}\to 0}\frac{((I-\delta\theta^{B_k}_{B_k'},\times)R^{B_k}_W-R^{B_k}_W)(p^W_{B_{k+1}}-p^W_{B_k}-v^W_{B_k}\triangle t_k+\frac{1}{2}g^W\triangle t_k^2)}{\delta\theta^W_{B_k}}\\ &=[R^{B_k}_W(p^W_{B_{k+1}}-p^W_{B_k}-v^W_{B_k}\triangle t_k+\frac{1}{2}g^W\triangle t_k^2)]_X\end{aligned}$$

$$\frac{\delta\delta\beta^{B_k}_{B_{k+1}}}{\delta q^W_{B_k}} = [R^{B_k}_W(v^W_{B_{k+1}} - v^W_{B_k} + g^W \triangle t_k)]_X$$

$$\begin{aligned}
\frac{\delta\delta\theta^{B_k}_{B_{k+1}}}{\delta q^W_{B_k}} &= \lim_{\delta\theta^{B_k}_{B_k}\to 0} \frac{2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta^W_{B_k} \end{bmatrix})^{-1} \otimes q^W_{B_{k+1}}]_{xyz} - 2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_k} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix})^{-1} \otimes q^W_{B_{k+1}}]_{xyz}}{\delta\theta^W_{B_k}} \\
&= \lim_{\delta\theta^{B_k}_{B_k}\to 0} \frac{2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1}]_L [(q^W_{B_k})^{-1} \otimes q^W_{B_{k+1}}]_R \begin{bmatrix} 1 \\ -\frac{1}{2}\delta\theta^W_{B_k} \end{bmatrix}}{\delta\theta^W_{B_k}} \\
&= [(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1}]_L [[(q^W_{B_k})^{-1} \otimes q^W_{B_{k+1}}]_R]_{3x3} \\
&= [(q^W_{B_{k+1}})^{-1} \otimes q^W_{B_k}]_L [[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1}]_R]_{3x3}
\end{aligned}$$

$$\delta q = \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta^W_{B_k} \end{bmatrix}$$

$[\cdot]_{3x3}$ 3x3 block at the right-bottom of a matrix

J[1]

$$\begin{aligned}
\frac{\delta\delta\alpha^{B_k}_{B_{k+1}}}{\delta v^W_{B_k}} &= \lim_{\delta p^{B_k}_{B_k}\to 0} \frac{R^{B_k}_W(p^W_{B_{k+1}} - p^W_{B_k} - (v^W_{B_k} + \delta v^W_{B_k})\triangle t_k + \frac{1}{2}g^W \triangle t_k^2) - \hat{\alpha}^{B_k}_{B_{k+1}} - \delta\alpha^{B_k}_{B_{k+1}}}{\delta v^W_{B_k}} \\
&= -R^{B_k}_W \triangle t_k
\end{aligned}$$

$\alpha^{B_k}_{B_{k+1}} \approx \hat{\alpha}^{B_k}_{B_{k+1}} + J^{\alpha}_{b_a}\delta b_{a_k} + J^{\alpha}_{b_w}\delta b_{w_k}$ from IMU pre-integration

$$\frac{\delta\delta\alpha^{B_k}_{B_{k+1}}}{\delta b_{a_k}} = \lim_{\delta b_{a_k}\to 0} \frac{-(\hat{\alpha}^{B_k}_{B_{k+1}} + J^{\alpha}_{b_a}\delta b_{a_k}) + \hat{\alpha}^{B_k}_{B_{k+1}}}{\delta b_{a_k}} = -J^{\alpha}_{b_a}$$

$\beta^{B_k}_{B_{k+1}} \approx \hat{\beta}^{B_k}_{B_{k+1}} + J^{\beta}_{b_a}\delta b_{a_k} + J^{\alpha}_{b_w}\delta b_{w_k}$ from IMU pre-integration

$$\frac{\delta\delta\alpha^{B_k}_{B_{k+1}}}{\delta b_{w_k}} = \lim_{\delta b_{w_k}\to 0} \frac{-(\hat{\alpha}^{B_k}_{B_{k+1}} + J^{\alpha}_{b_g}\delta b_{w_k}) + \hat{\alpha}^{B_k}_{B_{k+1}}}{\delta b_{w_k}} = -J^{\alpha}_{b_w}$$

$$\frac{\delta\delta\theta^{B_k}_{B_{k+1}}}{\delta b_{w_k}} = \lim_{\delta b_{w_k}\to 0} \frac{2[(\hat{\gamma}^{B_k}_{B_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}J^{\gamma}_{b_w}\delta b_{w_k} \end{bmatrix})^{-1} \otimes (q^W_{B_{k+1}})^{-1} \otimes q^W_{B_{k+1}}]_{xyz} - 2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_{k+1}})^{-1} \otimes q^W_{B_{k+1}}]_{xyz}}{\delta b_{w_k}}$$

$$= \lim_{\delta b_{w_k} \to 0} \frac{2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_{k+1}})^{-1} \otimes q^W_{B_{k+1}}]_R \begin{bmatrix} 0 \\ -\frac{1}{2} J^{\gamma}_{b_w} \delta b_{w_k} \end{bmatrix}}{\delta b_{w_k}}$$

$$= -[[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_{k+1}})^{-1} \otimes q^W_{B_{k+1}}]_R]_{3x3} J^{\gamma}_{b_w}$$

$$\frac{\delta \delta \beta^{B_k}_{B_{k+1}}}{\delta v^W_{B_k}} = -R^{B_k}_W$$

$$\frac{\delta \delta \beta^{B_k}_{B_{k+1}}}{\delta b_{a_k}} = -J^{\beta}_{b_a}$$

$$\frac{\delta \delta \beta^{B_k}_{B_{k+1}}}{\delta b_{w_k}} = -J^{\beta}_{b_w}$$

$$\frac{\delta \delta b_a}{\delta b_{a_k}} = -I$$

$$\frac{\delta \delta b_w}{\delta b_{w_k}} = -I$$

J[2]

$$\frac{\delta \delta \alpha^{B_k}_{B_{k+1}}}{\delta p^W_{B_{k+1}}} = \lim_{\delta p^W_{B_{k+1}} \to 0} \frac{R^{B_k}_W (p^W_{B_{k+1}} + \delta p^W_{B_{k+1}} - p^W_{B_k} - v^W_{B_k} \triangle t_k + \frac{1}{2} g^W \triangle t_k^2) - \hat{\alpha}^{B_k}_{B_{k+1}} - \delta \alpha^{B_k}_{B_{k+1}}}{\delta p^W_{B_{k+1}}}$$
$$= R^{b_k}_W$$

$$\frac{\delta \delta \theta^{B_k}_{B_{k+1}}}{\delta q^W_{B_{k+1}}} = \lim_{\delta \theta^W_{B_{k+1}} \to 0} \frac{2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_{k+1}})^{-1} \otimes [q^W_{B_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \theta^W_{B_{k+1}} \end{bmatrix}]]_{xyz} - 2[(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_k})^{-1} \otimes q^W_{B_{k+1}}]_{xyz}}{\delta \theta^W_{B_{k+1}}}$$
$$= [(\hat{\gamma}^{B_k}_{B_{k+1}})^{-1} \otimes (q^W_{B_k})^{-1} \otimes q^W_{B_{k+1}}]_L]_{3x3}$$

J[3]

$$\frac{\delta \delta \beta^{B_k}_{B_{k+1}}}{\delta v^W_{B_{k+1}}} = R^{B_k}_W$$

$$\frac{\delta \delta \beta^{B_k}_{B_{k+1}}}{\delta b_{a_{k+1}}} = I$$

$$\frac{\delta \delta \beta^{B_k}_{B_{k+1}}}{\delta b_{w_{k+1}}} = I$$

Visual Measurement Residual for feature l in image j (first observed in image i)

In contrast to traditional pinhole camera models that define the reprojection errors on a generalized image plane, the **camera measurements** are defined on a **unit sphere**.

$$r_C(\hat{z}_l^{C_j}, X) = [b_1 b_2]^T (\hat{P}_l^{C_j} - \frac{P_l^{C_j}}{||P_l^{C_j}||})$$

C	Set of features at least observed twice
b_1, b_2	Arbitrarily orthogonal bases of tangent plane to $\hat{P}_l^{C_j}$

Unit vector of feature l in image j

$$\hat{P}_l^{C_j} = \pi_C^{-1}(\begin{bmatrix} \hat{u}_l^{C_j} \\ \hat{v}_l^{C_j} \end{bmatrix})$$

Based on first observation of feature l in image i

$$\begin{aligned} P_l^{C_j} &= R_B^C(R_W^{B_j}(R_{B_i}^W(R_C^B \frac{1}{\lambda_l}\pi_C^{-1}(\begin{bmatrix} u_l^{C_i} \\ v_l^{C_i} \end{bmatrix}) + p_C^B) + p_{B_i}^W) + p_W^{B_j}) + p_B^C \\ &= R_B^C(R_W^{B_j}(R_{B_i}^W(R_C^B \frac{1}{\lambda_l}\pi_C^{-1}(\begin{bmatrix} u_l^{C_i} \\ v_l^{C_i} \end{bmatrix}) + p_C^B) + p_{B_i}^W - p_{B_j}^W) - p_C^B) \end{aligned}$$

$$R_B^W P^B + P_B^W = P^W$$
$$P_W^B = -R_W^B P_B^W$$

$[u_l^{C_i}, v_l^{C_i}]^T$	First observation of feature l in image i
$[\hat{u}_l^{C_j}, \hat{v}_l^{C_j}]^T$	Observation of the same feature l in image j
π_C^{-1}	Back projection from pixel to unit vector

$X = [p_{B_i}^W, q_{B_i}^W][p_{B_j}^W, q_{B_j}^W][p_C^B, q_C^B]\lambda_l$ State vector

Jacobian

$$\frac{\delta r_C}{\delta X} = \frac{\delta r_C}{\delta P_l^{C_j}} \frac{\delta P_l^{C_j}}{\delta X}$$

Chain rule

$$\frac{\delta r_C}{\delta P_l^{C_j}} = \begin{bmatrix} \frac{1}{z_l^{C_j}} & 0 & \frac{x_l^{C_j}}{(z_l^{C_j})^2} \\ 0 & \frac{1}{z_l^{C_j}} & -\frac{y_l^{C_j}}{(z_l^{C_j})^2} \end{bmatrix}$$

$$\frac{\delta r_C}{\delta P_l^{C_j}} = [b_1 b_2]^T \frac{\delta(\frac{P_l^{C_j}}{||P_l^{C_j}||} - \hat{P}_l^{C_j})}{\delta P_l^{C_j}}$$

$$= \frac{\delta \frac{P_l^{C_j}}{||P_l^{C_j}||}}{P_l^{C_j}}$$

$$= \frac{1}{||P_l^{C_j}||} I - \frac{P_l^{C_j} \frac{\delta ||P_l^{C_j}||}{\delta P_l^{C_j}}}{||P_l^{C_j}||^2}$$

$$= \begin{bmatrix} \frac{1}{||P_l^{C_j}||} - \frac{x^2}{||P_l^{C_j}||^3} & -\frac{xy}{||P_l^{C_j}||^3} & -\frac{xz}{||P_l^{C_j}||^3} \\ -\frac{xy}{||P_l^{C_j}||^3} & \frac{1}{||P_l^{C_j}||} - \frac{y^2}{||P_l^{C_j}||^3} & -\frac{yz}{||P_l^{C_j}||^3} \\ -\frac{xz}{||P_l^{C_j}||^3} & -\frac{yz}{||P_l^{C_j}||^3} & \frac{1}{||P_l^{C_j}||} - \frac{z^2}{||P_l^{C_j}||^3} \end{bmatrix} \text{with } P_l^{C_j} = [x, y, z]^T$$

$$J[0]_{3x6} = \frac{\delta P_l^{C_j}}{\delta[p_{B_i}^W, q_{B_i}^W]} = [R_B^C R_W^{B_j} - R_B^C R_W^{B_j} R_{B_i}^W [[\frac{1}{\lambda_l} R_C^B \hat{P}_l^{C_i} + P_C^B]_X]]$$

$$\frac{\delta P_l^{C_j}}{\delta q_{B_i}^W} = \lim_{\delta\theta_{B_i}^{B_i} \to 0} \frac{R_B^C (R_W^{B_j} (R_{B_i}^W exp(\delta[\theta_{B_i}^{B_i}]_X)(R_C^B \frac{1}{\lambda_l} \hat{P}_l^{C_i} + P_C^B) + P_{B_i}^W - P_{B_j}^W) - P_C^B) - P_l^{C_j}}{\delta\theta_{B_i}^{B_i}}$$

$$= \lim_{\delta\theta_{B_i}^{B_i} \to 0} \frac{R_B^C R_W^{B_j} R_{B_i}^W (I + \delta[\theta_{B_i}^{B_i}]_X)(R_C^B \frac{1}{\lambda_l} \hat{P}_l^{C_i} + P_C^B)}{\delta\theta_{B_i}^{B_i}}$$

$$= -R_B^C R_W^{B_j} R_{B_i}^W [[\frac{1}{\lambda_l} R_C^B \hat{P}_l^{C_i} + P_C^B]_X]$$

$$J[1]_{3x6} = \frac{\delta P_l^{C_j}}{\delta[p_{B_j}^W, q_{B_j}^W]} = [-R_B^C R_W^{B_j} R_B^C ([R_W^{B_j} (R_{B_i}^W (R_C^B \frac{1}{\lambda_l} \hat{P}_l^{C_i}) + P_{B_i}^W - P_{B_j}^W)]_X)]$$

$$\frac{\delta P_l^{C_j}}{\delta q_{B_j}^W} = \lim_{\delta\theta_{B_i}^{B_i}\to 0} \frac{R_B^C(exp(\delta[\theta_{B_i}^{B_i}]_X)R_W^{B_j}(R_{B_i}^W(R_C^B\frac{1}{\lambda_l}\hat{P}_l^{C_i} + P_C^B) + P_{B_i}^W - P_{B_j}^W) - P_C^B) - P_l^{C_j}}{\delta\theta_{B_i}^{B_i}}$$

$$= \lim_{\delta\theta_{B_i}^{B_i}\to 0} \frac{R_B^C((I-\delta[\theta_{B_i}^{B_i}]_X)R_W^{B_j}(R_{B_i}^W(R_C^B\frac{1}{\lambda_l}\hat{P}_l^{C_i} + P_C^B) + P_{B_i}^W - P_{B_j}^W))}{\delta\theta_{B_i}^{B_i}}$$

$$= R_B^C([R_W^{B_j}(R_{B_i}^W(R_C^B\frac{1}{\lambda_l}\hat{P}_l^{C_i}) + P_{B_i}^W - P_{B_j}^W)]_X)]$$

$$J[2]_{3x6} = \frac{\delta P_l^{C_j}}{\delta[p_C^B, q_C^B]} = [R_B^C(R_W^{B_j}R_{B_j}^W - I)$$

$$[R_B^C(R_W^{B_j}(R_{B_i}^W P_C^B + P_{B_i}^W - P_{B_j}^W) - P_C^B)]_X$$

$$+[R_B^C R_W^{B_j} R_{B_i}^W R_C^B \frac{1}{\lambda_l}\hat{P}_l^{C_i}]_X$$

$$-R_B^C R_W^{B_j} R_{B_i}^W R_C^B[\frac{1}{\lambda_l}\hat{P}_l^{C_i}]_X]$$

$$\frac{\delta P_l^{C_j}}{\delta q_C^B} = \lim_{\delta\theta_{C'}^C\to 0} \frac{exp(\delta\theta_{C'}^C)R_B^C(R_W^{B_j}(R_{B_i}^W(R_C^B exp(\delta\theta_{C'}^C)\frac{1}{\lambda_l}\hat{P}_l^{C_i} + P_C^B) + P_{B_i}^W - P_{B_j}^W) - P_C^B) - P_l^{C_j}}{\delta\theta_{C'}^C}$$

$$= \lim_{\delta\theta_{C'}^C\to 0} \frac{(I-\delta[\theta_{C'}^C]_X)R_B^C(R_W^{B_j}R_{B_i}^W R_C^B(I-\delta[\theta_{C'}^C]_X)\frac{1}{\lambda_l}\hat{P}_l^{C_i} + R_W^{B_j}R_{B_i}^W P_C^B + R_W^{B_j}(P_{B_i}^W - P_{B_j}^W) - P_C^B)}{\delta\theta_{C'}^C}$$

$$= [R_B^C(R_W^{B_j}R_{B_i}^W P_C^B + R_W^{B_j}(P_{B_i}^W - P_{B_j}^W) - P_C^B)]_X$$

$$+[R_B^C R_W^{B_j} R_{B_i}^W R_C^B \frac{1}{\lambda_l}\hat{P}_l^{C_i}]_X - R_B^C R_W^{B_j} R_{B_i}^W R_C^B[\frac{1}{\lambda_l}\hat{P}_l^{C_i}]_X]$$

$$J[3]_{3x1} = \frac{\delta P_l^{C_j}}{\delta\lambda_l} = [\frac{1}{\lambda_l^2}R_B^C R_W^{B_j} R_{B_i}^W R_C^B \hat{P}_l^{C_i}]$$

Lean real-time Optimization at camera rate

Lightweight motion-only visual-inertial bundle adjustment to boost the state estimation to camera rate. Same cost function, but only optimize the poses and velocities of a fixed number of latest IMU states resulting in 5ms vs 50ms.

IMU Propagation

Propagate directly the latest VIO estimate with the set of most recent IMU measurements to achieve IMU rate performance.

Failure Detection and Recovery

Failure due to violent illumination changes or severely aggressive motion is unavoidable. Active failure detection and recovery improves the practicability of the system. Failure detection is an independent module that detects unusual output from the estimator: number of features being tracked in the latest frame is less than a certain threshold, large discontinuity in position or rotation between last two estimator outputs, or large change in bias or extrinsic parameter estimation. Once a failure is detected, the system switches back to the initialization phase. Once successfully initialized, a new and separate segment of the pose graph is created.

Global State Estimator

Pose Graph Optimization

Visual-Inertial Odometry (VIO) is accumulating drift in the global 3D position (x,y,z) and the rotation around the gravity direction (yaw). A **4-DOF pose graph optimization** provides **globally consistent poses**.

Pose Graph

Vertex

When a **keyframe** is marginalized out from the sliding window in the VIO optimization, it is added to the pose graph. A keyframe becomes a **vertex** in the pose graph, which connects to other vertices via **edges**.

Sequential edge

A keyframe establishes several sequential edges to previous keyframes. A sequential edge represents the **relative pose transformation between two keyframes** provided by the sliding window from the VIO optimization.

Newly marginalized keyframe i and previous keyframe j

$p^i_{i,j} = R_i^{W^{-1}}(p_j^W - p_i^W)$ Relative position

$\psi_{ij} = \psi_j - \psi_i$ Relative yaw angle

Loop closure edge

If the newly marginalized keyframe has a loop closure, a corresponding **relative transformation** (same as in the sequential edge) is added as an edge **between the keyframe and the loop closure frame** provided by the relocalization.

Optimization

$$\min_{p,\psi}\{\sum_{(i,j)\in S}||r_{i,j}||^2+\sum_{(i,j)\in L}p(||r_{i,j}||^2)\}$$

$X=[p_1^W,\psi_1,\cdots p_n^W,\psi_n]$	**State vector**
S	Set of all sequential edges
L	Set of all loop closures
$p(s)$	Huber norm

The Huber norm is used to reduce the impact of any possible wrong loop closures.

The **optimization** is implemented using the **Ceres** library.

Relative Pose Transformation Model

$$r_{i,j}(p_i^W,\psi_i,p_j^W,\psi_j)=\begin{bmatrix}R(\phi_i,\theta_i,\psi_i)^{-1}(p_j^W-p_i^W)-p_{ij}^i\\ \psi_j-\psi_i-\psi_{ij}\end{bmatrix}$$

ϕ_i,θ_i	Roll and pitch angles from VIO

Relocalization

Visual-Inertial Odometry (VIO) is accumulating drift in the global 3D position (x,y,z) and the rotation around the gravity direction (yaw). Relocation provides **loop closures tightly-coupled with the VIO optimization** to reduce the drift.

Loop Detection

Loop detection identifies places that have already been visited. Loop detection is based on **DBoW2**, a **bag-of-words place recognition** approach, described in Galvez-Lopez et al 2012.

The previously detected **corners** are described by the **BRIEF descriptor**. As roll and pitch are observable, there is no need to rely on rotation-invariant descriptors. An extra 500 corners are detected for the detection of loops to achieve a higher recall rate. The descriptors of an image are converted to **visual words** using a pre-trained vocabulary tree and represented as a word-of-bags vector **to query the visual database**. The visual database contains all the BRIEF descriptors for the past **keyframes**, while the raw images are discarded to reduce memory. Loop closure candidates have to pass **temporal** and **geometrical consistency checks**.

Feature correspondences are found by **BRIEF descriptor matching** plus **outlier detections**: 1) 2D-2D **fundamental matrix test with RANSAC**, and 2) 3D (from sliding window) -2D **PnP test with RANSAC**. The number of inliers are used as thresholds.

Bag-Of-Words

Image database composed of a **hierarchical bag of words** with **direct** and **inverse indices**.

A hierarchical **visual vocabulary** tree is created offline from training images. A rich set of **FAST keypoints with BRIEF descriptors** are extracted from the training images. The descriptors are discretized into k **binary clusters** by performing **k-medians clustering**. The clusters form the first **level of nodes** in the **vocabulary tree**. Subsequent levels are created by repeating the operation with the descriptors associated to each node, up to n times. The w leaves of the tree represent the words of the vocabulary. Each word is given a weight according to its relevance in the training images, basically the inverse of its frequency. Frequent words get a lower weight as they are less discriminative.

FAST (Features from Accelerated Segment Test)

FAST, described in Rosten and Drummond 2006, detects corner-like points by **comparing the intensity of some pixels** in a **Bresenham circle of radius 3 (16 pixels) to the center point**.

A **decision tree** is learned to classify points, based on the pixels in the circle, as corners. A segment test that identifies corners, by identifying a continuous arc with a certain number of pixels that have a higher or lower intensity than the center point, is used for training. Non-maximal suppression (removing adjacent corners) is implemented using a score function that is based on the sum of absolute differences between the pixels in the contiguous arc and the center point.

BRIEF (Binary Robust Independent Elementary Features)

BRIEF, described in Calonder et al 2010, provides a **binary vector** where each bit is the result of an **intensity comparison** between a **given pair of pixels in a patch around the keypoint**.

The intensity patches are smoothed with a Gaussian to reduce noise. The pairs of pixels are selected **randomly** offline. The BRIEF descriptor is parametrized by the size of the patch and the number of comparisons.

$B^i(p) = \begin{cases} 1, if I(p+a_i) < I(p+b_i) \\ 0, otherwise \end{cases} \quad \forall i \in [1...b]$	i_th bit
p	Position of keypoint
$a_i, b_i \in [-\frac{s}{2}, \frac{s}{2}]$	2D offset of the i-th test
s	Patch size
b	Number of comparisons = bits
$a_i, b_i \sim N(0, \frac{1}{25}s^2)$	Random sampling

The **Hamming distance** between two descriptors can be calculated with a **xor** operation.

The **BRIEF descriptors** are used with a patch size of $s = 48$, $b = 256$ number of bits and the following random sampling $a_i \sim N(0, \frac{1}{25}s^2)$ and $b_i \sim N(a_i, \frac{4}{625}s^2)$, which places the pairs close together.

An image is converted into a **bag-of-words vector**, basically a histogram of all words found in the image. The binary descriptors from the image traverse the tree from the root to the leaves, by selecting at each level the intermediate nodes that minimize the Hamming distance.

Inverse index

For each **word in the vocabulary**, the list of **images**, where the word can be found, with the **corresponding weights of the word in those images**, are stored. The indirect index is updated when a new image is added to the database. The inverse index allows to **limit the image search** to images that have some words in common.

Direct index

For each **image** we store the **nodes** from the **vocabulary tree**, for the different levels, that are **ancestors of the words in the image** with all their associated local features. The direct index with the vocabulary tree is used as a means to approximate nearest neighbors in the BRIEF descriptor space. The direct index is updated when a new image is added to the database. The direct index allows to **speed up the geometrical verification** by computing correspondences only between those features that belong to the same words, or to words with common ancestors at level l.

Similarity between two bag-of-words vectors

$s(v_1, v_2) = 1 - \frac{1}{2}|\frac{v_1}{|v_1|} - \frac{v_2}{|v_2|}|$ L_1-score $[0, ...1]$

$v_1, v_2 \in \mathbb{R}^w$	Bag-of-words vectors

Database query

An **image** is converted to a **bag-of-words vector**. The database query provides a **list of matching candidates** and associated **scores**.

$\eta(v, v_i) = \dfrac{s(v, v_i)}{s(v, v + \Delta t)}$	**Normalize scores** by the best score we expect to obtain
$s(v, v + \Delta t)$	Bag-of-word vector from previous image

Skip images where the score to the previous image (best score we expect to obtain) is too low or the images do not have enough features in common. Keep matching candidates that are above a certain **threshold**.

Match grouping

To prevent **images close in time** to compete, they are **grouped into islands** and treated as a **single match**. Island scores are calculated as the sum of the scores from the matching candidates, thereby favoring long sequences. The island with the highest score is selected.

Temporal consistency

The matching island needs to be **consistent with i previous queries**, such that the **intervals** of the matching islands are **close to operlap**. The matching candidate with the highest score in the island is selected.

Geometrical consistency

The geometrical consistency between two matching images is tested using a **fundamental matrix test with RANSAC** that yields at least 12 inliers.

The calculation of the feature correspondences between the two images can be accelerated using the direct index. The comparison of two features can be limited to the features that are associated to the same nodes at a given level l in the vocabulary tree. With $l = 0$ only features from the same word are compared, resulting in highest speedup but fewer correspondences.

Tightly-coupled VIO Integration

The relocalization aligns the current sliding window from the VIO optimization to the past poses.

Feature correspondences are established **between loop closure frames** and the **current keyframe** and then tightly integrated into the **VIO optimization**. Multiple observations of multiple features are directly used for relocalization.

The loop closures are added as another term to the Visual-Inertial Bundle Adjustment's optimization problem.

Loop Closure Model

The loop closure model is the same as the **vision model**, with the only difference that the pose of the loop closure frame is treated as a constant. The loop closure frame is taken from the pose graph.

$$\sum_{l,v} p(||r_C(\hat{z}_l^{C_v}, X, \hat{q}_v^W, \hat{p}_v^W)||^2_{P_l^{C_v}})$$

$\hat{q}_v^W, \hat{p}_v^W$	Pose of loop closure frame
v	Loop closure frame

Pose-Graph Optimization

A keyframe is added into the pose-graph when it is marginalized out from the VIO sliding window. If there is a loop between this keyframe and any other past keyframe, the **loop closure constraints**, formulated as **4 DOF relative rigid body transforms**, are also added to the pose-graph.

Annex

Ordinary Differential Equations (ODE)

[Wikipedia]

$$\dot{x}(t) = f(t, x(t))$$
$$x(t_0) = x_0$$

Any **ordinary differential equation (ODE) of order N** can be represented as a **system of first-order ODEs**.

$$x^{(N)}(t) = f(t, x(t), \dot{x}(t), \cdots, x^{(N-1)}(t))$$
$$z_1(t) = x(t), z_2(t) = \dot{x}(t), \cdots, z_N(t) = x^{(N-1)}(t)$$

$$\dot{z}(t) = \begin{bmatrix} \dot{x}(t) \\ \vdots \\ x^{N-1}(t) \\ x^{N}(t) \end{bmatrix} = \begin{bmatrix} \dot{z}_1(t) \\ \vdots \\ \dot{z}_{N-1}(t) \\ \dot{z}_N(t) \end{bmatrix} = \begin{bmatrix} z_2(t) \\ \vdots \\ z_N(t) \\ f(t, z_1(t), \cdots, z_N(t)) \end{bmatrix}$$

Numerical Integration

Runge-Kutta (RK) Numerical Integration

[Wikipedia]

Integration of nonlinear first-order differential equations over a time interval Δt in order to convert them to a **differences equation**.

$$\dot{x} = f(t, x(t))$$
$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(\tau, x(\tau))d\tau$$

Euler method

[Wikipedia]

The Euler method assumes that the **derivative** $f(.)$ is **constant over the interval**.

$$x_{n+1} = x_n + \Delta t \cdot f(t_n, x_n)$$

The local error (error per step) is proportional to the square of the time interval and the global error (error at a given time) is proportional to the time interval.

Midpoint method

[Wikipedia]

The Midpoint method assumes that the **derivative** $f(\cdot)$ is the one **at the midpoint of the interval**.

1. Use Euler method to integrate until the midpoint

$$x(t_n + \frac{1}{2}\Delta t) = x_n + \frac{1}{2}\Delta t \cdot f(t_n, x_n)$$

2. Use this value to evaluate the derivative at the midpoint

$$x_{n+1} = x_n + \Delta t \cdot f(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}\Delta t \cdot f(t_n, x_n))$$

The local error (error per step) is proportional to the cube of the time interval and the global error (error at a given time) is proportional to the square of the time interval. While computational more expensive than Euler's method, the midpoint method's error generally decreases faster as $\Delta t \to 0$.

Hamilton Quaternions

[Wikipedia]

A comprehensive overview of quaternions and rotation in 3D with their proper use in estimation can be found in Sola 2017.

Quaternions are used to represent rotations in three dimensions. Compared to Euler angles, quaternions do not have the problem of a gimbal lock.

Hamilton Quaternions

$$q = q_w + q_x i + q_y j + q_z k = q_w + q_v = \begin{bmatrix} q_w \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ q_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \in \mathbb{H}$$

$$i^2 = j^2 = k^2 = ijk = -1$$

Eigen, ROS, Google Ceres use Hamilton quaternions. Eigen used [x,y,z,w], a format equivalent to the homogeneous vector in the projective 3Dspace.

Sums

$$p \pm q = \begin{bmatrix} p_w \pm q_w \\ p_x \pm q_x \\ p_y \pm q_y \\ p_z \pm q_z \end{bmatrix}$$

$p + q = q + p$ commutative

$p + (q + r) = (p + q) + r$ associative

Products

$$p \otimes q = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix} = \begin{bmatrix} p_w q_w - p_v^T q_v \\ p_w q_v + q_w p_v + p_v q_v \end{bmatrix}$$

$p \otimes q \neq q \otimes p$ not commutative

$p \otimes (q \otimes r) = (p \otimes q) \otimes r$ associative

$p \otimes (q + r) = p \otimes q + p \otimes r$ distributive

$(p + q) \otimes r = p \otimes r + q \otimes r$

$p \otimes q = [p]_L \cdot q = [q]_R \cdot p$ matrix product

$$[q]_L = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & -q_z & q_y \\ q_y & q_z & q_w & -q_x \\ q_z & -q_y & q_x & q_w \end{bmatrix} \qquad [q]_R = \begin{bmatrix} q_w & -q_x & -q_y & -q_z \\ q_x & q_w & q_z & -q_y \\ q_y & -q_z & q_w & q_x \\ q_z & q_y & -q_x & q_w \end{bmatrix}$$

$$[q]_L = q_w I + \begin{bmatrix} 0 & -q_v^T \\ q_v & [q_v]_X \end{bmatrix} \qquad [q]_R = q_w I + \begin{bmatrix} 0 & -q_v^T \\ q_v & -[q_v]_X \end{bmatrix}$$

$v \times w = |v|_\times w = -|w|_\times v$ cross-product matrix

$$[v]_X = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$ skew operator

Identity

$$q_1 = 1 = \begin{bmatrix} 1 \\ 0_v \end{bmatrix}$$

Conjugate

$q* = q_w - q_v = \begin{bmatrix} q_w \\ -q_v \end{bmatrix}$

$(p \otimes q)^* = q^* \otimes p^*$ distributive

Norm

$||q|| = \sqrt{q \otimes q*} = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2}$

$||p \otimes q|| = ||q \otimes p|| = ||p||||q||$

Inverse

$q \otimes q^{-1} = q^{-1} \otimes q = q_1$

$q^{-1} = q^*/||q||^2$

Unit Quaternions

$||q|| = 1$

$q^{-1} = q*$ inverse = conjugate

$q = \begin{bmatrix} cos\theta \\ u \cdot sin\theta \end{bmatrix}$

$u = u_x i + u_y j + u_z k$ unit vector

Pure Quaternions

$q = \begin{bmatrix} 0 \\ q_v \end{bmatrix}$

$q = -q^*$ equal to minus its conjugate

Rotation (unit quaternion)

$x' = q \otimes x \otimes q*$ double product

$x = xi + yj + zk = \begin{bmatrix} 0 \\ x \end{bmatrix}$ pure quaternion

q unit quaternion

$q \otimes q* = 1$ constraint

$q_A^C = q_A^B \otimes q_B^C$ composition, not commutative

Rotation matrix

$$R(q) = \begin{bmatrix} 1-2q_z^2-2q_y^2 & -2q_zq_w+2q_yq_x & 2q_yq_w+2q_zq_x \\ 2q_xq_y+2q_wq_z & 1-2q_z^2-2q_x^2 & 2q_zq_y-2q_xq_w \\ 2q_xq_z-2q_wq_y & 2q_yq_z+2q_wq_x & 1-2q_y^2-2q_x^2 \end{bmatrix}$$

Double Cover on the Manifold of SO(3)

SO(3) is the group of rotations around the origin under the operation of composition.
The angle Θ between a quaternion q and the identity quaternion $q_1 = [1,0,0,0]$ in 4D space is half the angle Φ rotated by the quaternion in 3D space.

Ordinary Differential Equation (rotation)

$q \otimes q* = 1$ rotation constraint

$\dfrac{d(q^* \otimes q)}{dt} = \dot{q}^* \otimes q + q^* \otimes \dot{q} = 0$ differentiation product rule + derivative of 1 = 0

$q^* \otimes \dot{q} = -(\dot{q}^* \otimes q) = -(q^* \otimes \dot{q})^*$ first term to other side + conjugate distributive property

$q^* \otimes \dot{q} = \Omega = \begin{bmatrix} 0 \\ \Omega \end{bmatrix}$ a quaternion equal to minus its conjugate means that it is a pure quaternion

$\dot{q} = q \otimes \Omega$ left multiply by q + rotation constraint

$w = 2\Omega$ vector of angular velocities,
factor 2 due to double cover on the manifold of SO(3)

$\dot{q} = \dfrac{1}{2} q \otimes w$

Lie Group and Algebra

[Wikipedia]

A focused overview of the Lie theory as it applies to state estimation in robotics can be found in Sola 2019.

Lie Group

Manifold M = smooth (=differentiable) curved (hyper-) surface

Group = a set on a manifold M with a **composition operation** $\circ$

$X \circ Y \in M$ **Closure**

$\varepsilon \circ X = X \circ \varepsilon = X$	**Identity** ε
$X^{-1} \circ X = X \circ X^{-1} = \varepsilon$	**Inverse** X^{-1} = Group Constraint
$(X \circ Y) \circ Z = X \circ (Y \circ Z)$	**Associativity**
$X, Y, Z \in M$	

Group Actions: transforming elements of another set V

$\cdot : M \times V \to V : (X, v) \to X \cdot v$	Action of $X \in M$ on $v \in V$
$\varepsilon \cdot v = v$	**Identity**
$(X \circ Y) \cdot v = X \cdot (Y \cdot v)$	**Compatibility**

Lie Algebra

$T_X M$	**Tangent space** to M at X = **vector space** $\mathbb{R}^m$
m	Dimension of M

The **manifold** in a Lie group **looks the same at every point** (e.g. sphere), therefore all **tangent spaces at any point have the same structure**.

The **structure** of the Lie algebra can be found by **time-differentiating the group constraint**.

$X^{-1} \circ X = X \circ X^{-1} = \varepsilon$	**Inverse** X^{-1} (for multiplicative groups)
$X^{-1} \circ \dot{X} + \dot{X}^{-1} \circ X = 0$	**Time derivative** at X
$v^\wedge = X^{-1} \circ \dot{X} = -\dot{X}^{-1} \circ X$	Elements of the Lie algebra
$\dot{X} = X v^\wedge \in T_X M$	**Tangent space** at X

Lie algebra $T_\varepsilon M$ is the **tangent space** to the **Lie group's manifold** M **at the identity** ε

$\dot{X} = v^\wedge \in T_\varepsilon M$	**Lie algebra**

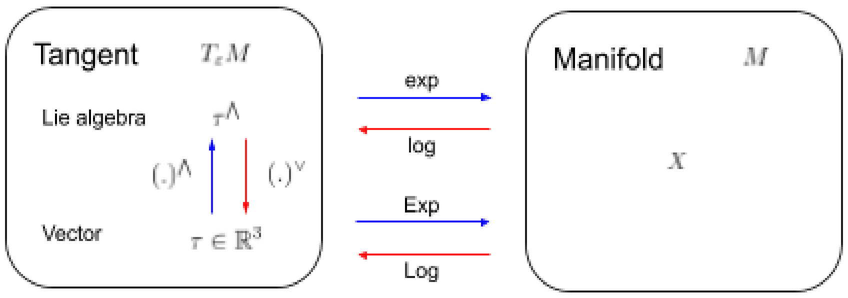
The elements of the Lie algebra can be expressed as **linear combination of base elements** E_i.

$\tau = vt \in \mathbb{R}^m$	**Vector**
$\mathbb{R}^m \to T_\varepsilon M : \tau \to \tau^\wedge = \sum_{i=1}^{m} \tau_i E_i$	**Hat**
$T_\varepsilon M \to \mathbb{R}^m : \tau^\wedge \to (\tau^\wedge)^\vee = \tau = \sum_{i=1}^{m} \tau_i e_i$	**Vee**

(Nearly) all operations in the Lie group, which is curved and nonlinear, have an exact equivalent in the Lie algebra, which is a linear vector space.

Mappings $exp()$ and $log()$ warp and unwarp elements of the **Lie algebra to/from** elements of the **Lie group**. The **exponential map** maps a **straight path** $v^\wedge t$ through the origin on the Lie algebra to a path $exp(v^\wedge t)$ around the manifold, which runs along the respective **geodesic**.

$\tau^\wedge = v^\wedge t \in T_\varepsilon M$	**Point in Lie algebra**
$X \in M$	Element in Lie group
$T_\varepsilon M \to M : \tau^\wedge \to X = exp(\tau^\wedge)$	**exp**
$M \to T_\varepsilon M : X \to \tau^\wedge = log(X)$	**log**
$\mathbb{R}^m \to M : \tau \to X = Exp(\tau)$	**Exp**
$M \to \mathbb{R}^m : X \to \tau = Log(X)$	**Log**


Tangent $T_\varepsilon M$
Lie algebra $\tau^\wedge$
$(.)^\wedge$
$(.)^\vee$
Vector $\tau \in \mathbb{R}^3$
exp
log
Exp
Log
Manifold M
X

3D Rotation Group $SO(3)$

Manifold = 3D unit sphere in $\mathbb{R}^4$
Tangent Space identified by **3D rotation vectors**

Manifold	$SO(3)$ Special Orthogonal Matrices in 3D
Composition operation	
Size	9 parameters
Dimension	3 DOF

$R \in \mathbb{R}^{3x3}$	**Rotation matrix**
$R_B^W R_C^B$	**Matrix multiplication**
$R^T R = I$	**Group constraint**
I	Identity matrix
$R^{-1} = R^T$	Inverse = Transpose
$R^T R = I$	
$R^T \dot{R} + \dot{R}^R R = 0$	**Time derivative**
$R^T \dot{R} = -(R^T \dot{R})^T$	Negative of transposed matrix
$R^T \dot{R} = [\omega]_X$	Skew-symmetric matrix

$\dot{R} = R[\omega]_X \in T_R SO(3)$ **Tangent space**

$\dot{R} = [\omega]_X \in T_\varepsilon SO(3)$ **Lie algebra**

$R = I$

$[\omega]_X = \omega_x E_x + \omega_y E_y + \omega_z E_z$

$$E_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Base elements / generators

$$E_y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

$$E_z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$\omega \in \mathbb{R}^3$ **Vector of rotation rate**

$\mathbb{R}^3 \to SO(3) : \omega \to \omega^\wedge = [\omega]_X$ **Hat**

$SO(3) \to \mathbb{R}^3 : [\omega]_X \to [\omega]_X^\vee = \omega$ **Vee**

$\dot{R} = R[\omega]_X \in T_R SO(3)$ **Ordinary Differential Equation (ODE)**

$R(t) = R_0 exp([\omega]_X t) \in SO(3)$ **Integration**

$R_0 = I$

$\theta u = \omega t \in \mathbb{R}^3$

θ Rotation angle

$u = u_x i + u_y j + u_z k$ Unit axis

$$R = exp([\theta u]_X) = \sum_k \frac{\theta^k}{k!}([u]_X)^k \in SO(3)$$

Exponential map

$$R = exp([\theta u]_X) = I + [u]_X sin\theta + [u]_X^2(1 - cos\theta)$$

Rodrigues rotation formula

$R = Exp(\theta u) = exp([\theta u]_x) \in \mathbb{R}^{3x3}$ **Exp**

$$\theta u = Log(R) = \frac{\theta(R - R^T)^\vee}{2sin\theta} \in \mathbb{R}^3$$

Log

$$\theta = cos^{-1}(\frac{trace(R) - 1}{2})$$

Rotation Action: rotating a vector v in 3D space by an angle θ around the unit axis u

$\cdot : R^{3x3} \times V \to V : (R, v) \to R \cdot v$ Action of $R \in R^{3x3}$ on $v \in V$

$R \cdot v = R \cdot v$ Rotation as **matrix vector multiplication**

Unit Quaternions Group S^3

Manifold = 3D unit sphere in $\mathbb{R}^4$
Tangent Space identified by **3D rotation vectors** (same as 3D Rotation Group, same Jacobians)

S^3 is a double cover of $SO(3)$ with q and $-q$ representing the same rotation. The first cover has a positive real part $q_w > 0$.

Manifold	S^3
Composition operation	$\otimes$
Size	4 parameters
Dimension	3 DOF

$$q = q_w + q_x i + q_y j + q_z k = q_w + q_v = \begin{bmatrix} q_w \\ q_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \in \mathbb{H}$$

$$q = \begin{bmatrix} cos(\theta/2) \\ u \cdot sin(\theta/2) \end{bmatrix}$$ **Unit quaternion**

θ Rotation angle

$u = u_x i + u_y j + u_z k$ Unit axis

$||q|| = 1$

$$p \otimes q = \begin{bmatrix} p_w q_w - p_x q_x - p_y q_y - p_z q_z \\ p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \end{bmatrix}$$ **Quaternion multiplication**

$q^* q = 1$ **Group constraint**

$$1 = \begin{bmatrix} 1 \\ 0_v \end{bmatrix}$$ Identity

$q^{-1} = q^*$ Inverse = Conjugate

$$q^* = q_w - q_v = \begin{bmatrix} q_w \\ -q_v \end{bmatrix}$$

$q^* q = 1$

$q^* \dot{q} + \dot{q}^* q = 0$ **Time derivative**

$q^* \dot{q} = -(q^* \dot{q})^*$

$q^* \dot{q} \in \mathbb{H}_P$ Pure quaternion

$vu \in \mathbb{H}_P$

$u = u_x i + u_y j + u_z k$	Pure unit quaternion
v	Norm
i, j, k	Base elements of the Lie algebra
$\dot{q} = qvu \in T_q S^3$	**Tangent space**
$\mathbb{R}^3 \to S^3 : \Theta \to \Theta^\wedge = 2\Phi$	**Hat**
$S^3 \to \mathbb{R}^3 : \Phi \to \Phi^\vee = \Theta/2$	**Vee**

$\dot{q} = qvu \in T_q S^3$ **Ordinary Differential Equation (ODE)**

$q = q_0 exp(vtu)$ **Integration**

$q_0 = 1$

$\Phi = \Phi u = vtu \in S^3$ Elements of the Lie algebra

$$q = exp(\Phi u) = \sum \frac{\Phi^k}{k!} u^k \in S^3$$

Exponential map

$$q = exp(\Phi u) = cos(\Phi) + usin(\Phi)$$

Closed form

$$q = Exp(\theta u) = cos(\theta/2) + usin(\theta/2) \in \mathbb{H}$$

Exp (same as 3D Rotation Group)

$$\theta u = Log(q) = 2 q_v \frac{arctan(||q_v||, q_w)}{||q_v||} \in \mathbb{R}^3$$

Log

if $q_w < 0 \quad q = -q$

Rotation Action: rotating a vector v in 3D space by an angle θ around the unit axis u

$\cdot : \mathbb{H} \times V \to V : (q, v) \to q \cdot v$	Action of $q \in \mathbb{H}$ on $v \in V$
$v = v_x i + v_y j + v_z k$	Vector in 3D space
$q \cdot v = q \otimes v \otimes q*$	Rotation as **double quaternion product**

$$R(q) = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

Optimization

Singular Value Decomposition (SVD)

[Wikipedia]

$$H = U \Sigma V^T \in \mathbb{R}^{mxn}$$

$U \in \mathbb{R}^{mxm}$ Left-singular vectors, with $UU^T = I$

$u_i = U_{:,i}$	Column orthonormal eigenvectors of HH^T
$V \in \mathbb{R}^{nxn}$	Right-singular vectors, with $VV^T = I$,
$v_i = V_{:,i}$	Column orthonormal eigenvectors of H^TH
$\Sigma = diag\{\sigma_1, , \sigma_n\} \in \mathbb{R}^{mxn}$	ordered, with σ_n being the smallest value
$\sigma_i > 0$	Singular values = square roots of eigenvalues of HH^T and H^TH
$\sigma_i = 0$	V_i null-space of HH^T and H^TH
$r = rank(H)$	Number of non-zero elements in Σ
$null(H)$	Columns in V corresponding to zero elements in Σ $HV_{r+1:n} = 0$

Invert matrix

$H^{-1}H = 1$
$H = U\Sigma V^T$
$H^{-1} = V\Sigma^{-1}U^T$ with $\Sigma^{-1} = diag\{1/\sigma_1, , 1/\sigma_n\}$ with $\sigma_i > 0$

QR Factorization

[Wikipedia]

$A = QR \in \mathbb{R}^{nxn}$

$Q \in \mathbb{R}^{nxn}$	Rotation matrix $Q^TQ = 1$
$R \in \mathbb{R}^{nxn}$	Upper triangular matrix

Cholesky Factorization

[Wikipedia]

$A = LL^T \in \mathbb{R}^{nxn}$	Symmetric and positive definite matrix
$L \in \mathbb{R}^{nxn}$	Lower triangular matrix

The Cholesky factorization replaces solving a complex linear system by solving two simple linear systems using a triangular matrix.

$Ax = b$
$LL^Tx = b$

$Ly = b$	Forward substitution
$L^Tx = y$	Back substitution

Variation

$A = LDL^T \in \mathbb{R}^{nxn}$	Symmetric and positive definite matrix
$DL^T x = y$	Back substitution

Linear Homogeneous Least Squares

[Wikipedia]

$Hx = 0$	Homogeneous linear system
x	Parameters
$H \in \mathbb{R}^{mxn}$	Linear measurement model
m	Number of measurements
n	Number of parameters x

Squared Error

$$e^2 = (Hx)^2$$

Minimize the Square Error Criterion

$\hat{x} = \min_x L(x)$ subject to $||x|| = 1$

$$\begin{aligned} L(x) \quad &= e^T e \\ &= (Hx)^T (Hx) \\ &= x^T H^T H x \end{aligned}$$

Compute the partial derivative with respect to the parameters x, set to 0 and solve for extremum.

$$\frac{\delta L(x)}{\delta x}|_{x=\hat{x}} = 2H^T H \hat{x} = 0$$
$$(H^T H)\hat{x} = 0$$

Singular Value Decomposition

$$H = U \Sigma V^T \in \mathbb{R}^{mxn}$$

$(H^T H)\hat{x} = 0$	$\hat{x}$ is null-space of $H^T H$
$\hat{x} = V_n$	V_n corresponds to smallest singular value, closest to null-space

Linear Non-Homogeneous Least Squares

[Wikipedia]

Measurement Model

$z = Hx + v$

z	Measurements
x	True values for parameters
$v \sim N(0, \sigma_v^2)$	Measurement **noise**, assumption: **zero-mean Gaussian**
$H \in \mathbb{R}^{mxn}$	**Linear measurement model**
m	Number of measurements z
n	Number of parameters x

Squared Error

$z - Hx$	Measurement residual
$e^2 = (z - Hx)^2$	

Minimize the Square Error Criterion

$\hat{x} = \min_x L(x)$ subject to $||z|| \neq 0$

$$\begin{aligned} L(x) \quad &= e^T e \\ &= (z - Hx)^T (z - Hx) \\ &= z^T z - 2z^T Hx + x^T H^T Hx \end{aligned}$$

Compute the partial derivative with respect to the parameters x, set to 0 and solve for extremum.

$$\frac{\delta L(x)}{\delta x}|_{x=\hat{x}} = -2Hz + 2H^T H\hat{x} = 0$$

$\hat{x} = (H^T H)^{-1} H^T z$	$(H^T H)^{-1}$ must exist, which requires $m \geq n$ with H of maximum rank $rank(H) = min(m, n) = n$
$(H^T H)^{-1} H^T$	Pseudo-inverse
$\hat{x} = H^{-1} z$	Unique solution, when $m = n$
	No solution, when $m < n$

Singular Value Decomposition

$$H = U\Sigma V^T \in \mathbb{R}^{mxn}$$

$$\hat{x} = (H^T H)^{-1} H^T z$$
$$(H^T H)^{-1} H^T = ((V\Sigma^T U^T)(U\Sigma V^T))^{-1}(V\Sigma^T U^T) = V\Sigma^{-1}U^T$$
$$\Sigma^{-1} = diag\{\frac{1}{\sigma_1},,\frac{1}{\sigma_n}\} \in \mathbb{R}^{nxm}$$
$$\hat{x} = V\Sigma^{-1}U^T z$$

Cholesky Factorization

$$(H^T H)\hat{x} = H^T z$$
$$A\hat{x} = b$$
$$A = H^T H$$
$$b = H^T z$$

$$Ax = b$$
$$LL^T x = b$$

$Ly = b$ Forward substitution

$L^T x = y$ Back substitution

Weighted Least Squares

[Wikipedia]

Measurement Model

$$z = Hx + v$$

$v \sim N(0, R)$ Measurement noise, assumption: zero mean normal distribution

$$R = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_m^2 \end{bmatrix}$$

Independent noise terms with different variance

Weighted Squared Error

$$e^2 = \frac{(z - h(x))^2}{\sigma^2} = \frac{(z_1 - H_{1,*}x)^2}{\sigma_1^2} + \frac{(z_2 - H_{2,*}x)^2}{\sigma_2^2} \cdots$$

Minimize the Square Error Criterion

$$\hat{x} = \min_x L(x)$$

$$\begin{aligned} L(x) \quad &= e^T R^{-1} e \\ &= (z - Hx)^T R^{-1} (z - Hx) \\ &= z^T R^{-1} z - 2z^T R^{-1} Hx + x^T H^T R^{-1} Hx \end{aligned}$$

Compute the partial derivative with respect to the parameters x, set to 0 and solve for extremum.

$$\frac{\delta L(x)}{\delta x}|_{x=\hat{x}} = -2R^{-1}Hz + 2H^T R^{-1} H\hat{x} = 0$$
$$\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} z$$

Probabilistic Maximum Likelihood

[Wikipedia]

$$\hat{x} = \max_x p(z|x)$$

Same estimates as weighted least squares.

Central Limit Theorem: Sum of different **errors** tend to be **Gaussian**

Conditional measurement likelihood as Gaussian Probability Density Function

$$p(z_i|x) = N(z_i; x, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2}(\frac{z_i - x}{\sigma_i})^2}$$

z_i	measurement
x	true values for parameters is the **mean**
σ_i^2	measurement noise **variance**

$$p(z|x) = N(z_1; x, \sigma_1^2) \times \cdots \times N(z_m; x, \sigma_m^2) = \frac{1}{(\sqrt{2\pi})^m \prod_i^m \sigma_i} e^{-\frac{1}{2}\sum_i^m (\frac{z_i - x}{\sigma_i})^2}$$

Maximizing is equivalent to minimizing the negative log.

$$\hat{x} = \min_x -log(p(z|x)) = \min_x -\frac{1}{2}\sum_i^m (\frac{z_i - x}{\sigma_i})^2$$

$$\hat{x} = \min_x -\frac{1}{2}(z-x)^T\Sigma^{-1}(z-x)$$

Maximizing the log Likelihood of independent Gaussian probability distributions is equivalent to minimizing least squares.

Recursive (Online) Least Squares

[Wikipedia]

Measurement Model

$z_t = H_t x + v_t$	x is fixed
z_t	Measurements at time t
$v_t \sim N(0, R_t)$	Measurement noise at time t

$$R_t = \begin{bmatrix} \sigma_{1_t}^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_{m_t}^2 \end{bmatrix}$$

Update step at time t

$\hat{x}_t = \hat{x}_{t-1} + K_t(z_t - H_t\hat{x}_{t-1})$ **Linear recursive update**

Probabilistic Formulation

$L(x) = \mathbb{E}[(x_t - \hat{x}_t)^2] = \sigma_t^2$	**Minimizes covariance**
$\quad = Trace(P_t)$	Covariance estimator

$$P_t = (I - K_t H_t)P_{t-1}(1 - K_t H_t)^T + K_t R_t K_t^T$$

$K_t = \dfrac{P_{t-1}H_t^T}{H_t P_{t-1} H_t^T + R_t}$	**Gain matrix**
$P_t = (I - K_t H_t)P_{t-1}$	**Covariance**

Nonlinear Least Squares

[Wikipedia]

Unknown Model

$$z = h(\Theta, x) + v$$

$h(\Theta, x)$	**Nonlinear measurement model**
Θ	Model parameters
z, x	Measurements

Squared Error

$e^2 = (z - h(\Theta, x))^2$	Difference between actual and predicted measurement

Minimize the Square Error Criterion

$$\hat{\Theta} = \min_{\Theta} L(\Theta)$$

$$\begin{aligned} L(\Theta) &= e^T e \\ &= (z - h(\Theta, x))^T (z - h(\Theta, x)) \\ &= z^T z - 2z^T h(\Theta, x) + h(\Theta, x)^T h(\Theta, x) \end{aligned}$$

Unknown State

$$z = h(x) + v$$

$h(x)$	**Nonlinear measurement model**
x	Parameters
z	Measurements
$\Omega = R^{-1}$	**Information matrix**

$$\Omega = \begin{bmatrix} \frac{1}{\sigma_1^2} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sigma_m^2} \end{bmatrix}$$

Squared Error

$e^2 = (z - h(x))^2$	Difference between actual and predicted measurement

Minimize the Square Error Criterion

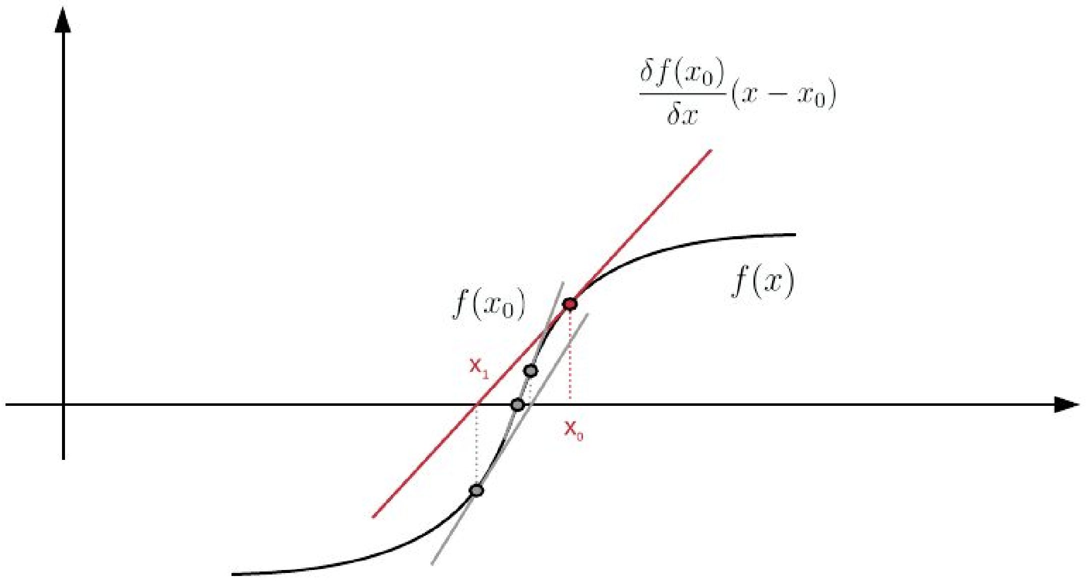
$$\hat{x} = \min_x L(x)$$

$$\begin{aligned} L(x) \quad &= e^T \Omega e \\ &= (z - h(x))^T \Omega (z - h(x)) \\ &= z^T \Omega z - 2z^T \Omega h(x) + h(x)^T \Omega h(x) \end{aligned}$$

Newton

[Wikipedia]

Newton's method determines x for $f(x) = 0$ **iteratively**, by approximating $f(x)$ by its first-order Taylor expansion and calculating x_k for $f(x_k) = 0$.



$$f(x) = 0$$

Approximation by first-order Taylor expansion

$$f(x) \approx f(x_k) + \frac{\delta f(x_k)}{\delta x}(x - x_k)$$

$$f(x_k) + \frac{\delta f(x_k)}{\delta x}(x_{k+1} - x_k) = 0$$

$$J(x_k) = \frac{\delta f(x_k)}{\delta x} = \begin{bmatrix} \frac{\delta f_1(x_k)}{\delta x_1} & \dots & \frac{\delta f_1(x_k)}{\delta x_n} \\ \vdots & & \vdots \\ \frac{\delta f_n(x_k)}{\delta x_1} & \dots & \frac{\delta f_n(x_k)}{\delta x_n} \end{bmatrix}$$

Jacobian, first-order partial derivative

$$x_{k+1} = x_k - \frac{f(x_k)}{J(x_k)}$$

Applying **Newton**'s method to **minimization and maximization problems**

$$\frac{\delta f(x)}{\delta x} = J(x) = 0$$

$$x_{k+1} = x_k - \frac{J(x_k)}{H(x_k)}$$

$$H(x_k) = \begin{bmatrix} \frac{\delta^2 f(x_k)}{\delta x_1^2} & \frac{\delta^2 f(x_k)}{\delta x_1 \delta x_2} & \dots & \frac{\delta^2 f(x_k)}{\delta x_1 \delta x_n} \\ \frac{\delta^2 f(x_k)}{\delta x_2 \delta x_1} & \frac{\delta^2 f(x_k)}{\delta x_2^2} & \dots & \frac{\delta^2 f(x_k)}{\delta x_2 \delta x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\delta^2 f(x_k)}{\delta x_n \delta x_1} & \frac{\delta^n f(x_k)}{\delta x_2^2} & \dots & \frac{\delta^2 f(x_k)}{\delta x_n^2} \end{bmatrix}$$

Hessian, second-order partial derivative

Newton's method can be applied to solve **iteratively nonlinear least squares**, but it requires the calculation of the Hessian, which may be difficult or expensive.

Gauss-Newton

[Wikipedia]

Gauss-Newton's method solves **iteratively nonlinear least squares**.

$$\hat{x} = \min_x L(x)$$

$$L(x) \quad = e^T \Omega e$$

$$= (z - h(x))^T \Omega (z - h(x))$$
$$= z^T \Omega z - 2 z^T \Omega h(x) + h(x)^T \Omega h(x)$$

Linearization by first-order Taylor expansion

$$h(x) \approx h(x_k) + \frac{\delta h(x_k)}{\delta x}(x - x_k)$$

$$L(x) = z^T \Omega z - 2 z^T \Omega (h(x_k) + \frac{\delta h(x_k)}{\delta x}(x - x_k))$$
$$+ (h(x_k) + \frac{\delta h(x_k)}{\delta x}(x - x_k))^T \Omega (h(x_k) + \frac{\delta h(x_k)}{\delta x}(x - x_k))$$

Compute the partial derivative with respect to the parameters x, set to 0 and solve for extremum.

$$\frac{\delta L(x)}{\delta x} = 2 \frac{\delta h(x_k)^T}{\delta x} \Omega (h(x_k) + \frac{\delta h(x_k)}{\delta x}(x_{k+1} - x_k)) - 2\Omega \frac{\delta h(x_k)^T}{\delta x} z = 0$$

$$2 \frac{\delta h(x_k)^T}{\delta x} \Omega (h(x_k) + \frac{\delta h(x_k)}{\delta x}(x_{k+1} - x_k)) - 2\Omega \frac{\delta h(x_k)^T}{\delta x} z = 0$$

$$x_{k+1} = x_k + \frac{H^T \Omega (z - h(x_k))}{(H^T \Omega H)}$$

$(H^T H)^{-1}$ must exist, requires $m \geq n$

$$H = \frac{\delta h(x_k)}{\delta x} = \begin{bmatrix} \frac{\delta h_1(x_k)}{\delta x_1} & \cdots & \frac{\delta h_1(x_k)}{\delta x_n} \\ \vdots & & \vdots \\ \frac{\delta h_n(x_k)}{\delta x_1} & \cdots & \frac{\delta h_n(x_k)}{\delta x_n} \end{bmatrix}$$

Jacobian, first-order partial derivative at last estimate x_k

Cholesky factorization

$$x_{k+1} = x_k + \triangle x$$
$$H^T \Omega H \triangle x = H^T \Omega (z - h(x_k))$$
$$A \triangle x = b$$
$$A = H^T \Omega H$$
$$b = H^T \Omega (z - h(x_k))$$

$A\triangle x = b$	
$LL^T\triangle x = b$	
$Ly = b$	Forward substitution
$L^T\triangle x = y$	Back substitution

The Gauss-Newton method is an approximation of Newton's method, where the term involving the Hessian is approximated by the Jacobian ignoring the second-order derivative terms. Therefore the convergence is not guaranteed and requires that the residual values are small in magnitude, at least around the minimum, and that the functions are only mildly nonlinear.

Levenberg-Marquardt

[Wikipedia]

Adding a damping factor

$$x_{k+1} = x_k + \frac{H^T\Omega(z - h(x))}{(H^T\Omega H + \lambda I)}$$

λ	**Damping factor**, adjusted at each iteration
I	Identify matrix

Cholesky factorization

$$x_{k+1} = x_k + \triangle x$$
$$(H^T\Omega H - \lambda I)\triangle x = H^T\Omega(z - h(x_k))$$
$$A\triangle x = b$$
$$A = (H^T\Omega H - \lambda I)$$
$$b = H^T\Omega(z - h(x_k))$$

$A\triangle x = b$	
$LL^T\triangle x = b$	
$Ly = b$	Forward substitution
$L^T\triangle x = y$	Back substitution

The Levenberg-Marquardt method is an approximation of Newton's method, where the term involving the Hessian is approximated by a multiple of the identity matrix. Levenberg-Marquardt is actually a combination of two other minimization methods: the gradient descent method and the Gauss-Newton method. Levenberg-Marquardt is more robust than Gauss-Newton. It can be used when the Jacobian does not have maximum rank and its pseudoinverse does not exist.

Robust Kernels

Large outliers are problematic for the assumption that the error is Gaussian.

Robust M-Estimator (instead of squared function)

Assumes non-Gaussian distributed noise, intuitively PDF with heavy tails.

$p(e) = \exp(-\rho(e))$ Kernel function

$X = \min_X \sum_i \rho(e_i(X))$ Minimize negative log likelihood

$\rho(e) = e^2$ Gaussian

$\rho(e) = |e|$ L1-norm (absolute values)

$\rho(e) = \begin{cases} \frac{e^2}{2} & if|e| < c \\ c(|e| - \frac{c}{2}) & otherwise \end{cases}$ Huber norm

Tukey, Cauchy, Blake-Zisserman, Corrupted Gaussian, …
depends on **outlier distribution**

Use weighted least squares to implement robust estimation

$X = \min_X \frac{1}{2} \sum_i w_i ||e_i(X)||^2$ Weighted least square

$$\frac{1}{2}\frac{\delta(w_i e_i^2(x))}{\delta x} = w_i e_i(x) \frac{\delta e_i(x)}{\delta x} = 0$$

$X = \min_X \sum_i \rho(e_i(X))$ Robust estimation

$$\frac{\delta(\rho(e_i(x)))}{\delta x} = \rho'(e_i(x)) \frac{\delta e_i(x)}{\delta x} = 0$$

$$w_i e_i(x) = \rho'(e_i(x))$$

$$w_i = \frac{1}{e_i(x)} \rho'(e_i(x))$$

Kernel also requires changes to Jacobian.

RANdom SAmple Consensus (RANSAC)

RANSAC is a trial-and-error approach to estimate parameters of a model from data that contains a significant number of outliers. In the process of fitting the model, all data points are evaluated and classified as either inlier or outlier, thereby providing **outlier detection**.

RANSAC iterates a certain number of times through the following steps:

1. Select random sample points (hypothetical inliers)
2. Fit a model
3. Test all points against the model
4. Count inliers
5. Determine the best set of points in terms of maximum number of inliers

RANSAC assumes fitting a single model.

The number of points sampled corresponds to the minimum number of data points required to fit the model. RANSAC works typically well up to 10 parameters.

The total number of iterations t can be calculated based on a probability threshold to be certain that at least one set of random points were sampled that are all inliers and thereby the right model was found.

$(1-o)^s$	Probability of drawing s inliers
$1-(1-o)^s$	Probability of failing to draw s inliers
$1-p=(1-(1-o)^s)^t$	Probability of failing over t iterations
$t=\dfrac{log(1-p)}{log(1-(1-o)^s)}$	

p	Probability that at least one set of random points were sampled that are all inliers
s	Number of points sampled
o	Ratio of outliers to data points

Perception

A comprehensive overview and in-depth descriptions of perception algorithms can be found in Foestner and Wrobel 2016.

Homogeneous Coordinates

Use of **homogeneous coordinates** allows to represent rotation and translation in a single matrix.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Euclidean to homogeneous coordinates

$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \end{bmatrix}$	Homogeneous to euclidean coordinates
$\begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix}$	Translation
$\begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$	Rotation
$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$	Translation and rotation = pose

Coordinate Frames

$(.)^W$	World frame (fixed)
$(.)^B$	Body frame
$(.)^C$	Camera frame
B_k	Body frame at the moment in time when the k^{th} image was taken
C_k	Camera frame at the moment in time when the k^{th} image was taken

$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$	Point
X, Y, Z	3D Euclidean coordinates
$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{22} & r_{22} & r_{23} \\ r_{33} & r_{32} & r_{33} \end{bmatrix}$	Rotation matrix (3 parameters = rotation around X, Y, Z axes) $R^T R = 1$ and $det(R) = 1$ right hand
$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$	Translation (3 parameters)
t_x, t_y, t_z	3D coordinates

World to Camera

$$P^C = R^C_W P^W + t^C_W$$

Use of **homogeneous coordinates** allows to represent rotation and translation in a single matrix and to perform a coordinate frame transformation using a matrix vector multiplication.

$$P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Point in homogeneous coordinates

$$P^C = M_W^C P^W$$

$$M_W^C = \begin{bmatrix} & R_W^C & & t_W^C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera to World

$$M_C^W = \begin{bmatrix} & {R_W^C}^T & & -{R_W^C}^T t_W^C \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Camera to Body to World

$$M_C^W = M_B^W M_C^B$$

R, C from 3 corresponding points

$P_i^C = R(P_i^W - C)$

$P_i^C = d_i p_i$	3D points in camera frame
P_i^W	3D points in world frame
R	Rotation matrix
C	Origin of camera frame in world frame

$\{v_1^W = \vec{P_1^W P_2^W}, v_1^C = \vec{P_1^C P_2^C}\}$	2 pairs of non-collinear vectors
$\{v_2^W = \vec{P_1^W P_3^W}, v_2^C = \vec{P_1^C P_3^C}\}$	
$\{v_3^W = v_1^W \times v_2^W, v_3^C = v_1^C \times v_2^C\}$	3rd pair = orthogonal vectors (right-hand)

$v_i^C = Rv_i^W$ Solve for R using the determinant

$$R = v_1^C \frac{v_2^W \times v_3^W}{|v_1^W v_2^W v_3^W|}^T + v_2^C \frac{v_1^W \times v_3^W}{|v_1^W v_2^W v_3^W|}^T + v_3^C \frac{v_1^W \times v_2^W}{|v_1^W v_2^W v_3^W|}^T$$

$|v_1^W v_2^W v_3^W|$ Determinant

Enforce orthogonal constraints on rotation matrix

$R = K^{-1}P_{1:3}$	Rotation matrix
$R = U\Sigma V^T$	Singular Value Decomposition
$\Sigma = diag(\sigma_1, \sigma_2, \sigma_3)$	
$R' = UV^T$	Singular Values = 1
$C = P_1 - R'^T P_1^C$	

Pinhole Camera Model

[Wikipedia]

$u = f\frac{X}{Z}$ and $v = f\frac{Y}{Z}$	Pinhole model
$I(u, v)$	Image
u, v	Pixel coordinates
$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$	Pixel in homogeneous coordinates
$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = LK[R\|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$	**Camera Projection** (defined up to a scale factor) *Point in world frame to pixel in image*
λ	Scale factor
$P = K[R\|t] \in \mathbb{R}^{3x4}$	Direct Linear Transform by an Affine Camera defined by a homogeneous projection matrix P (11 parameters + $scale$)
$[R\|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{22} & r_{22} & r_{23} & t_y \\ r_{33} & r_{32} & r_{33} & t_z \end{bmatrix}$	**Extrinsic Parameters** (6 parameters) *Point in world frame to point in camera frame*
t	Origin of world frame in the camera frame
$C = -R^T t$	Origin of camera frame in world frame

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Intrinsic Camera Parameters (5 parameters)
Point in camera frame to pixel in image

c_x, c_y	Optical center or principle point in image
$f_x = m_x * f, f_y = m_y * f$	Focal length in terms of pixels
$s = sl * f$	Skew factor between x and y axis (often 0)
m_x, m_y	Pixel scale factors
sl	Slant of image plane
f	Focal length

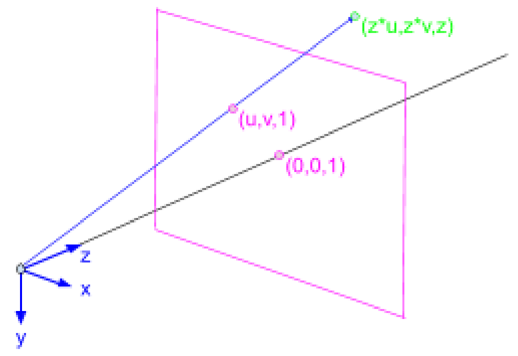
$$L(u, v) = \begin{bmatrix} 1 & 0 & \triangle u(u, v, k, p) \\ 0 & 1 & \triangle v(u, v, k, p) \\ 0 & 0 & 1 \end{bmatrix}$$

Lens Deformation Mapping
(incl. all non-linear factors
from point in world frame to pixel in image)
Pixel in image to pixel in image

$$u^d = u\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + 2p_1uv + p_2(r^2 + 2u^2)$$
$$v^d = v\frac{1 + k_1r^2 + k_2r^4 + k_3r^6}{1 + k_4r^2 + k_5r^4 + k_6r^6} + 2p_2uv + p_1(r^2 + 2v^2)$$

$k = [k_1, k_2[, k_3[k_4, k_5, k_6]]]$	**Radial Distortion** Parameters
$p = [p_1, p_2]$	**Tangential Distortion** Parameters
r	Distance of u, v from the optical center

Projective Plane



Use of **homogeneous coordinates** allows to represent relationships between the image plane and the 3D world elegantly using linear algebra.

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$ Point in the image

$$\overrightarrow{p} = \begin{bmatrix} Z * u \\ Z * v \\ Z \end{bmatrix}$$ Corresponding ray in the 3D world

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}^T \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$ Line in the image

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}^T \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0$$ Corresponding plane in the 3D world

$$l = k * \begin{bmatrix} a \\ b \\ c \end{bmatrix} = p_1 \times p_2$$ Line in the image by two points

$p = l_1 \times l_2$ Point in the image by intersection of two lines

Direct Linear Transform [linear intrinsics and extrinsics]

P_i Known points in the world frame

p_i Known pixels in the image corresponding to the points

$p_i = K[R|t]P_i$ Projection of a point to a pixel in the image

$P = K[R|t] \in \mathbb{R}^{3x4}$ Projection matrix

$p_i = PP_i$

Determine Projection Matrix P (11 parameters)

Direct solution, but not statistically optimal

$$p_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} P_i$$

requires at least n = **6 x 3D points**

2 measurements u, v per 3D point

Transform into linear system

Option 1

$$p_i = PP_i$$
$$[p_i]_x PP_i = 0$$

Cross product of a vector with itself = 0
Multiply out scale factor

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}_\times \begin{bmatrix} [p_{11} & p_{12} & p_{13} & p_{14}]P_i \\ [p_{21} & p_{22} & p_{23} & p_{24}]P_i \\ [p_{31} & p_{32} & p_{33} & p_{34}]P_i \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & -1 & v_i \\ 1 & 0 & -u_i \\ -v_i & u_i & 0 \end{bmatrix} \begin{bmatrix} P_i^T & 0_{1x4} & 0_{1x4} \\ 0_{1x4} & P_i^T & 0_{1x4} \\ 0_{1x4} & 0_{1x4} & P_i^T \end{bmatrix} \begin{bmatrix} [p_{11} & p_{12} & p_{13} & p_{14}]^T \\ [p_{21} & p_{22} & p_{23} & p_{24}]^T \\ [p_{31} & p_{32} & p_{33} & p_{34}]^T \end{bmatrix} = 0$$

$$\begin{bmatrix} 0_{1x4} & -P_i^T & v_i P_i^T \\ P_i^T & 0_{1x4} & -u_i P_i^T \\ -v_i P_i^T & u_i P_i^T & 0_{1x4} \end{bmatrix} \begin{bmatrix} [p_{11} & p_{12} & p_{13} & p_{14}]^T \\ [p_{21} & p_{22} & p_{23} & p_{24}]^T \\ [p_{31} & p_{32} & p_{33} & p_{34}]^T \end{bmatrix} = 0$$

$$h_i = \begin{bmatrix} 0 & 0 & 0 & 0 & -X_i & -Y_i & -Z_i & -1 & v_i X_i & v_i Y_i & v_i Z_i & v_i \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i X_i & -u_i Y_i & -u_i Z_i & -u_i \\ -v_i X_i & -v_i Y_i & -v_i Z_i & -v_i & u_i X_i & u_i Y_i & u_i Z_i & u_i & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$p = vec(P^T) = \begin{bmatrix} p11 \\ p12 \\ p13 \\ p14 \\ \cdots \\ p31 \\ p32 \\ p33 \\ p34 \end{bmatrix}$$

Vectorization of the matrix P

$$\begin{bmatrix} h_1 \\ \cdots \\ h_n \end{bmatrix} p = Hp = 0$$

Stacking up n measurements
h_i is only of rank 2

Option 2

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} [p_{11} & p_{12} & p_{13} & p_{14}]P_i \\ [p_{21} & p_{22} & p_{23} & p_{24}]P_i \\ [p_{31} & p_{32} & p_{33} & p_{34}]P_i \end{bmatrix}$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_i}{z_i} \\ \frac{y_i}{z_i} \\ 1 \end{bmatrix}$$

Homogeneous to Euclidean coordinates

$$u_i = \frac{[p_{11}p_{12}p_{13}p_{14}]P_i}{[p_{31}p_{32}p_{33}p_{34}]P_i}$$

$$v_i = \frac{[p_{21}p_{22}p_{23}p_{24}]P_i}{[p_{31}p_{32}p_{33}p_{34}]P_i}$$

$$-P_i^T[p_{11}p_{12}p_{13}p_{14}] + u_i P_i^T[p_{31}p_{32}p_{33}p_{34}] = 0$$

$$-P_i^T[p_{21}p_{22}p_{23}p_{24}] + v_i P_i^T[p_{31}p_{32}p_{33}p_{34}] = 0$$

$$h_i = \begin{bmatrix} -X_i & -Y_i & -Z_i & -1 & 0 & 0 & 0 & 0 & u_iX_i & u_iY_i & u_iZ_i & u_i \\ 0 & 0 & 0 & 0 & -X_i & -Y_i & -Z_i & -1 & u_iX_i & v_iY_i & v_iZ_i & v_i \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ \cdots \\ h_n \end{bmatrix} p = Hp = 0$$

Stacking up n measurements

Solve linear homogeneous least squares

$\min_p (Hp)^2$ subject to $||p|| = 1$

$$H = U\Sigma V^T$$

Singular Value Decomposition
rank(H) = 11

$$\hat{p} = V_{:12}$$

No solution, if all points on a plane, e.g Z = 0

Construct matrix P from vector $\hat{p}$

Determine K, R, t

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = K[R|t] = [KR|Kt]$$

$$KR = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

K is an upper triangular matrix
R is a rotation matrix

$A = KR$

$A^{-1} = (KR)^{-1} = R^{-1}K^{-1} = R^T K^{-1}$

$A^{-1} = QR$ **QR-Factorization**

$R^T = Q$ Rotation matrix

$R = Q^T$

$K^{-1} = R$ Upper triangular matrix

$K = \dfrac{R^T}{R^T_{33}}$ Homogeneity normalization

$$Kt = \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

$$t = K^{-1} \begin{bmatrix} p_{14} \\ p_{24} \\ p_{34} \end{bmatrix}$$

Depending on the coordinate system you want to use, the signs of the (1,1) and (2,2) elements of K and R may need to be inverted to properly deal with having placed the image plane in front of the camera.

Zhang's Method [linear intrinsics]

Checkerboard calibration target on a **plane**.

For **each image of the checkerboard**, define the world frame so that the X and Y axes define the plane and therefore all points on the checkerboard have $Z = 0$.

$$scale * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{22} & r_{22} & t_y \\ r_{33} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$ Eliminate everything related to Z

Determine Homography P (8 parameters) for each checkerboard image

$$p_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} P_i$$

requires at least n = **4 x points on checkerboard**
2 measurements u, v per point

Solve the same way as in Direct Linear Transform (except now points are on a plane)

Determine K from all checkerboard images

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = [p_1, p_2, p_3] = K[r_1, r_2, t]$$

$K^{-1}[p_1, p_2, p_3] = [r_1, r_2, t]$ — K is invertible

$r_1 = K^{-1}p_1$

$r_2 = K^{-1}p_2$

$r_1^T r_2 = 0$ — r_1, r_2, r_3 orthogonal basis

$p_1^T K^{-T} K^{-1} p_2 = 0$

$||r_1|| = ||r_2|| = 1$ — r_1, r_2, r_3 have unit length

$p_1^T K^{-T} K^{-1} p_1 = p_2^T K^{-T} K^{-1} p_2$

$p_1^T K^{-T} K^{-1} p_1 - p_2^T K^{-T} K^{-1} p_2 = 0$

$A = K^{-T} K^{-1}$ — Symmetric and positive definite matrix (6 unique parameters)

Transform into linear system

$p_1^T A p_2 = 0$

$p_1^T A p_1 - p_2^T A p_2 = 0$

$a = vec(A^T)$ — Vectorization of the matrix A, but only of the 6 unique parameters

$h_{12}^T a = 0$

$h_{11}^T a - h_{22}^T a = 0$

$$h_{ij} = \begin{bmatrix} p_{1i}p_{1j} \\ p_{1i}p_{2j} + p_{2i}p_{1j} \\ p_{3i}p_{1j} + p_{1i}p_{3j} \\ p_{2i}p_{2j} \\ p_{3i}p_{2j} + p_{2i}p_{3j} \\ p_{3i}p_{3j} \end{bmatrix}$$

$$\begin{bmatrix} h_{12}^T \\ h_{11}^T - h_{22}^T \\ \cdots \\ h_{12}^T \\ h_{11}^T - h_{22}^T \end{bmatrix} a = Ha = 0$$

Stacking up **at least 3 checkerboard images**

Solve linear homogeneous least squares

$\min_a (Ha)^2$ subject to $||a|| = 1$

$H = U\Sigma V^T$
$\hat{a} = V_{:6}$

Singular Value Decomposition

Construct matrix A from vector $\hat{a}$

$A = LL^T \in \mathbb{R}^{nxn}$
$L \in \mathbb{R}^{nxn}$
$L = K^{-T}$
$K = L^{-T}$

Cholesky Factorization
Lower triangular matrix

Lens Deformation Mapping [+ nonlinear intrinsics]

$p_i = LK[R|t]P_i$ — **Camera Projection**
$h(k, p, K, R, t, P_i) = LK[R|t]P_i$ — **Unknown nonlinear measurement model**
$\Theta = [k, p, K, R, t]$ — Model parameters
p_i, P_i — Measurements

$$\min_{k,p,K,R_n,t_n} \sum_{I_k}^{images} \sum_{i}^{points} (p_{i_{I_k}} - h(k, p, K, R_{I_k}, t_{I_k}, P_{i_{I_k}}))^2$$

Solve as Nonlinear Least Squares using Levenberg-Marquardt
initialized with Zhang’s method and nonlinear parameters = 0.

Undistorted Points = Projection Vectors

The inverse projection for a pinhole camera model from points in the image to projection vectors in the camera frame is calculated by the inverse lens deformation mapping followed by the inverse intrinsic camera projection.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = scale * K^{-1} L^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

The **inverse lens deformation mapping** is calculated iteratively, because $L(u, v)$ is unknown when u, v is unknown.

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = (L(u, v)))^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Initial guess

$$\begin{bmatrix} u_{k+1} \\ v_{k+1} \\ 1 \end{bmatrix} = (L(u_k, v_k))^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Inverse intrinsic camera projection

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

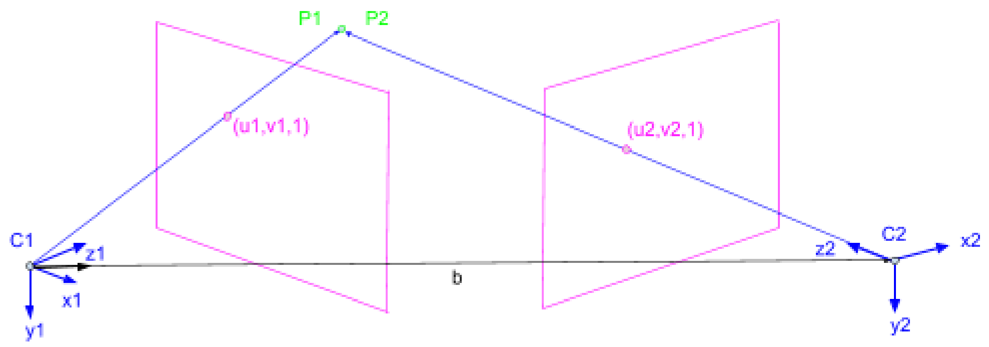
Normalized to $Z = 1$

$$\begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \\ 1 \end{bmatrix}$$

Essential Matrix (E)

[Wikipedia]

$$E = [b]_x R \in \mathbb{R}^{3x3}$$



$C_1 = K_1[I_{3x3} \quad 0_3]$	Camera 1 Pose (origin)
$C_2 = K_2[R \quad t]$	Camera 2 Pose (from origin)
$b = t$	Baseline between two camera poses *only known up to a scale factor*
P_1	3D Point from Camera 1
$P_2 = RP_1 + b$	Same 3D Point from Camera 2
$P_2 - b = RP_1$	Vector in Epipolar Plane
$b \times P_2 = [b]_x P_2$	Normal to Epipolar Plane
$[.]_x$	Matrix representation of a cross product

$$[b]_x = \begin{bmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{bmatrix}$$

$$\begin{aligned} 0 &= (P_2 - b)^T [b]_x P_2 = (RP_1)^T [b]_x P_2 \\ &= P_1^T R^T [b]_x P_2 \\ &= -P_2^T [b]_x R P_1 \\ &= P_2^T E P_1 \end{aligned}$$

Essential matrix (epipolar line constraint)

Fundamental Matrix (F)

[Wikipedia]

$$F = K_2^{-T} E K_1^{-1} \in \mathbb{R}^{3x3}$$

$p_1 = K_1 P_1$	Pixel in Image 1 corresponding to 3D Point
$p_2 = K_2 P_2$	Pixel in Image 2 corresponding to 3D Point
$0 = P_2^T E P_1$	Essential Matrix
$= p_2^T K_2^{-T} E K_1^{-1} p_1$	
$= p_2^T F p_1$	**Fundamental matrix** (epipolar line constraint)

$p_2^T F$	**Epipolar Line** in Image 1 Possible projections of a pixel from Image 2
$Fe_1 = 0$	**Epipole** in Image 1 Projection of optical center from Camera 2

Essential Matrix using the 5-Point Algorithm

The calculation of the essential matrix E from $p_2^T K_2^{-T} E K_1^{-1} p_1 = 0$ with a minimum of 5-points (minimal case), described in Li and Hartley 2006 provides a simpler and less heuristic algorithm than the state-of-the-art-work by Nister 2004.

$p_2^T K_2^{-T} E K_1^{-1} p_1 = 0$	**Epipolar line constraint**
$p_1,\ p_2$	Corresponding pixels (projections of the same point)
$q_2^T E q_1 = 0$	
$q_1 = K_1^{-1} p_1$	Assuming calibrated cameras
$q_2^T = p_2^T K_2^{-T}$	

Transform epipolar line constraint as linear homogeneous system

$$Qe = 0$$

$e = vec(E^T)$ Vectorization of the matrix E
9 unknown parameters

$$Q = \begin{bmatrix} q_{1_x} q_{2_x} & q_{1_y} q_{2_x} & q_{1_z} q_{2_x} & q_{1_x} q_{2_y} & q_{1_y} q_{2_y} & q_{1_z} q_{2_y} & q_{1_x} q_{2_z} & q_{1_y} q_{2_z} & q_{1_z} q_{2_z} \\ \dots & & & & & & & & \end{bmatrix}$$

Stacking 5 points
Rank of Q is typically 5

Solve linear homogeneous system

$Q = U \Sigma V^T$	Singular Value Decomposition
$V_{:6}, V_{:7}, V_{:8}, V_{:9}$	Bases of the null space

$vec(E_1) = V_{:6},\ vec(E_2) = V_{:7},\ vec(E_3) = V_{:8},\ vec(E_4) = V_{:9}$

$E = xE_1 + yE_2 + zE_3 + wE_4$	E parametrized by x, y, z
$w = 1$	E is a homogeneous matrix only defined up to a scale factor
$det(E) = 0$	**Cubic singularity condition**

$2EE^TE - trace(EE^T)E = 0$ **Cubic constraints**
9 constraints, one for each matrix element

Insert E parametrized by x, y, z into the cubic singularity condition and cubic constraints and transform as linear homogeneous system

$Mp = 0$
$M \in \mathbb{R}^{10x20}$ Coefficient matrix
$p = [x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3, x^2, xy, xz, y^2, yz, z^2, x, y, z, 1]^T$

Hidden Variable Technique
A technique to eliminate variables from a multivariate polynomial equation system

$C(z)p(x, y) = 0$ Consider z a hidden variable
$C(z) \in \mathbb{R}^{10x10}$ Coefficient matrix
$p(x, y) = [x3, y3, x2y, xy2, x2, y2, xy, x, y, 1]^T$

Solve 10th degree polynomial for z using the **companion matrix method**

$det(C(z)) = 0$ Hidden variable resultant

Calculate x and y as the null space of $C(z)$

Up to 10 possible essential matrices E

Choose essential matrix E by checking with the 5 points or more than 5 points when used with RANSAC using **point triangulation**.

Advantages over the 8-point algorithm:
- Higher computational efficiency with RANSAC
- Higher accuracy
- Works with planar scene, but not if base line is perpendicular to plane (e.g. landing)

Fundamental Matrix using the 8-Point Algorithm

[Wikipedia]

Calculate fundamental matrix F from $p_2^TFp_1 = 0$ with a minimum of 8-points

$p_2^TFp_1 = 0$ Fundamental Matrix

$F \in \mathbb{R}^{3x3}$ but $rank(F) = 2$

Transform into linear system

$$[u_i^2 v_i^2 1] \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_i^1 \\ v_i^1 \\ 1 \end{bmatrix} = 0$$

$$u_i^1 u_i^2 f_{11} + u_i^1 v_i^2 f_{21} + u_i^1 f_{31} + v_i^1 u_i^2 f_{12} + v_i^1 v_i^2 f_{22} + v_i^1 f_{32} + u_i^2 f_{13} + v_i^2 f_{23} + f_{33} = 0$$

$$\begin{bmatrix} u_1^1 u_1^2 & u_1^1 v_1^2 & u_1 & v_1^1 u_1^2 & v_1^1 v_1^2 & v_1^1 & u_1^2 & v_1^2 & 1 \\ \dots \\ u_8^1 u_8^2 & u_8^1 v_8^2 & u_8^1 & v_8^1 u_1^2 & v_8^1 v_1^2 & v_8^1 & u_8^2 & v_8^2 & 1 \\ \dots \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

$Hf = 0$

Solve linear homogeneous least squares

$\min_x (Hf)^2$ subject to $||f|| = 1$

$H = U \Sigma V^T$ **Singular Value Decomposition**

$\hat{f} = V_{:8}$

Constructing F from $\hat{f}$

Applying constraint $rank(F) = 2$

$F = U \Sigma V^T$ **Singular Value Decomposition**

Σ' $rank(F) = 2$ enforced by setting last element to 0

$F = U \Sigma' V^T$ Reconstruct F from Singular Value Decomposition

Relative Camera Pose from the Essential Matrix

$F = K_2^{-T} E K_1^{-1}$ Fundamental Matrix

$E = K_2^T F K_1$ Essential Matrix

$E = [b]_x R$

$$[R|b] \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = b$$

Epipole in Image 2 by projecting the optical center

$b^T E = 0$ Epipole is the nullspace of E

$$E = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

Singular Value Decomposition

$U = [u_1 u_2 u_3]$ with u_3 left nullspace of E

$b = u_3$ or $b = -u_3$ 2 solutions as scale factor is unknown

Hartely 1992 shows how to factorize a matrix E that is the product of an orthogonal matrix R and a skew-symmetric matrix $[b]_x$ using the singular value decomposition of that matrix.

$$\begin{aligned} E = [b]_x R &= (U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} U^T)(UYV^T) \\ &= U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} YV^T \end{aligned}$$

$$\Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} Y$$

$$Y = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} or \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$R = U \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \quad \text{or} \quad R = U \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

If $det(R) = -1$ then $b = -b$ and $R = -R$

4 possible camera poses

$$[R|b] = [UYV^T|u_3] or [UY^TV^T|u_3] or [UYV^T|-u_3] or [UY^TV^T|-u_3]$$

Cheirality check (reconstructed points must be in front of the cameras)

Choose camera pose by **point triangulation**,
confirming that a point is in front of both cameras $Z > 0$

The baseline or translation between the cameras can only be resolved up to a scale factor.

Point Triangulation from Multiple Images

P	Unknown point in the world frame
$p_1,\ p_2$	Known pixels in image 1 and 2 corresponding to same point P

$p_1 = K_1[R_1\|t_1]P$	Projection of a point P to a pixel in image 1
$[p_1]_x K_1[R_1\|t_1]P = 0$	Cross product of a vector with itself = 0
$p_2 = K_2[R_2\|t_2]P$	Projection of the same point P to a pixel in image 2
$[p_2]_x K_2[R_2\|t_2]P = 0$	

$$\begin{bmatrix} [p_1]_x K_1[R_1|t_1] \\ [p_2]_x K_2[R_2|t_2] \\ \cdots \end{bmatrix} P = 0$$

Stack Multiple Images

Solve linear homogeneous least squares

$\min_P (HP)^2$ subject to $||P|| = 1$

$H = U\Sigma V^T$ **Singular Value Decomposition**

$$\hat{P} = V_{:4}$$

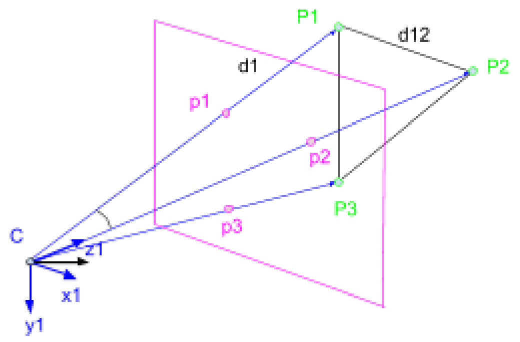
$$\hat{P} = \begin{bmatrix} \frac{\hat{P}_x}{\hat{P}_w} \\ \frac{\hat{P}_y}{\hat{P}_w} \\ \frac{\hat{P}_z}{\hat{P}_w} \\ 1 \end{bmatrix}$$

Normalize to homogeneous coordinates

Camera Pose from Perspective-n-Point (PnP)

[Wikipedia]

P3P



$d_i p_i = [R\|t] P_i$	Camera Projection
P_i	Known 3D point
p_i	Known corresponding point in the ideal image
d_i	Unknown distance of 3D point from camera

3 Triangles given 3 Points

$d_i^2 + d_j^2 - 2d_i d_j cos\alpha_{ij} = d_{ij}^2$	**Cosine law**
α_{ij}	Known angle between points in the image
d_{ij}	Known distance between 3D points

$$d_2^2 + d_3^2 - 2d_2 d_3 cos\alpha_{23} = d_{23}^2$$

$a = \frac{d_2}{d_1}$ and $b = \frac{d_3}{d_1}$

$$(ad_1)^2 + (bd_1)^2 - 2ad_1 bd_1 cos\alpha_{23} = d_{23}^2$$

$$d_1^2(a^2 + b^2 - 2abcos\alpha_{23}) = d_{23}^2$$

$$d_1^2 = \frac{d_{23}^2}{a^2 + b^2 - 2abcos\alpha_{23}}$$

Solve for a and put into other 2 equations

$$d_1^2 = \frac{d_{13}^2}{1 + b^2 - 2bcos\alpha_{13}}$$

Set 2 equations equal

$$d_1^2 = \frac{d_{12}^2}{a^2 + 1 - 2acos\alpha_{12}}$$

$$A_4 b^4 + A_3 b^3 + A_2 b^2 + A_1 b + A_0 = 0$$

Get and solve 4th degree polynomial for b

$$d_1^2 = \frac{d_{13}^2}{1 + b^2 - 2bcos\alpha_{13}}$$

Solve for d_1

$$d_3 = bd_1$$

Solve for d_3

$d_2^2 + d_3^2 - 2d_2 d_3 cos\alpha_{23} = d_{23}^2$ Solve for d_2

4 possible solution

Tilting triangle P_1, P_2, P_3 along the 3 sides keeps the angles between points in the image and the distances between 3D points the same and therefore provides 3 other solutions.

Initial guess or 4th point to remove ambiguity

Determine R, C from corresponding points P_i^C in camera and P_i in world frame

The solution is unstable if the projection center lies on a cylinder defined by the 3D points.

P6P

Direct Linear Transform to determine projection matrix P from 6 3D points

Determine R, t from projection matrix

$P = K[R|t]$ Projection matrix

$[R|t] = K^{-1}P$ Known camera matrix K

Enforce orthogonal constraints on rotation matrix

$R = K^{-1}P_{1:3}$ Rotation matrix

$R = U\Sigma V^T$ Singular Value Decomposition

$\Sigma = diag(\sigma_1, \sigma_2, \sigma_3)$

$R' = UV^T$ Singular Values = 1

Resolve scale ambiguity of translation

$t = K^{-1}R_4/\sigma_1$ Translation

EPnP

An efficient PnP algorithm with complexity O(n) is described in Lepetit et al. 2009.

The coordinates of the 3D reference points are expressed as a weighted sum of four non-coplanar virtual control points in the camera coordinate system, thereby reducing the unknowns to those four control points.

$$P_i^W = \sum_{j=1}^{4} \alpha_{ij} C_j^W$$

3D points defined by 4 control points in world frame

$$C_k^W, k = 1, 2, 3, 4$$

Control points in world frame

$$\sum_{j=1}^{4} \alpha_{ij} = 1$$

Uniquely defined homogeneous **barycentric coordinates**, estimated from 3D points in world frame

The non-coplanar control points can be chosen at random, but choosing one point being the centroid of the 3D points and the others forming a base that aligns with the principle axis of the 3D points increases the stability of this method.

$$P_i = \sum_{j=1}^{4} \alpha_{ij} C_j$$

3D points defined by the same 4 control points in camera frame

$$C_k, k = 1, 2, 3, 4$$

Same control points in camera frame

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} C_{x_j} \\ C_{y_j} \\ C_{z_j} \end{bmatrix}$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_i}{z_i} \\ \frac{y_i}{z_i} \\ 1 \end{bmatrix}$$

Projection of the 3D points in camera frame to the ideal image
Homogeneous to Euclidean coordinates

$$\sum_{j=1}^{4} \alpha_{ij} C_{x_j} - \alpha_{ij} u_i C_{z_j} = 0$$

$$\sum_{j=1}^{4} \alpha_{ij} C_{y_j} - \alpha_{ij} v_i C_{z_j} = 0$$

$$h_i = \begin{bmatrix} \alpha_{i1} & 0 & -\alpha_{i1} u_i & \alpha_{i2} & 0 & -\alpha_{i2} u_i & \alpha_{i3} & 0 & -\alpha_{i3} u_i & \alpha_{i4} & 0 & -\alpha_{i4} u_i \\ 0 & \alpha_{i1} & -\alpha_{i1} v_i & 0 & \alpha_{i2} & -\alpha_{i2} v_i & 0 & \alpha_{i3} & -\alpha_{i3} v_i & 0 & \alpha_{i4} & -\alpha_{i4} v_i \end{bmatrix}$$

$$c = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix}$$

$$\begin{bmatrix} h_1 \\ \cdots \\ h_n \end{bmatrix} c = Hc = 0$$

Stacking up n points
4 points minimum

Solve linear homogeneous least squares
and keep solution with smallest reprojection error

$\min_c (Hc)^2$ subject to $||c|| = 1$

$H = U\Sigma V^T$ **Singular Value Decomposition**

$$\hat{c} = \sum_{i=1}^{N} \beta_i V_i$$

Linear combination of right singular vectors
corresponding to $N = 1, ...4$ null singular values
$N = 1$ for perfect data from 6 points

Calculate linear combination β_i for $N = 1, 2, 3, 4$

$$||\hat{c}_i - \hat{c}_j||^2 = ||C_i^W - C_j^W||^2$$

Same 6 distances between the 4 control points
in camera frame and world frame

$N = 1$

$$||\beta_1 V_{1_i} - \beta_1 V_{1_j}||^2 = ||C_i^W - C_j^W||^2$$

$$\beta_1 = \frac{\sum_{i,j \in [1:4]} ||V_{1_i} - V_{1_j}|| ||C_i^W - C_j^W||}{\sum_{i,j \in [1:4]} ||V_{1_i} - V_{1_j}||^2}$$

$N = 2, 3, 4$

$$||(\beta_1 V_{1_i} + \beta_2 V_{2_i} + \beta_3 V_{3_i} + \beta_4 V_{4_i}) - (\beta_1 V_{1_j} + \beta_2 V_{2_j} + \beta_3 V_{3_j} + \beta_4 V_{4_j})||^2$$
$$= ||C_i^W - C_j^W||^2$$

Solve using linearization and relinearization techniques

Refine solution using **Gauss-Newton**

$$\hat{\beta} = \min_\beta \sum_{i<j} ||\hat{c}_i - \hat{c}_j||^2 - ||C_i^W - C_j^W||^2$$

Determine R, C from corresponding points C_i in camera and C_i^W in world frame

Minimize Reprojection Error

Statistical optimum iterative method is based on Levenberg-Marquardt minimizing the reprojection error, that is the sum of squared distances between the observed projections and the projected points, like in the bundle adjustment.

Bundle Adjustment

[Wikipedia]

$$\hat{p}_{ji} = K_j[R_j|t_j]P_i$$
$$\hat{p}_{ji} = K_j R(q_j)[I_{3x3} - C_j]P_i$$
$$\hat{p}_{ji} = K_j R(q_j)[P_i - C_j]$$

Camera Projection for camera j

P_i — Unknown 3D point i

3 parameters per point

$$K_j = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Unknown Camera Calibration (5)

$$R(q_j) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Unknown Camera Rotation represented by quaternions (4)

$$= \begin{bmatrix} 1-2q_z^2-2q_y^2 & -2q_zq_w+2q_yq_x & 2q_yq_w+2q_zq_x \\ 2q_xq_y+2q_wq_z & 1-2q_z^2-2q_x^2 & 2q_zq_y-2q_xq_w \\ 2q_xq_z-2q_wq_y & 2q_yq_z+2q_wq_x & 1-2q_y^2-2q_x^2 \end{bmatrix}$$

C_j — Unknown Camera Origin (3)

12 parameters per camera

$$\hat{p}_{ji} = \begin{bmatrix} \hat{x}_{ji} \\ \hat{y}_{ji} \\ \hat{z}_{ji} \end{bmatrix}$$

Projected point in image of camera j

$$\begin{bmatrix} \hat{x}_{ji} \\ \hat{y}_{ji} \\ \hat{z}_{ji} \end{bmatrix} = \begin{bmatrix} f_xr_{11}+sr_{21}+c_xr_{31} & f_xr_{12}+sr_{22}+c_xr_{32} & f_xr_{13}+sr_{23}+c_xr_{33} \\ f_yr_{21}+c_yr_{31} & f_yr_{22}+c_yr_{32} & f_yr_{23}+c_yr_{33} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}[P_i - C_j]$$

$$\begin{bmatrix} \hat{u}_{ji} \\ \hat{v}_{ji} \end{bmatrix} = \begin{bmatrix} \frac{\hat{x}_{ji}}{\hat{z}_{ji}} \\ \frac{\hat{y}_{ji}}{\hat{z}_{ji}} \end{bmatrix}$$

Homogeneous to Euclidean coordinates

$$\begin{bmatrix} u_{ji} \\ v_{ji} \end{bmatrix}$$

Observed points in image of camera j

2 measurements per observations (per point per camera)

Squared Error

$$e^2 = \sum_j \sum_i || \begin{bmatrix} u_{ji} \\ v_{ji} i \end{bmatrix} - \begin{bmatrix} \hat{u}_{ji} \\ \hat{v}_{ji} \end{bmatrix} ||^2$$

Reprojection error

Minimize the Square Error Criterion

$$\begin{aligned} \hat{q}, \hat{C}, \hat{P} &= \min_{q,C,P} \sum_j \sum_i || \begin{bmatrix} u_{ji} \\ v_{ji} i \end{bmatrix} - \begin{bmatrix} \hat{u}_{ji} \\ \hat{v}_{ji} \end{bmatrix} ||^2 \\ &= \min_{q,C,X} \|z - h(R(q), C, P)\|^2 \end{aligned}$$

$$z_{ji} = \begin{bmatrix} u_{ji} \\ v_{ji} \end{bmatrix}$$

Measurements

$$h(R(q_j), C_j, P_i) = \begin{bmatrix} \frac{\hat{x}_{ji}}{\hat{z}_{ji}} \\ \frac{\hat{y}_{ji}}{\hat{z}_{ji}} \end{bmatrix}$$

Measurement Model

Solve **nonlinear least squares** with iterative **Levenberg-Marquardt**

$$x_{k+1} = x_k + \Delta x$$

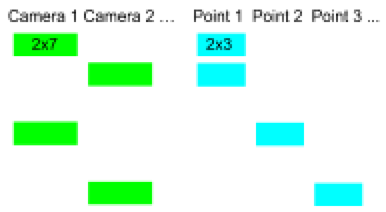
$$\Delta x = \frac{H^T \Omega (z - h(x))}{(H^T \Omega H + \lambda I)}$$

Jacobian H is **highly sparse** and can be arrange in **special ordered blocks**
assuming camera calibration is known

$$H_{ji} = [\frac{\delta h(R(q_j), C_j, P_i)}{\delta R_j} \frac{\delta R_j}{\delta q_j} \frac{\delta h(R(q_j), C_j, P_i)}{\delta C_j} \frac{\delta h(R(q_j), C_j, P_i)}{\delta P_i}]$$

2x10 = 2x9 9x4 2x3 2x3

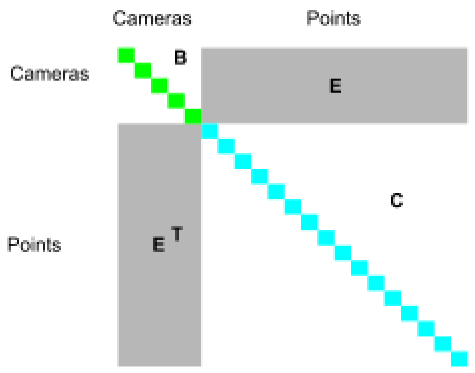
$$H = \begin{bmatrix} \frac{\delta h(R(q_1),C_1,P_1)}{\delta R_1} \frac{\delta R_1}{\delta q_1} \frac{\delta h(R(q_1),C_1,P_1)}{\delta C_1} & \cdots & \cdots & 0 & \frac{\delta h(R(q_1),C_1,P_1)}{\delta P_1} & 0 & 0 \\ 0 & \frac{\delta h(R(q_2),C_2,P_1)}{\delta R_2} \frac{\delta R_2}{\delta q_2} \frac{\delta h(R(q_2),C_2,P_1)}{\delta C_2} & \cdots & 0 & \frac{\delta h(R(q_2),C_2,P_1)}{\delta P_1} & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\delta h(R(q_1),C_1,P_2)}{\delta R_1} \frac{\delta R_1}{\delta q_1} \frac{\delta h(R(q_1),C_1,P_2)}{\delta C_1} & \cdots & \cdots & 0 & 0 & \frac{\delta h(R(q_1),C_1,P_2)}{\delta P_2} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \frac{\delta h(R(q_2),C_2,P_3)}{\delta R_2} \frac{\delta R_2}{\delta q_2} \frac{\delta h(R(q_2),C_2,P_3)}{\delta C_2} & \cdots & 0 & 0 & 0 & \frac{\delta h(R(q_2),C_2,P_3)}{\delta P_3} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$



*2 * Observations x 7 * Cameras + 3 * Points, with many more points than cameras*

Computational expensive inversion of a huge matrix
$(H^T \Omega H + \lambda I)^{-1}$

Inversion of a sparse matrix with special blocks B, E, C can be calculated efficiently
B and (huge) C are block diagonal matrices



$$(H^T \Omega H + \lambda I)^{-1} \Delta x = H^T \Omega (z - h(x))$$
$$\begin{bmatrix} B & E \\ E^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} e_c \\ e_p \end{bmatrix}$$

Block structured linear system
solved using Gaussian elimination

$$(B - EC^{-1}E^T)\Delta x_c = e_c - EC^{-1}e_p$$

Reduced system

$$\Delta x_p = C^{-1} e_p - E^T \Delta x_c$$

Back substitution

Only the inversion of C is required, which can be done very efficiently
by calculating the inverse of each of the block diagonal matrices.

$$\frac{\delta h(R(q_j), C_j, P_i)}{\delta R_j} = \begin{bmatrix} \frac{\hat{z}_{ji}\frac{\delta \hat{x}_{ji}}{\delta R_j} - \hat{x}_{ji}\frac{\delta \hat{z}_{ji}}{\delta R_j}}{\hat{z}_{ji}^2} \\ \frac{\hat{z}_{ji}\frac{\delta \hat{y}_{ji}}{\delta R_j} - \hat{y}_{ji}\frac{\delta \hat{z}_{ji}}{\delta R_j j}}{\hat{z}_{ji}^2} \end{bmatrix}$$

$$\frac{\delta \hat{x}_{ji}}{\delta R_j} = [f_x(X_i - C_{x_j}) | s(Y_i - C_{y_j}) | c_x(Z_i - C_{z_j})]$$

$$\frac{\delta \hat{y}_{ji}}{\delta R_j} = [0|f_y(Y_i - C_{y_j})|c_y(Z_i - C_{z_j})]$$

$$\frac{\delta \hat{z}_{ji}}{\delta R_j} = [0|0|(Z_i - C_{z_j})]$$

$$\frac{\delta R_j}{\delta q_j} = \begin{bmatrix} \frac{\delta r_{11}}{\delta q_j} \\ \frac{\delta r_{12}}{j} \\ \cdots \\ \frac{\delta r_{33}}{\delta q_j} \end{bmatrix}$$

$$\frac{\delta r_{11}}{\delta q_j} = [0| - 4q_y| - 4q_z|0]_, \ \frac{\delta r_{12}}{\delta q_j} = [2q_y|2q_x| - 2q_w| - 2q_z]_, \ \frac{\delta r_{13}}{\delta q_j} = [2q_z|2q_w|2q_x|2q_y]$$

$$\frac{\delta r_{21}}{\delta q_j} = [2q_y|2q_x|2q_w|2q_z]_, \ \frac{\delta r_{22}}{\delta q_j} = [-4q_x|0| - 4q_z|0]_, \ \frac{\delta r_{23}}{\delta q_j} = [-2q_w|2q_z|2q_y|2q_x]$$

$$\frac{\delta r_{31}}{\delta q_j} = [2q_z| - 2q_w|2q_x| - 2q_y]_, \ \frac{\delta r_{32}}{\delta q_j} = [2q_w|2q_z|2q_y|2q_x]_, \ \frac{\delta r_{33}}{\delta q_j} = [-4q_x| - 4q_y|0|0]$$

$$\frac{\delta h(R(q_j), C_j, P_i)}{\delta C_j} = \begin{bmatrix} \frac{\hat{z}_{ji}\frac{\delta \hat{x}_{ji}}{\delta C_j} - \hat{x}_{ji}\frac{\delta \hat{z}_{ji}}{\delta C_j}}{\hat{z}_{ji}^2} \\ \frac{\hat{z}_{ji}\frac{\delta \hat{y}_{ji}}{\delta C_j} - \hat{y}_{ji}\frac{\delta \hat{z}_{ji}}{\delta C_j}}{\hat{z}_{ji}^2} \end{bmatrix}$$

$$\frac{\delta \hat{x}_{ji}}{\delta C_j} = -[f_x r_{11} + s r_{21} + c_x r_{31}|f_x r_{12} + s r_{22} + c_x r_{32}|f_x r_{13} + s r_{23} + c_x r_{33}]$$

$$\frac{\delta \hat{y}_{ji}}{\delta C_j} = -[f_y r_{21} + c_y r_{31}|f_y r_{22} + c_y r_{32}|f_y r_{23} + c_y r_{33}]$$

$$\frac{\delta \hat{z}_{ji}}{\delta C_j} = -[r_{31}|r_{32}|r_{33}]$$

$$\frac{\delta h(R(q_j), C_j, P_i)}{\delta P_i} = \begin{bmatrix} \frac{\hat{z}_{ji}\frac{\delta \hat{x}_{ji}}{\delta P_i} - \hat{x}_{ji}\frac{\delta \hat{z}_{ji}}{\delta P_i}}{\hat{z}_{ji}^2} \\ \frac{\hat{z}_{ji}\frac{\delta \hat{y}_{ji}}{\delta P_i} - \hat{y}_{ji}\frac{\delta \hat{z}_{ji}}{\delta P_i}}{\hat{z}_{ji}^2} \end{bmatrix}$$

$$\frac{\delta \hat{x}_{ji}}{\delta P_i} = [f_x r_{11} + s r_{21} + c_x r_{31}|f_x r_{12} + s r_{22} + c_x r_{32}|f_x r_{13} + s r_{23} + c_x r_{33}]$$

$$\frac{\delta \hat{y}_{ji}}{\delta P_i} = [f_y r_{21} + c_y r_{31}|f_y r_{22} + c_y r_{32}|f_y r_{23} + c_y r_{33}]$$

$$\frac{\delta \hat{z}_{ji}}{\delta P_i} = [r_{31}|r_{32}|r_{33}]$$

Kalman Filters

Kalman Filter

[Wikipedia]

The Kalman filter is the **Best Linear Unbiased Estimator (BLUE)**. The algorithm is recursive and consists of two steps.

Dynamic Model discretized in time domain t

$$x_t = F_t x_{t-1} + B_t u_t + w_t$$

F_t	**state-transition or motion model**
x_t	state
B_t	control-input model
u_t	control-input
$w_t \sim N(0, Q_t)$	process noise assumed zero mean multivariate normal distribution
Q_t	process noise covariance Q_t process noise covariance

$$z_t = H_t x_t + v_t$$

H_t	**measurement model** mapping true state space to observed **multi-sensors** space
$v_t \sim N(0, R_t)$	measurement noise assumed zero mean multivariate normal distribution
R_t	measurement noise covariance

1. Prediction step

Estimate the current state and covariance (uncertainties) based on dynamic model and process noise.

$\hat{x}_{t\|t-1} = F_t x_{t-1\|t-1} + B_t u_t$	(a priori) state estimate
$P_{t\|t-1} = F_t P_{t-1\|t-1} F_t^T + Q_t$	(a priori) state estimate covariance

$\hat{x}_{t|t-1}$ represents the estimate of x at time t given observation at time t-1

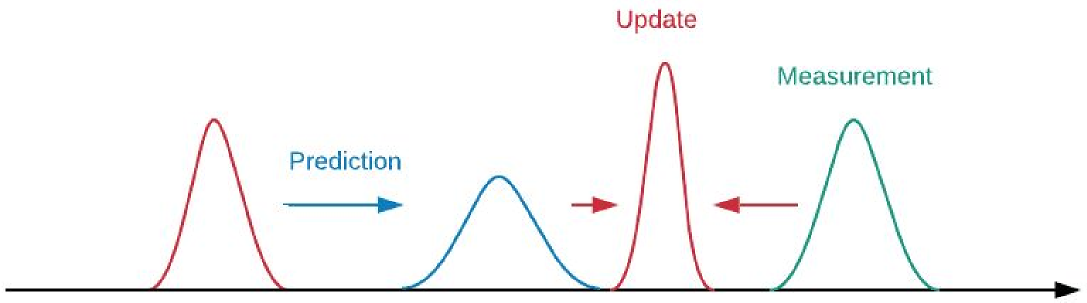
2. Update step

State estimates are updated based on measurements and sensor noise using a weighted average.

$$K_t = \frac{P_{t|t-1}H_t^T}{H_t P_{t|t-1} H_t^T + R_t}$$ optimal Kalman gain

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H_t \hat{x}_{t|t-1})$$ (a posteriori) state estimate

$$P_{t|t} = (I - K_t H_t) P_{t|t-1}$$ (a posteriori) state estimate covariance



$$N(\hat{x}_{t-1|t-1}, P_{t-1|t-1}) \quad N(\hat{x}_{t|t-1}, P_{t|t-1}) \quad N(\hat{x}_{t|t}, P_{t|t}) \quad N(\hat{z}_t, R_t)$$

Extended Kalman Filter (EKF)

[Wikipedia]

The extended Kalman filter is the de facto standard for nonlinear state estimators, when the state transition model is well defined. The extended Kalman filter is the **nonlinear version** of the Kalman filter, which **linearizes using a first order Taylor expansion** about the current estimate.

Linearized Dynamic Model

motion model

$$x_t = f(x_{t-1}, u_t, w_t)$$
$$\approx f(\hat{x}_{t-1}, u_t, 0) + \frac{\delta f}{\delta x}|_{\hat{x}_{t-1}, u_t, 0}(x_{t-1} - \hat{x}_{t-1}) + \frac{\delta f}{\delta w}|_{\hat{x}_{t-1}, u_t, 0} w_t$$

measurement model

$$z_t = h(x_t, v_t)$$
$$\approx h_t(\hat{x}_t, 0) + \frac{\delta h}{\delta x}|_{\hat{x}_t, 0}(x_t - \hat{x}_t) + \frac{\delta h}{\delta v}|_{\hat{x}_t, 0} v_t$$

1. Prediction Step

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}, u_t, 0)$$ (a priori) state estimate

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + G_t Q_t G_t^T$$ (a priori) state estimate covariance

$$F_t = \frac{\delta f}{\delta x}|_{\hat{x}_{t-1|t-1},u_t,0}$$
$$G_t = \frac{\delta f}{\delta w}|_{\hat{x}_{t-1|t-1},u_t,0}$$

2. Update Step

$$K_t = \frac{P_{t|t-1}H_t^T}{H_t P_{t|t-1} H_t^T + M_t R_t M_t^T}$$ optimal Kalman gain

$$H_t = \frac{\delta h}{\delta x}|_{\hat{x}_{t|t-1},0}$$
$$M_t = \frac{\delta h}{\delta v}|_{\hat{x}_{t|t-1},0}$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - h(\hat{x}_{t|t-1},0))$$ (a posteriori) state estimate

$$P_{t|t} = (I - K_t H_t)P_{t|t-1}$$ (a posteriori) state estimate covariance

Error-State Extended Kalman Filter (ES-EKF)

A comprehensive introduction to Error-State Kalman Filters can be found in Sola 2017.

$\hat{X} = X + \delta X$, $\hat{X}$ true-state, X nominal-state, δX **error-state**

The Error-State Extended Kalman Filter **estimates the error state directly** and uses it as a **correction for the nominal-state**. Only the **error-state** is **linearized**.

An **error-state system** is always **operating close to the origin**, far from possible parameter singularities, etc., providing a guarantee that the **linearization validity holds at all times**. The **error-state** is always **small**, meaning that all **second-order products are negligible** and consequently making the computation of **Jacobians very easy** and **fast**. **Error dynamics** are **slow** because all large-signal dynamics have been integrated in the nominal-state, therefore small-signal error-state Kalman Filter corrections can be applied at a **lower rate** than the predictions.

Linearized Dynamic Model

$$x_t = f(\hat{x}_{t-1}, u_t, 0) + F_t(x_{t-1} - \hat{x}_{t-1}) + G_t w_t$$
$$x_t - f(\hat{x}_{t-1}, u_t, 0) = F_t(x_{t-1} - \hat{x}_{t-1}) + G_t w_t$$
$$\delta x_t = F_t \delta x_{t-1} + G_t w_t$$ **error-state**

$$z_t = h(\hat{x}_t, 0) + H_t(x_t - \hat{x}_t) + M_t v_t$$
$$z_t = h(\hat{x}_t, 0) + H_t \delta x_t + M_t v_t$$

1. Prediction Step

$\hat{x}_{t\|t-1} = f(\hat{x}_{t-1\|t-1}, u_t, 0)$	(a priori) **nominal-state** estimate
$\delta\hat{x}_{t\|t-1} = F_t \delta\hat{x}_{t-1\|t-1} + G_t w_t$	**(a priori) error-state estimate**
$P_{t\|t-1} = F_t P_{t-1\|t-1} F_t^T + G_t Q_t G_t^T$	(a priori) **state** estimate **covariance**

2. Update Step, if measurement is available

$K_t = \dfrac{P_{t\|t-1} H_t^T}{H_t P_{t\|t-1} H_t^T + M_t R_t M_t^T}$	optimal Kalman gain
$\delta\hat{x}_{t\|t} = K_t(z_t - h(\hat{x}_{t\|t-1}, 0))$	**(a posteriori) error-state**
$\hat{x}_{t\|t} = \hat{x}_{t\|t-1} + \delta\hat{x}_{t\|t}$	(a posteriori) state estimate
$P_{t\|t} = (I - K_t H_t) P_{t\|t-1}$	(a posteriori) state estimate covariance

Pose Graph

Graph Construction (front-end)

$x_i \;\; i \in [1 \cdots n]$	**Node** = pose (R and t in homogeneous coordinates) = State vector
$x_i^{-1} x_j$	(Current) relative pose transformation x_i to x_j

Measurements (e.g. odometry, ICP, loop closures)

$\langle z_{ij}, \Omega_{ij} \rangle$	**Edge** = soft constraint
z_{ij}	Measured relative pose transformation from x_i to x_j
$\Omega_{ij} = \Sigma^{-1}$	Information matrix (inverse covariance matrix)

Graph Optimization (back-end)

Error

$e_{ij}(x) = m2v(z_{ij}^{-1}(x_i^{-1} x_j))$	**Relative pose transformation error**
$m2v()$	Extract $(x, y, z, pitch, roll, yaw)^T$ vector from matrix

Solve **nonlinear least squares** with iterative **Levenberg-Marquardt**

$$x_{k+1} = x_k + \Delta x$$
$$\Delta x = \frac{H^T \Omega e(x)}{(H^T \Omega H + \lambda I)}$$

Jacobian H is **highly sparse** and can be arranged for efficient processing

$$H_{ij} = \frac{\delta e_{ij}(x)}{\delta x} = (0 \cdots \frac{\delta e_{ij}(x_i)}{\delta x} \cdots \frac{\delta e_{ij}(x_j)}{\delta x} \cdots 0)$$

Need to fix coordinate system > fix one x_i by eliminating rows and columns of x_i

Information Matrix

$\Sigma = (H^T \Omega H)^{-1}$	(Dense) covariance matrix
$(\cdot)_{[a,b]}$	Block a, b in a matrix
$\Sigma_{[j,j]}$	Block j, j contains the covariance matrix of the states/poses
$\Omega = \Sigma^{-1}$	Information matrix

Relative Uncertainty between x_i **and** x_j (useful to identify candidates for loop closures)

$H^T \Omega H$	Fix x_i by eliminating rows and columns of x_i
$\Sigma = (H^T \Omega H)^{-1}$	Covariance matrix
$\Sigma_{[j,j]}$	Block j, j contains covariance matrix of x_j with respect to x_i
$\Omega_{ij} = (\Sigma_{[j,j]})^{-1}$	Information matrix between x_i and x_j

References

The design and implementation of VINS-Mono is described in Qin et al. 2018 with source code provided on Github. More details about the equations can be found in Wu 2019.

T. Qin, P. Li, and S. Shen; **VINS-Mono: A robust and versatile monocular visual-inertial state estimator**; IEEE Trans. Robot.; 2018
VINS-Mono Github

Yibin Wu; **Equations Derivation of VINS-Mono**; CVPR 2019

Perception algorithms are implemented using the **CamOdoCal** library.

Lionel Heng, Bo Li, and Marc Pollefeys; **CamOdoCal: Automatic Intrinsic and Extrinsic Calibration of a Rig with Multiple Generic Cameras and Odometry**; 2013
CamOdoCal GitHub

Image analysis and **perception algorithms** are implemented using the **OpenCV 3.3.1** library.

Bradski, G.; **The OpenCV Library**; Dr. Dobb's Journal of Software Tools; 2000
OpenCV Homepage GitHub

Place recognition is implemented using **DBoW2**.

D. Galvez-Lopez and J. D. Tardos; **Bags of binary words for fast place recognition in image sequences**; IEEE Transactions on Robotics; 2012
DBoW2 Github

Optimization algorithms are implemented using the **Ceres 1.14** library and the **Eigen 3.3.3** library.

Sameer Agarwal and Keir Mierle and Others; **Ceres Solver**

Guennebaud and Benoit Jacob and others; **Eigen**; 2010

B. D. Lucas and T. Kanade; **An iterative image registration technique with an application to stereo vision**; IJCAI 1981

Chris Harris & Mike Stephens; **A combined corner and edge detector**; 1988

R. Hartley; **Estimation of Relative Camera Positions for Uncalibrated Cameras**; ECCV 1992

J. Shi and C. Tomasi; **Good features to track**; CVPR 1994

Jean-Yves Bouguet; **Pyramidal Implementation of the Lucas Kanade Feature Tracker**; 2000

R. Hartley and A. Zisserman; **Multiple view geometry in computer vision**; Cambridge university press, 2003

D. Nister; **An efficient solution to the five-point relative pose problem**; IEEE Transactions on Pattern Analysis and Machine Intelligence; 2004

H. Li and R. Hartley; **Five-Point Motion Estimation Made Easy**; International Conference on Pattern Recognition (ICPR); 2006

E. Rosten and T. Drummond; **Machine learning for high-speed corner detection**; European Conference on Computer Vision; 2006

V. Lepetit, F. Moreno-Noguer, and P. Fua; Epnp: **An accurate o (n)solution to the pnp problem; International journal of computer vision**; 2009

M. Calonder, V. Lepetit, C. Strecha, and P. Fua; **BRIEF: Binary Robust Independent Elementary Features**; European Conference on Computer Vision; 2010

W. Foestner and B.P. Wrobel; **Photogrammetric Computer Vision;** Springer 2016

J. Solà; **Quaternion kinematics for the error-state Kalman filter**; 2017

J. Solà, J. Deray, D. Atchuthan; **A micro Lie theory for state estimation in robotics**; 2019