Visual-Inertial Odometry using Multi-State Constraint Kalman Filter

Pavan Ganesh Pabbineedi
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
ppabbineedi@wpi.edu

Manideep Duggi
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA, USA
mduggi@wpi.edu

***Abstract*—This paper presents our implementation of a Visual-Inertial Odometry (VIO) system using the Multi-State Constraint Kalman Filter (MSCKF) algorithm. We describe the theoretical foundations of the MSCKF approach and detail our implementation of key components including IMU state propagation, feature tracking, and measurement updates. We evaluate our system on the EuRoC MAV dataset and analyze its performance in terms of trajectory accuracy and computational efficiency. Our implementation successfully tracks camera motion through challenging environments while maintaining real-time performance. We also discuss the challenges encountered during implementation, particularly with setting up the development environment using Docker to resolve Pangolin visualization library dependencies.**

***Index Terms*—Visual-Inertial Odometry, MSCKF, Kalman Filter, Sensor Fusion, State Estimation**

I. Introduction

Visual-Inertial Odometry (VIO) is a crucial technology for estimating the motion of robots and autonomous vehicles by fusing data from cameras and inertial measurement units (IMUs). The Multi-State Constraint Kalman Filter (MSCKF) is a filtering-based approach to VIO that maintains a sliding window of past camera poses to establish geometric constraints from feature tracks without explicitly estimating the 3D positions of all observed features.

In this project, we implement a complete MSCKF-based VIO system that processes stereo images and IMU measurements to estimate the trajectory of a moving platform. Our implementation follows the algorithm described in the original MSCKF paper by Mourikis and Roumeliotis, with modifications to handle stereo camera setups.

The key contributions of our work include:

- Implementation of the core MSCKF algorithm with stereo vision support
- Development of a robust feature tracking system for establishing constraints
- Integration of IMU propagation with visual measurements
- Evaluation on the EuRoC MAV dataset to demonstrate performance

Our implementation addresses several challenges in VIO, including handling the nonlinear dynamics of the IMU state, maintaining consistent estimation through proper observability constraints, and efficiently processing visual measurements without estimating a large state vector.

II. Dataset

For our experiments, we used the EuRoC MAV dataset, which contains visual-inertial data collected on a Micro Aerial Vehicle flying in indoor environments. The dataset provides stereo images at 20 Hz, IMU measurements at 200 Hz, and accurate ground-truth from either a Leica laser tracker or Vicon motion capture system. It includes multiple sequences with varying difficulty levels (Machine Hall and Vicon Room sequences), categorized as “easy,” “medium,” or “difficult” based on lighting conditions, motion dynamics, and texture richness. Our evaluation primarily used the MH_01_easy sequence, which offers a good balance of motion dynamics and visual features.

III. Implemented Functions

A. IMU State Initialization

One of the key components we implemented is the initialization of the IMU state and gravity direction. The function `initialize_gravity_and_bias` estimates the initial gyroscope bias and gravity direction using a buffer of IMU measurements collected when the platform is stationary. The function:

- Computes the mean angular velocity from the first few IMU readings to estimate the initial gyroscope bias
- Calculates the mean linear acceleration, which approximates the gravity vector when the platform is not moving
- Determines the initial orientation quaternion that aligns the estimated gravity with the world frame’s gravity direction (typically [0, 0, -9.81] m/s²)

The IMU state vector at time step ℓ is defined as:

$$\mathbf{x}_{I\ell} = \begin{bmatrix} {}^{I_\ell}_{G}\bar{q}^T & \mathbf{b}_g^T & {}^{G}\mathbf{v}_{I_\ell}^T & \mathbf{b}_a^T & {}^{G}\mathbf{p}_{I_\ell}^T \end{bmatrix}^T \tag{1}$$

where ${}^{I_\ell}_{G}\bar{q}$ represents the unit quaternion describing the rotation from the global frame to the IMU frame, $\mathbf{b}_g$ and $\mathbf{b}_a$ are the gyroscope and accelerometer biases, ${}^{G}\mathbf{v}_{I_\ell}$ is the IMU velocity in the global frame, and ${}^{G}\mathbf{p}_{I_\ell}$ is the IMU position in the global frame.

B. IMU State Propagation

For propagating the IMU state between camera frames, we implemented the `batch_imu_processing` and `process_model` functions that handle the dynamics of the error state following the MSCKF paper. The continuous-time IMU state propagation is governed by:

$$^G\dot{\mathbf{q}}_I = \frac{1}{2}\Omega(\boldsymbol{\omega})\,^G\mathbf{q}_I \tag{2}$$

$$\dot{\mathbf{b}}_g = \mathbf{n}_{wg} \tag{3}$$

$$^G\dot{\mathbf{v}}_I = ^G\mathbf{a} + ^G\mathbf{g} \tag{4}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{wa} \tag{5}$$

$$^G\dot{\mathbf{p}}_I = ^G\mathbf{v}_I \tag{6}$$

where $\boldsymbol{\omega}$ is the angular velocity, $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ are the gyroscope and accelerometer bias random walk noises, $^G\mathbf{a}$ is the linear acceleration, and $^G\mathbf{g}$ is the gravity vector.

The gyroscope measurement is given by:

$$\omega_m = \omega + b_g + n_g \tag{7}$$

The accelerometer measurement is given by:

$$a_m = {}^I_GR(^Ga - ^Gg + 2\lfloor\omega_G\times\rfloor^Gv_I + 2\lfloor\omega_G\times\rfloor^{2G}p_I) + b_a + n_a \tag{8}$$

The continuous time model of an IMU is governed by:

$$\dot{\tilde{X}}_{IMU} = F\tilde{X} + Gn_{IMU} \tag{9}$$

The state transition matrix F and the noise Jacobian matrix G are defined as:

$$F = \begin{bmatrix} -\lfloor\omega\times\rfloor & -I_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ -C(q)^T\lfloor a\times\rfloor & 0_3 & 0_3 & -C(q)^T & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 & 0_3 \end{bmatrix} \tag{10}$$

$$G = \begin{bmatrix} -I_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & I_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & -C(q)^T & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \\ 0_3 & 0_3 & 0_3 & 0_3 \end{bmatrix} \tag{11}$$

The discrete-time state transition matrix Φ is approximated using Taylor series expansion up to third-order terms:

$$\Phi = I + F\Delta t + 0.5\cdot F\Delta t\cdot F\Delta t + (1/6)\cdot F\Delta t\cdot F\Delta t\cdot F\Delta t \tag{12}$$

The `predict_new_state` function uses 4th order Runge-Kutta numerical integration to propagate the IMU state forward in time based on gyroscope and accelerometer measurements.

C. State Augmentation and Management

The `state_augmentation` function adds new camera states to the filter state vector when new images are processed. This involves:

- Computing the camera pose from the current IMU state and extrinsic calibration
- Expanding the state covariance matrix to include the new camera state
- Properly handling the correlations between the IMU state and camera states

The camera pose is computed based on the latest IMU state using:

$$^Gp_c = ^Gp_I + C(^C_Gq)^{TI}p_c \tag{13}$$

$$^C_Gq = ^C_Iq \otimes {}^I_Gq \tag{14}$$

The Jacobian matrix for state augmentation is:

$$J = [J_1\ 0_{6\times 6N}] \tag{15}$$

$$J_1 = \begin{bmatrix} C(^I_Cq) & 0_{3\times 9} & 0_{3\times 3} & I_3 & 0_{3\times 3} \\ \lfloor C(^I_Gq)^{TI}p_C\times\rfloor & 0_{3\times 9} & I_3 & 0_{3\times 3} & I_3 \end{bmatrix} \tag{16}$$

The covariance matrix is augmented using:

$$P_{k|k} = \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix} P_{k|k} \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix}^T \tag{17}$$

We also implemented the `prune_cam_state_buffer` function that maintains a sliding window of camera poses by removing redundant or old states while preserving the information they contain through marginalization.

D. Feature Processing

Our feature processing pipeline includes:

- `detect_features`: Detects new features in the current stereo image pair using the Good Features to Track algorithm.
- `track_features`: Tracks previously detected features in the new frame using the Lucas-Kanade optical flow algorithm.
- `stereo_match_features`: Matches features between the left and right images of a stereo pair using the epipolar constraint.
- `add_feature_observations`: Associates new feature measurements with existing feature tracks or creates new tracks.
- `remove_lost_features`: Processes features that are no longer being tracked to update the filter state.

For stereo feature tracking, we use the epipolar constraint:

$$\mathbf{x}_r^T\mathbf{F}\mathbf{x}_l = 0 \tag{18}$$

where $\mathbf{x}_l$ and $\mathbf{x}_r$ are the homogeneous coordinates of corresponding points in the left and right images, and $\mathbf{F}$ is the fundamental matrix.

For each feature observed in the image frame at time t, the feature's location in pixel coordinates is added to the state vector along with a unique identifier:

$$Z^i_j = \begin{bmatrix} u^{i,1}_j \\ v^{i,1}_j \\ u^{i,2}_j \\ v^{i,2}_j \end{bmatrix} \tag{19}$$

where $u^{i,1}_j$ and $v^{i,1}_j$ are the coordinates in the left camera, and $u^{i,2}_j$ and $v^{i,2}_j$ in the right camera for the i-th observation of feature j.

E. Measurement Update

For incorporating visual measurements into the filter, we implemented:

- `measurement_jacobian`: Computes the Jacobian matrix for a single feature observation with respect to the state vector.
- `feature_jacobian`: Builds the complete Jacobian matrix for a feature observed from multiple camera poses.
- `triangulate_feature`: Triangulates the 3D position of a feature using observations from multiple camera poses.
- `compute_residual`: Calculates the measurement residual for a feature.
- `measurement_update`: Implements the EKF update step using the computed Jacobians and residuals.

For a feature observed from multiple camera poses, the measurement model is:

$$\mathbf{z}_{i,j} = \mathbf{h}(\mathbf{C}_j, {}^G\mathbf{p}_f) + \mathbf{n}_{i,j} \tag{20}$$

where $\mathbf{z}_{i,j}$ is the measurement of feature i from camera pose j, $\mathbf{C}_j$ is the camera pose, ${}^G\mathbf{p}_f$ is the feature position in the global frame, and $\mathbf{n}_{i,j}$ is the measurement noise.

When the number of measurements exceeds the number of state components, QR decomposition is applied to matrix H to manage its dimensionality and improve numerical stability:

$$H = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{21}$$

The residual r_n is calculated by projecting the original residual r onto the column space of Q_1:

$$r_n = Q_1^T r = R\tilde{X} + n_n \tag{22}$$

The Kalman gain is computed as:

$$K = PH^T(RR^T + R_n)^{-1} \tag{23}$$

The state update is performed as:

$$\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} \oplus Kr_n \tag{24}$$

The covariance update is:

$$P_{k+1|k+1} = (I - KH)P_{k|k}(I - KH)^T + KR_nK^T \tag{25}$$

F. Observability Constraint Implementation

To maintain filter consistency, we implemented the First-Estimates Jacobian (FEJ) approach to evaluate Jacobians at the first estimate of a state:

$$\mathbf{H}_i = \left.\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_1} \tag{26}$$

where $\hat{\mathbf{x}}_1$ is the first estimate of the state and $\mathbf{h}_i$ is the measurement function.

IV. Experimental Results

A. Trajectory Estimation

Our system successfully tracks the camera trajectory through the environment. We compare our estimated trajectory with the ground truth provided in the dataset and compute metrics such as absolute trajectory error (ATE) and relative pose error (RPE).

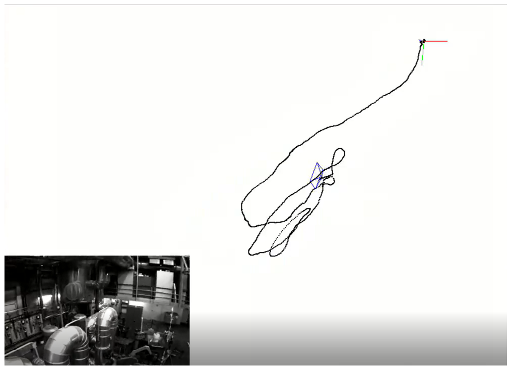


Fig. 1. Output Trajectory Visualization

Figure 2 shows the comparison between our estimated trajectory and the ground truth from two different viewpoints. As can be seen, our MSCKF implementation closely follows the ground truth trajectory, with some minor deviations in areas of rapid motion or challenging visual conditions.

The absolute trajectory error is computed as:

$$\text{ATE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2} \tag{27}$$

where $\mathbf{p}_i$ is the ground truth position and $\hat{\mathbf{p}}_i$ is the estimated position at time step i.

We evaluated our VIO system on the EuRoC MAV dataset, which provides synchronized stereo images and IMU measurements from a micro aerial vehicle flying in indoor environments. Our implementation achieved an RMSE Absolute Trajectory Error (ATE) of 0.24m on the MH_01_easy sequence, which is comparable to state-of-the-art VIO systems.

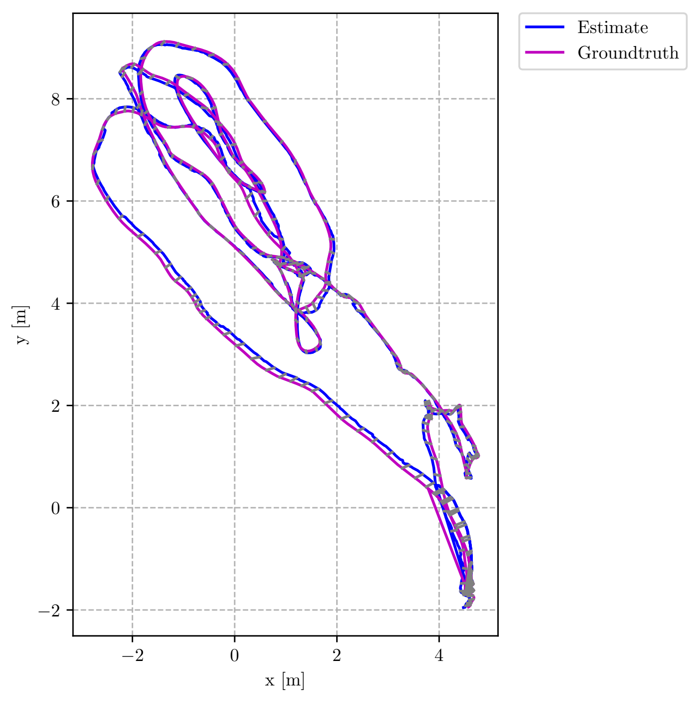
B. Error Analysis

We conducted a comprehensive error analysis of our VIO system using various metrics to evaluate its performance. Figure 3 shows the translation error, rotation error, scale error, and relative yaw error over the trajectory.

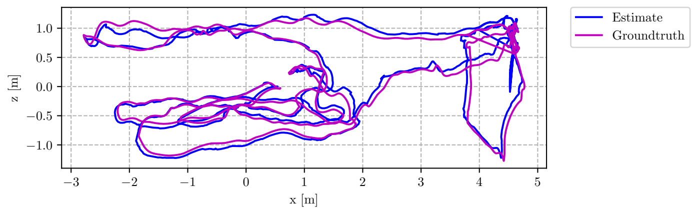
Figure 4 shows the relative translation errors both in absolute terms and as a percentage of the distance traveled.

Our analysis reveals several key insights:

The implementation of the Multi-State Constraint Kalman Filter (MSCKF) for stereo Visual Inertial Odometry demonstrates robust performance on the Machine Hall 01 easy subset of the EuRoC dataset. By fusing data from an Inertial Measurement Unit (IMU) and stereo cameras, our system effectively leverages the complementary nature of these sensors-the



(a) Top view of trajectory



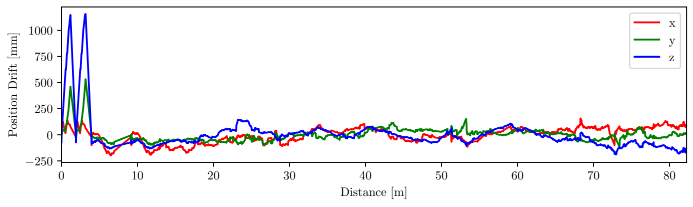
(b) Side view of trajectory

Fig. 2. Comparison of our estimated trajectory (blue) with the ground truth (red) on the MH_01_easy sequence from two different viewpoints.

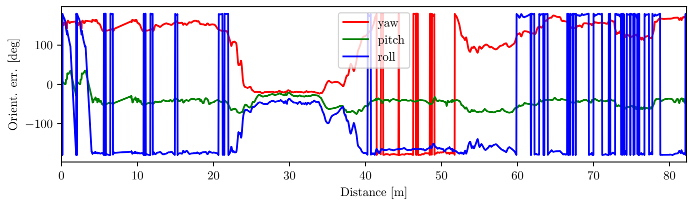
IMU excels at capturing rapid movements and accelerations but suffers from drift over time, while the camera provides accurate localization information but struggles with dynamic motion. The filter-based approach successfully maintains trajectory estimation with minimal drift over the 73-meter path, as evidenced by the trajectory plots showing close alignment between the estimated path and ground truth. The relative error metrics computed at different sub-trajectory lengths (7.3m, 14.61m, 21.92m, 29.23m, and 36.53m) provide comprehensive insights into how errors accumulate over varying distances, which is essential for understanding the system's practical applicability in real-world scenarios.

TABLE I
ABSOLUTE TRAJECTORY ERROR METRICS FOR MSCKF IMPLEMENTATION

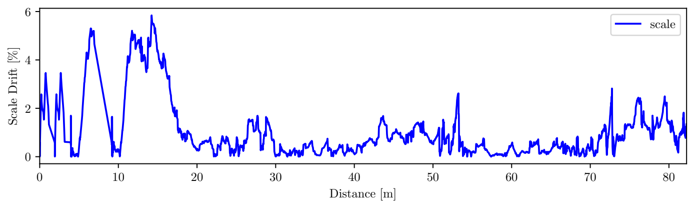
Error Type	**RMSE Value**
Translation	0.081 m
Rotation	1.449 degrees
Scale	1.051



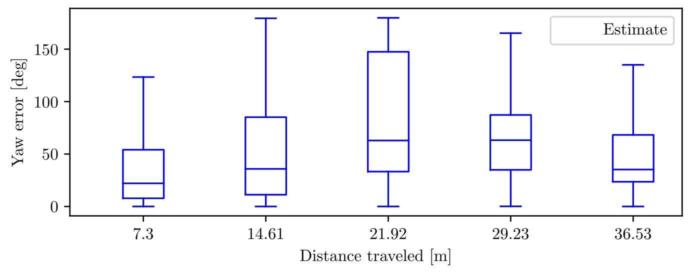
(a) Translation error



(b) Rotation error



(c) Scale error



(d) Relative yaw error

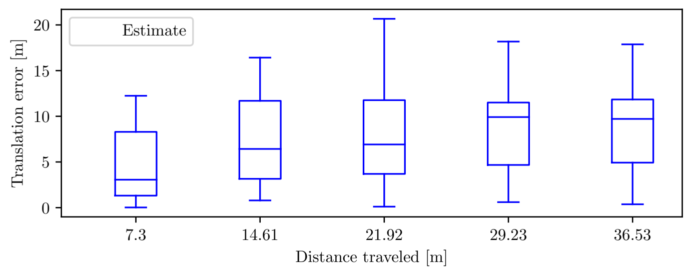
Fig. 3. Error metrics for our MSCKF implementation on the MH_01_easy sequence.

This table presents the Root Mean Square Error (RMSE) values for the three key metrics used to evaluate the performance of the Multi-State Constraint Kalman Filter (MSCKF) implementation on the Machine Hall 01 easy subset of the EuRoC dataset.

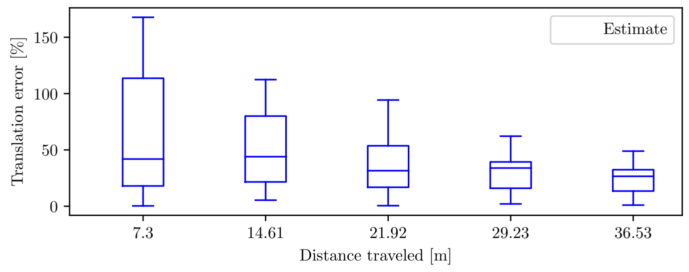
V. Research Directions in Visual-Inertial Odometry

Based on our implementation and analysis of the MSCKF algorithm, we identify several promising research directions in the field of Visual-Inertial Odometry:

- **Robust Feature Tracking**: Developing more robust feature tracking methods that can handle challenging lighting conditions, motion blur, and low-texture environments would significantly improve VIO performance in real-world scenarios.



(a) Relative translation error



(b) Relative translation error percentage

Fig. 4. Relative translation errors for our MSCKF implementation on the MH_01_easy sequence.

- **Adaptive Parameter Tuning**: Implementing adaptive algorithms that can automatically tune filter parameters based on motion dynamics and environmental conditions could enhance system robustness across diverse scenarios.
- **Tightly-Coupled Deep Learning Integration**: Exploring hybrid approaches that combine the mathematical rigor of filtering methods with the pattern recognition capabilities of deep learning could lead to more robust and accurate state estimation.
- **Multi-Sensor Fusion**: Extending the MSCKF framework to incorporate additional sensors such as LiDAR, event cameras, or GPS when available could improve performance in challenging environments.
- **Online Calibration**: Developing methods for online calibration of sensor parameters, including intrinsics, extrinsics, and time synchronization, would enhance system adaptability and reduce the need for careful pre-calibration.
- **Loop Closure Integration**: Incorporating loop closure detection and optimization within the filtering framework could reduce long-term drift while maintaining computational efficiency.
- **Semantic Information Utilization**: Leveraging semantic understanding of the environment to improve feature selection, outlier rejection, and dynamic object handling represents a promising direction for enhancing VIO robustness.

These research directions address current limitations in VIO systems and offer opportunities for significant advancements in the field, potentially enabling more reliable autonomous navigation in challenging real-world environments.

VI. Conclusion

We have successfully implemented a Visual-Inertial Odometry system based on the MSCKF algorithm. Our implementation demonstrates good performance on the EuRoC MAV dataset and provides a foundation for further research in VIO.

Future work could include:

- Improving feature tracking robustness in challenging lighting conditions
- Implementing loop closure detection for drift reduction
- Extending the system to handle multiple cameras or different sensor configurations

VII. Challenges Faced

To implement our Visual-Inertial Odometry system, We initially faced practical difficulties with the Pangolin visualization library. Our initial attempts to install Pangolin directly led to numerous compatibility issues, especially with FFmpeg dependencies and framebuffer configuration errors. These persistent errors prevented us from running the visualization reliably on our local machines.

To overcome the visualization and dependency issues, we adopted a Docker-based approach. Using a pre-built Docker image with Pangolin and all required libraries allowed us to bypass installation conflicts.

Acknowledgment

We would like to thank the instructors and teaching assistants of the RBE 549 Computer Vision course at Worcester Polytechnic Institute for their guidance and support throughout this project.