Evaluating Extended Kalman Filter Performance for Dead Reckoning on a Ground Rover using IMUs and Star Tracker

Erin Hong
Robotics & Autonomous Systems
Stanford School of Engineering
Los Angeles, USA
erin.y.hong@gmail.com

Alexander Ivanov
Robotics & Autonomous Systems
Stanford SCPD
Sunnyvale, California
alex1999@stanford.edu

Karthik Pythireddi
Robotics & Autonomous Systems
Stanford SCPD
Newark, California
karthikpythireddi@gmail.com

Ryan Udell
Dept. of Electrical Engineering
Stanford School of Engineering
Stanford, USA
ryan.udell@stanford.edu

***Abstract*—In this paper, the team demonstrates state estimation for a ground robot (also called a ground rover or vehicle) using inertial measurement units (IMUs) and a star tracker. After reviewing the current state-of-the-art state estimation algorithms through a literature review, the team models an autonomous rover and implements the Extended Kalman Filter (EKF) to compare its benefits and drawbacks. The team tested the feasibility of the EKF estimation solution in several scenarios, including differential data rates and multiple trajectories, in which sensor information is received asynchronously.**

***Index Terms*—autonomous ground vehicle, dead reckoning, inertial measurement unit, star tracker, Extended Kalman Filter**

I. Introduction

Robot autonomy depends on the ability to localize and navigate in a given environment. Extraterrestrial rovers typically operate in GPS-denied environments; thus, it is important to use a sensor suite and state estimation techniques for autonomous movements. The team is interested in analyzing a ground robot that is operating in a GPS-denied environment, relying on a dead reckoning algorithm using inertial measurement units (IMUs) and a star tracker to estimate its state. In Section II, the team presents the current state-of-the-art research in dead reckoning using the aforementioned sensors, as well as various state estimation algorithms. In Section III, the team formulates the problem statement and proposed work. The team details the dynamics and measurement model in Sections III-B and III-C. Lastly, the team displays the estimation results and conducts an analysis based on the results in Sections V and VI, respectively.

II. Literature Review

A. Dead Reckoning and IMUs

Ground rovers typically use an IMU, which contains a 3-axis accelerometer and gyroscope that are used for localization purposes. The dynamic state of the system is determined by the inertial sensors, where gyroscopes measure the rates of rotation, and accelerometers are used to measure the linear acceleration with respect to the inertial frame, as illustrated in [1].

Dead reckoning is used in a wide variety of applications. It estimates the current pose of the robot using the previously determined position along with the speed and heading angle [2]. From the work done by F.Shkurti, I. Rekleitis, M.Scaccia and G.Dudek in 2011 [1], wheel encoders were used to provide the wheel rotational rate for the purposes of localization. This approach suffers from both systematic and non-systematic errors, which add to the unbounded final position error accumulated over time. In the paper by Borenstein J., Feng L., [14] a method called gyrodometry is introduced. In this method, odometry errors when traveling over a uneven surface were compensated by a gyroscope and accelerometers.

As per the work done in [1], IMU-based inertial navigation systems have high-bandwidth output but short-lived stability; this is due to the noisy signals which often lead to trajectory drift as time progresses. Micro-electro-mechanical-systems (MEMS) IMUs have been used in many applications due to their low cost, small size, and low power consumption. However, they need frequent calibration to get good repeatability. Usually, accurate calibration is used to model the error sources using Kalman Filter-based algorithms. In another paper [4], the researchers considered a WiFi-based localization method, which they treated as global localization with respect to a local map of the indoor environment. Since WiFi signals at a given robot position keeps altering, this triggers errors in the robot's trajectory. By introducing the dynamic constraints of a wheeled robot in a geofenced environment, they minimized the errors and discontinuities using state estimation processes such as the EKF, UKF, and a particle filter with odometry and gyroscopes.

In other applications such as underwater robots (UWR), vision-based sensory data is fused with the IMU. As per the work done by Bong, Woo-sung, Woo-jin and Kwang-Ryul in 2011 [5], UWR often struggles with noise, interference, and distortion of the visual light spectrum because of the various reasons such as spectrum absorption by the water as well as the scattering of the visible light spectrum by the suspended plankton. Also, the visibility of the light decreases as the depth increases and, in order to counter the effects of the vision-based pose estimation, the researchers fused the IMU data with vision data for a better robot pose estimate. The IMU provides a noisy estimate of the vehicle's dynamics in strong underwater currents, and the monocular camera provides the corrective motion estimates. Since the IMU data propagates noise, the researchers considered the frame-to-frame visual features and used bundle adjustment to account for the influence of noise. The difference between the estimated 3D position and true position determines the corrected state and covariance.

B. Dynamics and Sensor Models

To simulate and properly test the state estimation algorithms, a dynamics model of the lunar rover needs to be developed. The kinematics, dynamics, and sensor models of the robot are dependent on the rover configuration and the simplifying assumptions that are made. When simulating the effects of a rover traversing over various different surfaces, models can be used to approximate the change in environment that the rover is experiencing. For example, complex dynamics of wheel-bedrock contact and wheel-soil interaction can be taken into consideration to formulate better estimates of the vehicles position when using the dead-reckoning navigational approach [6], [16]. Using these more complex models allows for more accurate performance of the state estimation algorithms.

Additionally, it is important to take into account sensor models that will be used to represent the measurements available to the autonomous vehicle. Papers such as [6] compute odometry errors using encoders on the wheels of a differential drive mobile robot while forgoing the use of accelerometer and gyroscope measurements. Research to incorporate multiple measurements and develop sensor models for rovers in a GPS-denied environment has been extensively researched [5], [8], [14].

Another example of a localization method develops a model using quaternions to determine an exact location of a lunar rover using a single star measurement via a star tracker [9]. This star tracker works best with a wide field-of-view and a high resolution that can work independently of Earth or a satellite network. Pitch and roll measurements of the rover can be combined with the quaternions found via the star tracker to estimate the exact location of a rover on a lunar surface or any planetary body.

C. State Estimation Algorithms

Previous studies have shown that a low error can be obtained when combining the EKF with a vision algorithm to estimate the position of a single lunar rover [10]. An EKF is applied to input IMU data to estimate the Euler angles which are then combined with the vision distance data via spherical coordinate systems. The application of the EKF combined with a vision algorithm can also be applied to many other applications such as drones and airplanes.

Additionally, a team has developed a state estimation architecture that utilized two cascaded Kalman Filters rather than use a single Kalman Filter [11]. In this research, a tractor is using GPS as well as radar and gyroscope to estimate its position and velocity; however, since the estimation states are not highly coupled with the tractor dynamics, they were able to implement separate Kalman Filters to prevent dynamics modeling errors from corrupting the navigation states.

Another state estimation algorithm that is useful in this application is a particle filter. A particle filter is a robust, Monte Carlo method that uses weighted particles to approximate a probabilistic distribution of a dynamic system model. The filter has been shown to be able to compute dead reckoning problems and diagnose faults simultaneously [12]. In previous works, Kalman Filters were used to generate fault models; however, as the number of fault models increased, the number of Kalman Filters increased as well. Particle filters can be deployed to minimize this increase in complexity, but these filters still have their own drawbacks including a loss of diversity in sampling importance.

D. State Estimation Extensions

In addition to the traditional methods of using the EKF for IMU and dead reckoning state estimation, there are other methods that are being explored such as using artificial intelligence. A research team has recently investigated using recurrent neural networks (RNNs) in addition to the EKF [13]. By training on image data from a ground rover, the RNN is able to adjust process and measurement noise covariance matrices as well as adapt parameters for the EKF.

III. Problem Statement

The team modeled a single agent ground rover and estimated its position and orientation by using measurements from inertial measurement units (IMUs) and star tracker. More specifically, a rover was modeled using the simplified bicycle kinematic model, in which the front wheel is steered, rather than using a more simply differential drive model [17]. Due to the simplified kinematics, significant process noise had to be used in order to account for dynamic effects that were not modeled. This process noise modeled the nonlinear dynamics of the rover's position, velocity, and steering and pitch angles. Additionally, sensor models were developed to incorporate noise in order to represent the inaccuracies in the measurements. The output of the sensor model was used as the input to the EKF update step in order to compare and contrast the performance of different trajectories and sensor data rates.

The main focus of the paper is to explore the EKF's robustness and compare the results from various trajectories. These comparisons included how accurately the estimated state was able to follow the true trajectory. In addition to the EKF, the UKF and PF were also attempted. These three algorithms are the standard for many of the most robust models on the commercial market as described in the Section II. The team also suggests additional future work to extend this project into a more robust experiment for a potential continuation of the project beyond the course.

A. Coordinate Frames

Since the topic considers a rover in a GPS-denied environment, it is necessary to establish multiple coordinate frames to represent the problem. In addition to the inertial (fixed) frame, the team has defined two additional frames: center frame (C) and front wheel frame (F). These coordinate frames are depicted in Figure 1. The center (C) frame is attached to the midpoint between the two wheels. The origin of C is considered to the position of the rover in the inertial frame and the location at which jerk and acceleration act on the body. The F frame is attached to the front steering wheel and is the point at which the steering acceleration acts.

B. State, Control Inputs, & Dynamics

The team set up the proposed rover problem using a 12-state vector containing the position, velocity, and acceleration of the rover in addition to the rover's pitch (θ), pitch angular velocity ($\dot{\theta}$), steering angle (δ), steering angular velocity ($\dot{\delta}$), and heading angle (ψ). The robot position, pitch angle and its rate, and heading angle are expressed in the inertial frame while steering angle and its rate are in the robot local frame. The state vector is defined in Equation 1.

$$x = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ a_x \\ a_y \\ \theta \\ \dot{\theta} \\ \delta \\ \dot{\delta} \\ \psi \end{bmatrix} \tag{1}$$

During the development of the model, the team considered using acceleration along the x-axis of the body frame as an input. However, to simplify the model, the team decided to input jerk and climbing angular acceleration to the robot (C) frame and input a steering angular acceleration in the front

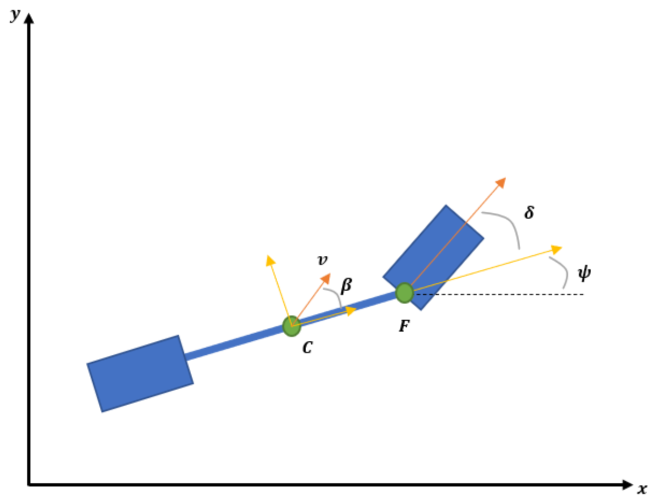

Fig. 1. Rover Reference Frame

wheel (F) frame. A climbing acceleration is included to control travel in the z direction in this 3D problem. The control input is shown in Equation 2.

$$u = \begin{bmatrix} j \\ \ddot{\theta} \\ \ddot{\delta} \end{bmatrix} \tag{2}$$

Using the state vector and the control inputs defined in Equations 1 and 2, the team produced a linearized version of the dynamics model function shown in Equations 3 and 4.

$$\mathbf{x_{t+1}} = f(x_t, u_t) + W_t \tag{3}$$

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ v_{xt+1} \\ v_{yt+1} \\ a_{xt+1} \\ a_{yt+1} \\ \theta_{t+1} \\ \dot{\theta}_{t+1} \\ \delta_{t+1} \\ \dot{\delta}_{t+1} \\ \psi_{t+1} \end{bmatrix} = \begin{bmatrix} v_{xt}\Delta t cos(\psi) + x_t \\ v_{xt}\Delta t sin(\psi) + y_t \\ v_{xt}\Delta t sin(\theta) + z_t \\ a_{xt}\Delta t + v_{xt} \\ a_{yt}\Delta t + v_{yt} \\ j_t\Delta t cos(\beta) + a_{xt} \\ j_t\Delta t sin(\beta) + a_{yt} \\ \dot{\theta}\Delta t + \theta_t \\ \ddot{\theta}\Delta t + \dot{\theta}_t \\ \dot{\delta}\Delta t + \delta_t \\ \ddot{\delta}\Delta t + \dot{\delta}_t \\ \frac{\sqrt{v_x^2+v_y^2} cos(\beta) tan(\delta)}{L}\Delta t + \psi_t \end{bmatrix} + W_t \tag{4}$$

In the above equation, $\beta = tan^{-1}(\frac{l_r tan(\delta)}{L})$, where L is the length of the wheel base and l_r is the length from the C frame to the rear wheel.

Utilizing the linearized model, the team took the Jacobian of the model to find the A matrix shown in Equations 5 through 8 to be used in the EKF filter.

$$A_{pos} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \Delta t cos(\psi_t) & 0 & \Delta t sin(\theta_t) \\ 0 & \Delta t sin(\psi_t) & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & v_{xt}\Delta t cos(\theta_t) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -v_{xt}\Delta t sin(\psi_t) & v_{yt}\Delta t cos(\psi_t) & 0 \end{bmatrix} \tag{5}$$

$$A_{vel/accel} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \Delta t & 0 & 1 & 0 \\ 0 & \Delta t & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

$$A_{angle} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma \\ 0 & 0 & 0 & 0 & \epsilon \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & \zeta \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

In the above equation, γ, ϵ, and ζ equals the following:

$$\gamma = \frac{v_x(v_x^2+v_y^2)^{-\frac{1}{2}}cos(\beta)tan(\delta)}{L}$$
$$\epsilon = \frac{v_y(v_x^2+v_y^2)^{-\frac{1}{2}}cos(\beta)tan(\delta)}{L}$$
$$\zeta = \frac{\sqrt{v_x^2+v_y^2}L^2sec(\delta)}{(L^2+l_r^2tan^2(\delta))\sqrt{L^2+l_r^2tan^2(\delta)}}\Delta t$$

$$A = \begin{bmatrix} A_{pos}^T \\ A_{vel/accel}^T \\ A_{angle}^T \end{bmatrix} \tag{8}$$

C. Sensors & Measurement Model

For the sensor model, the team considered two IMUs, each of which consists of an accelerometer and a gyroscope. In the measurement model, one IMU is located at the point C, for which the accelerometer measures acceleration with respect to the body, and the gyroscope measures the angular velocities for pitch and steering. The second IMU is located at point F and rotates with the front wheel; this gyroscope measures the steering rate ($\dot{\delta}$). The star tracker is located at C and provides inertial pitch and heading angles. The locations of the sensors can be seen in Figure 1. The IMUs and star tracker take into account turn-on bias and noise as represented by b and V respectively.

To simulate higher fidelity performance characteristics, the team decided to reference flight hardware to model for the simulation. For the IMU, the LN-200S from Northrop Grumman was selected as it was used on previous Mars Rovers [15]. As for the star tracker, the team chose the ST-16RT2 from Rocket Lab [16]. The specification sheets were used to get the bias and noise values to include in the model.

The accelerometer is described by a simple sensor model given by Equation 9, and the gyroscope is given by Equation 10

$$a_{measured} = \begin{bmatrix} a_{a_x} + b_{a_x} \\ a_{a_y} + b_{a_y} \end{bmatrix} + V_t \tag{9}$$

$$\omega_{measured} = \begin{bmatrix} \dot{\delta} + b_{\dot{\delta}} \\ \dot{\theta} + b_{\dot{\theta}} \end{bmatrix} + V_t \tag{10}$$

The team combined Equations 9 and 10 to produce equation 11, which combines the accelerometer model and the gyroscope model into single measurement model.

$$IMU_{measured} = \begin{bmatrix} a_{measured} \\ \omega_{measured} \end{bmatrix} \tag{11}$$

As mentioned before, the IMUs provide measurements at the respective points C and F to simplify the measurement model by measuring the states directly. The star tracker provides pitch angle (θ) and the heading angle (ψ). The final measurement vector includes six measurements: linear accelerations in the x and y directions of the C frame, the steering angular rate, the pitch angular rate, the pitch angle and the steering angle. The final measurement model equation is shown in Equation 12.

$$y_{measured} = \begin{bmatrix} a_{a_x} + b_{a_x} \\ a_{a_y} + b_{a_y} \\ \dot{\delta} + b_{\dot{\delta}} \\ \dot{\theta} + b_{\dot{\theta}} \\ \theta_t \\ \psi_t \end{bmatrix} + V_t \tag{12}$$

The team took the Jacobian of the measurement model to find the C matrix shown in equation 13.

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T + V_t \qquad (13)$$

IV. Methods and Approach

A. Algorithms Used

The team used the EKF algorithm to compare and contrast the drawbacks and benefits of the location of a star tracker. The EKF generalizes the prediction step by using the linearized nonlinear function through Taylor series expansion and corresponding Jacobian matrices [14]. The EKF is used as the counterpart the the Kalman Filter for nonlinear dynamical systems, and its pseudocode is seen in the following section. An extension of this project is to utilize a UKF and PF. More information about these algorithms and the potential project extension can be read in the section VII.

B. Code Setup

To prepare for the simulation, the code was developed in Python3 with a main script along with several helper functions. The simulation was set for 100 seconds with a loop rate of 100Hz. Commands for jerk and angular acceleration of the steering and climbing angle were manually specified through trial and error. The kinematic bicycle model assumption is only valid for a slow steering vehicle and velocities slower than 5 meters per second.

In the base case, the team ran the EKF for different trajectories with and without the star tracker. The circular and straighter trajectories were used to compare the robustness of receiving accurate angle measurements.

In the second case, the team ran the EKF with the IMU and star tracker outputting measurements at the same and different rates. Running the simulation with the same data rate is not a realistic or high-fidelity scenario as star trackers typically output at much slower rates than IMUs; however, star trackers are more accurate since they can output orientation directly rather than integrating measurements and compounding errors exhibited in a gyroscope. In the scenario of differential data rates, the star tracker is set up to output measurements at 2Hz. To accommodate the different data rates, the main loop is modified to have an if-statement in which the modulo operator is performed to check if a star tracker measurement should be generated. The team computed another C_{noST} based on a measurement model where the rover does not receive angles from the star tracker.

For the implementation of the EKF, Algorithm 1 describes the pseudo-code of the EKF algorithm used in this paper.

Algorithm 1 Extended Kalman Filter

```
0: Initialize $X_0 \sim N(\mu_{0|0}, \sum_{0|0})$
0: Predict Mean:
0: $\mu_{t|t-1} = f(\mu_{t-1|t-1}, u_{t-1})$
0: Predict Covariance:
0: $\sum_{t|t-1} = A_{t-1} \sum_{t-1|t-1} A_{t-1}^T + Q_{t-1}$
0: Kalman Gain:
0: $K_t = \sum_{t|t-1} C_t^T (C_t \sum_{t|t-1} C^T + R_t)^{-1}$
0: Update Mean:
0: $\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - g(\mu_{t|t-1}))$
0: Update Covariance:
0: $\sum_{t|t} = \sum_{t|t-1} - K_t C_t \sum_{t|t-1}$
0: where $A_t$ and $C_t$ are the Jacobians
0: =0
```

V. Results

Several scenarios were investigated during the development of this model and simulation. The team explored the following cases:

- Case 1: Rover traveling in a circular vs. straighter trajectory with and without a star tracker
- Case 2: Rover receiving IMU and star tracker measurements at the same rate and different rates

A. Case 1: Circular vs. Straighter Trajectory with and without a Star Tracker

In the original simulation, different types of trajectories were tested with and without the use of a star tracker. The star tracker outputs accurate angles that track pitch and heading states. For the circular trajectories, the rover performed similarly with and without a star tracker; however, in the run with the star tracker, there is less drift from the true heading and steering states as seen in Figures 2, 3, 4, and 5.

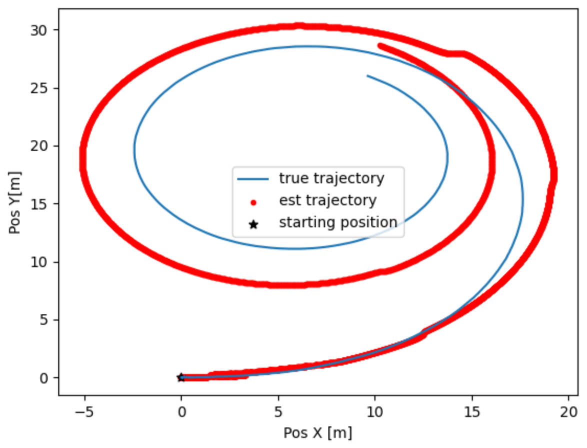

Fig. 2. Circular Trajectory with Star Tracker

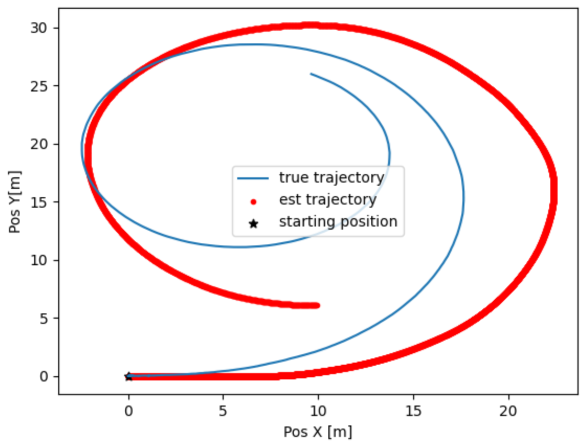

Fig. 3. Circular Trajectory without Star Tracker

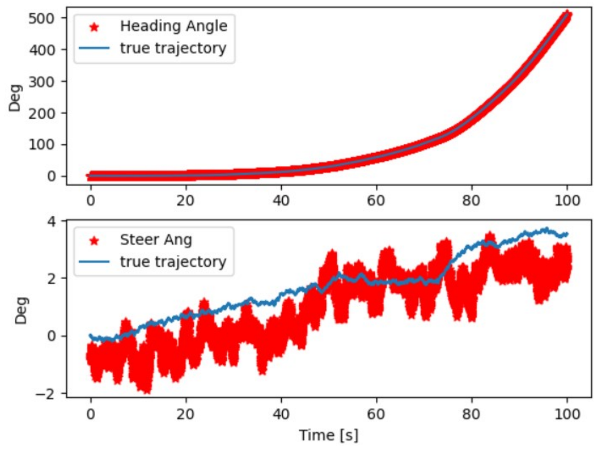

Fig. 5. Circular Trajectory without ST: Heading and Steering Angles

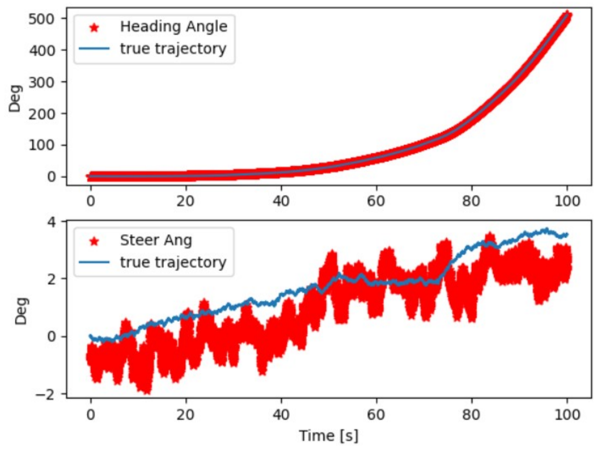

Fig. 4. Circular Trajectory with ST: Heading and Steering Angles

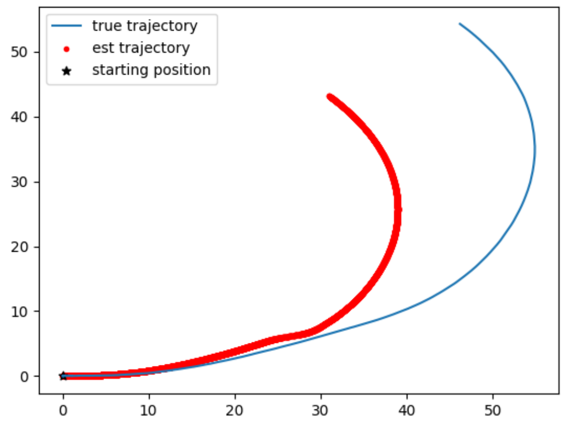

Fig. 6. Straight Trajectory with Star Tracker

When the rover went on a straighter trajectory, the star tracker measurements made a significant impact. As seen in Figures 6, 7, 8, and 9 the trajectory starts off with a straighter path and then starts turning. In the run without the star tracker, the estimated trajectory veers off from the truth very quickly since moving straight does not induce an angular rate, thus rendering the gyroscope useless. In the simulation where the star tracker is used, the rover is able to track the true trajectory during the straighter portion and then veers off the truth as expected with dead reckoning.

B. Case 2: Sensor Measurements at Same and Different Rates

As mentioned before, the code was originally set to run the measurement model of the IMUs and star tracker outputting measurements at the same rate, in which case gyro errors are not allowed to accumulated over time. Receiving more accurate star tracker information at the same frequency ensures that the rover will accurately estimate the change in attitude of the rover.

In order to create a more realistic model, the team modified the loop to include the ability to receive star tracker measurements at a slower rate (2Hz). When the data rates are not the same, specifically with the star tracker at a lower rate, the rover's estimated states diverged as expected with dead reckoning. However, it can be seen in Figures 10 and 11 that the angle estimates jump back close to the truth before diverging again. The star tracker provides observability into angles that are unable to be observed by the IMUs. The updates that are measured by the star tracker drive down the covariances related to the angular states, but they provide minimal correction for the positional estimates.

Additionally, the heading angle tracks well despite the tracker not being on at the same rate as the IMUs for the differential rate case with the star tracker on both the circular (Figure 12) and the straight trajectories (Figure 13)

VI. Analysis and Discussion

Due to the fact that the simplified kinematic bicycle model neglects many of the real world nonlinear dynamics, the

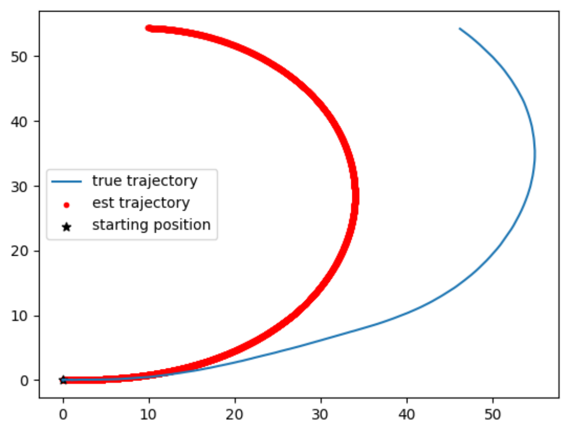

Fig. 7. Straight Trajectory without Star Tracker

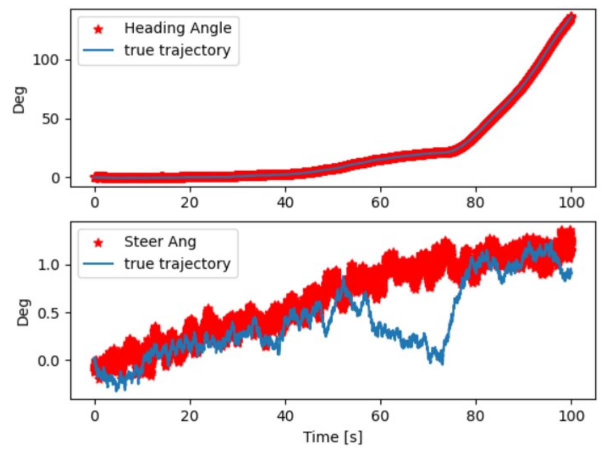

Fig. 8. Straight Trajectory with ST: Heading and Steering Angles

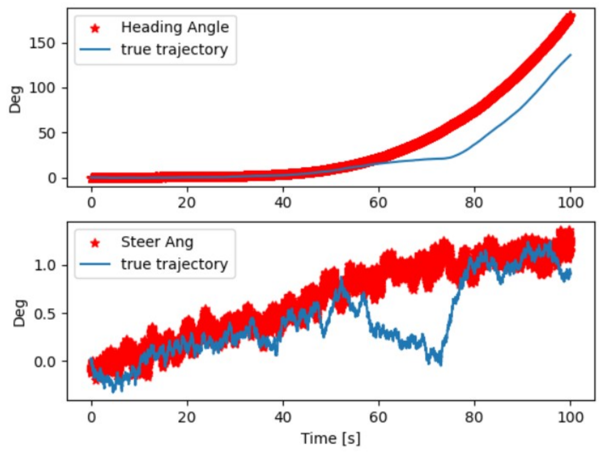

Fig. 9. Straight Trajectory without ST: Heading and Steering Angle

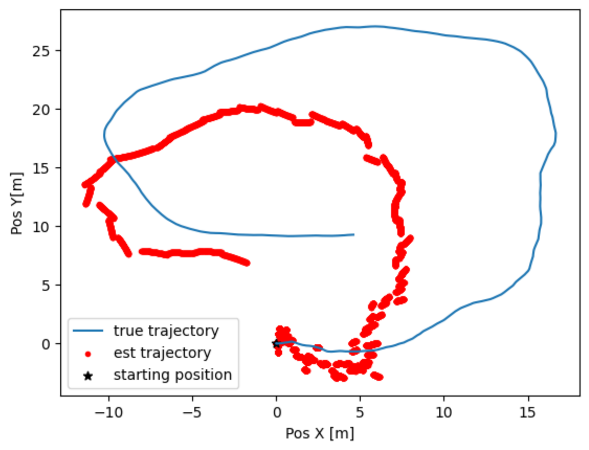

Fig. 10. Differential Data Rate with Circular Trajectory

process and measurement covariance matrices needed to be tuned carefully. The process noise was tuned in order to encapsulate nonlinear dynamics such as wheel slip, friction, and drag; however, the amount of noise was limited due to the instability of the model. The measurement covariances were tuned based off of the specification sheets for the IMUs and star tracker [15] [16].

When analyzing the results of dead reckoning, it is demonstrated that, as time passes, the EKF is able to track the angles much better than the translational states. Although the star tracker only accurately measures the steering angle, this measurement allows the rover to more confidently estimate the translational states of the trajectory despite the errors and noise from the IMUs. This is exemplified in Figures 10 and 11, where the the estimated trajectory deviates in position from the true trajectory but tracks its general trend.

To demonstrate the dead reckoning response, the team created plots with error ellipses overlaying the trajectory. When the trajectory is more circular, the tracking tends to be worse than the straighter trajectory as seen by the faster growing error ellipses in the circular path. The growing error ellipses in the circular path can be explained by the fact that the star tracker is not able to measure the heading angle quickly enough. The constant steering induces a rate of change on the heading angle which is used to determine the inertial positions of the rover. In the straighter case, the heading angle changes less rapidly, and this allows the rover to more accurately follow the path despite the sparse star tracker updates. The circular and straight trajectory error ellipses are found in Figures 14 and 15.

The results shown in this paper are more idealistic than normal dead reckoning algorithms. Further considerations need to be taken in order to model the scale factors and in-run bias that exist in normal IMU sensors. Modeling the in-run biases of the accelerometer and gyroscopes will degrade the performance of the algorithm significantly the longer the simulation runs.

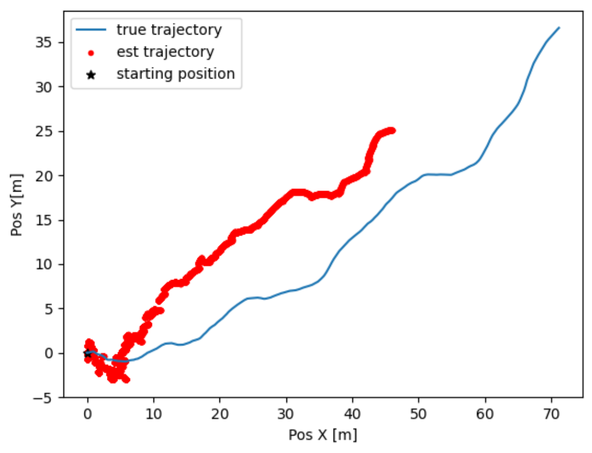

Fig. 11. Differential Data Rate with Straight Trajectory

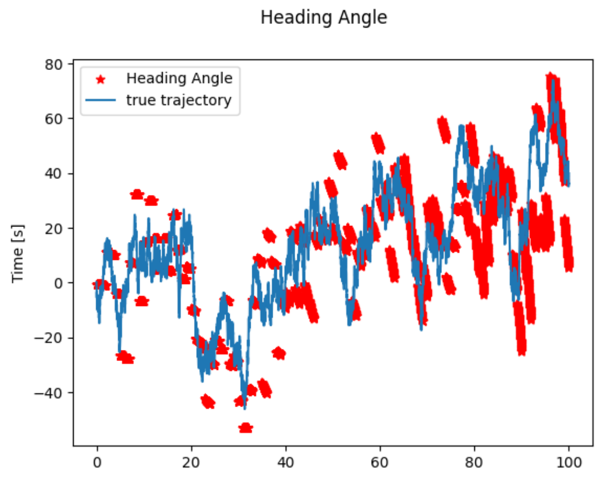

Fig. 13. Differential Data Rate with Straight Trajectory: Heading Angle

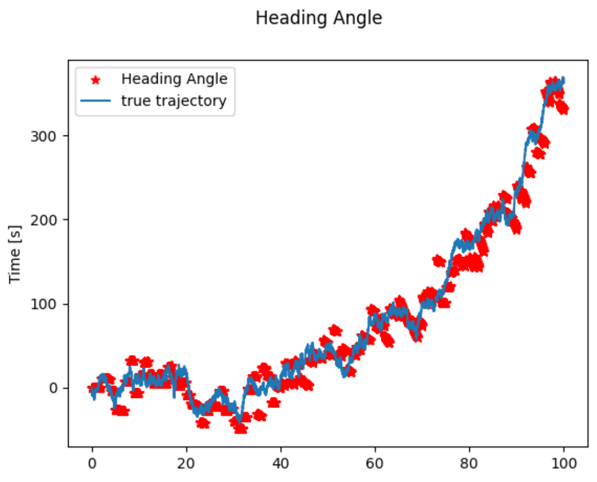

Fig. 12. Differential Data Rate with Circular Trajectory: Heading Angle

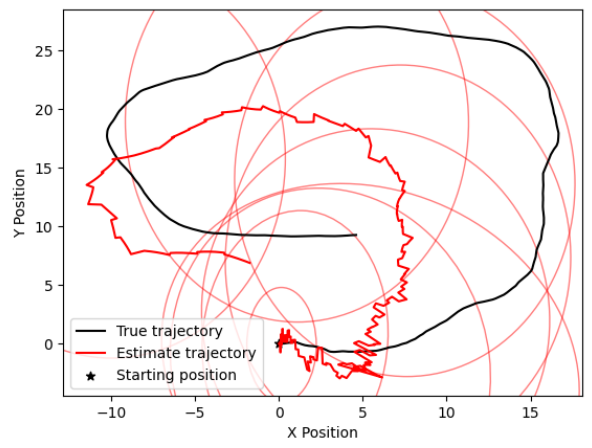

Fig. 14. Error Ellipses of Circular Trajectory

VII. Further Extensions

The UKF and particle filter were both considered but only briefly utilized in this project. An extension of this project is to compare the UKF and particle filter results with the results of the EKF. As described previously, the team decided to implement the EKF with different measurement model inputs to demonstrate the changes in the tracking rather than compare filters directly. The extension of using the UKF and particle filter is briefly described below.

The UKF addresses the divergence, overconfidence, and the need for a gradient issues of the EKF but is more computationally intensive. The UKF performs a ”stochastic linearization through the use of a weighted statistical linear regression process [14].” The UKF uses what is known as the ”Unscented Transformation” to extract ”sigma points” from a Gaussian function. The sigma points each have weights associated to them to be used when computing the mean and recovering the Gaussian. After each sigma point is predicted using the nonlinear dynamical function, an inverse transform is performed to obtain the predicted mean and covariance from the sigma points. In the update step, the same unscented transform is performed, but now, the sigma point measurement update is found, and expected measurement is calculated, and the Gaussian estimation equations are used to find the update mean and covariance.

In addition to addressing the divergence, overconfidence, and the need for a gradient issues of the EKF, the Particle Filter also addresses the uni-modality nature of the UKF and EKF. The PF represents ”the posterior by a set of random state samples drawn from this posterior [14].” By representing a function as a set of randomly drawn samples, the PF algorithm is not bound to any particular distribution. The PF initializes a particle set with a set of randomly drawn samples from the previous posterior or prior and predicts each sample using the nonlinear dynamics of the problem. The algorithm then updates the weights and normalizes them which

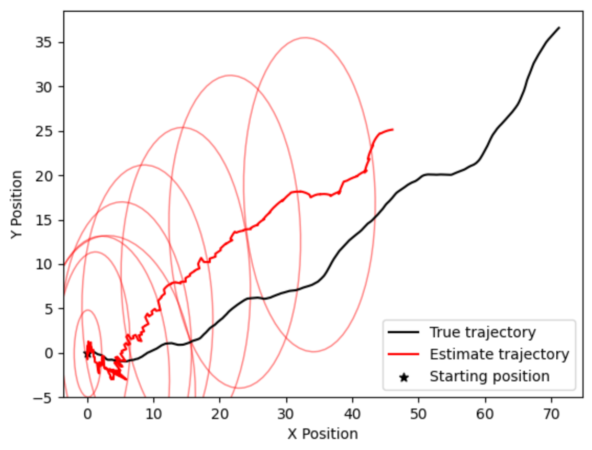

Fig. 15. Error Ellipses of Straight Trajectory

is calculated by finding the probability of the measurement under the particle set. Finally, to ensure that unlikely samples are removed and that the algorithm maximizes points in areas with higher weights, importance re-sampling is completed. Importance re-sampling reorganizes the new samples with new weights by dividing the particles into bins based on the weights, and the probability of drawing each particle is determined by its importance weight. An attempt was made in using the particle filter for the dynamics model described in the paper; however, due to inaccuracies, the trajectory did not track well and further analysis would be required in an extension to fully understand the results.

VIII. Conclusion

This paper documents an EKF implemented on a rover model, which is simplified using the kinematics bicycle model. The team measured the current position, velocity, acceleration, steering angle, pitch, and heading angle using IMUs and a star tracker. The resultant filter plots in Section V demonstrates multiple trajectories with and without the star tracker to show the benefits of a star tracker measurement of the heading angle and pitch. Additionally, the team addressed the star tracker providing measurements at a slower rate than the IMUs. This project demonstrates the benefits and drawbacks of an EKF for a ground rover with IMUs and a star tracker. In the further extensions section, the team provided further details on possible extensions for this project to be completed at a later date. The EKF was shown to be a descent filter for this problem in the initial time steps before diverging.

IX. Contributions

All team members contributed equally to the completion of this project. The work was divided appropriately and all team members believe that the project was a great learning experience.

References

[1] F. Shkurti, I. Rekleitis, M. Scaccia and G. Dudek, "State estimation of an underwater robot using visual and inertial information," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 2011, pp. 5054-5060.

[2] X. Niu, Y. Wu and J. Kuang, "Wheel-INS: A Wheel-Mounted MEMS IMU-Based Dead Reckoning System," in IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 9814-9825, Oct. 2021.

[3] Borenstein, J., and Feng, L., "Gyrodometry: A new Method for Measuring, Comparing, and Correcting Dead-Reckoning Errors in Mobile Robots", Proceeding of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota-April 1996, pp. 423-428

[4] C. Chen, W. Chai, A. K. Nasir and H. Roth, "Low cost IMU based indoor mobile robot navigation with the assist of odometry and Wi-Fi using dynamic constraints," Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium, Myrtle Beach, SC, USA, 2012, pp. 1274-1279.

[5] Cho, BS., Moon, Ws., Seo, WJ. et al. A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding. J Mech Sci Technol 25, 2907–2917 (2011).

[6] J. Borenstein and Liqiang Feng, "Measurement and correction of systematic odometry errors in mobile robots," in IEEE Transactions on Robotics and Automation, vol. 12, no. 6, pp. 869-880, Dec. 1996, doi: 10.1109/70.544770.

[7] Zhou, Feng & Arvidson, Raymond & Bennett, Keith & Trease, Brian & Lindemann, Randel & Bellutta, P. & Iagnemma, Karl & Senatore, Carmine. (2014). Simulations of Mars Rover Traverses. Journal of Field Robotics. 31. 10.1002/rob.21483.

[8] Conner, David & Kedrowski, Philip & Reinholtz, Charles & Bay, John. (2000). Improved dead-reckoning using caster wheel sensing on a differentially steered 3-wheeled autonomous vehicle. Proceedings of SPIE - The International Society for Optical Engineering. 10.1117/12.417296.

[9] D. A. Sigel and D. Wettergreen, "Star tracker celestial localization system for a lunar rover," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 2007, pp. 2851-2856.

[10] H. Kang, J. An and J. Lee, "IMU-Vision based Localization Algorithm for Lunar Rover," 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Hong Kong, China, 2019, pp. 766-771.

[11] D. M. Bevly and B. Parkinson, "Cascaded Kalman Filters for Accurate Estimation of Multiple Biases, Dead-Reckoning Navigation, and Full State Feedback Control of Ground Vehicles," in IEEE Transactions on Control Systems Technology, vol. 15, no. 2, pp. 199-208, March 2007.

[12] Z. Duan, Z. Cai, and H. Min. Robust Dead Reckoning System for Mobile Robots Based on Particle Filter and Raw Range Scan. Sensors 2014.

[13] H. Zhou, Y. Zhao, X. Xiong, Y. Lou and S. Kamal, "IMU Dead-Reckoning Localization with RNN-IEKF Algorithm," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 11382-11387.

[14] S. Thrun, W. Burgard and D. Fox. Probabilistic Robotics. MIT Press. 2006.

[15] LN-200S Inertial Measurement Unit (IMU) - Northrop Grumman. https://www.northropgrumman.com/wp-content/uploads/LN-200S-Inertial-Measurement-Unit-IMU-datasheet.pdf. June 7, 2023.

[16] Second Generation Star Tracker (ST-16RT2). https://www.rocketlabusa.com/assets/Uploads/Star-trackers-product-sheet.pdf. June 7, 2023.

[17] Rajamani, Rajesh. *Vehicle Dynamics and Control*, 2nd ed., p20-24. 2012.