State Estimation of Pipe Inspection Gauge using EKF and UKF

Sujet Phodapol
sujet@kth.se

Egill Milan Gunnarsson
emgun@kth.se

***Abstract*—State estimation is one of the challenges for the system that cannot directly monitor the states and works in an unforeseen environment. Pipe Inspection Gauge (PIG) is an intriguing example since it operates at a deep level in a pipe beneath the sea. Due to the data transmission limitation (i.e. cannot use the GPS for tracking), the robot can rely solely on the onboard sensor to estimate the states. In this project, the standard techniques Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are utilized to estimate the position and the speed of the robot. Two sensors data are used in this project: encoder, and flow speed sensor. We experiment the estimators with various scenarios to highlight the benefits and drawbacks of each. First, we demonstrate the advantage of sensor fusion over a single sensor. The performance of EFK and UKF is then demonstrated in a variety of circumstances. Also, the slip detection algorithm is proposed and investigated. Finally, joint parameter estimation using EKF will be implemented.**

Index terms— Pipe Inspection Gauge, State Estimation, Extended Kalman Filter, Unscented Kalman Filter, Sensor Fusion

I. Introduction

Pipeline Inspection Gauge (PIG), as depicted in Fig. 1, is a tool that has been used in the oil and gas industry for a long time for survey and maintenance. The inspection tool's working principle is that the robot is propelled by the fluid in the pipeline along the pipe from one site to the other site to collect the structure data, including corrosion of the pipeline surface.

The challenge of this robot comes with poor data collection, including incorrect corrosion position. These shortcomings might result in a significant waste of time and resources throughout the maintenance procedure. The main causes of poor data collecting are the fluctuation of the external fluid, which causes the robot's velocity to fluctuate dramatically, and the slippage of the sensors, which results in incorrect position mapping.

The recent approach handles the problem by designing the active PIG [1], which uses the turbine to directly drive the PIG. Another approach is the passive PIG [2], which uses the morphology of the tool to change the pressure difference in order to control the tool's velocity. However, in order to develop an efficient closed-loop speed controller, both methods require a reliable velocity estimator.

Another factor that makes the state estimation of this robot complicated is the operational environment. PIG is commonly used in the pipeline under the ocean, where GPS transmission is not possible. As a result, we cannot rely on satellite data to do robot localization. Normally, robot localization and state estimation in a limited GPS signal place can be done using landmarks [3]; however, in the long pipeline, it is difficult to specify proper landmarks and the robot also only operate in one direction, so we cannot also use the loop closure to improve the localization precision. Therefore, we can use only data from the onboard sensors of the PIG, encoders, inertial measurement unit (IMU) and flow sensor, to estimate states.

One last challenge is that, in the real system, some parameters are impossible or difficult to measure such as the friction constant. Therefore, we will require the method to approximate these values.

Our goal in this project is to create and implement the state estimator for the PIG and design the slip detection to handle the slipping issue. Although we can simply use data from one sensor to estimate the states, in practice, there are several methods that make use of multiple sensors to get a better state estimation (i.e. Sensor fusion). Due to the fact that the motion of the PIG can be represented with the nonlinear equation, we select two nonlinear estimators: Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) as the methods to estimate this system. First, We will demonstrate the benefit of using multiple sensors compare to one sensor. Subsequently, we will investigate and compare the performance of EKF and UKF in various scenarios. The slip model will then be introduced into the system, and the slip detection will be designed to reject outlier data. Lastly, we show how to estimate the parameter using the EKF framework.

The contributions of this project can be summarised as follow:

- We show how to implement EKF and UKF as a sensor fusion framework and the advantages over one sensor estimation.
- We demonstrate the behavior and performance of EKF and UKF in different inputs.
- We propose a slip detection algorithm that may be used to reject outliers and prevent the error from slipping occurrence.
- We manifest the implementation of the joint EKF for parameter estimation.

The outline of this report is organized as follows. The related works in the state estimation are examined in the following section. In part 3, we will go through EKF and UKF's physical model, sensor model, and implementation technique. Part 4 will present the findings of the experiment to answer the contributions. Lastly, the conclusion from the experiment will conclude in part 5.

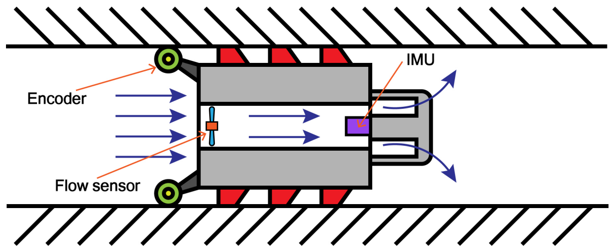

Fig. 1. Pipe Inspection Gauge: the robot consists of three sensors: encoders, IMU and flow sensor. Blue arrows show the direction of the flow through the robot.

II. Related work

While Extended Kalman Filter is one of the most commonly used estimators for nonlinear systems, Unscented Kalman filter is another novel estimator that can handle the nonlinearities of the system [4].

There are several studies that use KF as a method for position and speed estimation in different systems. For example, the work from J.K. HWANG [5] explains how to estimate the vehicle's speed by combining wheel speed and accelerometer sensor with a conventional Kalman Filter. For the more complex systems (i.e. nonlinear system), EKF and UKF are widely used. The paper [6] shows how to implement the EKF for speed estimation in permanent magnet synchronous motor. For speed estimation, apart from the basic UKF, several types of unscented transform (UT) are developed to generate sigma points to use in state estimation for nonlinear system [7].

For the PIG system, the work from Salazar [8] uses only data from encoders to estimate the speed of the tool. The result is then compared with the supervisory data from the external pressure sensor. Despite the fact that the result has an acceptable tiny inaccuracy, the system is evaluated in a low-noise control environment. Our project extends the framework using EKF and UKF to combine multiple sensors (i.e. encoder and flow sensor) to estimate the states of the PIG. The slip detection is also added to the estimator. With this framework in place, the estimator will be able to better deal with unexpected events.

Several frameworks for parameter estimation are proposed. The study from Blanchard [9] describes how to estimate the parameter of the roll plane modelling vehicle using EKF. The work [10] shows that UKF can also use for parameter estimation with superior convergence performance. In our project, we choose EKF, which is more simple to implement, as the method for this estimation.

III. Method

A. PIG equation of motion model

The equation motion of PIG [2] can be represented by the force interaction between the fluid and the robot, including Force from pressure difference (F_p), the change of momentum of the external fluid (F_m) and friction force (f):

$$m\ddot{x} = F_P + F_m - f \tag{1}$$

From the configuration of the PIG, the pressure difference can be computed as a function of the velocity of the bypass fluid through the PIG (V_{BP}), resulting in:

$$m\ddot{x} = \frac{1}{2}k\rho V_{BP}^2 A + \rho A(V_F - \dot{x})(\dot{x} + \frac{V_F}{2}) - f \tag{2}$$

Where $m, x, \dot{x}, \ddot{x}$ and A are the mass, position, velocity, acceleration, and cross-sectional area of PIG, respectively. k is the pressure loss coefficient based on the configuration of PIG, which is assumed to be constant in this project. ρ and V_F are the density and the velocity of the external fluid. With the conservation of mass, the bypass velocity of the fluid can be represented as:

$$V_{BP} = (V_F - \dot{x})\frac{D^2}{d^2} \tag{3}$$

Where D and d are the diameter of the pipe and the hole of the PIG. In this project, to avoid the piecewise differential problem in Extended Kalman Filter, we model the friction with Coulomb friction model [11] by modelling friction as:

$$f = \mu_f tanh(\dot{x}) \tag{4}$$

Where μ_f is the constant of the friction. Therefore, we can govern the state space description of the system as follow:

$$\dot{x} = v \tag{5}$$

$$\dot{v} = \frac{k\rho AD^4}{2md^4}(V_F - v)^2 + \frac{\rho A}{m}(V_F - v)(v + \frac{V_F}{2}) - \frac{\mu_f tanh(v)}{m} \tag{6}$$

Where x and v are the position and speed of the gauge. These two states are chosen to be measured by sensor, encoders. Therefore, the output can be defined as follow:

$$y = \begin{bmatrix} x & v \end{bmatrix}^T \tag{7}$$

In order to simulate, we need to transform from continuous time to discrete time. Thus the derivation is defined by the basic forward difference method as follow:

$$\dot{x} = \frac{x(k+1) - x(k)}{\Delta t} \tag{8}$$

B. Extended Kalman Filter

EKF is the linearized version of the conventional Kalman Filter (KF). According to Probabilistic Robotics [4], EKF uses the first-order Taylor's expansion to deal with the nonlinear system. The algorithm consists of two main steps: the prediction and the update step.

1) Prediction Step: In this step, we can predict the state of the system from the linearized motion model as follows:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \tag{9}$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \tag{10}$$

From eq. 5 and 6, we can derive the linearized function in Jacobian matrices as:

$$G_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 + \left[\frac{k\rho AD^4}{2md^4}(v - V_F) + \frac{\rho A}{m}(\frac{V_F}{2} - 2v) - \frac{\mu_f \text{sech}^2(v)}{m} \right] \Delta t \end{bmatrix} \tag{11}$$

The measurement model can derive from eq.7 as follow:

$$H_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{12}$$

2) Update Step: Then, we can compute Kalman gain and update the state with the following procedure:

$$K_t = \bar{\Sigma} H_t^T (H_t \bar{\Sigma} H_t^T + Q_t)^{-1} \tag{13}$$
$$\mu_t = \bar{\mu}_t + K(z_t - h(\bar{\mu}_t)) \tag{14}$$
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \tag{15}$$

C. Unscented Kalman Filter

Although the EFK is a good enough estimator for the slightly nonlinear system, UKF is introduced to handle a greatly nonlinear system. Compare to conventional EKF, instead of linearization, UKF use sigma points and unscented transform to approximate the nonlinear system. According to [4], the unscented transform algorithm can be implemented as follow:

1) Generating Sigma point: For the system with n-dimensional Gaussian distribution with mean (μ) and covariance matrix (Σ), sigma points ($\chi^{[i]}$) are generating by the following equations with total $2n$ + 1 points:

$$\chi^{[0]} = \mu \tag{16}$$
$$\chi^{[i]} = \mu + (\sqrt{(n+\lambda)\Sigma})_i \quad \text{for } i = 1,...,n \tag{17}$$
$$\chi^{[i]} = \mu - (\sqrt{(n+\lambda)\Sigma})_{i-n} \quad \text{for } i = n+1,...,2n \tag{18}$$
$$\lambda = \alpha^2(n+\kappa) - n \tag{19}$$

Where α and κ are scaling factor. In practical consideration of computing efficiency, we can use Cholesky factorization [12] to compute $\sqrt{\Sigma}$ as follow:

$$\Sigma = LL^T \tag{20}$$

2) Weight computing: Weights for the mean (w_m) and covariance (w_c) that will use to recover new Gaussian are computed as follows:

$$w_m^{[0]} = \frac{\lambda}{n+\lambda} \tag{21}$$
$$w_c^{[0]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{22}$$
$$w_m^{[i]} = \frac{1}{2(n+\lambda)} \quad \text{for } i = 1,...,2n \tag{23}$$
$$w_c^{[i]} = w_m^{[i]} \tag{24}$$

From [13], the recommended range of parameters are $\kappa \geqslant 0$, $\alpha \in [10^{-4}, 1]$ and $\beta = 2$ is the optimal value for the Gaussian distribution. Therefore, we choose $\alpha = 0.001$, $\kappa = 0$ and $\beta = 2$.

3) Unscented transform: The sigma points are then transformed through nonlinear function g. New mean (μ') and covaraince (Σ') are governed from the following equation:

$$Y^{[i]} = g(\chi^{[i]}) \tag{25}$$
$$\mu' = \sum_{i=0}^{2n} w_m^{[i]} Y^{[i]} \tag{26}$$
$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} (Y^{[i]} - \mu')(Y^{[i]} - \mu')^T \tag{27}$$

The main algorithm of Unscented Kalman Filter is then implemented in the same way as the conventional Kalman Filter with unscented transform

4) Prediction step: Like Kalman Filter, this step is used to compute the belief of the state ($\bar{\mu}, \bar{\Sigma}$), but, instead of using mean, we use sigma points that compute from unscented transform as follows:

$$\chi_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}}) \tag{28}$$
$$\bar{\chi}^*_{t-1} = g(u_t, \chi_{t-1}) \tag{29}$$
$$\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\chi}^{*[i]}_{t-1} \tag{30}$$
$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\chi}^{*[i]}_{t-1} - \bar{\mu}_t)(\bar{\chi}^{*[i]}_{t-1} - \bar{\mu}_t)^T + R_t \tag{31}$$

Then, we do the same procedure to compute the belief measurement from the belief ($\bar{\mu}, \bar{\Sigma}$) as follows:

$$\bar{\chi}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t}) \tag{32}$$
$$\bar{Z}_t = h(\bar{\chi}_t) \tag{33}$$
$$\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{Z}_t^{[i]} \tag{34}$$
$$S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{Z}_t^{[i]} - \hat{z}_t)(\bar{Z}_t^{[i]} - \hat{z}_t)^T + Q_t \tag{35}$$
$$\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\chi}_t^{[i]} - \bar{\mu}_t)(\bar{Z}_t^{[i]} - \hat{z}_t)^T \tag{36}$$

5) Update step: We compute the Kalman gain and update the states (μ, Σ) from the innovation as following equations:

$$K_t = \bar{\Sigma}_t^{x,z} S_t^{-1} \tag{37}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t) \tag{38}$$
$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T \tag{39}$$

D. Sensor Model

In this project, we have two main sensor types: three encoders and a flow sensor as shown in Fig.1. We use MATLAB to simulate the system (ground truth) and generate sensor data through the sensor model from the simulated system. Data from the flow sensor is used as an input (u) in the prediction step. Encoders are used in the measurement model. These sensor data will be used in sensor fusion (i.e. Kalman filter framework as shown in Fig.2) to estimate the states of the robot: position.

1) Encoder Model: Three encoders are used to measure the position and speed of the robot. By averaging the measurement, the noise can be reduced to some extent. From the simulated system, we can generate the encoder data for each encoder from the following equations:

$$number\,of\,ticks = \frac{[(x(k) - x(k-1)) + v_k \Delta t]ET}{2\pi R} \tag{40}$$

Where R and ET is the wheel radius and encoder ticks, which is 256 in this model. v_k is a white Gaussian measurement noise. Then, from the encoder data, we can compute the position and velocity from the following equations:

$$position = \frac{2\pi R}{3ET} \sum_{i=1}^{3} number\,of\,ticks_i \tag{41}$$

$$velocity = \frac{position}{\Delta t} \tag{42}$$

2) Flow Sensor Model: Flow sensor data is used as an input for the prediction step. Though in the real physical system, this data will have some noise, for simplicity, we assume this sensor data to be very accurate.

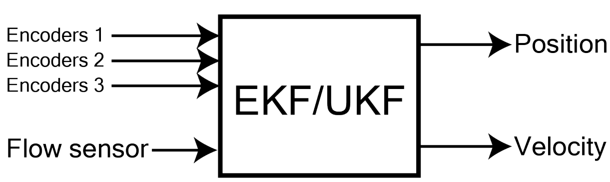

Fig. 2. The Kalman filter framework for state estimation.

E. Slip model

Since the robot is operating on an uneven pipe surface. One of the challenges in this system is the slipping of the encoder wheel. Slipping will result in the wrong data collection of the encoder. Thus if we include this data in the measurement model, the estimation will be wrong. To simulate this occurrence, we include some randomness in the data collection. When the slipping occurs, the encoder will detect much fewer ticks. To handle this scenario, we use the threshold, which is computed from the percent of total ticks from three encoders in order to accept or reject the data by the following rule:

$$number\,of\,ticks_i \leqslant threshold \times \sum_{i=1}^{3} number\,of\,ticks_i \tag{43}$$

Therefore, the final framework including the slip model is shown in Fig.3.

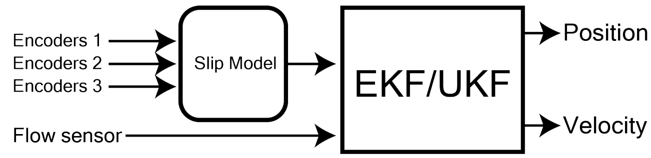

Fig. 3. The Kalman filter framework for state estimation with the slip detection

F. Parameter Estimation

Apart from state estimation, Kalman Filter can be also used to estimate parameters. In this project, we use EFK joint parameter estimation as a method to show this ability. The estimation is implemented by introducing parameter (Θ) to a state-space description. This procedure can be implemented by the fact that we assume the parameter to be constant, which can be described by the following equation:

$$\frac{d\Theta}{dt} = 0 \tag{44}$$

Then we recompute the Jacobian matrices (G and H) and follow the same procedure as normal state estimation EKF.

IV. Experiments

To understand the benefit and drawbacks of the EKF and UKF. We set up four experiments to answer the following question:

- Q1: Why do we need the KF for the sensor fusion?
- Q2: What difference between EKF and UKF in variety of inputs?
- Q3: How slip model has an effect on the estimation?
- Q4: How to estimate parameters using EKF?

By investigating these questions, we can understand more about the circumstances where we should use EKF or UKF. Moreover, we will study how the slipping affects the state estimation and how the simple rule can minimize the error.

In these experiments, we choose the process noise to be $w_p = 0.1$ and $w_v = 0.3$ and measurement noise, $v_k = 1$. The study is conducted by tuning the parameters (i.e. R and Q) to get the smallest error in one scenario and use the same parameters to test in other situations. For EKF, we use covariance matrices as follow:

$$R = \begin{bmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{bmatrix}, Q = \begin{bmatrix} 0.15^2 & 0 \\ 0 & 0.3^2 \end{bmatrix}$$

For UKF, we use covariance matrices as follow:

$$R = \begin{bmatrix} 0.1^2 & 0 \\ 0 & 0.1^2 \end{bmatrix}, Q = \begin{bmatrix} 0.23^2 & 0 \\ 0 & 0.3^2 \end{bmatrix}$$

The initial states value are:

$$x = \begin{bmatrix} 0 & 1 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

PIG is used as a system to evaluate the performance of estimators. The ground truth is simulated and sent to the sensor model, then the estimators will use the sensor data to do the estimation. The parameters for PIG is shown in Tab. IV.

TABLE I
PARAMETERS USED FOR PIG MODEL

Symbols	Value	Unit
m	427	kg
D	16	inch
d	8.32	inch
k	2.795	-
ρ	14.78	kg/m^3
A	0.095	m^2
μ_f	500	N

A. Experiment 1: Sensor Fusion

In this experiment, we demonstrate the benefit of the Kalman filter as a framework for sensor fusion to estimate the state of the system compared with the simple low pass filter. To implement the simple low pass filter, we can use the following equation:

$$q[k] = (1 - \kappa)q[k] + \kappa q[k - 1] \tag{45}$$

where κ is the weight between 0 and 1. The closer κ to 1 means that the lower cut off frequency, resulting in more smooth data. We choose $\kappa = 0.9$ to filter the encoder data in this experiment. From Fig.4, the result shows that estimated position using one sensor can yield an accepted result. On the contrary, as shown in Fig. 5, the raw velocity data has a large size of noise. Although the low-pass filter can smooth out the noise, the filter makes some delay to an estimation. With EKF and UFK, we combine two sensors, encoder and flow sensor data, together to estimate the states, resulting in better estimation with smaller delay as shown in Fig. 6 and smaller mean square error as shown in Tab. II.

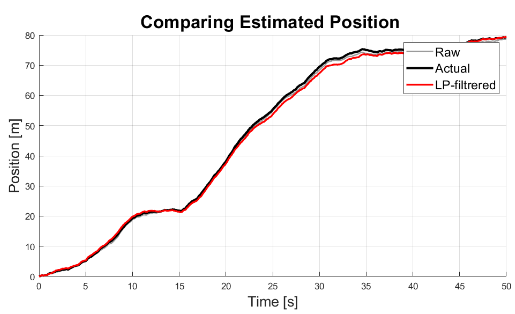

Fig. 4. Estimated Position using Low-pass filter

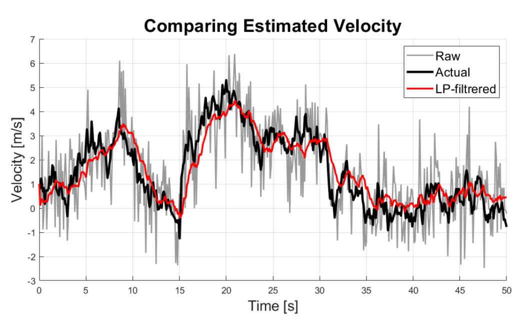

Fig. 5. Estimated Velocity using Low-pass filter

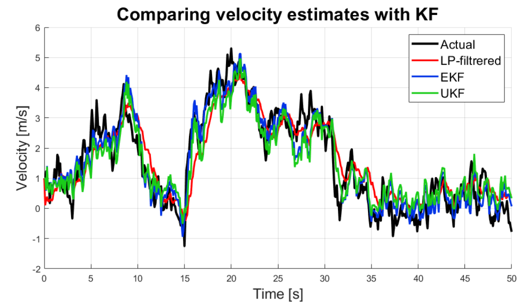

Fig. 6. Estimated Velocity using EKF and UKF

TABLE II
MEAN SQUARE ERROR (MSE) OF DIFFERENT METHODS

Estimators	LP	EKF	UKF
Position MSE	1.06	0.54	0.51
Velocity MSE	0.52	0.36	0.46

B. Experiment 2: State estimation with different input

In this section, we demonstrate the behavior of EKF and UKF in different working scenarios. We simulate by using different flow velocities (i.e. input). Five types of input as shown in Fig. 7, including constant, sine, sine with interruption, step and square wave input are used. We use the same tuning parameters R and Q for this experiment. The experiment is conducted three times with different random seeds in order to investigate the performance of each estimator, which are shown by mean square error (MSE).

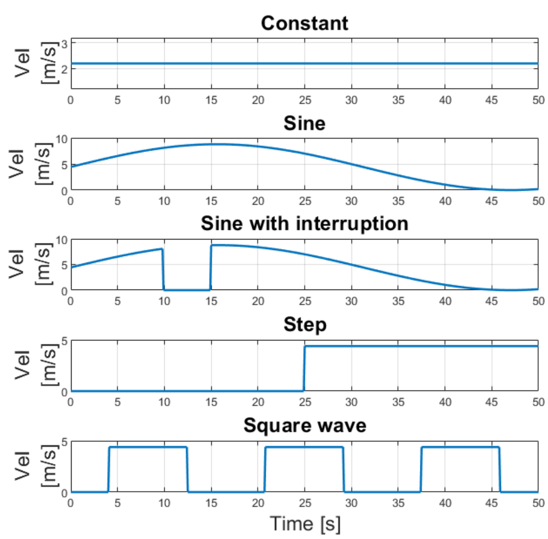

Fig. 7. Variety of input used in the experiment

The estimated position and velocity from each input are shown in Fig.8 and Fig.9, respectively. The result in Fig. 10 shows that UKF performs better in estimating the position of the robot; however, from Fig. 11, EKF has a better estimation for the velocity of the robot.

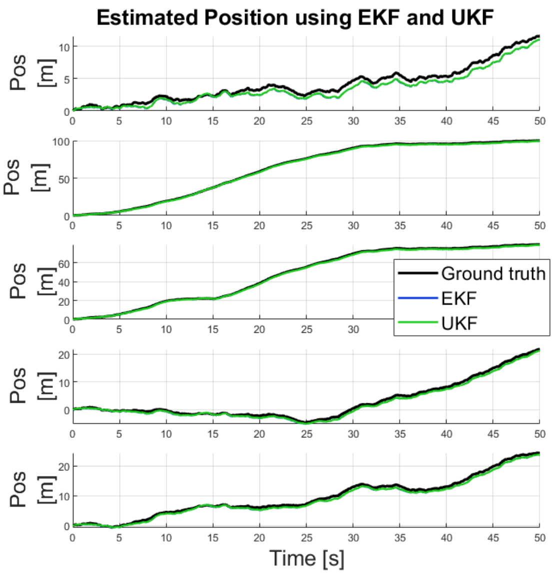

Fig. 8. Estimated position from a variety of inputs

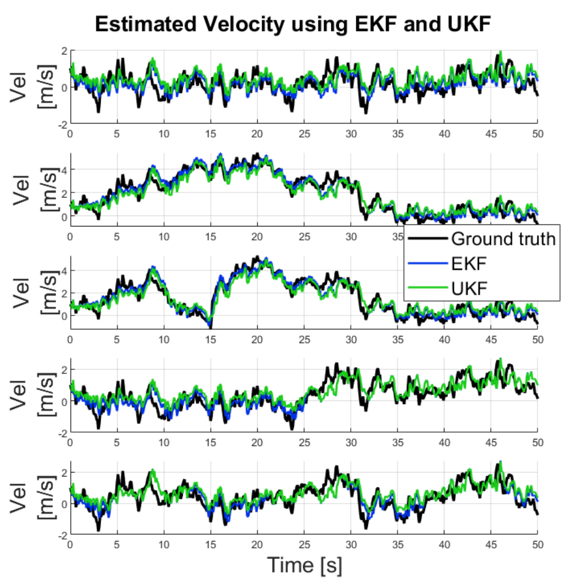

Fig. 9. Estimated velocity from a variety of inputs

C. Experiment 3: Slip model

In this experiment, we investigate one of the challenges of this system, the slipping occurrence. We simulate the event by introducing the likelihood for each encoder to be slipped if the slipping happens the number of ticks will be changed by the following rule:

$$number\,of\,ticks = number\,of\,ticks \times rand \times 0.5 \qquad (46)$$

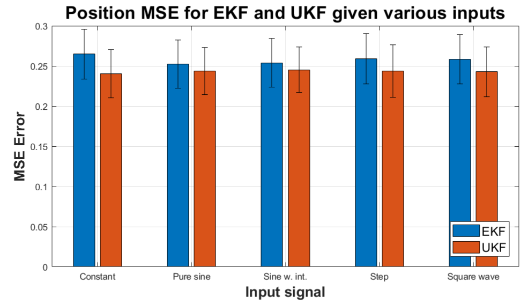

Fig. 10. Mean square error of position from a variety of inputs

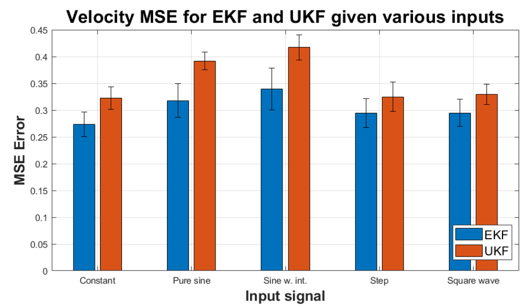

Fig. 11. Mean square error of velocity from a variety of inputs

The experiment is done by comparing the estimator with slip detection and without slip detection using sine with interruption input. As shown in Tab. III, the slip detection give a better result (i.e. lower MSE in both position and velocity). From Fig. 12, the experiment shows that the estimator without slip detection will accumulate the error in position, resulting in a very high error. On the other hand, from Fig. 13, slipping has a small impact on the velocity estimation, which can observe from a slightly change in MSE.

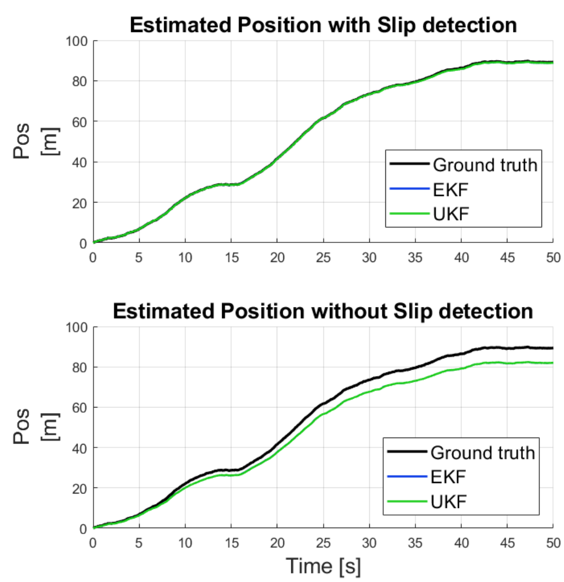

Fig. 12. Estimated position for the system with and without slip detection

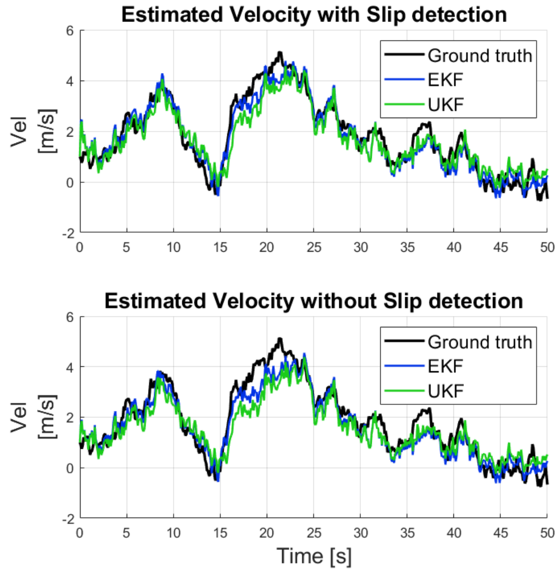

Fig. 13. Estimated velocity for the system with and without slip detection

TABLE III
ERROR FROM THE SLIPPING OCCURRENCE

Condition	Slip Detection		W/O Slip Detection	
	EKF	UKF	EKF	UKF
Position MSE	0.0535	0.0510	26.5955	26.5693
Velocity MSE	0.2284	0.3415	0.2782	0.4578

D. Experiment 4: Parameters Estimation

In this part, we demonstrate another benefit of the Kalman Filter framework, which is parameter estimation. We choose to estimate the friction constant (μ_f) using EKF. Thus we add the parameter as a new state to the state vector with the following equation:

$$\mu_{ft} = \mu_{ft-1} \tag{47}$$

We can derive the new Jacobian matrices for the process and measurement as:

$$G_t = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1+\beta\Delta t & -\frac{\tanh(v)}{m} \\ 0 & 0 & 1 \end{bmatrix} \tag{48}$$

$$\beta = \left[\frac{k\rho AD^4}{2md^4}(v - V_F) + \frac{\rho A}{m}(\frac{V_F}{2} - 2v) - \frac{\mu_f \text{sech}^2(v)}{m}\right] \tag{49}$$

$$H_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{50}$$

For simplicity in the tuning procedure, we assume all process and measurement noise to be zero. We choose tuning parameters and initial values to be

$$R = \begin{bmatrix} 0.1^2 & 0 & 0 \\ 0 & 0.1^2 & 0 \\ 0 & 0.1^2 & 10^2 \end{bmatrix}, Q = \begin{bmatrix} 0.15^2 & 0 \\ 0 & 0.5^2 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 & 1 & 300 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The result in Fig. 14 shows that the friction constant converges to the actual value, therefore we can approximate the parameter using the EKF framework.

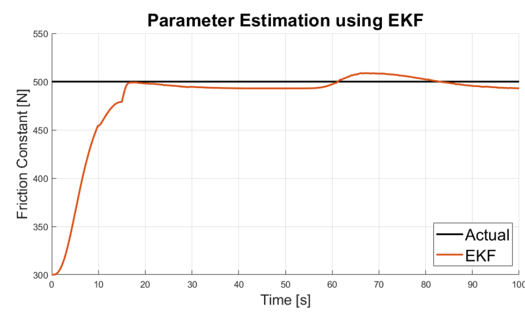

Fig. 14. Friction constant estimation using EKF

V. DISCUSSION AND CONCLUSION

In this project, we implement and investigate the behavior of EKF and UFK estimators. Also, we propose a mechanism for detecting and rejecting the occurrence of slips.

The first experiment shows that the multiple sensors framework improve the estimation process by reducing the noise, resulting in smaller error compare to a single sensor.

In the second experiment, UFk outperforms EFK in position estimation. In terms of velocity estimation, however, EKF shows superiority over UFK. The key advantage of EKF over UKF is that it is more computationally efficient due to the fact that UKF has more steps in the unscented transform. The performance of UKF should, in theory, be at least as accurate as EKF. Furthermore, because this method employs more points, sigma points, to estimate the nonlinear system (EKF uses only mean). Another advantage is that UKF does not have to compute the Jacobian matrix, which can be a time-consuming task in more complex systems. The reason that the experiment does not demonstrate a significant difference between the two methods may come from the fact that the system is not too nonlinear and from the numerical inaccuracy. Another reason can also come from the tuning parameters (Q and R), which should be retuned depending on the situation.

The third experiment shows that slip detection can reduce the error from the unexpected situation (i.e. slipping). In further work for slip detection, we will combine the inertial measurement unit (IMU) with the slipping model, thus we can compare the estimated speed from encoders and from IMU (integration of acceleration) to make the rejection rule.

The final experiment shows how to estimate the parameter from EKF framework. Although the result shows that this method is capable of estimating the parameter, when adding noise, tuning R and Q will be more complicated and the accuracy also lose. In future work, we will estimate the parameter using UKF framework.

REFERENCES

[1] Z. Hu and E. Appleton, "Dynamic characteristics of a novel self-drive pipeline pig," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 781–789, 2005.

[2] *Morphological Adaptation for Speed Control of Pipeline Inspection Gauges MC-PIG*, ser. Abu Dhabi International Petroleum Exhibition and Conference, vol. Day 1 Mon, November 15, 2021, 11 2021, d011S025R004. [Online]. Available: https://doi.org/10.2118/207403-MS

[3] C. Suliman, C. Cruceru, and F. Moldoveanu, "Mobile robot position estimation using the kalman filter," 2010.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005. [Online]. Available: https://books.google.se/books?id=2Zn6AQAAQBAJ

[5] J. K. Hwang, M. Uchanski, and C. K. Song, "Vehicle speed estimation based on kalman filtering of accelerometer and wheel speed measurements," *International Journal of Automotive Technology*, vol. 6, pp. 475–481, 10 2005.

[6] K. Shedbalkar, A. P. Dhamangaonkar, and A. B. Patil, "Speed estimation using extended kalman filter for pmsm," in *2012 International Conference on Emerging Trends in Electrical Engineering and Energy Management (ICETEEEM)*, 2012, pp. 433–435.

[7] A. Darvishi and A. Doroudi, "Continuous estimation of speed and torque of induction motors using the unscented kalman filter under voltage sag," *DYNA*, vol. 86, no. 208, pp. 37–45, jan 2019. [Online]. Available: https://doi.org/10.154462Fdyna.v86n208.65567

[8] A. Salazar, G. Lima, V. Freitas, R. Araújo, and A. Maitelli, "Pig's speed estimated with pressure transducers and hall effect sensor: An industrial application of sensors to validate a testing laboratory," *Sensors*, vol. 17, p. 2119, 09 2017.

[9] E. Blanchard, A. Sandu, and C. Sandu, "Parameter estimation method using an extended kalman filter," 01 2007.

[10] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[11] U. O. R. Lewis, *Wheel/Rail Interface Handbook*. Woodhead Publishing, 2009.

[12] S. Lü and R. Zhang, "Two efficient implementation forms of unscented kalman filter," *Control and Intelligent Systems*, vol. 39, 01 2011.

[13] N. Yongfang and Z. Tao, "Scaling parameters selection principle for the scaled unscented kalman filter," *Journal of Systems Engineering and Electronics*, vol. 29, no. 3, pp. 601–610, 2018.