AEM8442 Project: Final Report
Integration of Inertial Navigation System (INS) with LIDAR Based Localization

Siew, Peng Mun
siewx007@umn.edu

***Abstract*—Inertial Navigation System (INS) are frequently used and favored by the industry due to their high update frequency and them being insusceptible to jamming or outer interference. However, the state estimation error of a stand-alone INS tends to grow unbounded with time, especially for short propagation time and when a flat-Earth assumptions is used. Various studies has been conducted in this field to integrate the INS with an additional sensor in hope to bound the state estimation error. In this project, the INS is integrated with a Light Detection and Ranging (LIDAR) based localization using Extended Kalman Filter (EKF). After fusing the INS and LIDAR based localization, we are able to obtain better estimate of the vehicle velocity and a good estimate of the INS's error model.**

I. Introduction

Inertial Navigation System or INS is a self-contained navigation system. It typically comprises of a triad of accelerometers and a triad of gyros that are capable of measuring the 3-Dimensional acceleration and 3-Dimensional rotation of the platform. The position solution (position and pose in navigation frame) of the platform can then be calculated using these acceleration and rotation data.

As numerical integration routines are used to determine the position solution of the platform, the various measurement errors accumulate and cause the position solution error to not grow proportional to t, but at a higher proportionality. This can be easily shown by analyzing the error equation for a 1-Dimensional INS problem with platform tilt. Assuming that there is no gravity modeling errors and gyro scale factor errors. The position error solution is given by equation 1.

$$\delta x = \delta x_0 + \delta v_0 t + \frac{1}{2}k(v - v_0)t^2 + \frac{1}{2}(b_a - g\delta\theta_0)t^2 + \frac{1}{6}gb_g t^3 \quad (1)$$

where b_a is the accelerometer bias, b_g is the gryo bias and $\delta(\cdot)_0$ is the initial state error.

The position error grows unbounded with time and with a t^3 growth characteristic for gyro bias. For a 3-Dimensional case with roll, pitch and yaw, the position error growth characteristics become more complicated due to coupling between the Euler angles and position. But equation 1 gives a good intuition that for a higher dimensional case, the position errors will grow unbounded with a growth characteristics as high as t^3.

Since the early 90's, researchers have been looking into bounding the position error obtained from stand-alone INS, making it more reliable and suitable for long period operations. One of the proposed approaches is to integrate the INS with an additional localization system (auxiliary aiding system). Early works on integrating INS with Global Navigation Satellite System (GNSS) has shown very promising results and is able to yield bounded attitude and position errors.

Instead of fusing INS data with GNSS, this project focuses on integrating INS with LIDAR based localization in a structured environment. We assumed that we have a perfect LIDAR map of the environment annotated with landmarks of known position. The LIDAR scanner scans the environment at a frequency of 10Hz and will generate a point cloud of the environment in the platform's body frame. The relative position of the detected landmarks to the platform can then be calculated from the point cloud and the exact position of these detected landmark in the navigation frame can be obtained by matching the point clouds to the LIDAR map.

The Kalman filter has been traditionally used to fuse the INS with auxiliary aiding system. Kalman filter can produce the optimal state estimates when the system model and noise statistics are well known. It minimizes the expected value of the variance of the estimation error. Furthermore, it is computationally cheap and can be easily implemented on a microprocessor. In this project, an Extended Kalman Filter is used for sensor fusion between the INS and Lidar based localization. [5]

In this project, two different methods of integrating the INS with LIDAR based localization is examined. The first approach involves preprocessing the LIDAR measurements (relative position of platform to landmark) to obtain current attitude and position of the platform in the navigation frame and integrating that as a measurement input to the stand-alone INS. The second approach attempt to directly fuse INS data with the LIDAR measurements, in terms of distance to landmark from platform, elevation of landmark from the platform's body frame and the heading angle of landmark from the platform's body frame. Due to navigation errors, there will be a difference between the expected and true LIDAR measurements. The navigation error would deviate the platform from its true position resulting in a wrong expected LIDAR measurements to any particular landmark. This difference in measurements are then used as an observation equation for the Extended Kalman Filter (EKF).

An 8 minutes segment of true GNSS, INS and LIDAR measurements data obtained from the KITTI Vision Benchmark

Suite which was collected while driving around the mid-size city of Karlsruhe are used for this project. More information regarding the KITTI Vision Benchmark Suite will be provided in section II.

II. KITTI Vision Benchmark Suite

The KITTI dataset [3] was developed to provide a novel challenging benchmarks for the tasks of stereo, optical flow, visual odometry/SLAM and 3D object detection. It consists of over 6 hours of data collected by driving around in a mid sized town, in rural areas and on highways, but only about 25% (about 39.2km) of the total data are provided on-line for general public use. It was collected using a Volkswagen Passat equipped with 2 color and 2 grayscale camera, a high resolution Velodyne HDL-64E LIDAR scanner and a GNSS/IMU inertial navigation system (OXTS RT 3003 localization system). The GNSS and IMU data are collected at a rate of 100Hz, while the images and LIDAR scans are collected at a frequency of 10Hz.

Fig. 1. The recording platform equipped with four cameras, LIDAR scanner, GNSS and INS. [3]

The images and LIDAR scans are manually annotated for objects of interest in the form of 3D bounding box tracklets. The object of interest are divided into 8 classes; 'Car', 'Van', 'Truck', 'Pedestrian', 'Person (sitting)', 'Cyclist', 'Tram' and 'Misc (e.g., Trailers, Segways).' For the project, stationary cars are chosen as landmark of known location to localize the platform's position and attitude.

For this project, we will be using the IMU, GNSS data and point cloud obtained from a 8 minute segment of the KITTI Vision Benchmark Suite. We did not use any of the provided images and the GNSS information are used as a ground truth to evaluate the performance of our localization algorithm. The IMU data provided includes the roll, pitch and heading angle, North and East velocity, forward, leftward and upward velocity, x-axis, y-axis and z-axis acceleration, forward, leftward and upward acceleration, angular rate about x-axis, y-axis and z-axis, angular rate about forward axis, leftward axis and upward axis, and the position and velocity accuracy. The GNSS data provided (which we will be using as ground truth) uses real time kinematic (RTK) and has an open sky localization error of less than 5cm.

Figure 2 shows the trajectory plotted from that 8 minute segment of the data that is used in this project. The GNSS measurements are used to calculate the platform's position in the North-East-Down (NED) navigation frame based at its initial position.

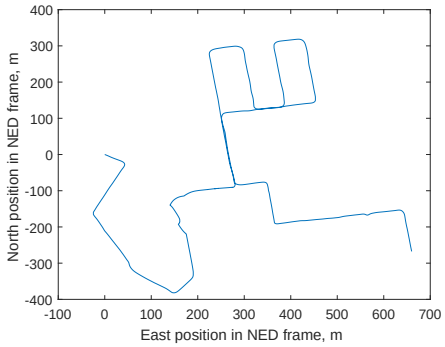

Fig. 2. Trajectory of the platform over the 8 minute segment.

Figure 3 shows the point cloud generated along part of the trajectory that was used for this project.

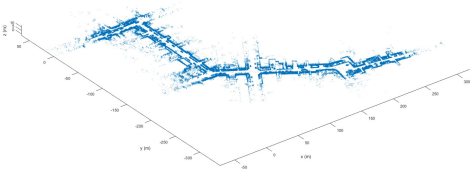

Fig. 3. Point clouds over a segment of the data in the local Mercator frame.

Figure 4 shows a part of the trajectory with landmark's position manually annotated in the NED frame. As the segment was taken from an urban environment, hence there are numerous landmarks present throughout the trajectory. Each landmark corresponds to a vehicle detected in the point cloud shown in figure 3.

III. Problem Formulation

For this project, two different approaches of integrating the INS with LIDAR measurements are carried out. Both approaches uses the same time update formulation but different measurement update approach.

For the first approach, the LIDAR measurements are used to solve for an estimated platform's position and attitude in the navigation frame, which are then fed as measurement inputs to the Extended Kalman Filter. In this approach, the preprocessing of LIDAR measurements can be treated as a perspective-n-point (PnP) problem, with n being the number of pseudo-landmark detected at every measurement update. A line of position can be obtained by using the distance, bearing and elevation measurement of each pseudo-landmark

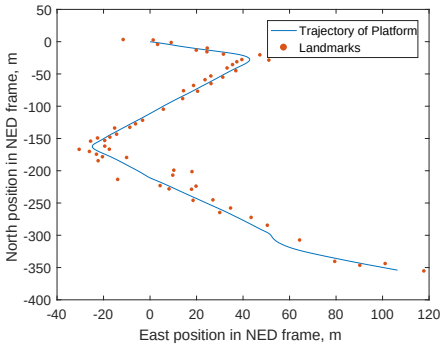

Fig. 4. Trajectory of the platform with annotated landmarks of known location.

with respect to the platform. Each line of position are all feasible position of the platform that satisfy that particular measurement. The intersection of these lines of position would be the estimated location and attitude of the vehicle. This method of estimating location and attitude is also known as position triangulation. Due to navigation error, the estimated position and attitude from PnP localization would differs from the estimated position and attitude obtained from INS solution. This difference in estimated position and attitude are used as an observation equation for the Extended Kalman Filter.

For the second approach, the LIDAR measurements are directly fused to the INS data without any preprocessing. Due to navigation errors, there will be a difference between the expected and true LIDAR measurements. The navigation error would deviate the platform from its true position resulting in a wrong expected LIDAR measurements to any particular landmark. This difference in measurements are then used as an observation equation for the Extended Kalman Filter.

In the subsequent subsections, more information on the two approaches are provided.

A. Time Update

As mentioned previously, both approaches uses the same time update formulation, where the estimated position and attitude of the platform is updated at a rate 100Hz using INS data (accelerometer and gyro data).

Upon receiving new IMU data, the accelerometer attitude are first determined using the equation 2.

$$\omega_{nb}^{b} = \omega_{ib}^{b} - C_{n}^{b}(\omega_{ie}^{n} + \omega_{en}^{n} + b_g) \tag{2}$$

where ω_{ib}^{b} is the gyro measurements, ω_{ie}^{n} is the Earth rotation rate given by equation 3, ω_{en}^{n} is the transport rate given by equation 4, C_{n}^{b} is the transformation matrix that maps vectors in the navigation frame to the body frame.

$$\omega_{ie}^{n} = \begin{bmatrix} \omega_{ie}^{n} cos(lat) \\ 0 \\ -\omega_{ie}^{n} sin(lat) \end{bmatrix} \tag{3}$$

$$\omega_{en}^{n} = \begin{bmatrix} \dfrac{V_E}{R_E + h} \\ -\dfrac{V_N}{R_N + h} \\ -\dfrac{V_E}{R_E + h} tan(lat) \end{bmatrix} \tag{4}$$

where R_E and R_N is the East-West and North-South radii of curvature based on the World Geodetic System 1984 (WGS-84) respectively.

The Euler angle of the platform at any particular time step is given by equation 5.

$$\begin{bmatrix} roll, \phi \\ pitch, \theta \\ yaw, \psi \end{bmatrix}_{(t_k)} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{(t_{k-1})} + \Delta t F_{b}^{n} \omega_{nb}^{b} \tag{5}$$

where F is the matrix that maps the angular rates in body frame into Euler angle rates and is given by equation 6.

$$F_{b}^{n} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & \dfrac{sin(\phi)}{cos(\theta)} & -\dfrac{cos(\phi)}{cos(\theta)} \end{bmatrix} \tag{6}$$

After solving for the accelerometer attitude, the platform's instantaneous speed is then computed using following equations (7, 8).

$$V_{t_k}^{n} = V_{t_{k-1}}^{n} + \delta t \dot{V} \tag{7}$$

$$\dot{V} = C_{b}^{n}(f_b + b_a) - (2\omega_{ie}^{n} + \omega_{en}^{n}) \times V_{t_{k-1}}^{n} + g^{n} \tag{8}$$

where f_b is the accelerometer measurements in the body frame, b_a is the accelerometer bias and g_n is the local gravity vector.

Finally, the geodetic location of the platform can be solved by integrating the platform's velocity vectors as shown in equation 9.

$$p_{t_k}^{g} = p_{t_{k-1}}^{g} + \delta t T_{n}^{g} V \tag{9}$$

where T is the matrix that map velocity vector in the navigation frame to the geodetic frame and is given by equation 10.

$$T_{n}^{g} = \begin{bmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & \dfrac{sin(\phi)}{cos(\theta)} & -\dfrac{cos(\phi)}{cos(\theta)} \end{bmatrix} \tag{10}$$

The covariance matrix, P of the system is given by equation 11.

$$P = \Phi P \Phi^{T} + Q_k \tag{11}$$

where Φ is the state error and bias transition matrix and Q_k is the process noise at time step k.

$$\Phi = exp(F \delta t) \tag{12}$$

As the time between GNSS update is small, the Schuler dynamics are ignored and we are assuming a simple additive bias and noise terms. Therefore, the navigation state error vector is given by equation 13 and the matrix F is as shown in equation 14. [1], [2]

$$\delta X = \begin{bmatrix} \delta p_N & \delta p_E & \delta p_D & \delta v_N & \delta v_E & \delta v_D & \delta\phi & \delta\theta & \delta\psi & \delta b_{a,N} & \delta b_{a,E} & \delta b_{a,D} & \delta b_{g,N} & \delta b_{g,E} & \delta b_{g,D} \end{bmatrix} \tag{13}$$

$$F \triangleq \begin{bmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-|g^n|/R_0 & 0 & 0 & 0 & 0 & 0 & 0 & -f_D & f_E & C^n_{b(1,1)} & C^n_{b(1,2)} & C^n_{b(1,3)} & 0 & 0 & 0 \\
0 & -|g^n|/R_0 & 0 & 0 & 0 & 0 & f_D & 0 & -f_N & C^n_{b(2,1)} & C^n_{b(2,2)} & C^n_{b(2,3)} & 0 & 0 & 0 \\
0 & 0 & 2|g^n|/R_0 & 0 & 0 & 0 & -f_E & f_N & 0 & C^n_{b(3,1)} & C^n_{b(3,2)} & C^n_{b(3,3)} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C^n_{b(1,1)} & -C^n_{b(1,2)} & -C^n_{b(1,3)} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C^n_{b(2,1)} & -C^n_{b(2,2)} & -C^n_{b(2,3)} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -C^n_{b(3,1)} & -C^n_{b(3,2)} & -C^n_{b(3,3)} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \omega_{ie,D} + \omega_{en,D} & -\omega_{en,E} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\omega_{ie,D} - \omega_{en,D} & 0 & \omega_{ie,N} + \omega_{en,N} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \omega_{en,E} & -\omega_{ie,N} - \omega_{en,N} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/t_g & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/t_g & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/t_g
\end{bmatrix} \tag{14}$$

B. Approach I: Fusing Preprocessed LIDAR Measurements

Assuming that the problem of associating LiDAR scans between subsequent frames has been solved and we know exactly which point cloud belong to which pseudo-landmark in a global frame. The estimated platform's position and attitude can be solved using a PnP formulation when there is 3 or more detected pseudo-landmark. The relative position of the pseudo-landmarks can be obtained from the LIDAR measurements.

For the scenario where there is exactly 3 detected pseudo-landmarks, assuming that the pseudo-landmark's position in navigation frame is given as $p^n_{L_i}, i = 1...3$, and the relative position observations to these landmarks from the platform as $p^b_{L_i}, i = 1...3$.

For each pseudo-landmark, the measurements and the map can be related by equation 15.

$$\begin{aligned} p^n_{L_1} &= p^n_R + C^n_b p^b_{L_1} \\ p^n_{L_2} &= p^n_R + C^n_b p^b_{L_2} \\ p^n_{L_3} &= p^n_R + C^n_b p^b_{L_3} \end{aligned} \tag{15}$$

where p^n_R is the platform's position in the navigation frame and C^n_b is the rotational matrix that transforms vector from the body frame to the navigation frame.

The platform's position can be eliminated from equation 15 by finding the difference between those three equations. For instance, the difference between equation 14.a and 14.b is taken, followed by taking the difference between equation 14.a and 14.c, we would arrive at equation 16.

$$\begin{aligned} p^n_{L_1} - p^n_{L_2} &= C^n_b p^b_{L_1} - C^n_b p^b_{L_2} \\ p^n_{L_1} - p^n_{L_3} &= C^n_b p^b_{L_1} - C^n_b p^b_{L_3} \end{aligned} \tag{16}$$

Let $p_{ij} = p_{L_i} - p_{L_j}$, then we would arrive at equation 17.

$$\begin{aligned} p^n_{12} &= C^n_b p^b_{12} \\ p^n_{13} &= C^n_b p^b_{13} \end{aligned} \tag{17}$$

The cross product of these two equations are obtained and used to form equation 18.

$$[p^n_{12} \quad p^n_{13} \quad p^n_{12} \times p^n_{13}] = C^n_b [p^b_{12} \quad p^b_{13} \quad p^b_{12} \times p^b_{13}] \tag{18}$$

The rotational matrix can be solved by rearranging equation 18.

$$C^n_b = [p^n_{12} \quad p^n_{13} \quad p^n_{12} \times p^n_{13}][p^b_{12} \quad p^b_{13} \quad p^b_{12} \times p^b_{13}]^{-1} \tag{19}$$

Note: The equation cannot be solved when $[p^b_{12} \quad p^b_{13} \quad p^b_{12} \times p^b_{13}]$ is not invertible (i.e. all of the landmarks lie on a line).

The Euler angles can then be obtained by decomposing the rotational matrix.

The platform's position can be solved by using equation 20.

$$p^n_R = \frac{1}{3} \sum_{i=1}^{3} (p^n_{L_i} - C^n_b p^b_{L_i}) \tag{20}$$

When there is more than 3 detected pseudo-landmarks, a least square formulation can be used to estimate the platform's position and rotational matrix.

$$\begin{Bmatrix} p^n_{12} & p^n_{13} & p^n_{12} \times p^n_{13} \\ p^n_{12} & p^n_{14} & p^n_{12} \times p^n_{14} \\ \vdots & \vdots & \vdots \end{Bmatrix} = C^n_b \begin{Bmatrix} p^b_{12} & p^b_{13} & p^b_{12} \times p^b_{13} \\ p^b_{12} & p^b_{14} & p^b_{12} \times p^b_{14} \\ \vdots & \vdots & \vdots \end{Bmatrix} \tag{21}$$

$$A = C^n_b B$$

$$C^n_b = AB^T(BB^T)^{-1} \tag{22}$$

The platform's position can then be solved using by using equation 23.

$$p^n_R = \frac{1}{n_{landmark}} \sum_{i=1}^{n_{landmark}} (p^n_{L_i} - C^n_b p^b_{L_i}) \tag{23}$$

The difference between platform's position and attitude obtained from INS solution and the PnP solution are then treated as measurement input to the Extended Kalman Filter's observation equation. Therefore, the innovation terms are given as equation 24.

$$Innov = \begin{Bmatrix} p^n_{ins} - p^n_{LIDAR} \\ \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{ins} - \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{LIDAR} \end{Bmatrix} \tag{24}$$

The H matrix (Observation matrix) is given by equation 25.

$$H = \begin{bmatrix} I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x6} \\ 0_{3x3} & 0_{3x3} & I_{3x3} & 0_{3x6} \end{bmatrix} \tag{25}$$

The Kalman gain is then computed using equation 26.

$$K = P^- H^T (H P^- H^T + R)^{-1} \tag{26}$$

The posterior covariance are updated using equation 27.

$$P^+ = (I - KH)P^- \tag{27}$$

The navigation state error vector, estimate platform's position and attitude from INS are then updated.

$$\delta X^+ = \delta X^- + K \times Innov \tag{28}$$

$$lat^+ = lat^- - \frac{\delta X^+_{(1)}}{R_N + h^-} \tag{29}$$

$$lon^+ = lon^- - \frac{\delta X^+_{(2)}}{(R_E + h^-)cos(lat^-)} \tag{30}$$

$$h^+ = h^- - X^+_{(3)} \tag{31}$$

$$v^+ = v^- - X^+_{(4:6)} \tag{32}$$

$$C_b^{n+} = [I - skew(X^+_{(7:9)})]C_b^{n-} \tag{33}$$

$$b_a^+ = b_a^- - X^+_{(10:12)} \tag{34}$$

$$b_g^+ = b_g^- - X^+_{(13:15)} \tag{35}$$

The R and Q matrix (noise variances) are a tuning parameters that need to be tuned based on the problem.

C. Approach II: Fusing Raw LIDAR Measurements

To fuse the LIDAR measurements (distance to landmark, bearing and elevation of landmark from platform's body frame) with the INS data, we need to first come up with a sensor measurement model to calculate the distance, bearing and elevation of the landmarks from the INS estimated position solution. This can be achieved by first converting the landmark's position from the navigation frame to the body frame using the following relationship and equation 36 - 37.

$$L^b = C_n^b (L^n - p^n)$$

$$\begin{bmatrix} L_N^b \\ L_E^b \\ L_D^b \end{bmatrix} = \begin{bmatrix} (L_N^n - p_N^n)cos(\theta)cos(\psi) + (L_E^n - p_E^n)cos(\theta)sin(\psi) - (L_D^n - p_D^n)sin(\theta) \\ (L_N^n - p_N^n)(sin(\phi)sin(\theta)cos(\psi) - cos(\phi)sin(\psi)) + (L_E^n - p_E^n)sin(\phi)sin(\theta)sin(\psi) + cos(\phi)cos(\psi)) - (L_D^n - p_D^n)sin(\phi)cos(\theta) \\ (L_N^n - p_N^n)(cos(\phi)sin(\theta)cos(\psi) - sin(\phi)sin(\psi)) + (L_E^n - p_E^n)cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi)) - (L_D^n - p_D^n)cos(\phi)cos(\theta) \end{bmatrix} \tag{36}$$

$$z_{INS} = \begin{bmatrix} d \\ azimuth \\ elevation \end{bmatrix}_{INS} = \begin{bmatrix} |L^b|_2^2 \\ atan\dfrac{L_E^b}{L_N^b} \\ atan\dfrac{L_D^b}{\sqrt{(L_N^b)^2 + (L_E^b)^2}} \end{bmatrix} \tag{37}$$

As the platform is a ground vehicle, small angle approximation is used for the roll and pitch angle and the derivative of the observation function with respect to each navigation state error vector is taken to generate the H matrix. Refer to the appendix section for the complete H matrix expression.

The innovation term is given by the differences between expected measurements and LIDAR measurements.

$$Innov = \left\{ \begin{bmatrix} d \\ azimuth \\ elevation \end{bmatrix}_{INS} - \begin{bmatrix} d \\ azimuth \\ elevation \end{bmatrix}_{LIDAR} \right\} \tag{38}$$

The Kalman gain, posterior covariance, navigation state error vector, estimated platform's position and attitude from INS are then updated using the same equations as in approach I (equation 26 - 35).

IV. Results and Discussion

A. Unaided INS

The following subsections covers the position and attitude solution obtained by only doing time propagation without any auxiliary aiding system.

The position solution obtained from a stand-alone INS diverges from the true solution as time passes, as shown in figure 5. Without any auxiliary aiding system, the system is not able to compensate for the inherent error within the measurements. As time passes, the errors start to accumulate.

Fig. 5. Estimated position solution using a stand-alone INS.

Even without any measurement update, the attitude errors seem to grows at a acceptable rate as shown in figure 6. However, any small error in attitude has an adverse effect in the position solution.

From figure 7 and 8, we can see how the error grows unbounded with time and at a much faster rate compared to errors in the attitude solution. The error in V_N solution has an

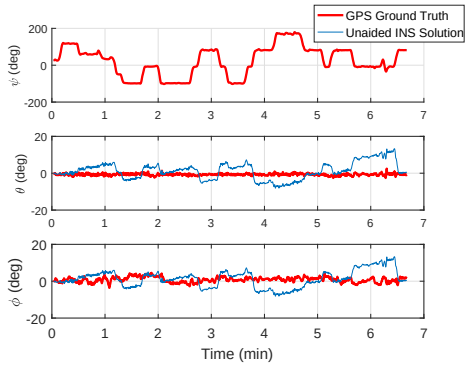

Fig. 6. Estimated attitude solution using a stand-alone INS.

almost cubic growth characteristics.

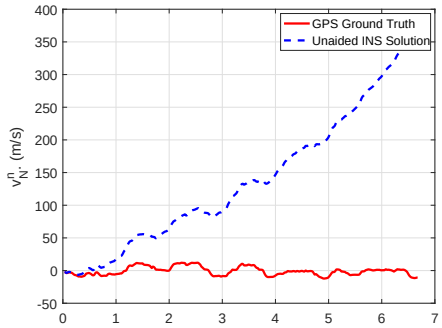

Fig. 7. Estimated V_N solution using a stand-alone INS.

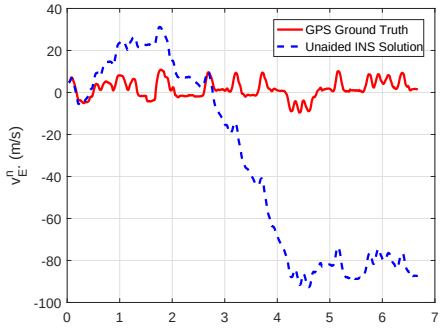

Fig. 8. Estimated V_E solution using a stand-alone INS.

B. Fusing Preprocessed LIDAR Measurements

The measurement noise covariance need to be manually tuned to obtain better state estimates. Without proper tuning, the results are as shown in figures 9 - 13. Although the sensor bias seems to have converged to a steady state value in figure 11, the estimated states still oscillates about the true states. The estimated position are able to trace the ground truth most of the time as shown in figure 9, with some oscillating deviation after taking turns. Strong oscillations are observed in the attitude

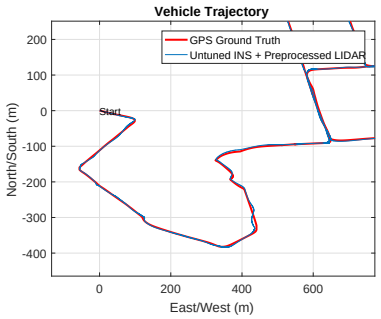

Fig. 9. Estimated position solution from untuned INS and processed LIDAR data fusion.

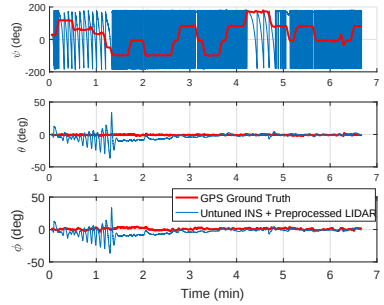

Fig. 10. Estimated attitude solution from untuned INS and processed LIDAR data fusion.

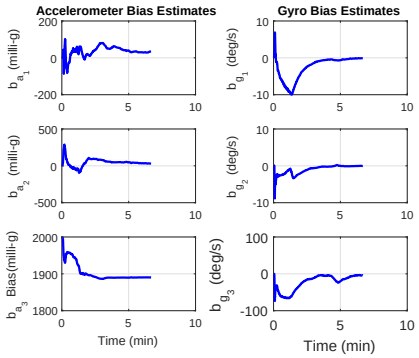

Fig. 11. Estimated sensor bias from untuned INS and processed LIDAR data fusion.

solution as shown in figure 10. There is consistent estimation error in the North and East velocity vector as well, as shown in figure 12 and 13.

After tuning the measurement noise covariance matrix, better state estimation are obtained as shown in figure 14 - 18. Little to no deviation are observed in the estimated position as shown in figure 14.

The estimated attitude solution no longer oscillates as in the previous case and it is able to trace the ground truth rather good as shown in figure 15. The attitude change is consistent with the ground track shown in figure 5.

The sensor bias history are shown in figure 16. It converges to a steady state value after approximately 3 minutes. The steady state bias value are shown in table I.

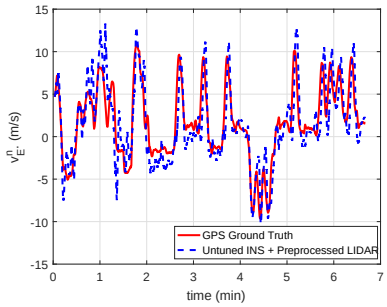

Fig. 12. Estimated V_E solution from untuned INS and processed LIDAR data fusion.

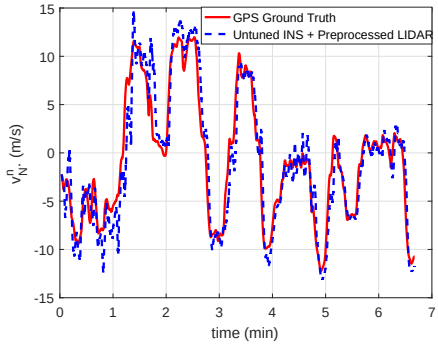

Fig. 13. Estimated V_N solution from untuned INS and processed LIDAR data fusion.

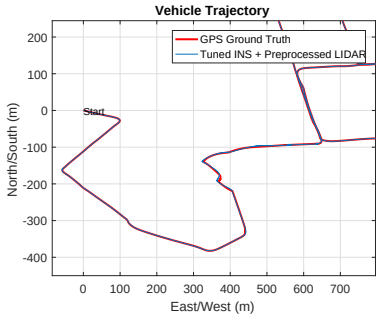

Fig. 14. Estimated position solution from tuned INS and processed LIDAR data fusion.

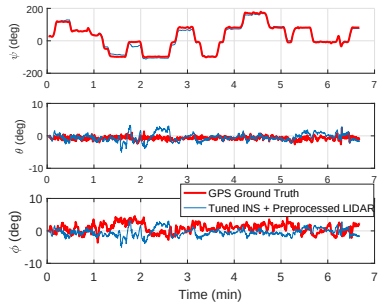

Fig. 15. Estimated attitude solution from tuned INS and processed LIDAR data fusion.

TABLE I
STEADY STATE SENSOR BIAS ESTIMATES

State	Bias Estimate
$b_{a,N}$	-6.4107 milli-g
$b_{a,E}$	28.4387 milli-g
$b_{a,D}$	1998.1 milli-g
$b_{g,N}$	-0.0027 deg/s
$b_{g,E}$	0.0008 deg/s
$b_{g,D}$	-0.0080 deg/s

Even at steady state, it is noticed that the bias estimate still varies slightly as time passes, this is due to the fact that the sensor errors are modeled as a simple additive biases and other form of biases, such as scale factor errors, misalignment error and non-orthogonality errors are ignored. A better model can be achieved by using a higher order error model.

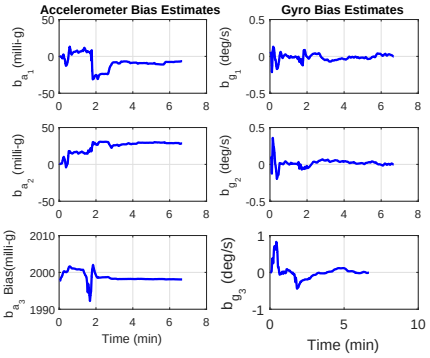

Fig. 16. Estimated sensor bias from tuned INS and processed LIDAR data fusion.

The estimated velocity vectors are able to trace the ground truth very well especially after the biases has reached a steady state value as shown in figure 17 and 18.

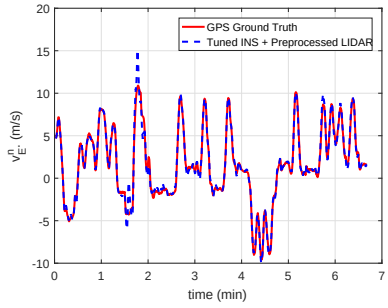

Fig. 17. Estimated V_E solution from tuned INS and processed LIDAR data fusion.

C. Fusing Raw LIDAR Measurements to INS

Unfortunately due to time constraint, the process and measurement noise was not successfully tuned to arrive at a converging solution using the second approach.

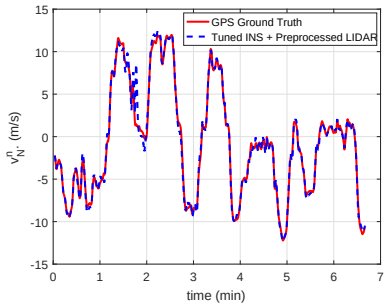

Fig. 18. Estimated V_N solution from tuned INS and processed LIDAR data fusion.

V. Conclusion

The estimation error of stand-alone INS grows unbounded with time as shown in figure 5 - 8. It is possible to bound these errors and obtain a better state estimation by integrating the INS with auxiliary aiding devices, such as GNSS or sighting devices (Camera, LIDAR, etc). In the project, two different approaches of integrating INS with LIDAR data are presented and one of the approach was successfully implemented and compared to the performance of a stand-alone INS.

In the implementation, the INS is integrated with preprocessed LIDAR data using an Extended Kalman Filter formulation. The relative position between pseudo-landmark and the platform are used to generate a set of estimated position and pose which are then integrated with the INS data. In our dynamic model, a simple additive bias and noise term is assumed and thus resulting in a 15 state error vector. The platform's pose and position is then solved by propagating and updating the 15 state error vector.

The process and measurement covariance need to be manually tuned for the EKF to perform optimally. Some of the newer researches are looking into formulating an adaptive Kalman Filter to remove the need for manual tuning.

The integrated INS and LIDAR based localization has shown satisfactory performances and is able to estimate the platform's pose, position and velocity with good accuracy. Better performances can be achieved by using a high order error model at the expense of computational speed.

References

[1] Scott Gleason; Demoz Gebre-Egziabher, *GNSS Applications and Methods*, Artech House, 2009.

[2] Paul D Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Artech House, 2013.

[3] Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun, *Vision meets Robotics: The KITTI Dataset*, International Journal of Robotics Research (IJRR), 2013.

[4] Billur Barshan and Hugh F. Durrant-Whyte, *Inertial Navigation Systems for Mobile Robots*, IEEE Transaction on Robotics and Automation, VOL. II, NO. 3, June 1995.

[5] Leandro Ribeiro Lustosa, Ronan Arraes Jardim Chagas and Jacques Waldmann, *Sighting Device-Aided Inertial Navigation: Fusion with Adaptive Kalman Filtering Techniques*, 21st International Congress of Mechanical Engineering, October 2011.

APPENDIX A
EXTRACTING DATA FROM KITTI DATASET

Listing 1. Reading Point Cloud Data from KITTI Dataset

```
base_dir  = 'D:\Research\Kitti\2011_09_30_drive_0028_sync\2011_09_30\2011
    _09_30_drive_0028_sync';
drawplot = 0;

x2 = nan(30000,601);
y2 = nan(30000,601);
z2 = nan(30000,601);

for frame = 0:600 % First 600 frames

fid = fopen(sprintf('%s/velodyne_points/data/%010d.bin',base_dir,frame),'rb');
velo = fread(fid,[4 inf],'single')';
velo = velo(1:5:end,:); % remove every 5th point for display speed
fclose(fid);
if drawplot == 1
plot3 (velo(:,1),velo(:,2),velo(:,3),'.','MarkerSize',0.5)
axis equal
axis ([-25,25,-25,25,-5,5])
view([-70 17])
pause (.5)
end
x2(1:size(velo(:,1)),frame+1) = velo(:,1);
y2(1:size(velo(:,1)),frame+1) = velo(:,2);
z2(1:size(velo(:,1)),frame+1) = velo(:,3);
end
```

Listing 2. Transforming Landmarks to body frame at each time time step

```
base_dir  = 'D:\Research\Kitti\2011_09_30_drive_0028_sync\2011_09_30\2011
    _09_30_drive_0028_sync';
load ('Kitti_obj.mat');
% load oxts data
oxts = loadOxtsliteData(base_dir);

% transform to poses
pose = convertOxtsToPose(oxts);

for i=1:1:length(pose)
Obj_body{i}= [[pose{i}(1:3,1:3)', -pose{i}(1:3,1:3)'*pose{i}(1:3,4);pose{i}(4,1:4)]*
    obj;obj(1:3,:)];
end
```

Listing 3. Extracting GNSS and INS data from KITTI dataset

```
imu_project = zeros(length(oxts),7);
gps_pos_lla_project = zeros(length(oxts),3);
for i = 1:length(oxts)
imu_project(i,1)= Ts_imu_sec(i);
imu_project(i,2:4) = oxts{i}(18:20);
imu_project(i,5:7) = -oxts{i}(12:14);
acc_temp(i,1:3) = -oxts{i}(15:17);
gps_pos_lla_project(i,1:3) = oxts{i}(1:3);
gps_vel_ned_project(i,1:2) = oxts{i}(7:8);
```

```
gps_vel_ned_project(i,3) = 0;
roll_project(i,1) = oxts{i}(4);
pitch_project(i,1) = oxts{i}(5);
yaw_project(i,1) = oxts{i}(6);
end
```

APPENDIX B
MATLAB CODE FOR APPROACH II (STATE INNOVATION FORMULATION, H MATRIX FORMULATION)

Listing 4. State innovation formulation for approach II

```
function stateInnov = meas_imu(landmark,pos,roll,pitch,yaw)
% landmark is a 6 by x matrix, row 1-3 are the distance, bearing and elevation
    measurement from the LIDAR respectively, while row 4-6 is the landmark position
    in the local NED frame
% pos is the platform estimated position in the local NED frame
% roll, pitch, yaw are the estimated attitude of the platform in the local NED frame
stateInnov = zeros(3*size(landmark,2),1);
for i = 1:size(landmark,2)
landmark(4,i) = landmark(4,i)-pos(1);
landmark(5,i) = landmark(5,i)-pos(2);
landmark(6,i) = landmark(6,i)-pos(3);
stateInnov(3*i-2,1) = (sqrt((landmark(4,i))^2+(landmark(5,i))^2+(landmark(6,i))^2))-
    landmark(1,i);
stateInnov(3*i-1,1) = (atan2(landmark(4,i)*(sin(roll)*cos(pitch)*cos(yaw)-cos(roll)*
    sin(yaw))+landmark(5,i)*(sin(roll)*sin(pitch)*sin(yaw)+cos(roll)*cos(yaw))+
    landmark(6,i)*(sin(roll)*cos(pitch)),landmark(4,i)*(cos(pitch)*cos(yaw))+landmark
    (5,i)*(cos(pitch)*sin(yaw))-landmark(6,i)*sin(pitch)))-landmark(2,i);
stateInnov(3*i,1) = (atan2(landmark(4,i)*(cos(roll)*sin(pitch)*cos(yaw)+sin(roll)*
    sin(yaw))+landmark(5,i)*(cos(roll)*sin(pitch)*sin(yaw)-sin(roll)*cos(yaw))+
    landmark(6,i)*(cos(roll)*cos(pitch)),sqrt((landmark(4,i)*(cos(pitch)*cos(yaw))+
    landmark(5,i)*(cos(pitch)*sin(yaw))-landmark(6,i)*sin(pitch))^2+(landmark(4,i)*(
    sin(roll)*cos(pitch)*cos(yaw)-cos(roll)*sin(yaw))+landmark(5,i)*(sin(roll)*sin(
    pitch)*sin(yaw)+cos(roll)*cos(yaw))+landmark(6,i)*(sin(roll)*cos(pitch)))^2)))-
    landmark(3,i);
end
end
```

Listing 5. H matrix formulation for approach II

```
function H = meas_matrix(landmark,pos,roll,pitch,yaw)
% Formulate H matrix
% landmark is a 6 by x matrix, row 1-3 are the distance, bearing and elevation
    measurement from the LIDAR respectively, while row 4-6 is the landmark position
    in the local NED frame
% pos is the platform estimated position in the local NED frame
% roll, pitch, yaw are the estimated attitude of the platform in the local NED frame
H = zeros(3*size(landmark,2),15);
for i = 1:size(landmark,2)
landmark(4,i) = landmark(4,i)-pos(1);
landmark(5,i) = landmark(5,i)-pos(2);
landmark(6,i) = landmark(6,i)-pos(3);
dist = sqrt((landmark(4,i))^2+(landmark(5,i))^2+(landmark(6,i))^2);
H(3*i-2,1:3) = [-landmark(4,i)/dist, -landmark(5,i)/dist, -landmark(6,i)/dist];

den2 = (landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(roll*pitch*sin(
    yaw)+cos(yaw))+landmark(6,i)*roll)^2+ (landmark(4,i)*cos(yaw)+landmark(5,i)*sin(
    yaw)-landmark(6,i)*pitch)^2;
```

```
H(3*i-1,1) = (landmark(5,i)+roll*(1+pitch^2)*(landmark(6,i))*cos(yaw)-pitch*(landmark(6,i))*sin(yaw))/den2;
H(3*i-1,2) = (-landmark(4,i)+roll*(1+pitch^2)*(landmark(6,i))*sin(yaw)+pitch*(landmark(6,i))*cos(yaw))/den2;
H(3*i-1,3) = ((-roll*(1+pitch^2)*landmark(4,i)-pitch*(landmark(5,i)))*cos(yaw)+(pitch*(landmark(4,i))-roll*(landmark(5,i))-roll*pitch^2*(landmark(5,i)))*sin(yaw))/den2;
H(3*i-1,7) = (landmark(6,i)+pitch*(landmark(4,i))*cos(yaw)+pitch*(landmark(5,i)+pos(2)-pos(2)*sin(yaw))*(-pitch*(landmark(6,i))+landmark(4,i)*cos(yaw)+landmark(5,i)*cos(yaw)))/den2;
H(3*i-1,8) = (roll*(landmark(4,i)*cos(yaw))^2-landmark(4,i)*landmark(6,i)*sin(yaw)+roll*(landmark(5,i)*sin(yaw))^2-cos(yaw)*(-landmark(5,i)*landmark(6,i)+2*roll*((landmark(5,i)+pos(2))*pos(1)+(landmark(4,i)+pos(1))*pos(2))*sin(yaw))+roll*((landmark(5,i))^2+((landmark(4,i)+pos(1))*(landmark(5,i)+pos(2))+pos(1)*pos(2))*sin(2*yaw)))/den2;
H(3*i-1,9) = -((landmark(4,i)+pos(1))^2+(landmark(5,i)+pos(2))^2-2*(landmark(4,i)+pos(1))*(pos(1))+pos(1)^2-2*(landmark(5,i)+pos(2))*(pos(2))+pos(2)^2+(pitch*(-landmark(4,i))+roll*landmark(5,i)+roll*pitch^2*(landmark(5,i)))*landmark(6,i)*cos(yaw)-(roll*(1+pitch^2)*(landmark(4,i))+pitch*(landmark(5,i)))*landmark(6,i)*sin(yaw))/den2;

den3 = (((pitch*(landmark(6,i))-landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))^2 +((roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+pitch*roll*sin(yaw))))^2)^1/2)*((pitch*(landmark(6,i))-landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))^2 +((roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+pitch*roll*sin(yaw))))^2+(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw)+landmark(5,i)*(-roll*cos(yaw)+pitch*sin(yaw))))^2);
H(3*i,1) = (-(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw))+landmark(5,i)*(-roll*cos(yaw)+pitch*sin(yaw)))*(-2*cos(yaw)*(pitch*(-landmark(6,i))+landmark(4,i)*cos(yaw)+landmark(5,i)*sin(yaw))+2*(-roll*pitch*cos(yaw)+sin(yaw))*(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw))))+2*(-pitch*cos(yaw)-roll*sin(yaw))*((pitch*landmark(6,i)-landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))^2+(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+pitch*roll*sin(yaw)))^2))/(2*den3);
H(3*i,2) = (-(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw))+landmark(5,i)*(-roll*cos(yaw)+pitch*sin(yaw)))*(-2*sin(yaw)*(pitch*(-landmark(6,i))+landmark(4,i)*cos(yaw)+landmark(5,i)*sin(yaw))+2*(-cos(yaw)-roll*pitch*sin(yaw))*(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw))))+2*(roll*cos(yaw)-pitch*sin(yaw))*((pitch*landmark(6,i)-landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))^2+(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw)))^2))/(2*den3);
H(3*i,3) = (-(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw))+landmark(5,i)*(-roll*cos(yaw)+pitch*sin(yaw)))*(2*pitch*(pitch*(-landmark(6,i))+landmark(4,i)*cos(yaw)+landmark(5,i)*sin(yaw))-2*roll*(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw))))-2*((pitch*landmark(6,i)-landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))^2+(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw)))^2))/(2*den3);
H(3*i,7) = (-(landmark(6,i)+pitch*landmark(4,i)*cos(yaw)+pitch*landmark(5,i)*sin(yaw))*(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw))+landmark(5,i)*(-roll*cos(yaw)+pitch*sin(yaw)))*(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw)))+(-landmark(5,i)*cos(yaw)+landmark(4,i)*sin(yaw))*((pitch*landmark(6,i)-landmark(4,i)*cos(yaw)-
```

```
    landmark(5,i)*sin(yaw))^2+(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-
    sin(yaw))+landmark(5,i)*(cos(yaw)+pitch*roll*sin(yaw)))^2))/(den3);
H(3*i,8) = (-(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw))+landmark(5,
    i)*(-roll*cos(yaw)+pitch*sin(yaw)))*(2*landmark(6,i)*(pitch*(landmark(6,i))-
    landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))+2*(roll*landmark(4,i)*cos(yaw)+
    roll*landmark(5,i)*sin(yaw))*(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(
    yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw))))+2*(landmark(4,i)*
    cos(yaw)+landmark(5,i)*sin(yaw))*((pitch*landmark(6,i)-landmark(4,i)*cos(yaw)-
    landmark(5,i)*sin(yaw))^2+(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-
    sin(yaw))+landmark(5,i)*(cos(yaw)+pitch*roll*sin(yaw)))^2))/(2*den3);
H(3*i,9) = (-(landmark(6,i)+landmark(4,i)*(pitch*cos(yaw)+roll*sin(yaw))+landmark(5,
    i)*(-roll*cos(yaw)+pitch*sin(yaw)))*(2*(landmark(5,i)*cos(yaw)-landmark(4,i)*sin(
    yaw))*(pitch*(-landmark(6,i))+landmark(4,i)*cos(yaw)+landmark(5,i)*sin(yaw))+2*(
    landmark(5,i)*(roll*pitch*cos(yaw)-sin(yaw))-landmark(4,i)*(cos(yaw)+roll*pitch*
    sin(yaw)))*(roll*landmark(6,i)+landmark(4,i)*(roll*pitch*cos(yaw)-sin(yaw))+
    landmark(5,i)*(cos(yaw)+roll*pitch*sin(yaw))))+2*(landmark(5,i)*(pitch*cos(yaw)+
    roll*sin(yaw))+landmark(4,i)*(roll*cos(yaw)-pitch*sin(yaw)))*((pitch*landmark(6,i
    )-landmark(4,i)*cos(yaw)-landmark(5,i)*sin(yaw))^2+(roll*landmark(6,i)+landmark
    (4,i)*(roll*pitch*cos(yaw)-sin(yaw))+landmark(5,i)*(cos(yaw)+pitch*roll*sin(yaw))
    )^2))/(2*den3);
end
end
```