Estimating Vehicle Kinematics Using Onboard IMU and Wheel Speed Sensors

Ross Alexander
Dept. of Aeronautics & Astronautics
Stanford University
Stanford, CA 94305
RBALEXAN@STANFORD.EDU

Rushil Goradia
Dept. of Electrical Engineering
Stanford University
Stanford, CA 94305
RGORADIA@STANFORD.EDU

Trey Weber
Dept. of Mechanical Engineering
Stanford University
Stanford, CA 94305
TPWEBER@STANFORD.EDU

***Abstract*—Autonomous passenger vehicles are being deployed on public roads in record numbers as a result of several fundamental advancements in sensing, perception, decision-making, control, and computational power. In particular, companies such as Tesla, Waymo, and Cruise have already logged millions of miles of operation without a human driver in control. Among many complex environments, urban autonomous driving scenarios pose significant challenges due to issues including unique road geometries, numerous road actors, multimodal actor behaviors, and sensor occlusions. As a result, accurate and efficient vehicle state estimation algorithms are of increasingly essential. However, highly accurate sensing technologies are often too expensive for the consumer market, so cheaper sensors must be used instead. In this work, we address the issue of vehicle state estimation using low-fidelity onboard inertial measurement units (IMUs) and wheel speed sensors. We construct a 3-degree-of-freedom nonlinear dynamical model of a vehicle and propose a measurement model consistent with an IMU and wheel speed sensors. We perform state estimation of the vehicle longitudinal velocity, lateral velocity, and yaw rate using nonlinear filter architectures such as the extended Kalman filter, unscented Kalman filter, and particle filter. In the last portion of this paper, we incorporate actuator delay in our simulator and demonstrate the necessity of using filter architectures with delay compensation.**

I. Introduction

With the rapid development and deployment of autonomous passenger vehicles on public roads in recent years, high-precision state estimation algorithms have become a critical piece of the autonomous vehicle stack. Notably, technological advancements have provided higher precision sensors, which ultimately reduces the design burden on the controls engineer. However, due to affordability constraints, autonomous vehicle control engineers may opt for cheaper and more extensive sensor suites, thus driving the need for high-precision state estimation.

Knowing an accurate representation of a system's kinematic state is fundamental for analysis and control of the system. Some common issues that arise in this domain are noise, modeling inaccuracies, or partial observability. Nearly all of these problems are present in autonomous vehicle development, as autonomous vehicles are complex systems operating in dynamic environments that equipped with a vast array of measurement capabilities including global positioning systems (GPSs), inertial measurement units (IMUs), wheel speed sensors, etc. Approaches to the state estimation problem

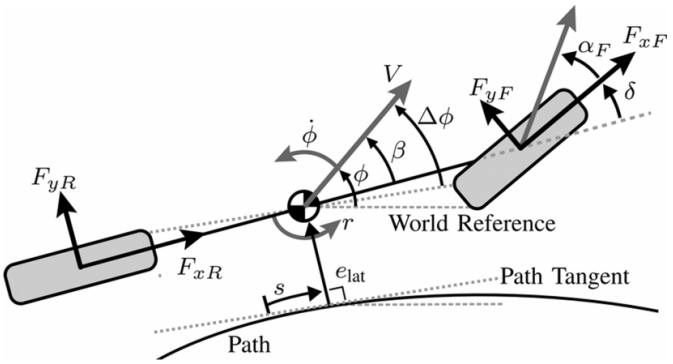

Fig. 1. Schematic of the dynamic bicycle model, depicting the pose of the vehicle with respect to a reference path in both the inertial, body-fixed, and path-fixed frames.

can be broadly categorized into model-based and model-free methods; this work will focus on model-based methods.

One of the most widely used methods of model-based state estimation is the Kalman filter (KF) and its many variations [1]. The KF is a stochastic state estimator that assumes that:

1) the system state is parameterized by a Gaussian distribution with a given mean and covariance,
2) the system state evolves with linear transition dynamics subject to Gaussian white noise, and
3) the system state is observed with linear measurement dynamics subject to Gaussian white noise.

The KF has been successfully applied in many vehicle applications. Venhovens and Naab used a KF architecture to estimate the velocity of surrounding vehicles that then allows for implementation of safety features such as lane-keeping and adaptive cruise control [2]. Specifically, they modeled longitudinal dynamics of other agents and used range measurement sensing (with assumed noise means and covariances). More interesting variations of the KF have also been successful for vehicle state estimation. Best et al. use an extended Kalman filter (EKF) to estimate the vehicle state (longitudinal velocity, lateral velocity, and yaw rate) and friction under extreme cornering maneuvers [3]. The authors demonstrate the benefits of considering nonlinearity by comparing the performance of a linear KF with a nonlinear EKF using a 4-degree-of-freedom (DoF) bicycle model and lateral acceleration sensing. Wenzel

et al. build on this idea of simultaneously estimating states and system parameters using a dual extended Kalman filter (DEKF) [4]. This approach allows partial separation of the parameter and state estimation problems, allowing them to run at different times and frequencies.

In this work, we seek to extend the success of modern state estimation techniques in vehicle dynamics applications by leveraging more-affordable, lower-fidelity sensors. In Section II, we introduce the nonlinear dynamics model and discuss several key components of the simulator. Section III details our sensor model and provides the general simulation and estimation workflow. In Section IV, we describe our simulation and estimation environment and demonstrate effective state estimation on simulated data, simulated data with actuator delay, and real-world data. Finally, in Sections V and VI we summarize our findings and provide directions for future work.

II. Background

In Sections II-A and II-B, we describe our choice of dynamics and measurement models for the vehicle state estimation problem.

A. Dynamics Model

In literature, there has been a wide range of success with modeling a typical four-wheeled vehicle as a two-wheeled, single-track vehicle, commonly referred to as a *dynamic bicycle model* [5]. The dynamic bicycle model consists of lumping the two tires on each axle together and representing the vehicle body as a rigid line connected by a single front and rear tire. The front tire is free to rotate, while the rear tire is fixed to the body axis. We only consider motion in the horizontal plane (no pitching, rolling, or vertical velocity). As a result, we can fully define the state of the vehicle with three degrees of freedom: longitudinal velocity U_x, lateral velocity U_y, and yaw rate r. The control inputs are the front steering angle δ and front and rear longitudinal tire forces corresponding to braking and throttle F_{xf} and F_{xr}, respectively.

$$\mathbf{x} = \begin{bmatrix} U_x, U_y, r \end{bmatrix}^\top \qquad \mathbf{u} = \begin{bmatrix} \delta, F_{xf}, F_{xr} \end{bmatrix}^\top \tag{1}$$

Using the definition of the state vector $\mathbf{x}$ and the control vector $\mathbf{u}$ in Equation (1) and performing an analysis of the forces on the vehicle (as depicted in Figure 1) we can write the dynamics of the system as:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{2}$$

$$\begin{bmatrix} \dot{U}_x \\ \dot{U}_y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}\left(F_{xf}\cos\delta - F_{yf}\sin\delta + F_{xr}\right) + rU_y \\ \frac{1}{m}\left(F_{xf}\sin\delta + F_{yf}\cos\delta + F_{yr}\right) - rU_x \\ \frac{1}{I_z}\left(aF_{xf}\sin\delta + aF_{yf}\cos\delta - bF_{yr}\right) \end{bmatrix}. \tag{3}$$

Here, m, I_z, a, and b, are the mass, moment of inertial, and distances from the center of mass (CoM) to front and rear wheels, respectively. We can further localize our vehicle to a reference path with an additional set of path coordinates: distance along the path s, lateral distance from path or error e, and heading angle difference with respect to path $\Delta\psi$. These path states cannot be measured directly since measurements can only be obtained in the inertial or body-fixed frames, not in the path frame. In this work, we assume the reference path is given.

In generating state trajectories, both in simulation and in real-world experiments, we produced control inputs from two low-level controllers. The first controller is the lookahead controller, which seeks to minimize error at some projected distance forward along the path using feedforward and feedback control. The control law is

$$\delta = -K_{la}(e + x_{la}\Delta\psi) + \delta_{ff} \tag{4}$$

where K_{la} is the lookahead feedback gain, x_{la} is the lookahead distance at which our error is calculated, and δff is the feedforward steering angle calculated from vehicle kinematics at steady state. The second controller is a standard feedforward and feedback control law for the longitudinal force, which is given by:

$$F_x = K_{long}(U_{x,des} - U_x) + ma_{x,des} + F_{drag} + F_{grade} + F_{roll} \tag{5}$$

where K_{long} is the longitudinal feedback gain, $U_{x,des}$ and $a_{x,des}$ are the desired speed and acceleration (given by the reference path), respectively, and F_{drag}, F_{grade}, and F_{roll} are feedforward terms to compensate for drag, grade, and rolling resistance forces, respectively.

Our lateral tire forces F_{yf} and F_{yr} come from the nonlinear Fiala brush tire model which is a physically-motivated model based on the deformation of portions of the tire tread, which are modeled as brushes [6]. The Fiala model provides a mapping from the tire slip angle to the lateral tire force. The front and rear tire slip angles, α_f and α_r, which are the angles of the body velocity vector with respect to the longitudinal axis of the tire, can be calculated directly from vehicle geometry and velocity states as follows

$$\alpha_f = \arctan\left(\frac{U_y + ar}{U_x}\right) - \delta \tag{6}$$

$$\alpha_r = \arctan\left(\frac{U_y - br}{U_x}\right). \tag{7}$$

Beyond the tire slip angles, the nonlinear Fiala model is parameterized by the normal force on the fire (F_z), the cornering stiffness (C_α), and the friction coefficient (μ). As shown in Figure 2, the Fiala model produces a lateral tire force that is essentially linear with tire slip angle for small tire slip angles and tapers off for larger tire slip angles due to the effects of tire force saturation.

B. Sensor Model

The concept of using wheel speed sensors and gyroscope data as the measurement updates has been widely used in robotics concepts [8], [9]. Our measurement model consists of these two sensors and is detailed below.

1) Wheel Speed Sensors: Depending on the drive axle of the vehicle, we expect different tires to experience differing amounts of tire slippage. Specifically, we expect the drive axle tires to have more longitudinal slip than the non-drive axle

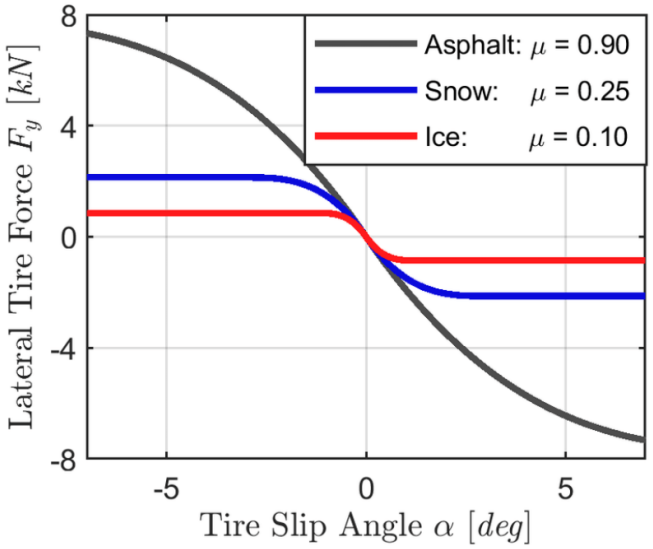

Fig. 2. Variation in lateral tire force as a function of tire slip angle for various driving surfaces according to the Fiala tire model. Depending on the driving surface and friction coefficient, there is a region of nearly linear dependence of lateral tire force on tire slip angle for small slip angles [7].

tires, therefore we would place the wheel speed sensors on the non-drive axle to obtain more accurate measurements. We can average the left and right wheel speeds to get an estimated wheel speed at the center of the rear axle for our dynamic bicycle model. This measurement update assumes that we do not have any rear wheel lateral slip, which we believe is an acceptable assumption for normal front wheel drive (FWD) vehicle operation[1] on paved roads with a vehicle brake bias towards the front.

2) Body Yaw Rate From IMU Gyroscope: Our second measurement is the angular rate measured from the gyroscope on the vehicle, which we will assume is the yaw rate of the vehicle. Note that this assumes our vehicle is on a flat plane and has negligible roll and pitch. In practice, this will probably not be true. Nevertheless, similar to the assumptions used when taking wheel speed measurements, we intend our estimation algorithm to be used in situations where this would be a highly accurate measurement of yaw rate.

As a result, we can define our measurement vector as

$$\mathbf{y} = \begin{bmatrix} \omega_{wheel} & \omega_{z,gyro} \end{bmatrix}^\top \tag{8}$$

and we can write our measurement dynamics as

$$\mathbf{y} = g(\mathbf{x}, \mathbf{u}) \tag{9}$$

$$\begin{bmatrix} \omega_{wheel} \\ \omega_{z,gyro} \end{bmatrix} = \begin{bmatrix} \frac{1}{r_{wheel}} U_x \\ r \end{bmatrix}. \tag{10}$$

[1]We simulate an FWD vehicle and our on-road experiments are conducted with an FWD vehicle.

III. Methodology

A. Estimation

We implemented four estimation architectures to examine the efficacy of our vehicle and sensor modeling:

- EKF: extended Kalman filter
- iEKF: iterated extended Kalman filter
- UKF: unscented Kalman filter ($\lambda = 2$)
- PF: particle filter ($N = 75$)

For each of the four filters, we made an implementation without actuator (plant) delay compensation and an implementation with actuator delay compensation. To do this, we chose a reasonable pure time delay to stagger the update steps of our filters, which was not necessarily identical to the time delay in the simulator or in the on-road testing. In all four cases, we used the parameters given in Table I, where $\tau_{actuator,mode}$ denotes the pure time delay used on each actuator, in either simulation or data (on-road) experiments.

TABLE I
Actuator Time Delay Compensation

$\tau_{\delta,sim}$	$\tau_{F_x,sim}$	$\tau_{\delta,data}$	$\tau_{F_x,data}$
100ms	200ms	50ms	100ms

Additionally, we selected the process and sensor noise covariances according to the OxTS manufacturer specs as well as performance tuning. For the EKF, iEKF, and UKF:

$$Q_{sim} = \begin{bmatrix} 5E-4 & 1 & 0 \\ 0 & 2E-5 & 0 \\ 0 & 0 & 1E-5 \end{bmatrix}, R_{sim} = \begin{bmatrix} 1 & 0 \\ 0 & 8.7E-5 \end{bmatrix} \tag{11}$$

$$Q_{data} = \begin{bmatrix} 5E-1 & 1 & 0 \\ 0 & 5E-4 & 0 \\ 0 & 0 & 1E-4 \end{bmatrix}, R_{data} = \begin{bmatrix} 1 & 0 \\ 0 & 8.7E-5 \end{bmatrix}. \tag{12}$$

And for the PF, the same noise statistics were used for both analyses[2]:

$$Q = \begin{bmatrix} 5E-1 & 1 & 0 \\ 0 & 5E-4 & 0 \\ 0 & 0 & 1E-4 \end{bmatrix}, R = \begin{bmatrix} 1E-1 & 0 \\ 0 & 1.7E-4 \end{bmatrix}. \tag{13}$$

B. Vehicle Simulator

To propagate forward our ground truth state in simulation, and for the predict step in the EKF, iEKF, UKF, and PF algorithms, we developed a vehicle simulator in Python that is publicly available on Github[3]. The vehicle simulator uses a first-order Euler discretization of the differential equations in the dynamics and measurement models given in Equations (2) and (9). Pseudo-code for the vehicle simulator is provided in Algorithm 1.

[2]The PF required a larger covariance in the sensor noise to avoid particle weight numerical issues.

[3]`https://github.com/rbalexan/vehicle-state-estimation`

Algorithm 1: Vehicle Simulator Step

function SimulateVehicle (x_0, u_{com}, path, dt, params);
Input : Current state (x_0), commanded actuator inputs (u_{com}), path information, dt, vehicle parameters
Output: Propagated state (x_1), actual actuator inputs (u_{act})
if *Using Actuator Delay* **then**
  Store current actuator commands and retrieve delayed commands
end
if $F_x < 0$ **then**
  Distribute braking forces to wheels according to brake proportioning
else
  Distribute throttle forces according to drive model (front-wheel drive)
end
Calculate longitudinal static weight transfer (moment balance)
Calculate slip angles at each wheel (α_f, α_r)
Calculate front and rear lateral tire forces with Fiala tire model (F_{yf}, F_{yr})
Get nonlinear state derivatives ($\dot{U}_x$, $\dot{U}_y$, $\dot{r}$)
Integrate state derivatives to get current state (x_1)
end function

C. Data Acquisition

In addition to demonstrating the effectiveness of our estimation architecture in simulation, we present results from an experiment conducted with "Niki", an automated Volkswagen GTI. Niki is equipped with a comprehensive sensor suite that includes wheel speed sensors and an OxTS-RT3000 (Figure 3) - a GNSS-aided inertial navigation system [10]. Data from the RT3000 serves as both the measurements and inputs to our estimator, as well as a baseline comparison, as the RT3000 uses state-of-the-art proprietary estimation algorithms, depicted schematically in Figure 4.

Fig. 3. OxTS-RT3000 Navigation Unit

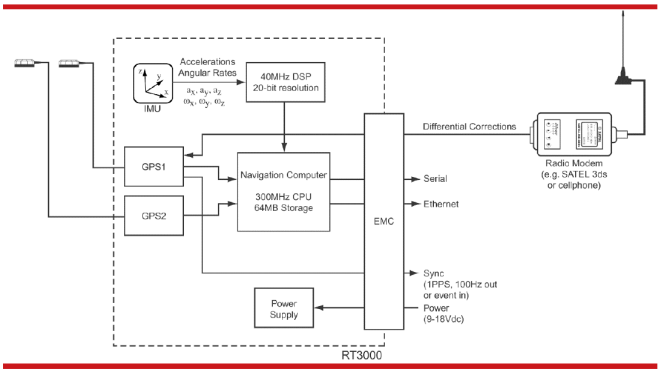

Fig. 4. OxTS-RT3000: Diagram of the estimation algorithm used as ground-truth in our experiments. The OXTS-RT3000 fuses a high-precision IMU with dual differential GPS for full state estimation [10].

IV. Experiments

To test our dynamics model, measurement model, simulation, and estimation, we designed an experiment that would subject the vehicle to forces and accelerations within the typical regime of everyday driving. The path is an oval consisting of 24 meter straight sections and 8.7 meter radius curves, which is depicted in Figure 5. These distances were chosen to allow us to test our algorithm on a real vehicle in Stanford University's Searsville parking lot. The desired speed profile, shown in Figure 6 was calculated using backwards integration from a desired lateral acceleration of 3 $\frac{m}{s^2}$ at the apex of each turn.

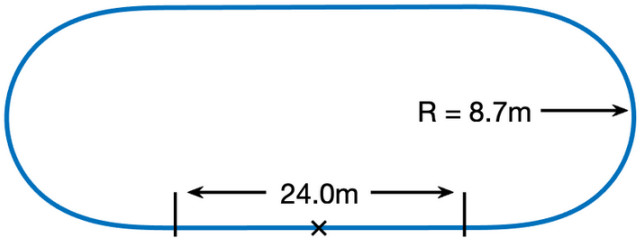

Fig. 5. Birds-eye view of the experimental test path.

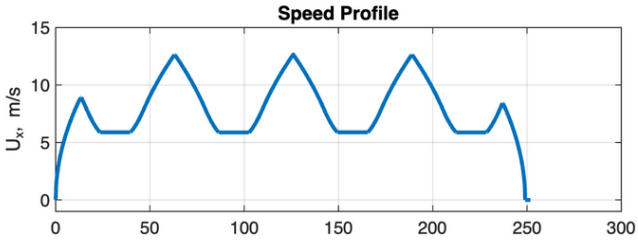

Fig. 6. Speed profile along the experimental test path. Low-level controller utilize these values as reference input. These values correspond to roughly 0.3g of acceleration and braking.

Note that while we did use data from this real experiment, we did not run it in real time on the car (or use the estimated states in the control law calculation). Our filters were designed and ran offline on the simulation data obtained from our experiments in a different course, ME227. In the future, these filtering architectures could be deployed to run in real time on the car for state estimation and control.

A. EKF Without Delay Compensation: Ideal Simulator and Real-World Data

In this section, we analyze the performance of the extended Kalman filter (EKF) on the ideal simulator and real-world dataset with no delay compensation in the filter. In Figure 7, we plot the three simulated state trajectories along with the three state estimates from the EKF with 95% confidence intervals. We observe excellent state estimation for all three states with initial state estimates converging very quickly. Throughout the trajectory, the state estimates are nearly identical to the true states and the confidence intervals include the true state and remain quite small, indicating high accuracy and precision.

In Figure 8 we show the three real-world state trajectories along with the three post-processed state estimates from the EKF with 95% confidence intervals. Again, we observe good state estimation throughout the trajectory. While the longitudinal velocity and yaw rate estimates are extremely accurate, there are a few issues with the lateral velocity estimates. In particular, there is potentially some erroneous noisy data that produces lateral velocity estimates that are far from the true state and do not recover until 3 seconds into the simulation. We do not know what caused this issue, but it can likely be reconciled in practice by adjusting the initialization of our state and estimated state. Relative to the simulated data, the real-world data is much more noisy and leads to very slightly poorer state estimates and slightly looser confidence bounds. Overall though, the EKF performs well for both cases.

B. EKF With and Without Delay Compensation: Non-Ideal Simulator

In this section, we analyze the performance of the extended Kalman filter (EKF) on the simulator with actuator delay with and without delay compensation in the filter. In the state estimates shown in Figure 9, we observe relatively good tracking of the simulator ground truth for the EKF without delay compensation and the EKF with delay compensation. However, we can see clearly that including delay compensation in the filter leads to more highly-accurate state estimates, even if the delay is not identical to the underlying process delay. The state estimates generated by the filter without delay compensation tend to lag behind the ground truth state estimates.

In the right portion of Figure 9, we show the error relative to ground truth. While the EKF without delay compensation leads to state estimates that are in error whenever the delayed actuator dynamics are excited, even the EKF with delay compensation that is not identical to the underlying process delay still experiences slight errors, albeit reduced error. This highlights the necessity for understanding delays in the system, modeling these delays, and including compensation for them in filtering algorithms. Further, adaptively learning these delay times online is likely a worthwhile endeavor since there will likely be slight parameter mismatch between simulation and reality.

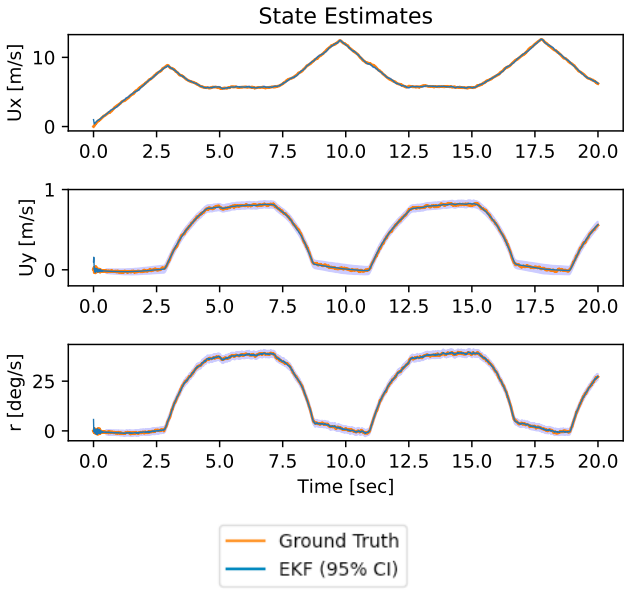

Fig. 7. Ground-truth state trajectories (*orange*) from the high-fidelity vehicle dynamics simulator and estimated state trajectories with ±95% confidence intervals from an extended Kalman filter (EKF) (*blue*).

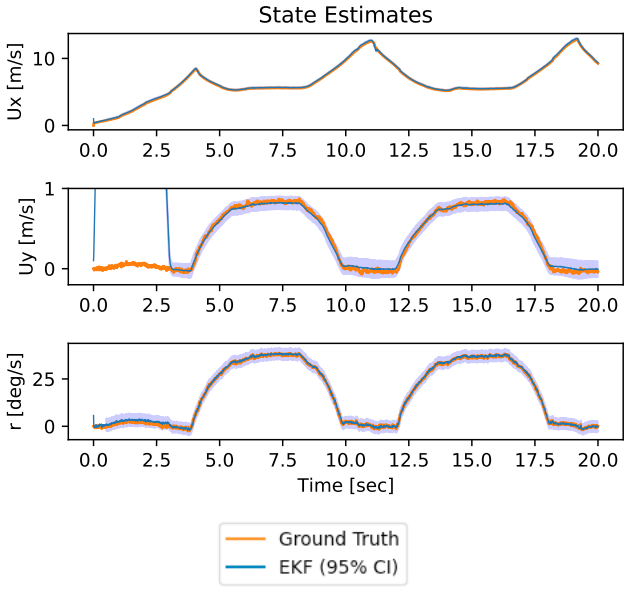

Fig. 8. Ground-truth state trajectories (*orange*) from OXTS-RT3000-recorded vehicle data and estimated state trajectories with ±95% confidence intervals from an extended Kalman filter (EKF) (*blue*).

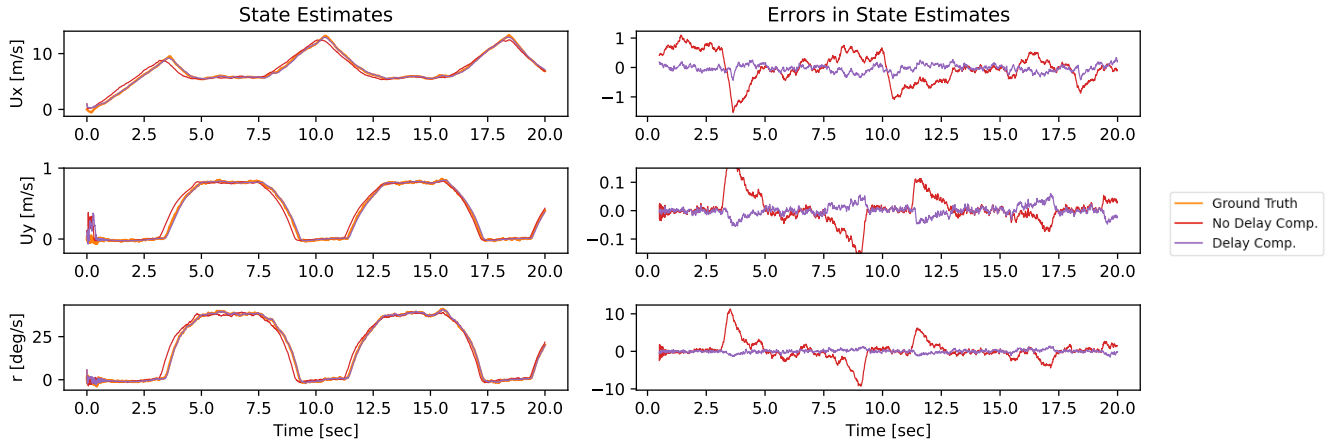

Fig. 9. *Left column:* Ground-truth state trajectories (*orange*) from the high-fidelity vehicle dynamics simulator with actuator delay and estimated state trajectories from an extended Kalman filter (EKF) without delay compensation (*red*) and an EKF with delay compensation (*purple*).
Right column: Errors of the estimated state trajectories from the ground-truth state trajectories.

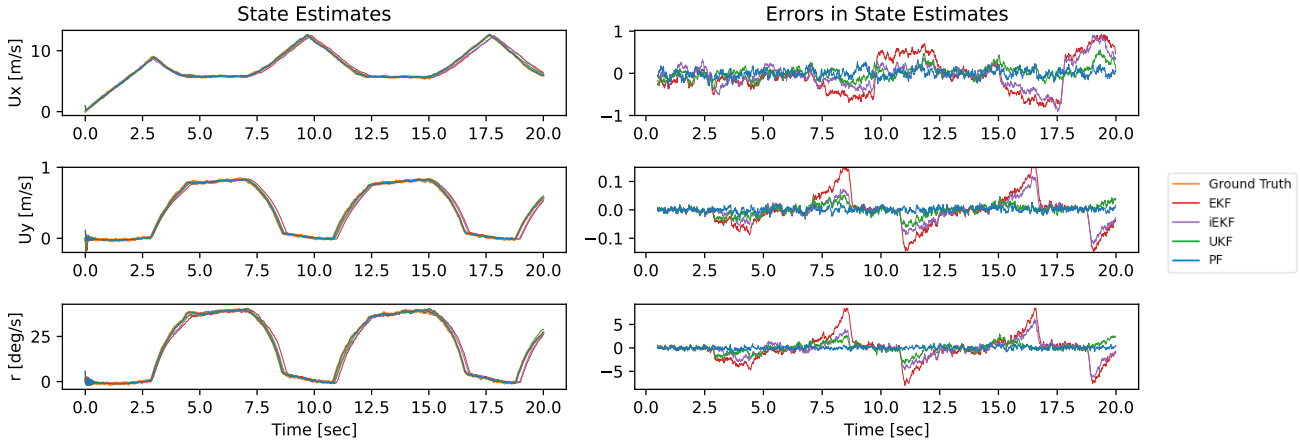

Fig. 10. *Left column:* Ground-truth state trajectories (*orange*) from the high-fidelity vehicle dynamics simulator and estimated state trajectories from an extended Kalman filter (EKF) (*red*), iterated extended Kalman filter (iEKF) (*purple*), unscented Kalman filter (UKF) (*green*), and particle filter (PF) (*blue*).
Right column: Errors of the estimated state trajectories from the ground-truth state trajectories.

C. All Filters Without Delay Compensation: Ideal Simulator

In this section we analyze all of the filter architectures without delay compensation for the ideal simulator (no delays). The purpose is to compare the filters in terms of their estimation error and draw conclusions about our model or measurements. We observe very good state estimation from all filters, with the Particle filter performing the best and EKF and iEKF performing the worst. In particular, the EKF does not track the true states well during transient changes in yaw rate and lateral velocity. This is expected, as our model is just an approximation of these dynamics and there is significant noise in our measurements. There is also some potential for adjusting the simulator parameters to make it a more realistic environment. The gains used in this case were the ones tuned for the real data case, and we can probably improve the simulated noise used in our experiments.

D. All Filters Without Delay Compensation: Real World Data

After testing in simulation, we ran our filter on data collected from the car, using the OxTS as ground truth. Here our filter does even better than simulation (given that we tuned our gains for the real data). Our forward velocity error is always under 0.25 m/s, which is accurate enough to use in different control architectures. An exception to this is the UKF which was extremely noisy on the road data. The iEKF also showed some initial divergence attributed to poor starting conditions for the filter, which will be fixed in future iterations time permitting. However, we were happy to see that even a simple EKF could give us good enough performance using our simple sensor suite.

E. Runtime

Table II shows a comparison of the average time taken for a single iteration of each filter, comparing both the simulation and real data cases. As expected there is not significant

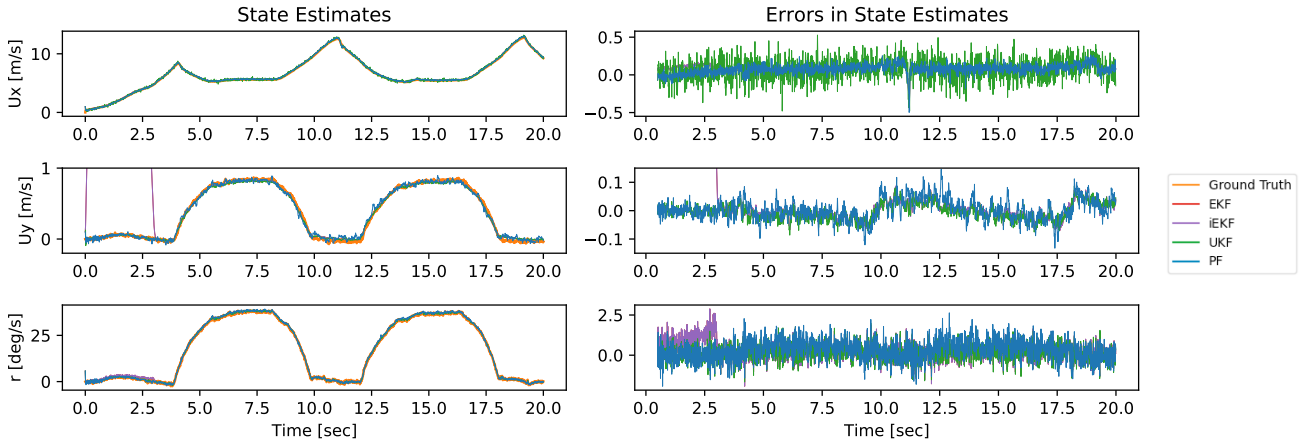

Fig. 11. *Left column:* Ground-truth state trajectories (*orange*) from OXTS-RT3000-recorded vehicle data and estimated state trajectories from an extended Kalman filter (EKF) (*red*), iterated extended Kalman filter (iEKF) (*purple*), unscented Kalman filter (UKF) (*green*), and particle filter (PF) (*blue*). *Right column:* Errors of the estimated state trajectories from the ground-truth state trajectories.

difference in processing times of simulation v/s real data, since our states and inputs remain the same between both cases. The EKF and iEKF are fairly equivalent in both cases, indicating that our comparison threshold may be too high, and there is room to improve this estimate further. The Unscented Kalman Filter is faster in comparison. As expected the particle filter takes the most amount of time to run.

When running these filters in real time on a production system we would want these filters to run at 100-200Hz, which would mean an iteration time of 5-10ms for our chosen filter implementation. This would be feasible for all filters apart from the particle filter. However, we believe there is room to improve the run-time for all of these filters, as expanded on in our future works section.

TABLE II
FILTER RUNTIMES PER ITERATION

Filter	**Avg. Iteration Time (ms)**	
	Simulation	*Real Data*
EKF	1.31	1.28
iEKF	1.31	1.29
UKF	0.82	0.68
PF	18.43	17.53

V. CONCLUSION

These results show the power of using an estimation algorithm with easily available and cheap sensors for estimation of vehicle kinematics. This estimator can run very fast, and hence is a good candidate for real time systems. An important application of this system is in developing fault tolerant safety architectures. For example, this system can be used to safely bring a vehicle to a stop in situations of loss of GPS or position localization. For a short duration, this system can be used to control the vehicle autonomously to a safe stop. Using a simple model and high integrity sensors with very few failure modes. Additionally, this filter could also be used to provide velocity estimates for a low level velocity controller in a multi-level control architecture.

VI. FUTURE WORK

As our research focused on what is needed to run our filter on a real time production level system, there are some next steps we would like to explore to make our filter more robust.

A. Accounting for measurement dropouts

High resolution wheel encoders can sometimes have strange artifacts associates with updates like measurement dropouts or out of order measurements. Specifically for this purpose we would like to implement a backtracking filter to handle this case

B. Better delay compensation

Our current delay compensation algorithm assumes a constant delay value for throttle, braking and steering. On real systems this delay can vary quite a lot between these 3, and even more if we have both a regenerative and friction braking ability. We would like to explore using separate delay values for the three, perhaps even building a filter to estimate these delays online.

C. Speed improvements

Even though our EKF implementation runs fast, we are still not at the performance needed to run at 100-200Hz on a real time system. Some easy areas for gain would be to rewrite our estimator in C++, as well as reduce linear algebra operations like matrix inversions by performing a LU factorization and doing a matrix solve instead.

VII. ACKNOWLEDGEMENTS

We would like to thank Professor Schwager and the CAs for a great experience this quarter, as well as helping us put together this final paper. Additionally, we would like to thank Professor Gerdes and the ME 227 staff for providing us with access to the class material as well as data from their vehicle Niki to use in our project.

REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, 1960.

[2] P. J. Venhovens and K. Naab, "Vehicle dynamics estimation using Kalman filters," *Vehicle System Dynamics*, vol. 32, no. 2, pp. 171–184, 8 1999.

[3] M. C. Best, T. J. Gordon, and P. J. Dixon, "Extended adaptive Kalman filter for real-time state estimation of vehicle handling dynamics," *Vehicle System Dynamics*, vol. 34, no. 1, pp. 57–75, 2000.

[4] T. A. Wenzel, K. J. Burnham, M. V. Blundell, and R. A. Williams, "Dual extended Kalman filter for vehicle state and parameter estimation," in *Vehicle System Dynamics*, vol. 44, no. 2, 2 2006, pp. 153–171.

[5] A. Mistri, "Planar vehicle dynamics using bicycle model," in *Lecture Notes in Mechanical Engineering*, 2019.

[6] E. Fiala, "Seitenkraften am rollenden Luftreifen," *V. D. I.*, vol. 96, pp. 973–979, 1954. [Online]. Available: http://ci.nii.ac.jp/naid/20001399772/en/

[7] J. P. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *Proceedings of the American Control Conference*, vol. 2019-July, 2019.

[8] J. J. Oh and S. B. Choi, "Vehicle velocity observer design using 6-D IMU and multiple-observer approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, 2012.

[9] I. Ashokaraj, P. Silson, and A. Tsourdos, "Application of an Extended Kalman Filter to Multiple Low Cost Navigation Sensors in Wheeled Mobile Robots," in *Proceedings of IEEE Sensors*, vol. 1, no. 2, 2002.

[10] Oxford Technical Solutions Ltd., "RT3000 v3," 5 2021.