TTK4250 Sensor Fusion Report

1 Introduction

State estimation is undoubtedly one of the most fundamental modules in applications within fields as robotics, navigation and autonomous systems. This will for many systems boil down to a filtering task, as pure model-based estimation usually lacks accurate enough representation of the model to predict the evolution of states, making the need for exteroceptive and interoceptive sensors necessary. Filtering out estimates of true states hidden sensor-data is then the key aspect of state estimation, often requiring fusion of multiple sensors. The Kalman filter plays a significant role in state estimation and sensor fusion when approaching the filtering problem from a Bayesian perspective, and is adapted to a variety of applications. This report will further investigate the use of variations of the Kalman filter for applications as target tracking, inertial navigation and simultaneous localization and mapping (SLAM).

2 IMM-PDAF

2.1 IMM-PDAF on simulated data

2.1.1 Initial tuning of an IMM-PDAF

Tuning an IMM-PDAF involves tuning the measurement noise r, process noise q, clutter rate λ, detection probability P_D and gate size g^2. A good initial guess of these parameters is important for speeding up the tuning process, and is achieved by first looking at the data at hand. As we have ground truth data available, we can compare the true trajectory against measurements to make a reasonable guess for some parameters.

Plotting the true trajectory with nearby measurements, initial values for P_D, g^2 and r can be made. We see from the simulated data that measurements close to and on top of the trajectory appear in almost every frame, so the probability of detection seems high. This also means the gate size can be chosen relatively small. The measurement noise can be estimated by calculating the sample standard deviation of the measurements from the true trajectory, using ground truth as mean for every frame. This method of guessing P_D, g^2 and r assumes that measurements originating from the target are well encapsulated by the "nearby" measurements, where "nearby" is a subjective bound we must choose.

The process noise q should reflect the strongest maneuvers expected in the data, which is observed by the true trajectory. As the chosen models are CV and CT, with process noise q_{CV} and q_{CT} respectively, the fit of these models to the trajectory must be evaluated to make an initial guess of q. It can be seen that the trajectory follows paths that are relatively straight (not necessarily constant velocity) or with constant turn, so the process noise is assumed low. In order for the IMM-PDAF to switch fast between the models, the process noise for the models is chosen sufficiently different from each other so one model gates a measurement with higher probabilistic density than others. The transitions are modeled as a Markov chain with transition probabilities $\boldsymbol{\pi}$, where each model is chosen to have a high probability of remaining in its state so that the IMM-PDAF doesn't switch too frequently between models, as the true trajectory has clear transitions between CV and CT.

Finally, the cardinality of clutter is modeled as Poisson distributed. Looking at the data again, it can be seen that the clutter is nearly uniformly spread in the tracking region, so we make use of the approximation,

$$\mathrm{E}\left[m_k\right] = \lambda V + P_D \approx \frac{1}{K}\sum_{k=1}^{K} m_k \tag{1}$$

as the expected number of measurements, where V is the grid-size, K is the total number of time-steps and m_k is the number of measurements at a given time-step k. An estimate can then be made by solving for λ.

2.1.2 Validating performance of parameters

The performance of the IMM-PDAF is largely validated by metrics as RMSE and filter consistency. After initializing the tuning parameters by Section 2.1.1, they were continuously tuned to produce best RMSE and

filter consistency.

To achieve smooth estimates of position, the measurement noise was kept magnitudes higher than the process noise. Increasing it too much gave an under-confident filter with much of the NEES below the 95 % confidence bound, while decreasing it too much gave an overconfident filter.

Setting the gate size too low made the filter miss crucial measurements, making it rely too much on predictions. E.g in one of the turns it misses an important measurement that otherwise makes the filter transition to a CT model, making it continue straight in a CV model, thus completely missing the trajectory. A size of 3σ captured most crucial measurements, and increasing the size had no effect on RMSE or consistency as measurements further away had too low weights in the filter compared to closer ones.

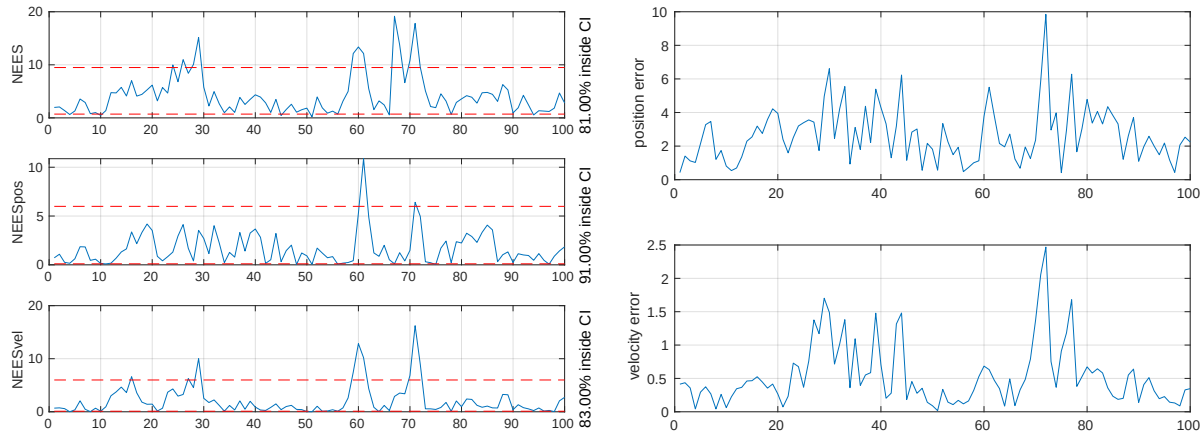
The detection probability does not affect the performance much except for extremely low values, and is set to a high value as this gave best RMSE. The clutter rate is then set according to Equation (1).

Tuning process noise followed same procedure as measurement noise, where both NEES and RMSE were considered. A lower process noise for the CV model, compared to the CT model, was found to give best RMSE. This may be because the CV regions of the trajectory are very straight, while the CT regions have some varying turn rate, giving more uncertainty. A low turn rate covariance on the CT model was found to give best performance, much due to the last turn in the trajectory. Too high values will blow up the estimated covariance in this turn, making the IMM-PDAF gate too many outlying measurements.

Values and plots of some performance metrics for the final tuning parameters are given in Table 1 and Figure 1. Worth mentioning is that these parameters were tuned to one specific simulation, and may therefore not be robust to other datasets. This may invalidate the use of NEES for filter consistency as we are making the filter consistent on only one simulation, not consistent in general. A better approach in terms of consistency would therefore be to tune over several simulations.

	ANEES ($r_1 = 1.63, r_2 = 2.41$)	RMSE
Position	1.680	3.121
Velocity	2.098	0.708

Table 1: *ANEES and RMSE for position and velocity, with lower and upper confidence bounds r_1 and r_2*



(a) *NEESes with confidence bounds*

(b) *Error of position and velocity for each time-step*

Figure 1: *Performance of IMM-PDAF with final tuning parameters*

2.2 IMM-PDAF on the real radar data set "Joyride"

2.2.1 IMM-PDAF vs single model EKF-PDAF

EKF with only CV or CT was tested for initial tuning and benchmarking with IMM. It was expected that EKF with a single model would perform worse than IMM due to the complex maneuvering that is not

encapsulated by either of the simple, linear models. However, after some tuning, the EKF for both models was, surprisingly, able to estimate the entire trajectory up to an acceptable level of RMSE. However, getting a larger proportion of the estimates inside the confidence interval than $\sim 50\%$ proved challenging within reasonable time. Both phenomena can best be explained by process noise. As the maneuvering of the boat is far from only CV or CT, the actual dynamics of the system are so uncertain that a high process noise component is necessary for the filter to correctly follow the target. This in turn results in NEES getting attenuated by the large covariance matrix $\boldsymbol{P}$.

2.2.2 Initial observations from dataset

To start tuning the Joyride dataset, an evaluation of the raw data was done to understand how the measurements were distributed to try to get a more intuitive understanding of a fitting clutter rate, detection rate, validation gate and measurement noise.

Firstly, the dataset is observed to be very challenging in terms of tracking, as the target is maneuvering very aggressively. This is a big difference from the simulated dataset, which has clear regions of CV and CT. This unpredictability should then be reflected in the choice of process noise, which intuitively should be higher than for the simulated dataset.

The assumption that clutter was uniformly distributed inside the sensor grid turned out to be wrong, as the measurements clearly follow the boat that was tracked, with only some noise appearing towards the end of the dataset in the top part of the grid. However, it was decided that the assumption of uniform distribution was still applicable when only considering gated measurements. Considering the scale of the grid and the number of gated measurements, it was concluded that λ would be a low number, far less than 1.

Most of the measurements were "well-behaved" in the way that gated measurements sufficiently follow the boat. This in turn meant that P_D could start off close to 1 and g^2 not too high.

Lastly, most measurements seemed trustworthy, that is, not clutter, encouraging a low measurement noise. However, it was observed occasional clutter off-track that prevented holding the measurement noise too low, especially in turns where the change of model is critical. Additionally, it was assumed that IMM would depend heavily on the measurements to correctly select correct model for the target, preventing having the measurement noise too high.

2.2.3 Tuning strategy

As discussed in Section 2.2.2, it was assumed that the measurements would be critical to follow the track without divergence, as the dynamics are so unpredictable. Initially, keeping the gate size as small as possible with a reduced detection probability to accommodate occasional lack of measurements was tested. Thus, P_D was set to 0.6 and $g^2 = 2^2$. The idea was to make the filter follow in the footsteps of the measurements. There was some clutter in turns that would throw the tracking off-track that hopefully would not be captured with a small enough gate size. As it was assumed that gated measurements would be correct detections, λ was set as low as $1 \cdot 10^{-10}$. However, it was not possible to find a proper compromise between having a small validation gate and reliably catching measurements when a mode change was necessary. It became evident that the sequence of measurements was not close enough and appearing frequently enough for the filter to follow them, and especially towards the end of the dataset, the track would go off-track and not get on-track again until the validation gate blew up large enough to catch a measurement.

Thus, a different strategy for tuning was required. After some testing, the chosen strategy was to make as good use of the three different models as good as possible. This is part to the fact that the previous method relied too much on the measurements, suppressing the entire filter framework available in IMM. The responsibility of each model was set to be:

CV: Responsible for keeping track of the boat under actual constant velocity. The variance of the process noise was thus set low.

CT: Responsible for tracking the boat in turn. To make it distinguishable from especially the CV model, a very low velocity process noise was set in conjunction with high turn rate noise. In practice, this makes the sigma ellipsis into a long, slim ellipsis that concentrates probability mass to the sides of the boat.

CVh: Responsible for picking up the boat under rapid acceleration. With only one process noise parameter, this was set as large as CT at its widest by eyeing it under simulation.

By setting the process noises different enough, IMM was better able to distinguish them and choose correct model accordingly, yielding better performance.

The gate size was initially increased to $g^2 = 3^2$ to get the filter to catch mode changes more easily. This in turn meant that λ had to be reconsidered, as $\lambda = 1 \cdot 10^{-10}$ was way too small with the increased number of gated measurements. A rough initial guess was derived from Equation (1), where the entire grid is considered, with V scaled accordingly. The grid size was taken to be the area covered by the radar, and rough readings resulted in the estimate $\lambda \approx 3 \cdot 10^{-7}$. However, as commented on in Section 2.2.2, assuming that measurements appear uniformly in the entire grid is an assumption that does not correlate with the actual dataset. Thus, a λ_k was calculated using $\lambda_k \approx \frac{m_k}{V_k}$ for every time step, where m_k is the number of gated measurements in time step k and V_k the volume of the validation gate at time step k, calculated with $V_k = \pi |g^2 \boldsymbol{S}_k|^{1/2}$. λ was thus estimated from $\lambda \approx \frac{1}{K} \sum_{k=1}^{K} \lambda_k$. This yielded the result $\lambda \approx 1.1 \cdot 10^{-5}$. All of this was calculated with a gate size of $g^2 = 3^2$, as this should contain 99.7% of all measurements given the uncertainty in the prior.

Testing with this estimate for λ showed decent performance, but occasional drifting. This is probably due to the filter suspecting too many clutter measurements and instead relying on the current mode. Decreasing λ to $1 \cdot 10^{-6}$ resulted in the best performance, with $1 \cdot 10^{-7}$ and lower causing the filter to behave "jerky", with a spikey track. This is probably a consequence of the filter not recognizing measurements as clutter correctly, resulting in the estimated track to hedge too reckless on gated measurements.

With the increased gate size, it was almost certain that a gated measurement should be from the boat, so P_D was increased to 0.95. This parameter proved satisfying for performance, given that the gate size g was at least 3, and was kept constant for the remaining of the tuning.

The $\boldsymbol{\pi}$ matrix, containing the conditional transition probabilities between modes, was not tuned terribly much. Experience from previous tuning also applied for the Joyride dataset, which meant that setting 0.95 as the probability for staying in a mode, and 0.025 for transitioning to either of the other modes, made the IMM well-behaved. In fact, keeping all other parameters constant, changing the transition probabilities made a negligable impact on the filter's performance. However, this made the modes covariance ellipsises overlap for the entirety of the lap, in addition to the mode probabilities quickly converge to approximately 0.33, in other words, all modes were equally likely. This was especially evident when the probability for remaining in a mode was set to 0.01. On the other side, setting the probability of remaining in a state too high, like 0.999, results in IMM never making a mode change, effectively making the filter in this case a pure CV filter.

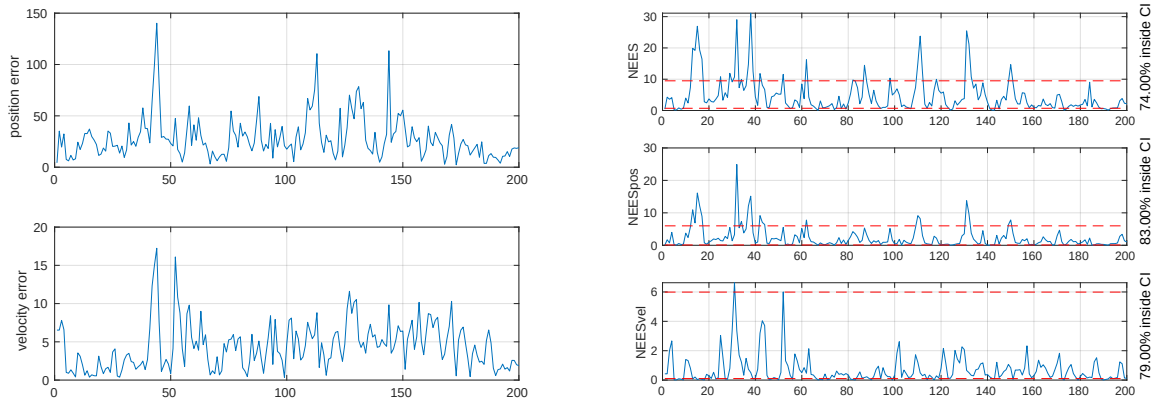
The final parameters can be found in Table 2, while ANEES and RMSE can be found in Table 3. As a note, the ANEES for velocity is below the CI lower bound. This is probably due to the observed behaviour of the IMM promoting the use of the CV and especially CVh, that have a high process noise in velocity. This makes the resulting filter underconfident in its velocity estimate.

Parameter	r	P_D	λ	g^2	q_{CV}	q_{CT}		q_{CVh}
Value	10^2	0.95	$2 \cdot 10^{-5}$	3^2	0.1^2	$\begin{bmatrix}0.0001^2 & 0.3^2\end{bmatrix}$		3^2

Table 2: *Table over final parameters for the IMM-PDAF filter for the Joyride data set.*

	ANEES ($r_1 = 1.73, r_2 = 2.29$)	RMSE
Position	2.3192	34.715
Velocity	0.6996	5.331

Table 3: *ANEES and RMSE for position and velocity, with lower and upper confidence bounds r_1 and r_2*



(a) Euclidean error. *(b) NEES.*

Figure 2: *Performance of the tuned IMM-PDAF for the Joyride data set.*

3 ESKF

3.1 ESKF on simulated data

Similar to the IMM-PDAF in Section 2, the process and measurement noise covariances $\mathbf{Q}_{\text{IMU}}$ and $\mathbf{R}_{\text{GNSS}}$ need to be tuned for the ESKF, now with IMU as control input to the filter. This allows us to describe the process uncertainty by noise parameters on the IMU, and with measurement uncertainty described by noise parameters on the GPS, $\mathbf{Q}_{\text{IMU}}$ and $\mathbf{R}_{\text{GNSS}}$ should be obtainable from the datasheets of the respective sensors. This is not possible for simulated data, but $\mathbf{Q}_{\text{IMU}}$ and $\mathbf{R}_{\text{GNSS}}$ can be initialized to typical values for these sensors as a reasonable first guess. As ground truth is available, $\mathbf{R}_{\text{GNSS}}$ was instead estimated by subtracting the ground truth from the GNSS measurements and calculating the empirical standard deviation of the resulting signal. This gave reasonable results, with equal variance in the plane and higher variance in altitude, as expected for a GNSS. A similar approach was used for tuning the white noise in $\mathbf{Q}_{\text{IMU}}$, where we exploit that the IMU is at rest for the first period of the simulation. This period is however not long enough to give good estimates of the sensor bias, which should be taken over several hours, and was therefore tuned by trial and error using the NEES and estimation errors. Sensor bias tends to be magnitudes lower than white noise, where around a tenth of the white noise standard deviation gave good results. As the ground truth is available, the filter was initialized with ground truth states as predictions and very low covariance to ensure good initial convergence. The bias is modeled as a Gauss-Markov process, with reciprocal time constants p_{acc} and p_{gyro}, but these are set to zero as the length of the dataset is only 15 minutes, giving a pure random walk that should suffice. The final parameters are shown in Table 4, with the ESKF performance shown in Figure 3.

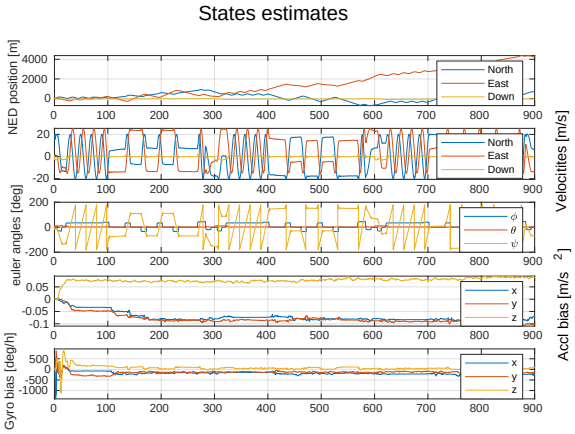
The use of NIS, NEES and RMSE for tuning and validation follows the analysis done in Section 2, although they were not used as actively when arriving at the values in Table 4. This is explained by the discussion above, were parameter values are estimated based on a purely statistical approach instead of the trial and error in Section 2. This isn't guaranteed to give the best results, and the results in Figure 3 are most likely not depicting the optimal solution, but further tuning by trial and error did not show significant increase in performance. Having a sensor give control input to the filter has therefore been shown to ease the tuning process, as the process noise is well explained by the properties of the sensor and not only on the uncertainty in the maneuvers of the object being estimated, which can be difficult to assess. This may be somewhat misleading at this point as the sensor data is simulated, and may therefore give noise that is more consistent with the noise models compared to an actual sensor.

The RMSE on state estimates are all low, as seen from Figure 3b, with estimated heading as poorest estimate. This doesn't come as a surprise, as heading can only be estimated by excitation in the yaw-axis of the gyroscope, whereas roll and pitch are decomposed by gravity and directly estimated by the accelerometer. The estimate of heading will therefore deviate when the vehicle is following a horizontal line or lying still,

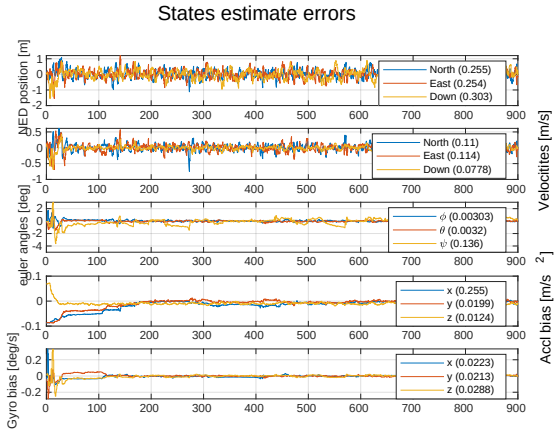
as the state becomes unobservable, and is subsequently a weakness of this filter. A compass or other sensors for estimating heading may therefore provide better estimates of heading for the whole trajectory.

Parameter	$q_{acc,n}$	$q_{acc,b}$	$q_{gyro,n}$	$q_{gyro,b}$	r_x	r_y	r_z	p_{acc}	p_{gyro}
Value	0.02^2	0.002^2	0.0008^2	0.00008^2	0.3^2	0.3^2	0.5^2	0	0

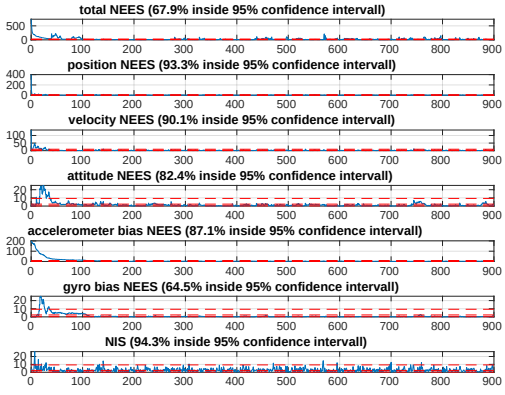
Table 4: *Table over final parameters for the ESKF on simulated data*



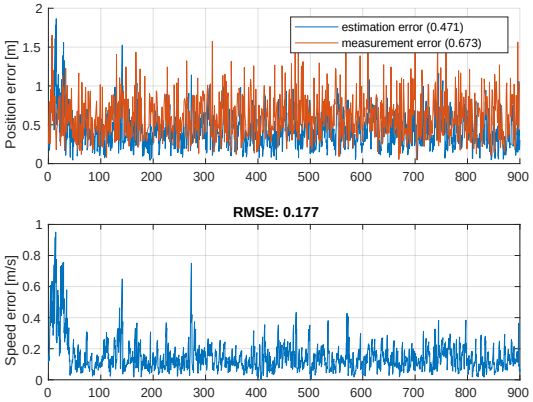
(a) *State estimates over time*



(b) *State estimate errors over time*



(c) *Filter consistency*



(d) *Position and velocity error with RMSE*

Figure 3: *Performance of the ESKF on simulated data with final tuning parameters*

3.2 ESKF on real data

3.2.1 Initial assessment

An initial test of the filter performance with the parameters from Section 3.1 was done. However, this resulted in suboptimal results, and a new investigation of initial parameters was done. With the sensors used now given, an initial guess on fitting noise parameters was derived from data in the respective datasheets. The

velocity accuracy for the used GNSS sensor is given as 0.05 m/s[3], so an initital guess for position noise was set to $r = \frac{0.05 \text{ m/s}}{1 \text{ Hz}} = 0.05$ m where 1 Hz is the sampling frequency.

For the IMU, the white noise parameters was found to be 0.06 m/s/$\sqrt{\text{hr}}$ for acceleration and 0.15 °/$\sqrt{\text{hr}}$ for angular velocity[2]. The following conversions was done to get the deviation in discrete time with correct unit:

$$\begin{aligned} \sigma_{\text{gyro}} &= 0.15 \text{ °}/\sqrt{\text{hr}} \\ &= 0.15 \text{ °}/\sqrt{\text{hr}} \cdot \frac{\pi \text{ rad}}{180°} \cdot \sqrt{\frac{1 \text{ hr}}{3600 \text{ s}}} \cdot \sqrt{250 \frac{1}{\text{s}}} \\ &\approx 6.90 \cdot 10^{-4} \text{ rad/s} \end{aligned} \qquad \begin{aligned} \sigma_{\text{acc}} &= 0.06 \text{ m/s}/\sqrt{\text{hr}} \\ &= 0.06 \text{ m/s}/\sqrt{\text{hr}} \cdot \sqrt{\frac{1 \text{ hr}}{3600 \text{ s}}} \cdot \sqrt{250 \frac{1}{\text{s}}} \\ &\approx 15.81 \cdot 10^{-3} \text{ m/s}^2 \end{aligned} \tag{2}$$

No numeric value for the bias variance is given in the datasheet for the STIM300. Figure 6-5 and Figure 6-13 in [2] give plots of the Allan variance for the gyroscope and accelerometer. The bias variance can be read from this plot by the intersection between the tangent to the log curve where the slope is 1/2 with where $\tau = 3$ s[1]. However, due to the short time scale of only 10^4 s ≈ 2.7 hours, this can unfortunately not be done, and was instead estimated from the white noise variance as a thousandth of its magnitude. It was initially assumed that bias was negligible over the time period of the time set, especially considering the quality of the STIM300, so the reciprocal time constant for both Gauss-Markov processes was set to 1 to suppress development of bias.

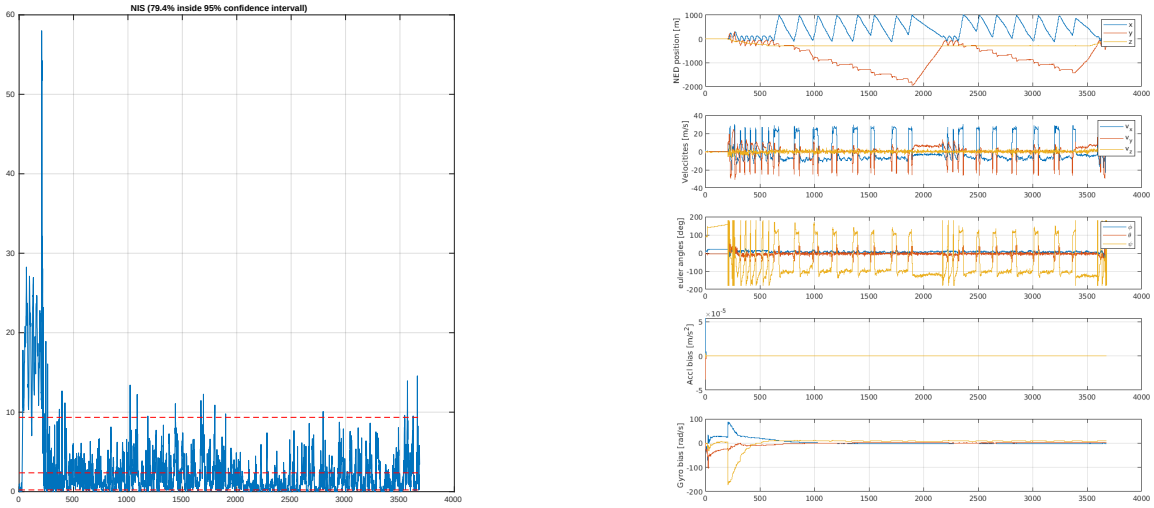
3.2.2 Tuning strategy

With the initial parameter values from Section 3.2.1, the filter had a decent performance. However, the the x- and y-coordinate of the pose drifted constantly, only to be drawn in to the correct pose by the regular GNSS update. The z-coordinate, however, of the pose was behaving reasonably smooth. In addition, roll and pitch were well-behaved, with yaw acting constantly up, either skipping between a constant $\pm 100°$, or looping from $-180°$ to $180°$ (in other words, the vehicle was spinning about its z-axis).

After investigating these results, complemented by the results from the simulated data, the following hypothesis was derived: roll and pitch are well-behaved due to the full observability of these axes from the gravity vector. This makes the ESKF much more capable of correctly estimating pitch and roll, resulting in their smooth behaviour. This propagates to good estimates of the z-coordinate, as the orientation of the z-axis is fully determined by pitch and roll, meaning the translation from body frame to world frame does not induce errors. This, however, is not the case for yaw. It is only observable by integrating angular velocity from the gyroscope, meaning yaw can only be estimated by exciting yaw motion. This makes the estimation much more sensitive to offsets and biases, as the error in heading accumulates with time because of the involved integration without proper bias estimation. This is probably what causes the vehicle to believe it is spinning. Additionally, this explains the observed drifts in the x- and y-directions: as the robot believes its heading is different from what it really is, the orientation of body frame is not aligned with the trajectory, hence the robot is prone to believe that it translates in a different direction than it actually is. This is only remedied by the periodic GNSS updates, which explains the saw-tooth looking estimated trajectory.

With this knowledge in hand, the following steps were taken: Obviously the system carried more bias, especially in gyro, than initially assumed. Thus, the reciprocal time constant was drastically reduced to make the ESKF more inclined to remove bias. However, making the estimation too aggressive would make it overestimate the bias, defeating the purpose. Still, the reciprocal time constant functions as an upper bound for the estimated bias, so for the final tuning, setting it as large as possible is preferred. In addition, the measurement noise was slightly inflated to make the ESKF weigh the integrated gyro update less in the update step. This was intended to make x and y drift less between updates from GNSS.

These changes resulted in a much smoother behaviour. The saw-tooth looking behaviour of the x- and y-coordinate in the plot shown in Figure 4b is explained by the shape of the trajectory, as the robot is zig-zagging back and forth in the x-direction before return to the origin, twice. The y-coordinate increases linearly when the robot turns, and remains stationary while making the stint where the x-coordinate changes. This causes the staircase looking plot. Acceleration bias was never a problem and was kept close to zero as it gave the best performance.



(a) Final NIS. *(b) Plot over final state estimates.*

Figure 4: *Final results for ESKF on real data.*

In Figure 4a, a large spike can be seen in the beginning of the dataset, before settling down. It can be seen in the start that the robot is rocked before being put to rest. Additionally, the NIS settles down when the robot starts to move. One possible explanation is that as the robot is at rest, a model mismatch occurs at the process model is engineered for a robot in motion. Additionally, considering the discussion above regarding observability, the system is not able to reliably estimate its state, as it "assumes" motion. Thus, small errors in linear and angular velocity makes the filter naively believe its moving, only to be corrected by the periodic GNSS measurements with its low measurement noise.

The NIS at $\sim$40% could also be improved. The data from `GNSSaccuracy` was studied in hope for a better estimate. Using this data directly would be wrong, as this describes the accuracy of the GNSS, not the precision. However, the standard deviation, calculated with `std(GNSSaccuracy)`, should be a good estimate. This revealed that the given GNSS module had closer to 0.5 m in standard deviation, ten times higher than the value predicted from the datasheet. Multiplying $\mathbf{R}_{\text{GNSS}}$ by 10 improved the NIS to almost 80%.

Parameter	r_x	r_y	r_z	q_{acc}	$q_{\text{acc, bias}}$	p_{acc}	q_{gyro}	$q_{\text{gyro, bias}}$	p_{gyro}
Value	0.5^2	0.5^2	0.5^2	0.0158^2	$(0.0158 \cdot 10^{-3})^2$	1	$(6.899 \cdot 10^{-3})^2$	$(6.899 \cdot 10^{-6})^2$	$4.0 \cdot 10^{-3}$

Table 5: *Table over final parameters for the ESKF on real data.*

3.2.3 Setting S_a and S_g to the identity matrix

A weakness of inertial navigation is the need to account for mounting, scaling and alignment errors, here between the IMU and body frame of the vehicle. This is artificial on simulated data, but is investigated to understand the effect of these errors. For the simulated data, looking at the state estimates shows practically no difference in the estimated state. However, the state estimate errors reveal a saw-tooth looking yaw error, in addition to an increased bias. Neither phenomena are no where to be seen with correct alignment. However, the biggest difference is possibly the large difference in NEES, where the total NEES falls from 67.9% inside the CI to 0%, while attitude falls from 82.4% to 3.99%. Both NEES curves indicate an overconfident filter. The best explanation for this is the fact that the aligning matrices S_a and S_b "presents" additional information for the filter, in the form of scale, orthogonality between sensor axes and orientation

between IMU frame and body frame of the robot. The scale factor is additional information that the filter would not be aware of otherwise. Rotation and orthogonality errors causes correlation between the different sensor readings, as, for example, gravity "contaminates" the readings of the other axes, without the filter being aware of this being gravity. To correctly estimate theses errors, the filter needs more information, but as it is not aware of this, this causes it to become overconfident.

For the real data, at first glance, the trajectory and state estimates seems fine, however, closer inspection and comparison with plots with correct S_a and S_b reveals that the heading ψ is off by 180°, as the heading has the opposite sign as the correct estimation. Additionally, the NIS is reduced from almost 80% to just over 30%. This tells us that detecting an IMU that is mounted incorrectly is hard to detect just based on the data.

In conclusion, the state estimates that come out of the filter seems reasonable even when the IMU is misaligned. However, a consistency analysis reveals the filter to be overconfident. This goes to show that correctly aligning the IMU is important for correct, consistent state estimation.

4 EKF SLAM

4.1 EKF SLAM on simulated data

EKF SLAM follows a similar prediction-update structure as IMM-PDAF and ESKF, and will thus also have process noise $\boldsymbol{Q}$ and measurement noise $\boldsymbol{R}$ as tuning parameters. Data association is similarly to IMM-PDAF a problem for SLAM, and is here handled by the Joint Compatibility Branch and Bound (JCBB) voting scheme, where we must set the boundaries for joint and individual compatibility between measurements and landmarks, α_{joint} and $\alpha_{\text{individual}}$.

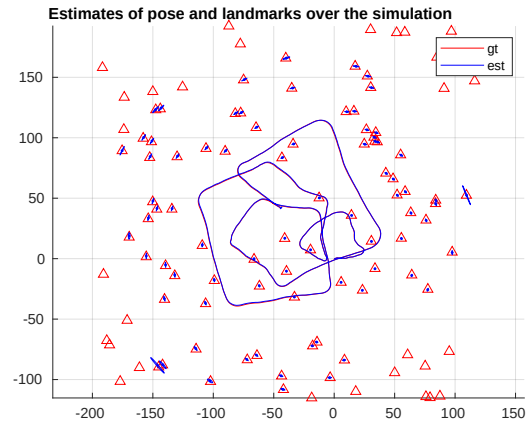
As there is no information about the simulated sensors, $\boldsymbol{Q}$ and $\boldsymbol{R}$ must be initialized in a similar manner as in Section 2. It is difficult to reason on an initial guess for $\boldsymbol{R}$ based on the data, as the robot may or may not receive measurements from different landmarks for each time-step, so calculating any representative statistic is not straightforward. $\boldsymbol{Q}$ can be reasoned on by looking at the true trajectory and odometry of the robot, chosen to encapsulate the strongest maneuvers observed. Typical for odometry-sensors is lower variance perpendicular to the direction of travel, e.g for wheel encoders, although this may not be the case for the simulation. The boundaries for joint and individual compatibility were initially set low to allow faster computation, but will be reasoned on later.

$\boldsymbol{Q}$ and $\boldsymbol{R}$ were further tuned by trial and error by visually comparing true trajectory and landmarks to estimates, until the estimates were decent. This way of tuning should give a lower RMSE, but has no guarantee in terms of filter consistency. NIS and NEES were further calculated to account for this, where the chi-square bounds for NIS were calculated for every time-step to compensate for the fact that the degrees of freedom vary with the cardinality of the measurements. NEES was only calculated for pose, and not landmarks as it was difficult to match estimated landmarks to true landmarks, so we have no guarantee for consistency in mapping for the tuned filter.

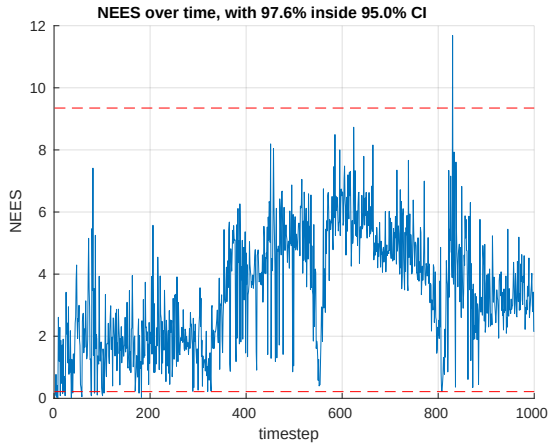
$\alpha_{\text{individual}}$ determines the individual compatibility of a landmark to each measurement. In practice, this corresponds to setting a gate size that excludes all landmarks that are not close enough. Based on experience from previous tuning, a gating size of 3 was chosen, equivalent to a 3σ bound that should in theory contain 99.7% of the measurements. As JCBB accepts a probability, this is in MATLAB computed with `1-chi2cdf(3^2, 2)`, where 2 comes from 2 degrees of freedom as the measurement is 2D (range and bearing). α_{joint} governs how well the joint probability of an association hypothesis should overlap with the landmarks it tries to match with, and can thus too be regarded as a gating procedure, only that the pdf now is multimodal. α_{joint}s from close to 1 to close to 0 was tested, and it was observed that increasing α_{joint} too high, especially between 0.9 and 1, EKFSLAM started running slow. This is probably due to the fact that JCBB struggles to make associations between new measurements and existing landmarks as it is too restricted by the tight gating test, thus bloating the state space with duplicate landmarks. Towards the end of the simulation, this makes especially the covariance matrix $\boldsymbol{P}$ costly to calculate. Ultimately, it was decided that $\alpha_{\text{joint}} = 0.05$ was a fitting value that made good associations while being computational feasible.

Parameter	q_x	q_y	q_ψ	r_r	r_θ	$\alpha_{\text{individual}}$	α_{joint}
Value	0.5^2	0.5^2	0.0524^2	0.05^2	0.035^2	0.0111	0.05

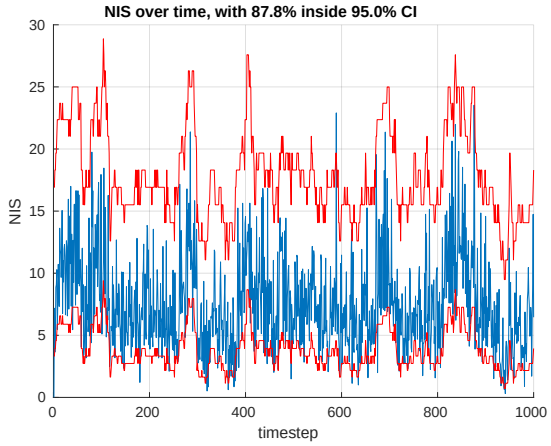
Table 6: *Table over final parameters for EKF SLAM on simulated data*



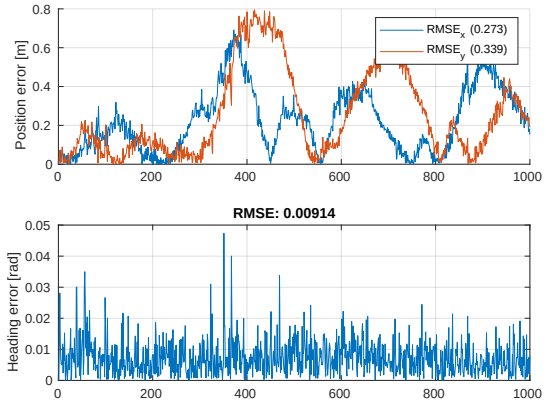
(a) State estimates for the whole duration



(b) NEES with confidence bounds



(c) NIS with confidence bounds

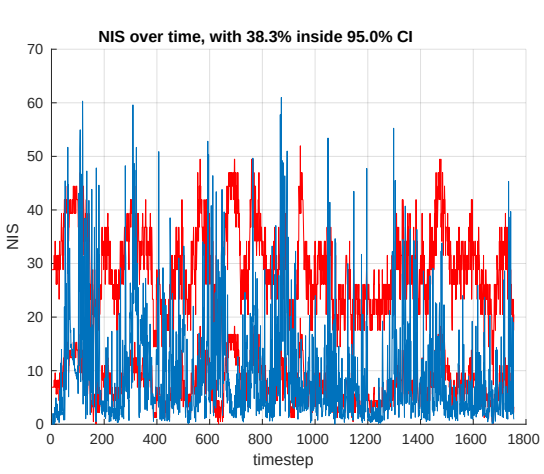


(d) Position and heading error with RMSE

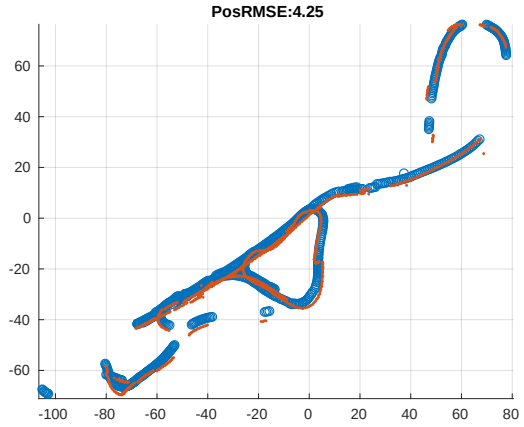
Figure 5: *Performance of EKF SLAM with final tuning parameters*

4.2 EKF SLAM on the Victoria Park dataset.

Initial tuning started with the same procedure as in Section 4.1. It was quickly realized that the magnitude of the parameter values had a large impact on the computational performance of the EKFSLAM, best explained by the much larger number of potential landmarks and associations in the Victoria Park dataset compared with the simulated. When $\boldsymbol{Q}$ and $\boldsymbol{R}$ both had a low magnitude, the simulation ran significantly slower. This is best explained by the fact that when both $\boldsymbol{R}$ and $\boldsymbol{Q}$ get too small, $\boldsymbol{S}$ also gets small. This concentrates probability mass towards the estimated measurement, which in turn shortens the confidence interval governed by $\alpha_{\text{individual}}$. Thus, JCBB is more inclined to assume a measurement is a new landmark, when in practice this is incorrect. With $\boldsymbol{R}$ =`diag([0.01, deg2rad(2)].^2)` and `sigmas=[0.05, 0.05, deg2rad(1)]`, $\boldsymbol{P}$ resulted in being 2057×2057, while changing $\boldsymbol{R}$ to `diag([0.1, deg2rad(2)].^2)` reduced $\boldsymbol{P}$ to 781×781.



(a) NIS.



(b) Map of trajectory on top of transformed GPS points with posrmse.

Figure 6: *Results of tuning of EKF SLAM to the Victoria Park dataset.*

The initial heading was estimated by looking at the first few data points from the GPS and fitting a line indicating the direction of motion. From this, 35.5° was used.

Tuning of NIS was done by considering alignment with GPS points together with having as high consistency as possible. Increasing the process and measurement noise yielded a trajectory that overlapped better with the GPS points, but decreased the NIS as the filter got too underconfident. Initially, the same $\alpha_{\text{individual}}$ and α_{joint} as in Section 4.1 yielded overlapping landmarks, and so it was tried to reduce these to lighten the computational load. This, however, reduced NIS quite drastically, and so increasing $\boldsymbol{R}$ was tried to remedy it. Counter-intuitvely, this in turn significantly increased run time. The best explanation for this is the fact that the recursive call of the JCBB down the interpretation tree only terminates if the algorithm recognizes that the remaining path does not improve the currently best association hypothesis or that joining a new association with the current hypothesis fails. Thus, with too large covariance on the innovations, bounding the solution of a certain branch fails, and the problem degenerates to an exhaustive search that grows exponentially with the number of associations in complexity.

These experiences lead to a suboptimal tuning as a compromise between filter consistency and computational feasibility. The results can be found in Table 7 and Figure 6.

It was attempted to calculate NEES with GPS as ground truth for additional information during tuning. However, as the calculated values were outside a useful scale at values ranging from 10^5 to 10^6, the attempt was aborted. The best explanation for this is that the GPS measurement is too varying and unpredictable to reliably be used as ground truth, and the inherent uncertainties it holds biased the calculated NEES to such a degree that it was not possible to draw any sensible conclusions from it.

Parameter	r_r	r_ϕ	q_x	q_y	q_ψ	$\alpha_{\text{individual}}$	α_{joint}
Value	$1 \cdot 10^{-2}$	$1.1 \cdot 10^{-2}$	$4.2 \cdot 10^{-2}$	$4.2 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-3}$

Table 7: *Table over final parameters for the EKF SLAM on the Victoria Park dataset.*

5 Conclusion

This report has considered a practical implementation and tuning of IMM-PDAF, ESKF and EKF SLAM, both for simulated data and real data. The purpose of the IMM-PDAF was to track a target which dynamics are not explained in a single, simple model. Detection probability, clutter rate, gate size and process and

measurement noise were key parameters to tune. By looking at the dataset to tune for, the detection probability and gate size was determined by looking at how measurements appear along the track and with what rate. The process and measurement noise play a key role in the fact that they help the IMM framework distinguish the different models by distributing the probability mass differently around the estimate, such that the weight for a given association in the Gaussian mixture that remains promotes the correct model.

The ESKF is used in an inertial navigation system that fusions interoceptive GNSS data with IMU data. The main advantage of this filter over the vanilla EKF is the fact that attitude estimation including composition of rotations can be done easily and free of singularities with a quaternion representation. This is all the while keeping the covariance matrix between state updates well-defined as the nominal state still only has three degrees of freedom from the three Euler angles. Tuning of the ESKF reduces to tuning of parameters that characterize an IMU, here white noise and bias of both the gyroscope and accelerometer, that can be considered independent sensors. Observability of the different body axes, in particular yaw, was discussed for both datasets as the trickiest to estimate, as it is only observable by integrating yaw rate measurements from the gyroscope.

EKF SLAM is a recursive SLAM method that involves data association of detected landmarks, per the fundamental SLAM problem it tries to solve. As EKF SLAM inherently suffers from divergence in the long-term, getting optimal results from the tuning proved a challenging, and was a not possible for the Victoria Park dataset. In addition, the actual implementation of how to compute the EKF update step is of special importance as the state space grows with time as landmarks are saved. As the covariance matrix grows quadratically with the states, computing the updated matrix correctly was different between running a simulation in a matter of seconds or minutes. The particular implementation of EKF SLAM used in this report uses JCBB for data association, and proper tuning of parameters for individual and joint compatability was here too the difference between an algorithm that is computational feasible or not. The parameters $\alpha_{\text{individual}}$ and α_{joint} showed a big influence on both how large the state space ended up being and how long it took for the JCBB to settle at an association hypothesis each time step.

References

[1] MathWorks. *Inertial Sensor Noise Analysis Using Allan Variance*. `https://se.mathworks.com/help/nav/ug/inertial-sensor-noise-analysis-using-allan-variance.html`. Accessed 22/11/2019.

[2] Sensonor. *STIM300*. `https://www.sensonor.com/media/1132/ts1524r9-datasheet-stim300.pdf`. Accessed 22/11/2019. 2013.

[3] u-blox. *u-blox 8 GNSS modules*. `https://www.u-blox.com/sites/default/files/MAX-8_DataSheet_%28UBX-16000093%29.pdf`. Accessed 22/11/2019. 2019.