P4 Ph1: Classical Visual-Inertial Odometry

Sarthak Mehta
Dept. of Robotics Engineering
Worcester Polytechnic Institute
smehta1@wpi.edu

Prasham Soni
Dept. of Robotics Engineering
Worcester Polytechnic Institute
psoni@wpi.edu

***Abstract*—This document presents the implementation of a Visual-Inertial Odometry (VIO) system using a classical approach. The developed pipeline is based on the Stereo Multi-State Constraint Kalman Filter (S-MSCKF) [2], which extends the mathematical framework of the original Multi-State Constraint Kalman Filter (MSCKF) [1].**

I. Introduction

In fast-moving robots like drones, it is very important to estimate position and motion accurately without using heavy or expensive hardware. Traditional monocular visual-inertial odometry methods either suffer from high computational cost or lack robustness under challenging conditions. The Stereo Multi-State Constraint Kalman Filter (S-MSCKF) addresses these challenges by combining stereo vision with a filter-based framework, providing greater resilience to lighting changes, low texture, and aggressive motion. By avoiding explicit 3D feature estimation and instead using the transformation between camera poses, it makes it much faster. S-MSCKF achieves real-time performance on lightweight onboard computers while maintaining high estimation accuracy.

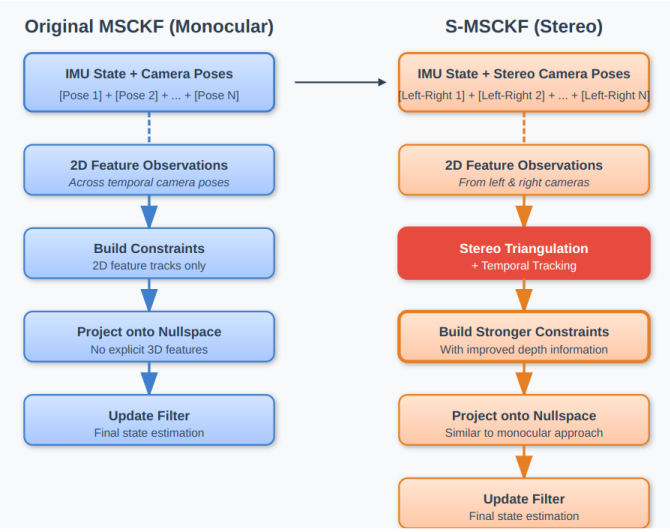

Fig. 1. MSCKF v/s S-MSCKF

As seen in above Fig. 1 the Stereo MSCKF (S-MSCKF) builds directly upon the original MSCKF by extending its monocular framework to stereo vision. While MSCKF relies solely on motion to recover depth, S-MSCKF uses stereo camera measurements to triangulate depth more accurately and immediately.

II. Phase I

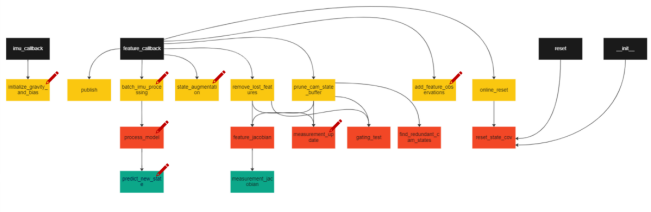

Fig. 2. Flowchart for S - MSCKF Implementation

As seen in Fig. 2. This is the flowchart that represents the implemented S - MSCKF. Out of these, in total of 7 functions were implemented.

A. Initialization of Gravity and Gyroscope Bias

At the beginning of the system, we do not know how the IMU is oriented with respect to the world, nor do we know its internal measurement biases. This creates uncertainty in interpreting the IMU readings correctly.

To solve this, we collect a few initial IMU measurements assuming the device is stationary. From these measurements:

- We estimate the gyroscope bias by averaging the measured angular velocities, assuming that true rotation should be close to zero.
- We estimate the direction and strength of gravity by averaging the measured accelerations.
- We then compute a rotation that aligns the sensed gravity direction with the true gravity direction $[0, 0, -g]$ in the world frame.

B. Batch IMU Processing

The batch IMU processing handles IMU measurements up to a specified `time_bound`. It loops through the IMU buffer and:

- Skips any IMU measurements that have already been processed.
- Processes each valid IMU measurement by applying the process model to predict the IMU state forward in time.
- Updates the state's timestamp and assigns a new unique ID after processing.

- Stops processing once it reaches measurements beyond the `time_bound`.
- Removes all IMU measurements that have been processed from the buffer.

This ensures that the IMU state is properly propagated up to the time of the next observation.

This ensures that the initial state is consistent with the inertial world, where gravity points straight down and the IMU biases are compensated for.

C. IV. PROCESS MODEL

We model the IMU propagation using the following continuous-time equations:

$$\dot{\mathbf{q}}_{IG}(t) = \frac{1}{2}\Omega(\boldsymbol{\omega}(t))\,\mathbf{q}_{IG}(t)$$
$$\dot{\mathbf{v}}_{IG}(t) = C(\mathbf{q}_{IG}(t))\,\mathbf{a}(t) + \mathbf{g}$$
$$\dot{\mathbf{p}}_{IG}(t) = \mathbf{v}_{IG}(t)$$

Here:

- $\mathbf{q}_{IG}(t)$ is the unit quaternion representing rotation from the global frame $\{G\}$ to the IMU frame $\{I\}$.
- $\boldsymbol{\omega}(t) = [\omega_x, \omega_y, \omega_z]^T$ is the gyroscope-measured angular velocity (after bias correction).
- $\mathbf{a}(t)$ is the accelerometer-measured linear acceleration (after bias correction).
- $\mathbf{g}$ is the gravity vector expressed in the global frame.
- $C(\mathbf{q})$ is the rotation matrix corresponding to quaternion $\mathbf{q}$.

The angular velocity matrix $\Omega(\boldsymbol{\omega})$ is given by:

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix}$$

where the skew-symmetric matrix $\lfloor\boldsymbol{\omega}\rfloor$ is:

$$\lfloor\boldsymbol{\omega}\rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Error-state dynamics

The error-state evolution in continuous time follows the linearized model:

$$\dot{\tilde{\mathbf{x}}}_{IMU} = F\tilde{\mathbf{x}}_{IMU} + G\mathbf{n}_{IMU}$$

where $\tilde{\mathbf{x}}_{IMU}$ is the IMU error state, and $\mathbf{n}_{IMU}$ represents the process noise.

The system matrices F and G are:

$$F = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor & -I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & -C(\mathbf{q})^\top \\ -C(\mathbf{q})^\top\lfloor\mathbf{a}\rfloor & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_3 & 0_{3\times3} \end{bmatrix}$$

$$G = \begin{bmatrix} -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & -C(\mathbf{q})^\top & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_3 \end{bmatrix}$$

where:

- I_3 is the 3×3 identity matrix,
- $0_{3\times3}$ is the 3×3 zero matrix.

Discrete-time approximation

The discrete-time transition matrix Φ is approximated by third-order Taylor expansion:

$$\Phi \approx I + F\Delta t + \frac{1}{2}(F\Delta t)^2 + \frac{1}{6}(F\Delta t)^3$$

where Δt is the time difference between IMU measurements.

The covariance is propagated as:

$$P_{k+1} = \Phi P_k \Phi^\top + Q$$

where Q is the discretized noise covariance.

D. Predict New State

During IMU propagation, we update the orientation, velocity, and position of the IMU based on gyroscope and accelerometer measurements using a 4th-order Runge-Kutta integration method.

Given:

- $\boldsymbol{\omega}$: bias-corrected gyroscope reading,
- $\mathbf{a}$: bias-corrected accelerometer reading,
- Δt: time interval between measurements.

Orientation Update: The quaternion dynamics are governed by:

$$\dot{\mathbf{q}}_{IG} = \frac{1}{2}\Omega(\boldsymbol{\omega})\mathbf{q}_{IG}$$

where the matrix $\Omega(\boldsymbol{\omega})$ is defined as:

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^\top & 0 \end{bmatrix}$$

and $\lfloor\boldsymbol{\omega}\rfloor$ is the skew-symmetric matrix of $\boldsymbol{\omega}$.

The quaternion is updated either using a trigonometric integration when $\|\boldsymbol{\omega}\|$ is large, or a linear approximation when $\|\boldsymbol{\omega}\|$ is very small.

Velocity and Position Update: The velocity and position dynamics are:

$$\dot{\mathbf{v}} = R(\mathbf{q})^\top\mathbf{a} + \mathbf{g}$$
$$\dot{\mathbf{p}} = \mathbf{v}$$

where $R(\mathbf{q})$ is the rotation matrix corresponding to the quaternion $\mathbf{q}$, and $\mathbf{g}$ is the gravity vector.

To numerically integrate these equations, we apply the classical 4th-order Runge-Kutta method:

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$\mathbf{p}_{k+1} = \mathbf{p}_k + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where the intermediate slopes k_1, k_2, k_3, k_4 are evaluated based on the current and intermediate estimates of velocity and position.

Summary: The updated orientation, velocity, and position after Δt are saved back to the IMU state. This ensures accurate and stable propagation even under fast rotations or accelerations.

E. State Augmentation

When a new image is received, a corresponding camera state is created based on the current IMU pose and known extrinsic calibration.

Given:

- R_{IC}: Rotation from IMU frame $\{I\}$ to camera frame $\{C\}$,
- $\mathbf{t}_{CI}$: Translation from camera frame to IMU frame,
- R_{WI}, $\mathbf{t}_{WI}$: Rotation and position of IMU in the world frame.

The camera pose in the world frame is:

$$R_{WC} = R_{IC} R_{WI}$$

$$\mathbf{t}_{CW} = \mathbf{t}_{WI} + R_{WI}^{\top} \mathbf{t}_{CI}$$

A new camera state consisting of $(R_{WC}, \mathbf{t}_{CW})$ is added to the state vector.

Covariance Augmentation: The Jacobian $J \in \mathbb{R}^{6\times 21}$ relating the new camera state to the IMU state is constructed as:

$$J = \begin{bmatrix} R_{IC} & 0 & 0 & 0 & I_3 & 0 \\ \lfloor R_{WI}^{\top} \mathbf{t}_{CI} \rfloor & 0 & 0 & I_3 & 0 & I_3 \end{bmatrix}$$

where: - $\lfloor \cdot \rfloor$ denotes the skew-symmetric matrix.

The augmented covariance matrix becomes:

$$P_{\text{aug}} = \begin{bmatrix} P & PJ^{\top} \\ JP & JPJ^{\top} \end{bmatrix}$$

where: - P is the existing state covariance.

Finally, the covariance is symmetrized:

$$P_{\text{aug}} = \frac{P_{\text{aug}} + P_{\text{aug}}^{\top}}{2}$$

This ensures numerical stability.

F. Adding Feature Observations

When new camera measurements arrive, feature observations are added to the map.

For each feature in the received measurement:

- **If the feature is new:**
  - Create a new feature object.
  - Store the stereo observation:

    $$\text{observation} = [u_0, v_0, u_1, v_1]$$

    where:
    - (u_0, v_0) are coordinates in the left image,
    - (u_1, v_1) are coordinates in the right image.
  - Insert the feature into the map server.
- **If the feature already exists:**
  - Add the new observation to the existing feature's history.

The **tracking rate** is updated as:

$$\text{tracking rate} = \frac{\text{number of tracked features}}{\text{current feature count} + \epsilon}$$

where ϵ is a small number to avoid division by zero.

G. Measurement Update

Given stacked feature observations, we perform a Kalman filter update step to correct the state estimate.

The stacked observation model is:

$$\mathbf{r} = H\tilde{\mathbf{x}} + \mathbf{n}$$

where:

- $\mathbf{r}$ is the stacked residual vector,
- H is the stacked Jacobian matrix,
- $\tilde{\mathbf{x}}$ is the state error,
- $\mathbf{n}$ is the measurement noise.

Step 1: Preconditioning: If H has more rows than columns, perform QR decomposition:

$$H = QR$$

where:

- Q is orthogonal,
- R is upper triangular.

Use the reduced system:

$$H_{\text{thin}} = R, \quad r_{\text{thin}} = Q^{\top} r$$

Otherwise, use H and r directly.

Step 2: Compute Kalman Gain: The innovation covariance is:

$$S = H_{\text{thin}} P H_{\text{thin}}^{\top} + R_n$$

where R_n is the observation noise.

The Kalman gain K is:

$$K = P H_{\text{thin}}^{\top} S^{-1}$$

Step 3: Update State Estimate: Compute the correction:

$$\delta \mathbf{x} = K r_{\text{thin}}$$

Apply the corrections to:

- IMU orientation, position, velocity, gyro bias, accelerometer bias,
- IMU-camera extrinsics,
- Camera states.

Small-angle approximations are used to update orientations:

$$\delta \mathbf{q} \approx [1, \frac{1}{2} \delta \boldsymbol{\theta}]$$

Step 4: Update Covariance: The covariance is updated as:

$$P^+ = (I - KH_{\text{thin}})P$$

Finally, the covariance matrix is symmetrized:

$$P^+ = \frac{P^+ + (P^+)^\top}{2}$$

H. Trajectory Error

TABLE I
ROTATION ERROR STATISTICS

Statistic	Value
Maximum	179.9704
Mean	170.7381
Median	172.7667
Minimum	145.9533
Number of Samples	2841
RMSE	170.8561
Standard Deviation	6.3476

TABLE II
SCALE ERROR STATISTICS

Statistic	Value
Maximum	4.0235
Mean	0.6907
Median	0.4659
Minimum	0.0001
Number of Samples	2841
RMSE	0.9895
Standard Deviation	0.7085

TABLE III
TRANSLATION ERROR STATISTICS

Statistic	Value
Maximum	0.1689
Mean	0.0734
Median	0.0703
Minimum	0.0090
Number of Samples	2841
RMSE	0.0809
Standard Deviation	0.0342

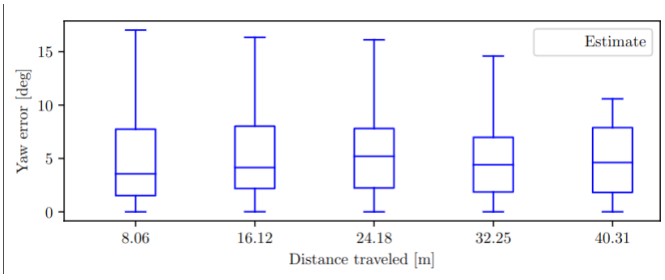

Fig. 3. Relative YAW Error

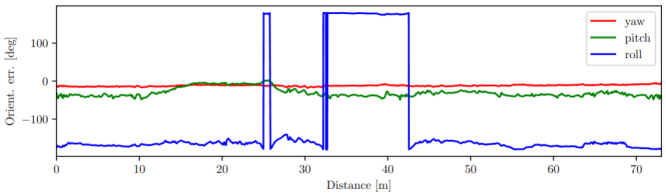

Fig. 4. Rotation Error

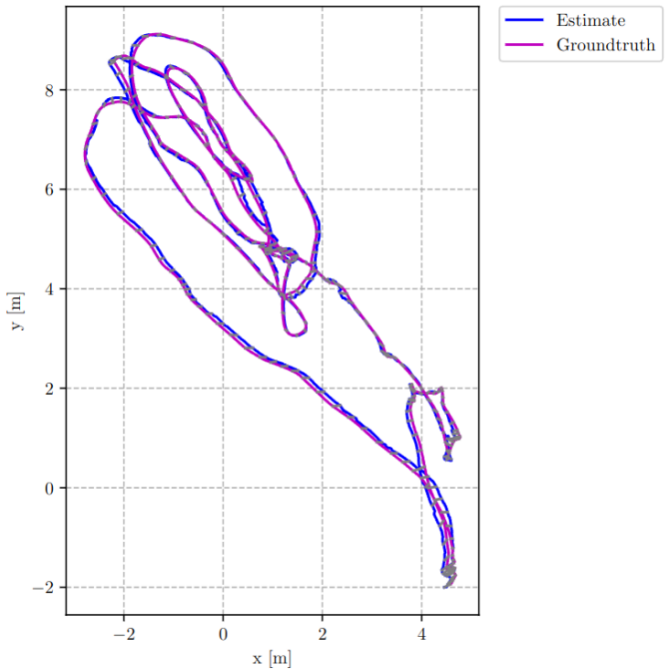

Fig. 5. Trajectory Top View

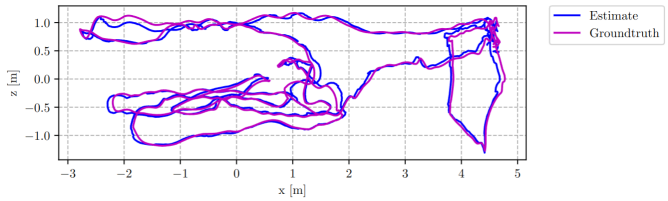

Fig. 6. Trajectory Side View

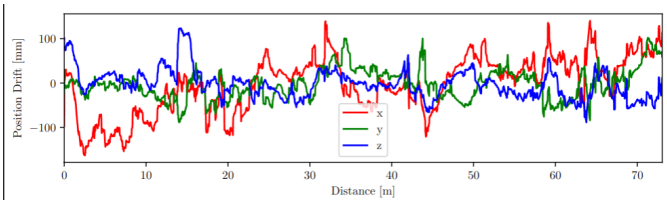

Fig. 7. Translation Error

REFERENCES

[1] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Rome, Italy, 2007, pp. 3565–3572.

[2] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robot. Autom. Lett.*, preprint version, accepted Dec. 2017, doi: 10.1109/LRA.2018.2793359.