RBE/CS 549 Computer Vision Project 4 Visual Inertial Odometry

Puneet Shetty
MS in Robotics
Worcester Polytechnic Institute
Email: ppshetty@wpi.edu

Edwin Clement
MS in Robotics
Worcester Polytechnic Institute
Email: eclement@wpi.edu

***Abstract*—This project develops an optimized Visual Inertial Odometry (VIO) system utilizing data from a stereo camera and an Inertial Measurement Unit (IMU) to enhance localization accuracy while minimizing computational demands. Traditional methods relying solely on stereo cameras are computationally intensive and prone to failures in high-speed scenarios due to motion blur. By integrating an IMU, our approach maintains high performance even under rapid movements and accelerations. This system employs a filter-based method using the Multi-State Constraint Kalman Filter (MSCKF), selectively tracking fewer features and leveraging mathematical techniques to determine the relative poses of cameras, thereby efficiently localizing the agent in its environment. The implementation details and performance metrics of this stereo VIO system are thoroughly documented in the subsequent sections.**

Index Terms*— Visual Inertial Odometry, Multi-State Constraint Kalman Filter, Inertial Measurement Unit, Aerial Systems, Perception & Autonomy, SLAM

I. INTRODUCTION

This project's main objective is to precisely extract scale and depth from imagery, which is a difficult issue because single-camera setups cannot offer depth information without previous environmental context. In order to get around these restrictions, we use a stereo camera system, which enables feature matching between its two points of view to estimate depth. Nevertheless, in situations where motion blur occurs—a frequent problem in robotic applications—feature matching on its own can be computationally taxing and inefficient.

We integrate an Inertial Measurement Unit (IMU) to address these issues. With its six degrees of freedom (DoF) can measure both linear and angular accelerations, this gadget performs exceptionally well in situations where typical cameras might struggle due to fast motions or vibrations. IMUs can give useful information in high-speed situations, but they can also drift, or accumulate mistakes over time.

A strong multi-modal fusion framework is formed by the synergy between the more stable but slower visual data from cameras and the high-frequency, drift-prone IMU readings. This fusion is essential for improving the camera's depth estimating skills and correcting drift in IMU data. By using this combined method, we hope to solve important problems in visual inertial odometry by providing an accurate and dependable way to determine the posture of the camera and subsequently retrace the depth from photos.

II. DATA

To evaluate our approach, we would use Machine Hall 01 easy (MH 01 easy), a subset of the EuRoC dataset. Utilizing a 6-DoF sensor mounted on a quadcopter following a predetermined path, the data was gathered. Vicon Motion capture technology, which provides sub-millimeter accuracy, served as the source of ground truth for the system.

III. IMPLEMENTED FUNCTIONS

The Multi-State Constraint Kalman Filter (MSCKF) implementation has initial code available. To finish the model's implementation, we made updates to the following routines in the msckf.py Python file.

A. Initializing Gravity and Bias in IMU

The `initialize_gravity_and_bias` function is essential for determining the Inertial Measurement Unit's (IMU) basic characteristics in relation to the world frame. First, the function averages the first few data from the IMU message buffer to get the angular and linear velocities of the IMU. These angular velocity values are averaged to set the gyroscopic bias, provide a baseline for future changes, and account for intrinsic sensor drift.

Concurrently, gravitational acceleration is used as a reference point to calculate the initial orientation of the IMU with respect to the world frame. Through the computation of the required rotation to synchronize the IMU's frame of reference, the function aligns the downward-pointing gravity vector with the normalized average of the linear acceleration measurements, which is assumed to equal the standard gravitational force of 9.81 m/s^2 with that of the world.

Quaternion form, which effectively captures the rotational difference between the IMU's original arbitrary orientation and the right alignment with the global frame, is used to express this rotational alignment. The orientation state of the IMU is then set to the computed quaternion. This quaternion is integrated with other crucial factors by the state vector, represented by the symbol X_I:

- $\mathbf{I}_q$: Quaternion representing the rotation from the inertial frame to the body frame (IMU frame).
- $\mathbf{b}_g$ and $\mathbf{b}_a$: Biases for the gyroscope and accelerometer, respectively.
- $\mathbf{G}_v^I$ and $\mathbf{G}_p^I$: Velocity and position of the body frame (IMU) in the inertial frame.

This comprehensive initialization ensures that subsequent navigational computations are accurately aligned.

B. Batch IMU processing

The `batch_imu_processing` function controls, within a given time limit, the messages in the IMU message buffer. Every IMU message is processed by applying the process model to the states that have been saved until the IMU timestamp coincides with the time boundary. The function applies the process model iteratively for each message inside the time bound, making sure that all IMU data up to the boundary is used to update the state as described in Section C. In order to preserve buffer efficiency and system accuracy, the function updates the current IMU identification to that of the next state when the time limit is reached and clears the buffer of any messages that are no longer needed.

C. Process model

The `process_model` function encapsulates the prediction step of the Extended Kalman Filter for an IMU within a visual-inertial odometry system. The goal of this function is to propagate the IMU's state forward in time based on the current state and the IMU measurements of angular velocities and accelerations.

1) **State Transition and Noise Covariance Matrices (F and G):**
   - The state transition matrix F captures the dynamics of how the state evolves over time due to system inputs and is defined by the IMU's kinematic equations. It is constructed as follows:

$$F = \begin{bmatrix} -[\boldsymbol{\omega}\times] & -I_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 \\ -C(\boldsymbol{q}^T)\frac{\partial[\boldsymbol{a}]\times}{\partial\boldsymbol{a}} & 0_3 & 0_3 & -C(\boldsymbol{q}^T) \\ 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 \end{bmatrix}$$

   - The noise covariance matrix G maps the process noise to the appropriate state variables and is constructed to reflect the dimensions and effects of gyroscopic and accelerometric noise.

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

2) **Matrix Exponential Approximation (ϕ):**
   - The matrix exponential of F, denoted as ϕ, is approximated using a third-order Taylor expansion. This is crucial for integrating the continuous-time system over a discrete time step, which in this case is performed using the Runge-Kutta method:

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

   Here, Δt is the time step, and F^2, F^3 represent the matrix F squared and cubed, respectively.

3) **State Propagation (Runge-Kutta Integration):**
   - The RK4 integration method is applied to the IMU's orientation, velocity, and position, using the corrected IMU measurements (gyroscopic and accelerometric data) to compute the new state estimates at time $t+\Delta t$.
   - The specific RK4 steps are not detailed here, but the process involves calculating intermediate states and then taking a weighted average of these to achieve the final state prediction.

4) **Covariance Propagation:**
   - The state covariance is updated using the equation:

$$P_{II_{k+1|k}} = \Phi_k P_{II_{k|k}} \Phi_k^T + Q_k$$

   Where $P_{II_{k+1|k}}$ is the covariance of the IMU state at the next time step, Φ_k is the discrete-time state transition matrix (approximated by ϕ), and Q_k is the process noise covariance, which can be integrated over the time step as shown previously.

In summary, the `process_model` function predicts the IMU's state at a future time instant based on the current state and the latest measurements. It updates both the state estimates and their associated uncertainties (covariances), preparing the filter for the next measurement update cycle.

D. Predict new state

The `predict_new_state` function within the visual-inertial odometry process model employs a 4th order Runge-Kutta numerical integration scheme to update the estimated IMU state, encompassing orientation, velocity, and position. This method strikes a balance between computational efficiency and the accuracy required for VIO applications. Discretization of the continuous state equations through Runge-Kutta integration allows for prediction of the new state at discrete time intervals.

The function is described as follows:

1) **Orientation Update**: The orientation of the IMU is represented by a quaternion $\mathbf{q}$. The integration of quaternions is performed in a way that preserves their norm, using the omega matrix $\Omega(\boldsymbol{\omega})$:

$$\Omega(\boldsymbol{\omega}) = \begin{bmatrix} -[\boldsymbol{\omega}\times] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}$$

2) **Velocity and Position Update**: Velocity and position are updated using the 4th order Runge-Kutta method:

$$k_1 = f(t_n, y_n)$$
$$k_2 = f\left(t_n + \frac{\Delta t}{2}, y_n + k_1\frac{\Delta t}{2}\right)$$
$$k_3 = f\left(t_n + \frac{\Delta t}{2}, y_n + k_2\frac{\Delta t}{2}\right)$$
$$k_4 = f(t_n + \Delta t, y_n + k_3\Delta t)$$

By substituting the functions of velocity and position in place of y, the updates for the IMU state are computed.

3) **Integration into `predict_new_state` Function**: The function follows these steps:
   - Compute angular velocity and acceleration for the Runge-Kutta integration.
   - Apply the 4th order Runge-Kutta integration to update the IMU's orientation using the Ω matrix.
   - Apply the same integration method to update the IMU's velocity and position based on their kinematic equations.
   - Convert the resulting orientation into quaternion form to maintain the unit norm property.
   - Update the velocity and position states with the new estimates from the Runge-Kutta integration.
   - Assign these new values as the current state for the subsequent state prediction.

This method advances the IMU's state in discrete steps with precision, adeptly handling the nonlinear nature of IMU state changes, making it highly suitable for precise motion tracking required in VIO tasks.

E. State Augmentation

In the `state_augmentation` function, we compute the state covariance matrix to propagate the uncertainty of the state. This process starts by obtaining the IMU and camera state values corresponding to the rotation from the IMU to the camera and the translation vector from the camera to the IMU. A new camera state is then added to the state server utilizing the initial IMU and camera states.

The state augmentation Jacobian, denoted as J, is updated as follows:

$$J_r = \begin{bmatrix} C\left(\mathbf{q}_I^\top\right) & 0_{3\times 9} & 0_{3\times 3} & I_3 & 0_{3\times 3} \\ -C\left(\mathbf{q}_I^\top\right)[\mathbf{p}_{c\times}] & 0_{3\times 9} & I_3 & 0_{3\times 3} & I_3 \end{bmatrix}$$

where $C(\mathbf{q}_I^\top)$ is the rotation matrix from the quaternion, and $[\mathbf{p}_{c\times}]$ is the skew-symmetric matrix of the camera position vector.

Next, we resize the state covariance matrix and propagate the covariance of the IMU state. The full propagation of the uncertainty is represented as:

$$P_{k+1|k} = \begin{bmatrix} P_{II_{k+1|k}} & \Phi_k P_{IC_{k|k}} \\ P_{IC_{k|k}}^\top \Phi_k^\top & P_{CC_{k|k}} \end{bmatrix}$$

Finally, the augmented covariance matrix $P_{k|k}$ is updated as:

$$P_{k|k} = \left(\frac{I_{21+6N} + \delta}{J}\right) P_{k|k} \left(\frac{I_{21+6N} + \delta}{J}\right)^T$$

Afterward, the updated covariance matrix is stored back in the server.

The function also updates the camera pose using the relationship between the IMU and camera frames. With the new camera pose, the function appends this pose to the state vector. The covariance matrix P, which encapsulates the system's uncertainty, is augmented using the Jacobian J obtained from the above calculations.

F. Add feature observations

The `add_feature_observations` In order to manage the feature data extracted from camera images, function is essential. Prior to processing each feature, this function determines the current IMU state ID, counts the total number of features, and receives a feature message as input. Subsequently, it adds each feature, one at a time, to the map server; features that are already there are updated with new observations, and those that are not are added with a unique feature ID. This makes sure that all features are consistently tracked across all states, and that the map server is updated as needed. In addition, the function computes the tracking rate, which is the ratio of tracked features to all features. This yields useful metrics regarding the effectiveness and precision of the feature tracking procedure.

G. Measurement update

The `measurement_update` function in a visual-inertial odometry system updates the state estimates and state covariance matrix utilizing the measurement model. This model incorporates the Jacobian matrix H and measurement noise to refine the state estimates based on observed measurements.

The measurement update process involves several steps:

1. Compute the Kalman gain:

$$K = P\tilde{H}^T\left(\tilde{H}P\tilde{H}^T + R_n\right)^{-1}$$

where K is the Kalman gain, P is the state covariance matrix, $\tilde{H}$ is the measurement Jacobian matrix, and R_n is the measurement noise covariance matrix.

2. Update the state estimate using the Kalman gain and the innovation vector r_n:

$$\Delta X = K r_n$$

where ΔX represents the change in the state estimate.

3. Update the state covariance matrix to reflect the reduced uncertainty after incorporating the measurement:

$$P_{k+1|k+1} = (I_s - KH)\, P_{k+1|k}\, (I_s - KH)^T + K R_n K^T$$

where I_s is the identity matrix of appropriate size.

4. QR decomposition is used to reduce the complexity of the Jacobian matrix H, thus reducing the computation required for the update:

$$H_x = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_h \\ 0 \end{bmatrix} = Q_1 R_1$$

where Q_1 and Q_2 are unitary matrices whose columns form bases for the range and nullspace of H_x, respectively, and R_h is an upper triangular matrix from the QR decomposition of H.

5. The final residual r is computed as the difference between the measurement z and the estimated measurement $\hat{z}$, which depends linearly on the state error:

$$r = z - h(\hat{x}, f) = H_x \tilde{X} + H_f \tilde{f} + n$$

where H_x is the Jacobian matrix mapping the state error to the measurement domain, H_f is the feature Jacobian, $\tilde{X}$ is the state error, $\tilde{f}$ is the feature position error, and n is the noise.

6. The camera poses are updated, and these poses are appended to the state vector. Accordingly, the state covariance matrix P is augmented to reflect the new information from the measurements.

7. The function ultimately projects the residual r onto the left nullspace of the matrix H_f, ensuring the residual is independent of the errors in the feature coordinates, allowing the EKF update to be performed optimally except for inaccuracy caused by linearization.

IV. Results & Analysis

Important insights into the visual-inertial odometry process are revealed by the Multi-State Constraint Kalman Filter (MSCKF) implementation findings, especially when using camera observations for recursive state estimation. It is noted that misleading local minima and inconsistent results can result from the non-linear character of the measurement model and the camera data's sensitivity to noise. Incorporating a feature depth parameterization into the measurement model efficiently addresses the issues arising from mistakes in linearization. Further restrictions on feature tracking over consecutive frames are seen; these are frequently caused by noise, a narrow field of vision, and occlusions, which might cause the feature tracking algorithm to malfunction.

Absolute Trajectory Error for Rotation (RMSE):
89.38995324911025
Absolute Trajectory Error for Translation (RMSE):
0.08209740661564874

The examination of data from the Machine Hall 01 easy (MH 01 easy) subset of the EuRoC dataset validates the practical implementation of these results. The trajectory output is in good agreement with the expected trajectory, supporting the applicability of the MSCKF technique in this situation. In addition to the code files, the related video output provides a visual depiction of the system's functionality, which facilitates the assessment and examination of the MSCKF implementation in an actual environment.

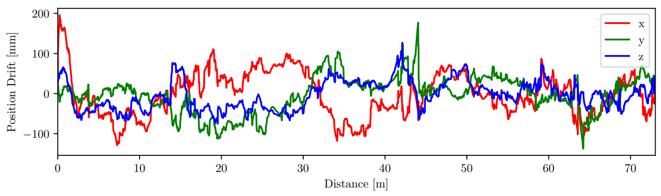

Fig. 1. Translation Error

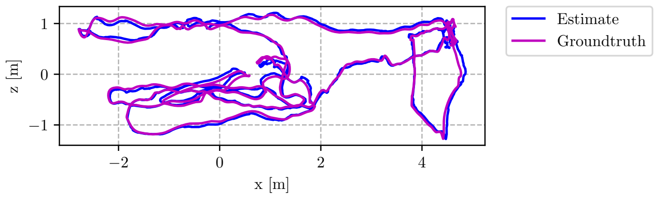

Fig. 2. Trajectory Comparison (Side)

V. Conclusion

In this project, we overcame many implementation obstacles and improved system performance by effectively implementing and optimizing important functionalities inside a stereo visual inertial odometry framework. To ensure accurate and computationally efficient posture prediction, we concentrated our efforts on integrating and optimizing different parts of the Multi-State Constraint Kalman Filter (MSCKF), such as state augmentation and exact feature tracking. The research successfully addressed issues with data fusion and system scalability, demonstrating the efficient application of sophisticated mathematical models and algorithmic techniques in real-time navigation scenarios. In addition to reducing the computing load, this study lays the groundwork for future advancements in optical inertial odometry systems.

References

[1] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," in *arXiv preprint arXiv:1712.00036*, 2018.

[2] Anastasios I. Mourikis and Stergios I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 3565-3572.

[3] Kumar Robotics, "msckf_vio," GitHub repository, [Online]. Available: https://github.com/KumarRobotics/msckf_vio.

[4] RBE 549, "Project 4: Spring 2024," course webpage, 2024. [Online]. Available: https://rbe549.github.io/spring2024/proj/p4/.

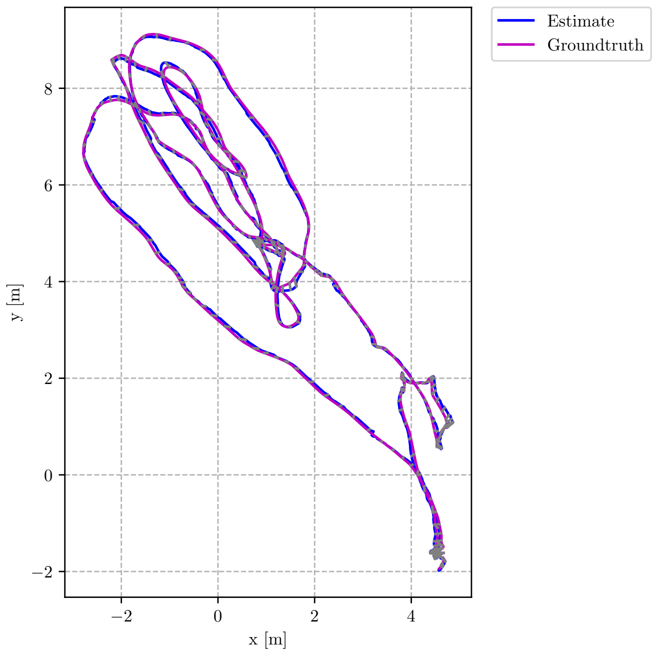

Fig. 3. Trajectory Comparison (Top)

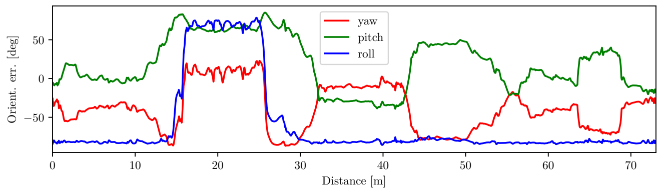

Fig. 4. Rotation Error

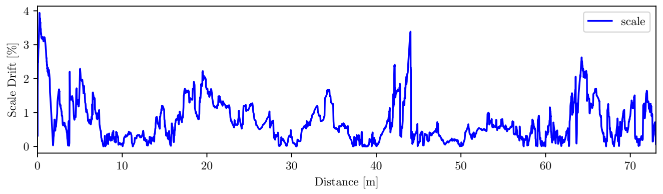

Fig. 5. Scale Error

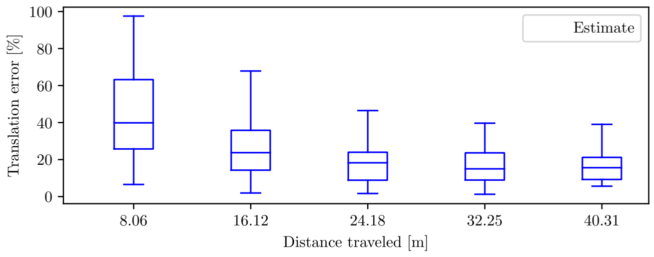

Fig. 6. Box graph of Translation Error (Percentage)

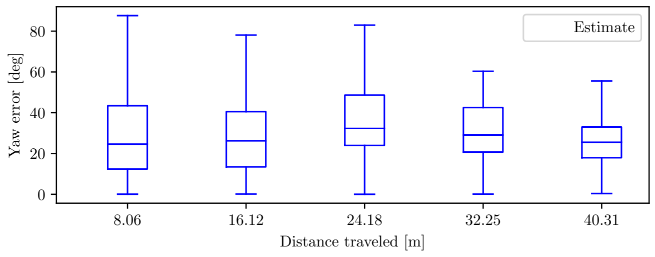

Fig. 7. Box Graph of Yaw Error