Project Report

1st Dipti V Motwani

I. Introduction

We explore the application of the Kalman filter to the localization problem in robotics, using a sample trajectory of a car navigating an outdoor environment from the KITTI dataset [7]. The Kalman filter is a recursive Bayesian estimator used to infer the state of a moving body by optimally fusing information from 1) noisy propogation of its dynamics in response to a control input ("dynamics model") and 2) one or more noisy sensor readings of its state ("observation model"). The state estimation is termed as "belief" over the robot state, and the uncertainties in the process dynamics and in the sensor measurement are termed "process noise" and "measurement noise" respectively.

The standard Kalman filter makes two assumptions: 1) modeling the distribution over the state vector, process noise and measurement noise as Gaussians (with the noise parameters being 0-mean Gaussians), and 2) linearity of the system dynamics and observation model. In practice, the Gaussian approximation works farily well for robot localization problems even if the true errors are only approximately Gaussian, unless one needs to maintain a multi-model belief state (for example, if we start with the belief that a robot can be in one of several rooms, resulting in a few distinct peaks of probability mass) or a highly non-Gaussian/skewed belief (for example, a robot navigating a long hallway with a door at one end: where our belief is likely that the robot is further along the hallway but still needs to be compatible with being near the start). In such scenarios, alternative methods such as particle filters or Gaussian Mixture Models may be required to accurately represent the robot's belief state. For our specific dataset of car localization in an outdoor setting, a multimodal or skewed belief state is not required and the Gaussian approximation is expected to work well.

The second Kalman assumption of linear dynamics is more restrictive and breaks down as soon as the robot follows a non-linear (curved) trajectory, since the evolution of the robot's state is related to the control input through nonlinear (e.g., trigonometric) functions of the robot's heading and motion. Two widely used techniques to handle nonlinearities are the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). However, tackling non-linearity is not the focus of this project, and we specially chose an approximately linear car trajectory (change in yaw less than 0.15 rad) from the KITTI dataset, so that a linear model is sufficiently accurate to allow us to focus on the development of Bayesian techniques for robotics. This trajectory data file is not included due to large size, but can be downloaded as `2011_09_26_drive_0001` at [7].

Specifically, the process and measurement noise parameters may not be known in many situations, and while specifications such as sensor accuracy may offer good starting priors, they are often hand-tuned for best accuracy of the estimated state against a known ground truth. We demonstrate the application of the Kalman filter for accurate localization of the car fusing GPS measurement and acceleration input, first hand-tuning the process and measurement noise, and then experimenting with Bayesian inference for adaptive tuning of these noise parameters using an Expectation Maximisation approach.

II. Problem Setup

We localize a car moving on a 2D plane with two translational degrees of freedom (in X and Y) and one rotational (yaw) in response to acceleration input measured by an Inertial Measurement Unit (IMU). This is a Constant Acceleration model in response to a deterministic control input (acceleration). The acceleration is measured in the frame of reference of the car (termed "body frame") and is transformed to the "world frame" to be in the same frame as the robot's coordinates.

The KITTI dataset provides accurate ground truth measurements for the X and Y location, velocities and orientation of the vehicle from a state-of-the-art Inertial Navigation System. To simulate noisy GPS sensor readings, we add synthetic noise (0-mean Gaussian) to the X and Y positions, while the precise X/Y location is treated as the ground truth measurement reference for our evaluation. We do not use measurement inputs for any other parameters (such as velocity or yaw) to mimic a real-world scenario where only GPS readings are available.

In order to propogate the dynamics from acceleration input to the X and Y position, we need to maintain a belief over the robot's X-Y coordinates as well as X-Y velocities, resulting in a 4x1-dimensional state vector. We also need to update the robot's yaw with time to be able to transform the acceleration input into the world frame. However, we update the yaw separately instead of keeping it as part of the state vector, so that the non-linear transformation from body to world frame can be done outside the Kalman filter operations, and the system dynamics themselves (from acceleration to velocity and position) remain linear. The disadvantage of this choice is that the yaw estimate does not get corrected by the covariances with the X and Y position updates, manifesting as a gradual drift in the yaw resulting from forward propogation only. However, as our selected trajectory involves a very small change in yaw (below 1.5 rad over the entire sequence), this is acceptable for our experiments.

III. Kalman Filter Formulation

A. Bayesian Motivation

The linear Kalman filter [1] can be understood in the framework of section 8.4 of the course textbook [2]. A conjugate normal-normal prior and likelihood pair result in a posterior whose mean is the weighted average of the prior and likelihood means, weighted by the inverse of the respective variances.

$$\theta|X \sim \mathcal{N}\left(\frac{\tau^2}{\sigma^2+\tau^2}X + \frac{\sigma^2}{\sigma^2+\tau^2}\mu, \frac{\sigma^2\tau^2}{\sigma^2+\tau^2}\right) \tag{1}$$

The posterior variance is smaller than both the prior and likelihood variances.
The Gaussian assumption of the Kalman filter over the state, and measurement and process parameters allows us to apply this same concept to update the posterior belief over the 4-dimensional state at time t, as a function of the prior belief at time (t-1) and the likelihood of seeing the measurement data, given the state.

$$\pi(\mathbf{x}_t \mid \mathbf{m}_t) \propto \pi(\mathbf{m}_t \mid \mathbf{x}_t)\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$$

If we rearrange the terms of the mean in Equation 1, it can be expressed as

$$\mu + \frac{\tau^2}{\sigma^2+\tau^2}(X-\mu) \tag{2}$$

This $\frac{\tau^2}{\sigma^2+\tau^2}$ functions like a "gain" for updating the prior mean μ with the data X and this term will be analogous to the Kalman gain term derived in section III.D below.
The variance term can also be expressed in terms of the above gain as:

$$(1 - \frac{\tau^2}{\sigma^2+\tau^2})\tau^2 \tag{3}$$

Again the variance of the posterior is smaller than both the prior and likelihood variances.

B. State and Measurement Vectors

The state vector $\mathbf{x}_t$ consists of the 2D position (p) and 2D velocity (v) in the world frame. The measurement vector $\mathbf{z}_t$ consists of the observed GPS positions. The control input $\mathbf{u}_{t-1}$ is the world-frame acceleration derived from the IMU.

$$\mathbf{x}_t = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}_t, \quad \mathbf{z}_t = \begin{bmatrix} z_{gps,x} \\ z_{gps,y} \end{bmatrix}_t, \quad \mathbf{u}_{t-1} = \begin{bmatrix} a_x \\ a_y \end{bmatrix}_{t-1} \tag{4}$$

C. System Matrices

The system assumes a Constant Acceleration model where the acceleration is provided as a deterministic control input, and process noise arises from unmodeled jerk (changes in acceleration). Detailed derivations of the below matrices and equations for the 1D case can be found under sec. 6.3.2 of [5]. Let Δt be the time step. The State Transition Matrix $\mathbf{A}$ propogates position and velocity over an interval Δt and the Control Input Matrix $\mathbf{B}$ propogates acceleration, according to the equations of motion $\Delta x = v\Delta t + (1/2)a\Delta t^2$.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{6}$$

The Process Noise Covariance matrix $\mathbf{Q}$ expresses how an error in the acceleration propogates to the position and velocity states: since position varies as $(1/2)a\Delta t^2$, the variance of position is a factor of $(1/4)\Delta t^4$ times the acceleration noise (Q[0,0] and Q[1,1]). Similarly since velocity = at, the variance of velocity is a factor of Δt^2 (Q[2,2] and Q[3,3]). This matrix is scaled by the scalar process variance $q = \sigma^2_{accel}$.
The Measurement Noise Covariance matrix $\mathbf{R}$ quantifies the measurement errors in X and Y position, which are treated as independent with 0

off-diagonal terms. **R** is scaled by the scalar measurement variance $r = \sigma^2_{meas}$.
As the system dynamics (when expressed in the world frame) and the sensor measurement are expected to be equally precise in X and Y (i.e. no reason to believe one direction is more accurate), we use a single parameter each for the process and measurement noise, for an isotropic covariance matrix.

$$\mathbf{Q} = q \cdot \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \tag{7}$$

$$\mathbf{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \tag{8}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{9}$$

The Measurement Matrix **H** represents the "observation model" mapping the state vector for comparison with the GPS measurement, and since the X/Y state and measurement map 1:1 in our case, **H** simply extracts the X/Y position from the first two rows of the state vector.

D. Kalman Filter Algorithm

The filter proceeds recursively in two steps: the Prediction (Time Update) and the Correction (Measurement Update).
1) Prediction (Time Update): The state is propagated forward using the motion model **A** and the IMU control input $\mathbf{u}_{t-1}$. (Where the notation $@t$ represents the estimate of the respective quantity at timestep t). The prediction step adds to the covariance matrix (increases uncertainty).

$$\hat{\mathbf{x}}_{t@t-1} = \mathbf{A}\hat{\mathbf{x}}_{t-1@t-1} + \mathbf{B}\mathbf{u}_{t-1} \tag{10}$$
$$\mathbf{P}_{t@t-1} = \mathbf{A}\mathbf{P}_{t-1@t-1}\mathbf{A}^\top + \mathbf{Q} \tag{11}$$

2) Correction (Measurement Update): This is the key Bayesian update step.The predicted state is corrected using the GPS measurement $\mathbf{z}_t$. The innovation measures the difference between the measured and predicted state, and the innovation covariance is the sum of the process and measurement covariances.
Similar to the single-variable case, we calculate a gain representing the ratio of the process variance as a proportion of the total innovation covariance. This "Kalman gain" is the factor by which the innovation is applied to the state update, and is analogous to the term highlighted in 2.

$$\text{Innovation:} \quad \mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_{t@t-1} \tag{12}$$
$$\text{Innovation Covariance:} \quad \mathbf{S}_t = \mathbf{H}\mathbf{P}_{t@t-1}\mathbf{H}^\top + \mathbf{R} \tag{13}$$
$$\text{Kalman Gain:} \quad \mathbf{K}_t = \mathbf{P}_{t@t-1}\mathbf{H}^\top\mathbf{S}_t^{-1} \tag{14}$$
$$\text{State Update:} \quad \hat{\mathbf{x}}_{t@t} = \hat{\mathbf{x}}_{t@t-1} + \mathbf{K}_t\mathbf{y}_t \tag{15}$$
$$\text{Covariance Update:} \quad \mathbf{P}_{t@t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_{t@t-1} \tag{16}$$

The process covariance matrix is multiplied by a factor of $(I - K)$, analagous to 3 in section III.A. The Corrections step decreases the magnitude of the covariance matrix (decreases uncertainty). Like in the single-variable case, the inverse magnitides of process and measurement variance represent how much the posterior trusts the process dynamics vs. the measurement in any timestep.

IV. Rough Hand-tuning of Noise Parameters

A. Tuning Process and Results

Since we inject random noise with stddev 1 to simulate GPS measurements from the ground truth data, we start with a measurement variance of 1 and experiment with values ranging from 0.1 to 100 on a log scale. Fixing measurement variance at 1, the process variance is also varied from 0.1 to 100 and performance is measured in terms of the error in the fused state estimate from the ground truth measurement at each time step. This error is plotted and visualized with the variance band predicted by the process covariance matrix, as it evolves in each time step (Fig. 2, 3). For a well-tuned filter, 68% of errors should lie within one standard deviation along the respective axis; a smaller or larger proportion of errors contained within these bands indicates an over- and under- confident filter respectively. To facilitate comparison using a single scalar metric we also calculate RMS error between state predictions and ground truth. Finally, we also calculate the mean of the X and Y errors to determine if there is a significant bias away from 0. This is relevant to identifying over-reliance on the process dynamics, because without a measurement update the state estimate drifts over time as errors accumulate. The rough tuning yielded best performance at process variance = measurement variance = 1.0 as summarized below (Tab. I).

TABLE I: Performance with best hand-tuned process variance = 1; measurement variance = 1

	X	**Y**
Mean Position Error	-0.0450	0.1068
Position Error Std	0.2491	0.3169
Filter Uncertainty	0.3917	0.3917
% Errors in +-1σ	87.9	73.8
Bias	-0.045	0.1068
Yaw Error	Mean: -0.0018 rad, Std: 0.0022 rad	
RMSE	0.4194	

B. Discussion

The position error standard deviations lie well within the $\pm 1\sigma$ bounds estimated by the filter. Bias in X errors are small, while bias in Y is more substantial. Note that despite not updating the velocity through a sensor measurement, the velocity estimates still get corrected by the position updates through the covariance matrix, and track the ground truth velocity very closely (Fig. 4). This is not true of the estimated yaw (Fig. 5), which shows a significant drift over the trajectory, since it is estimated separately from the rest of the state. [4] shows that a similar indirect correction of yaw is achieved when yaw is included in the state for non-linear dynamics modeling on the KITTI dataset.

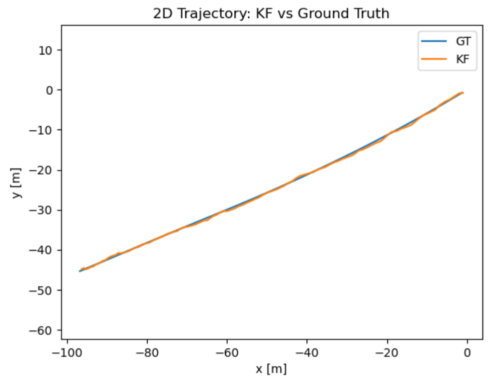

Fig. 1: Ground Truth and Filtered Trajectory

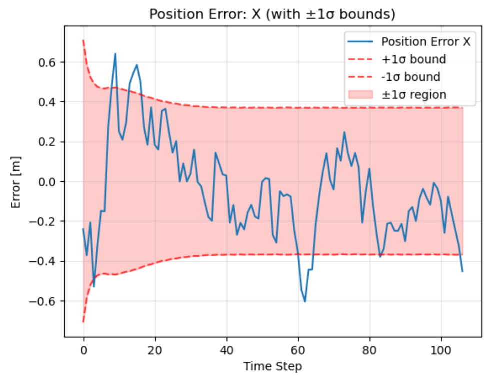

Fig. 2: X error with variance bands

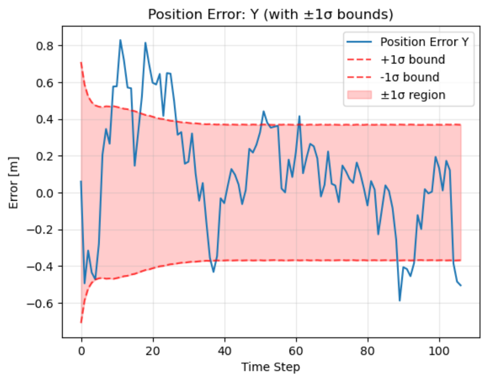

Fig. 3: Y error with variance bands

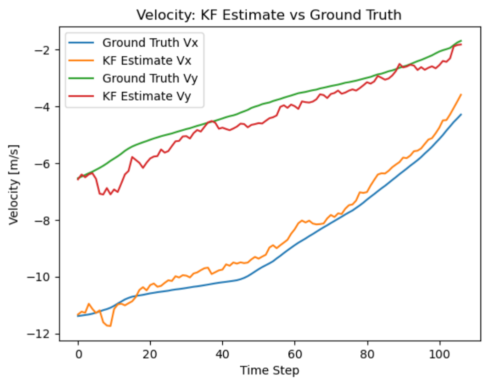

Fig. 4: Ground Truth and Filtered Velocity

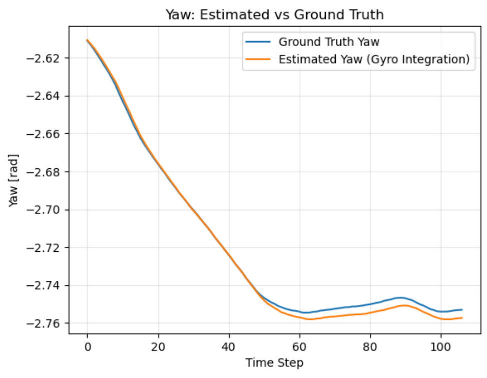

Fig. 5: Ground Truth and Filtered Yaw

V. Process-only and Measurement-only baselines

It is instructive to simulate how the Kalman filter's performance would compare to a non-Bayesian estimation of the state using either only the process dynamics (known as "dead reckoning") or only the measurement; we simulate these by setting the measurement variance or process variance to very large values in our algorithm respectively. Complete reliance on the process dynamics without any corrective feedback from measurements leads to a trajectory that drifts from the ground truth over time due to accumulation of systematic errors in a single direction (Fig. 6). On the other hand, reliance on the measuremnts alone results in a very noisy trajectory due to the high degree of fluctuation of a single measurement, but whose mean value closely tracks the ground truth mean (Fig. 7).

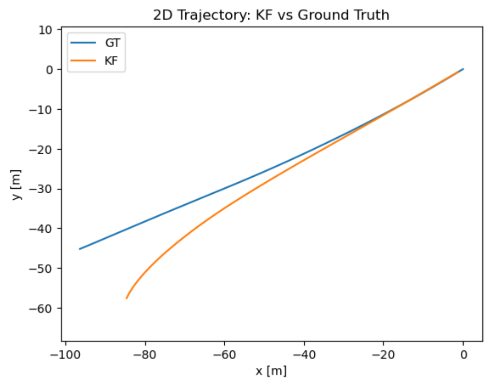

Fig. 6: Ground Truth and Filtered Trajectory: Dead Reckoning

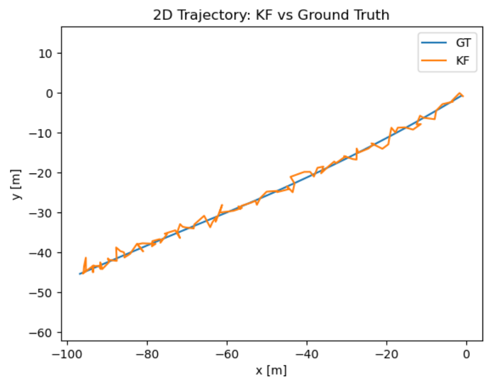

Fig. 7: Ground Truth and Filtered Trajectory: Measurement Only

VI. Learning Noise Parameters

An interesting question is whether we can learn a probability distribution over the noise parameters from data, by modeling them as random variables with a prior distribution. Ideally, to compute the full posterior for the parameters of a state-space model conditioned on the model and the observed data (measurements), we can compute the marginal posterior of parameters by integrating out all the states. As usual this integral is difficult in practice, and prior work ([3], Chap. 12) in Bayesian parameter estimation for state-space models suggests direct optimization or Expectation Maximisation (EM) algorithms for computing MAP estimates of parameters, or MCMC methods for generating approximations of their posterior distributions.

Here we experiment with the EM algorithm for estimating the process and measurement noise parameters. An Inverse Gamma prior is chosen for both variances so as to form a conjugate pair with the Normal likelihood of the residual errors used to update their posteriors, as explained below.

The modes of both priors are set at 1.0 to represent our lack of stronger conviction in either the process or the measurement. Initially, values of α and β are chosen to be small to make the priors weakly informative, since we assume we know little about the system's noise parameters at the start.

$$\beta/(\alpha+1) = 1 \implies \beta = 2; \alpha = 1$$

The EM algorithm alternates Expectation and Maximisation steps over a loop, starting from initial guesses. A general treatment of EM for Bayesian parameter estimation can be found in ([3], sec. 12.2.3), and the specific adaptation for our case is explained below:

For $i = 0$ to n iterations:

- **E-Step:** Given the current parameter estimates, compute the posterior distribution of the states conditioned on the complete measurement sequence. This involves running the Forward Kalman Filter followed by the Backward Rauch-Tung-Striebel (RTS) Smoother to obtain the smoothed state means and covariances for all time steps. Note, both the forward propagation and the backward smoothing are necessary to find the distribution of states that accounts for the full history of measurements (past and future) at every point in the trajectory.
- **M-Step:** Given the smoothed state trajectories from the E-step, update the parameters to maximize the likelihood of seeing these states. The process variance q is updated by calculating the sum of squared errors (SSE) of the velocity deviations (residuals between the smoothed velocity change and the expected IMU acceleration). Similarly, the measurement variance r is updated using the SSE of the position residuals (difference between GPS measurements and smoothed positions).
  Since we have placed Inverse-Gamma priors on the variances, and assume a normal likelihood of these residuals with mean 0 and the respective process or measurement variance, this becomes a Maximum A Posteriori (MAP)

estimate for the conjugate InverseGamma-Normal pair.
The update rules take the form of the mode of the posterior Inverse-Gamma distribution.

$$q^{(i+1)} = \frac{\beta^{q}_{prior} + \frac{1}{2}\sum_t ||\epsilon_{process}||^2}{\alpha^{q}_{prior} + \frac{1}{2}N_{samples} + 1} \tag{17}$$

$$r^{(i+1)} = \frac{\beta^{r}_{prior} + \frac{1}{2}\sum_t ||\epsilon_{meas}||^2}{\alpha^{r}_{prior} + \frac{1}{2}N_{samples} + 1} \tag{18}$$

Note, under the isotropic assumption we treat errors in X and Y the same, and they are simply summed into one SSE term each for the measurement and process noise.
Two simpler methods were tried with poor results before implementing the full EM algorithm above, which are documented in A and B below, but the reader may skip directly to C.

A. *Online parameter estimation using position error*

Initially, to pursue the goal of "online" tuning of the filter concurrently with the Kalman filtering for the trajectory prediciton, the SSE for acceleration was obtained online by directly taking the difference of expected and actual position (X/Y), after each step of the Kalman update, updating the process variance for the next update step. This led to extremely noisy and inflated estimates for the acceleration and process noise, as it was attempting to account for large errors from noisy measurements in a small time step, based on an acceleration error. Two key improvements were necessary: 1) calculate process residuals based on velocity deviations for more stable estimates, and 2) estimate the noise parameters in an offline, iterative fashion based on the complete trajectory. This offline method can still be useful to learn parameters from an iniital segment of a trajectory, and fix them at tuned values for subsequent trajectory prediciton.

B. *Without backward smoothing*

We tried estimating the measurement and process noise using the residuals directly from the Kalman filtered trajectory (without the RTS smoothing step above). This led to huge estimates for the process noise as, without smoothing, it tried to account for the huge jumps resulting from measurement noise. This lumps the measurement noise into the process noise, and consequently makes the Kalman filter ignore the process in favour of the measurements. The correct method should estimate both noise parameters from a smoothed trajectory to keep them separate, for which we employed RTS smoothing below.

C. *With RTS smoothing*

RTS smoothing implementation was borrowed from [6] "Fixed Interval Smoothing" and a detailed introduction can be found in sec 8.2 of [3]. Starting from the end of the trajectory, the RTS smoother updates each previous filtered state $\hat{x}_{t|T}$ (at time t) with an error term quantifying how much the predicted state at t+1 ($\hat{x}_{t+1@t}$) differs from the smoothed value at t+1 ($\hat{x}_{t+1|T}$):

$$\hat{x}_{t|T} = \hat{x}_{t@t} + \mathbf{C}_t \left(\hat{x}_{t+1|T} - \hat{x}_{t+1@t}\right) \tag{19}$$

where

$$\mathbf{C}_t = \mathbf{P}_{t@t}\mathbf{A}^{\top}\mathbf{P}^{-1}_{t+1@t} \tag{20}$$

$\mathbf{P}_{t@t}$ is the current uncertainty, $\mathbf{A}^{\top}$ the dynamics from t to t+1, and $\mathbf{P}^{-1}_{t+1@t}$ the inverse of the predicted uncertainty.
While the filtered state estimate is conditoned on the measurements upto its timestep, the smoothed solution is conditioned on the full measurement sequence. This serves to smooth out erratic fluctuations in the Kalman-filtered trajectory that result from fusing noisy measurement updates.
With this experiment, measurement noise remains on a similar scale as before, but the process noise reduces drastically as it is affected much less by the deviations from noisy measurement. This led to the opposite issue of estimated process noise being too small, to the extent of the Kalman filter ignoring the measurement and suffering from a high bias due to over-reliance on the process. Both cases with and without smoothing point to two practical challenges with the EM method for estimating noise parameters: an extremely low (or high) value of q can both cause a feedback loop, trusting the process too much (or too less), and driving the q estimate even lower (or higher), respectively. In our case, the trajectory can be explained both as a very straight line with high sensor noise (q small, r large) or vice-versa as a very wiggly line with low sensor noise (q large, r small). This points to the need for stronger priors on one or both of the parameters to guide the estimation.

D. *With RTS smoothing and stronger priors*

To increase the effective sample size of the priors, the magnitudes of alpha and beta hyperparameters were scaled by a factor of 100-200, while keeping the prior mode constant at 1. This effective sample size is of the same order of magnitude as the number of points in our trajectory (107). The best results, in terms of RMSE of the predicted trajectory from the ground truth, were obtained for the stronger prior on process noise (sample size =

200) leading to final process and measurement noise estimates of 0.67 and 0.89 respectively. The strength of the measurement noise prior did not seem to affect the results, so the below results are reported with the weak prior on measurement noise. However, even with the strong process noise prior, the RMSE was slightly higher than that of the hand-tuned filter - the new filter still exhibits a slight higher drift due to over-reliance on the process, indicating that process noise is still being under-estimated despite a strong prior at 1. Moreover, imposing a strong prior on the process noise requires some knowledge about an acceptable range of q for the system (though that could still be easier than hand-tuning in some cases).

E. *With RTS smoothing, stronger priors and acceleration bias*

Finally, we briefly experimented with adding a fixed bias parameter in the acceleration, estimated as the mean of the acceleration residuals in each iteration of the EM loop, in an attempt to correct the drift arising from low process variance by better modeling the acceleration. This however led to much poorer results, both with and without strong priors on the process noise, and was not explored further.

F. *Best Estimated Model*

In summary, the best RMSE on the Kalman filter trajectory was obtained using the parameters estimated from EM with RTS smoothing and a strong process noise prior. Plots showing the trajectory (Fig. 8), position errors with the estimated variances (Fig. 9, 10), mean errors and RMSE are summarized below (Tab. II). Note, velocity and yaw plots are omiited as they are similar to the hand-tuned filter.

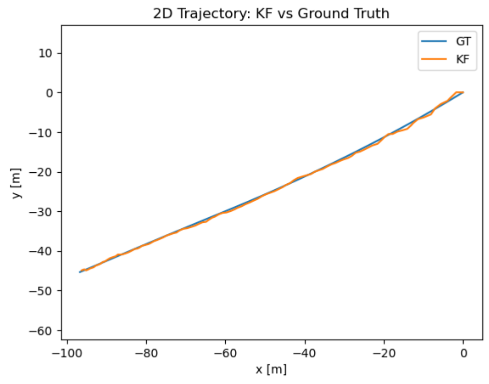

Fig. 8: Ground Truth and Filtered Trajectory

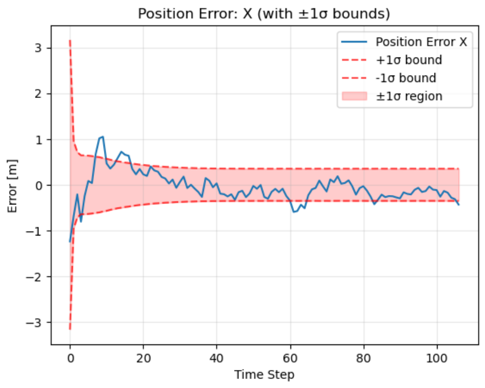

Fig. 9: X error shown with variance bands

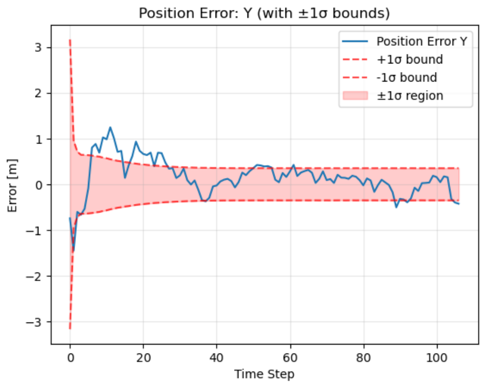

Fig. 10: Y error shown with variance bands

TABLE II: Performance with inferred process variance = 0.67; measurement variance = 0.89

	X	**Y**
Mean Position Error	-0.0734	0.1320
Position Error Std	0.3082	0.3606
Filter Uncertainty (mean σ)	0.3850	0.3850
% Errors in $\pm 1\sigma$	85.0	68.2
Bias	-0.0734	0.1320
RMSE	0.4978	

VII. Conclusion

We demonstrate the application of Kalman filters, a Bayesian filtering technique, for fusing process dynamics and sensor measurement for a robot localization problem, using car IMU and GPS data from the KITTI dataset as an example. We compare the localisation error against a ground truth trajectory, for both cases of hand-tuned noise parameters, versus parameters estimated in an EM loop using a filtered+smoothed trajectory. The parameter estimation approach requires a strong prior on the process noise to yield reasonable estimates, and shows slightly inferior performance than hand-tuning. This method could be considered where some information about a reasonable range for the process noise is known for prior elicitation, and where hand tuning may be harder.

VIII. Future Work

While we focused on a linear motion model for this project, it would be interesting to test the Extended Kalman Filter on a trajectory with significant curvature, with both the hand-tuned and EM-estimated parameters. It is expected that more accurate modeling of the non-linear dynamics will allow the process noise to take smaller values. The parameter estimation required a strong prior to be effective, and it is worth exploring if a different prior-likelihood pair could be more robust to some of the feedback loop issues seen during the EM (possibly using MCMC sampling methods if the pair is not conjugate). Separately, although our GPS noise was a synthetic Gaussian in this project, real-world data may contain heavy-tailed outliers. It would be interesting to observe if, in the presence of outliers in the sensor or IMU data, a different distribution family for the belief and/or noise parameters would yield better results (for ex, a Student's T).

IX. Program Files

- `LinearKF2D.py`: Linear Kalman Filter class with prediction and correction steps
- `EM_KalmanFilter.py`: EM algorithm for estimation of process and measurement noise parameters in the Kalman Filter
- `KF0.ipynb`: Notebook applying Kalman Filter on KITTI dataset trajectory
- `KF0_EMKF.ipynb`: Notebook applying Kalman Filter with EM for parameter estimation on KITTI dataset trajectory

References

[1] Kalman, R. E. (1960). "A new approach to linear filtering and prediction problems." *Journal of Basic Engineering*, 82(1), 35-45.

[2] Brani Vidakovic, *Engineering Biostatistics: An Introduction using MATLAB® and WinBUGS*.

[3] Simo Särkkä and Antti Nummenmaa, "Recursive noise adaptive Kalman filtering by variational Bayesian approximations," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 596–600, 2009.

[4] Motoki Kimura, "Kalman Filter with KITTI Example (Jupyter Notebook)," https://github.com/motokimura/kalman_filter_with_kitti/blob/d0983500d54fe673dc9a35301b2310ff2f9fb502/demo.ipynb

[5] Bar-Shalom, Y., Li, X. R., & Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation.* Wiley.

[6] Roger R. Labbe Jr., "Kalman and Bayesian Filters in Python: Smoothing," https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/13-Smoothing.ipynb

[7] KITTI Vision Benchmark Suite, "Raw Data," https://www.cvlibs.net/datasets/kitti/raw_data.php