IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of current version June 25, 2020.

Digital Object Identifier PLACEHOLDER

EEET 4075 Mechatronic System Design 2 - Project Report

MICHAEL J. DUKE[1]
[1]University of South Australia, Mawson Lakes, SA 5095 Australia (e-mail: dukmj002@mymail.unisa.edu.au)

This work was supported in part by the University of South Australia

ABSTRACT To determine an absolute position and orientation of a robot is not a simple task. Every sensor is subject to noise and errors of some sort, even systems such as the Global Navigation System used in many systems today is subject to large amount of noise and error, making their direct usage less useful. To overcome this, this report demonstrates a method which combines an instantaneous, fast updating, inertia and wheel encoder based estimate, with a slower, delayed beacon based reference position. These estimates are fused using an Extended Kalman Filter (EKF) which is modified to account for the delayed, multirate, multisensor system that is used by incorporating Alexander's Method. The results showed that while the modified EKF worked quite well in and of itself, despite a relatively low update rate, systemic errors introduced in the absolute position reference caused a fair amount of error while traveling at higher velocities.

INDEX TERMS Dead reckoning, Distance measurement, Indoor navigation, Kalman filters, Nonlinear filters, Robot motion.

I. INTRODUCTION

THE ability to determine a robot's position and orientation is an important requirement for a lot of mobile robotic systems. Without it, their capabilities are quite limited. To measure these directly however, slow, and often delayed measurements need to be taken, assuming they can be taken at all. Another method is to use faster, instantaneous dead reckoning techniques to estimate the position and orientation. Except all dead reckoning techniques inherently have accumulation error so the problem presents itself, How does one use the two together to get the best of both worlds?

This report uses a combination of methods described in the literature review below to achieve the following, to estimate an absolute pose using a beacon based navigation system, to estimate the relative pose using both an Inertial Measurement Unit (IMU) and wheel encoders, and to fuse these measurements in a way that offers the best compromise between accuracy, and efficiency using an Extended Kalman Filter (EKF) or some derivative of.

II. RELATED WORKS

Put broadly, localisation can be split into two categories; absolute, which measures the pose of a system directly; and relative, which uses a dead reckoning technique to estimate the pose of a system using other information. [1] investigates methods for both. They use inertial based localisation through the use an Inertial Measurement Unit (IMU) and wheel encoders updated at a high rate to estimate the pose of the robot in question. They then use a LiDAR image at specific places on its path and match that to an already known occupancy map to adjust said pose. Both of these estimates are then fused through the use of an EKF to give a final estimate.

Unfortunately, occupancy maps are beyond the scope of this experiment, but [2] describes a method for absolute pose estimation using triangulation from known landmarks. While their method uses particle filtering to calculate the estimated pose, the specifics of how the LiDAR is used in triangulation can be used in any filtering technique, including the EKF.

For the standard EKF to work effectively, a few assumptions are made; both the process and measurement noise are additive, and have a Gaussian distribution; the system is not heavily non-linear; and that measurements in y_k arrive at the same time, and pertain to the current state x_k [3]. The first two assumptions can really only be rectified by changing the system, usually unfeasible, or by using a different filtering algorithm altogether, such as the Unscented Kalman Filter or Particle Filter. However, both of these increase the computational complexity of the filtering.

The last assumption however, can be solved while still

using the EKF effectively. With the use of a multirate, multisensor system such as the one used within, [4] includes a number of methods with which to account for this delayed measurement. The most straightforward way to accomplish this is to simply go back to the state in which the delayed measurement pertains to and recalculate all states from that point till the current state. This is quite resource intensive, requiring all states and variables to be held in memory, and calculated twice. To overcome this, [5] proposed just using the linearized portions of the EKF, the Kalman Gain and Prediction Covariance matrices, to fuse the current (primary) measurement and the delayed (secondary) measurement. The advantages of this system is that it is very efficient, both in memory and computational complexity, nor does it introduce and systematic error to the system. The drawback to this solution is that the states between each secondary measurement are not optimal. A solution does exist to solve this optimality issue, which runs the recalculation part of [5]'s method in parallel to the primary EKF, and uses the parellel filter to reinitialize the primary filter. [4] shows that this method is as computationally expensive as just recalculating all states.

Being that the methods described pertain to fusing two sensors, [6] states that multiple EKFs can simply be cascaded, with the output of one EKF being used as the input to another. In this way, any number of measurements with differing update rates and delays can be fused at the cost of computational complexity. As keeping computational complexity to a minimum was a goal of this experiment, using Alexander's Method to fuse the delayed measurement with a standard EKF was the chosen method. While not optimal, it did present the best compromise between accuracy and complexity of the solutions presented. In a system where a higher update rate is likely to have more of an impact to system stability and accuracy than a more optimal filtering technique, it does appear the best choice.

III. METHODOLOGY

A. SYSTEM MODEL

As with most things in control theory, the first step is to determine the state transition model of the system. In the case of the TurtleBot, a lot of the lower level control over the robot is controlled directly by the servomotors themselves. The only variables that are able to be controlled are linear velocity in the x-direction of the bot's local frame (v), and angular velocity around the z-axis of the bot's local frame (ω). As such, the model in 1 was developed, where T is the sample period. This model uses a few assumptions and holonomic constraints to keep the model simple.

- The robot cannot move sideways or vertically.
- The wheels will never slip.
- The effects of inertia are negligible.

$$\boldsymbol{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + Tv\cos(\theta_{k-1}) \\ y_{k-1} + Tv\sin(\theta_{k-1}) \\ \theta_{k-1} + T\omega \end{bmatrix} \tag{1}$$

With the non-linear system model created, the Jacobian in 2 was calculated to be used for gain and error calculation by the EKF.

$$\boldsymbol{F}_k = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x_{k-1}}} = \begin{bmatrix} 1 & 0 & -Tv\sin(\theta_{k-1}) \\ 0 & 1 & Tv\cos(\theta_{k-1}) \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

B. DEAD RECKONING

With the state transition model taken care of, the observation model can be determined. In the case of this EKF, there is two observation models which will be referred as the primary observation model, which includes all of the faster dead reckoning measurements, and the secondary observation model, which includes the slower, absolute measurements.

The primary observation model in this implementation includes information taken from the *joint_states* topic and the *imu* topic. The *joint_states* topic has information regarding the position, velocity, and torque of each wheel. For the observation model, only the wheel positions (s_l and s_r) are used. The *imu* topic has information regarding the linear acceleration, angular velocity, and magnetic field orientation of the bot. For the observation model, only the angular velocity around the z axis was used, while acceleration was considered initially, due to the noise, and need for double integration for it to be used, it was not used. for s_l and s_r to be usable, a small amount of processing is used to convert the information to the change in distance traveled (δs_k) and the change of yaw $(\delta\theta_k)$, this is reflected in 3 and 4 and visualized in figure 1.

$$\delta s_k = \frac{(s_{r,k} - s_{r,k-1} + s_{l,k} - s_{l,k-1})\, r}{2} \tag{3}$$

$$\delta\theta_k = \frac{(s_{r,k} - s_{r,k-1} + s_{l,k} - s_{l,k-1})\, r}{b} \tag{4}$$

Where b is the wheelbase and r is the radius of the wheels, both in meters.

$$b = 0.287 \tag{5}$$

$$r = 0.033 \tag{6}$$

The angular velocity measurement from the *imu* topic (ω_k) can be used directly. Similar to the state transition model, both the non-linear model and the Jacobian of that model is needed for the EKF, which are presented in 7 and 8 respectively.

$$\boldsymbol{h}_k = \begin{bmatrix} \delta s_k \\ \delta\theta_k \\ \omega_k \end{bmatrix} = \begin{bmatrix} \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \\ \theta_k - \theta_{k-1} \\ \frac{\theta_k - \theta_{k-1}}{T} \end{bmatrix} \tag{7}$$

$$\boldsymbol{H}_k^\top = \begin{bmatrix} \frac{x_k - x_{k-1}}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}} & 0 & 0 \\ \frac{y_k - y_{k-1}}{\sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}} & 0 & 0 \\ 0 & 1 & \frac{1}{T} \end{bmatrix} \tag{8}$$

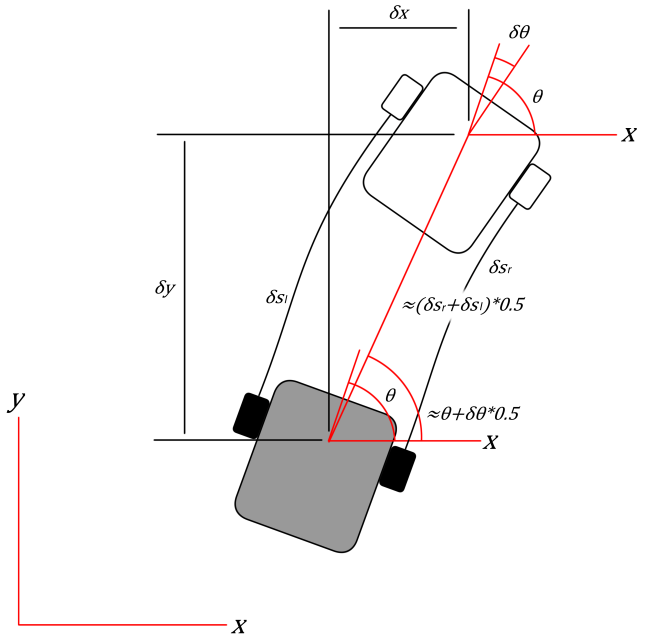


FIGURE 1: Geometric estimation of $\delta\theta$, δx, and δy based off δs_r and δs_l.

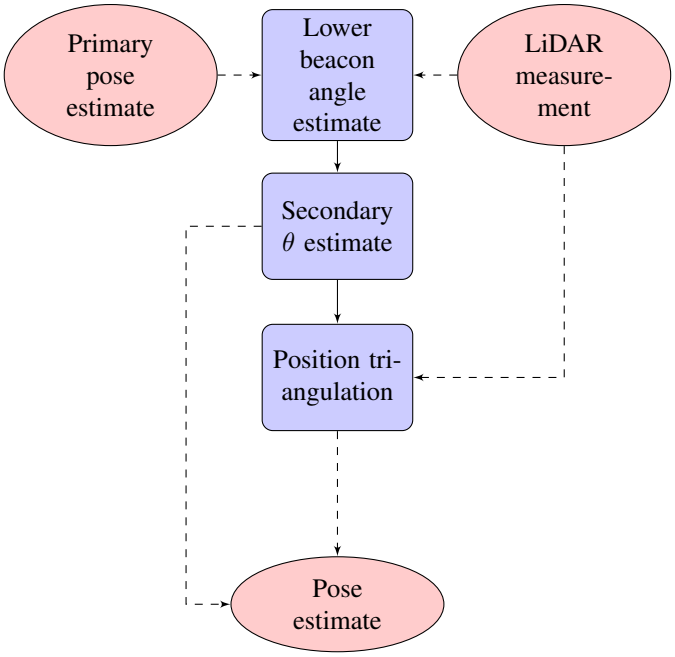


FIGURE 2: Secondary pose estimate procedure

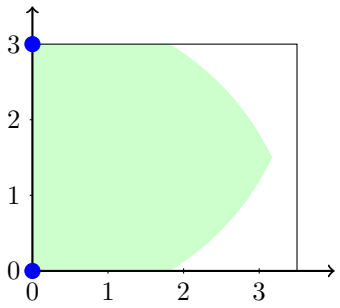


FIGURE 3: Left beacons showing measurable locations in green.

C. ABSOLUTE

The secondary measurement was implemented as a beacon based navigation system with a method similar to [2]. The main difference being that the number of beacons being measured is limited to only two. This is done to reduce the computation time at the expense of accuracy. The method used also does not allow for a definite single estimation position, but a pair of them. With the appropriate assumptions and control measures in place though, one of the positions can always be ruled out.

The simplified procedure for pose estimation using the LiDAR is demonstrated in figure 2 and first involves taking the pose estimate from the primary measurements to estimate where one of the beacons should be. This is done using 11, however, as the center of the LiDAR is offset from the estimate position by 0.064 m, the corrections in 9 and 10 need to be used first.

$$\hat{x} = \hat{x} - 0.064 \sin(\theta) \tag{9}$$

$$\hat{y} = \hat{y} - 0.064 \cos(\theta) \tag{10}$$

$$\hat{\theta}_{b1} = \text{atan2}\left((y_1 - \hat{y}), (x_1 - \hat{x})\right) \tag{11}$$

The choice of beacon is arbitrary, as long as the beacon is within the LiDAR maximum measurement range of 3.5 m. Being that the pose of the bot has been estimated previously, that can be used to determine which beacons can be used for triangulation. Figure 3 shows the locations in which two beacons on the left side of a 3.5 x 3 m field can be measured from. It shows that having a beacon in each corner means being able to cover the entire field. As such the beacons to be used can be calculated ahead of time simply by knowing which half of the field the bot thinks it is in. Assuming that the LiDAR is always in range of two beacons, and that the bot is restricted to being on only one side of each pair, it is possible to expand this method almost indefinitely to cover any space.

To estimate the position of the TurtleBot, the intersection of two circles is calculated using knowing their center coordinates and radii.

$$\begin{aligned}(x - x_1)^2 + (y - y_1)^2 &= r_1^2 \\ (x - x_2)^2 + (y - y_2)^2 &= r_2^2\end{aligned} \tag{12}$$

Rearranged, and with the assumption that $x_1 = x_2$ in both cases we get 13 through 16.

$$y = -\frac{r_1^2 - r_2^2 - y_1^2 + y_2^2}{2(y_1 - y_2)} \tag{13}$$

$$x = \pm \frac{\sqrt{\beta_1 \beta_2}}{2\left(y_1 - y_2\right)} \tag{14}$$

$$\beta_1 = \left(r_1 + r_2 + y_1 - y_1\right)\left(r_1 + r_2 - y_1 + y_1\right) \tag{15}$$

$$\beta_2 = \left(r_1 - r_2 + y_1 - y_1\right)\left(-r_1 + r_2 + y_1 - y_1\right) \tag{16}$$

If it is assumed that the bot cannot escape from the field at any point, one of the possible x positions can always be eliminated, leaving 17 if using the left beacons, and 18 if using the right beacons.

$$x = x_1 + \frac{\sqrt{\beta_1 \beta_2}}{2\left(y_1 - y_2\right)} \tag{17}$$

$$x = x_1 - \frac{\sqrt{\beta_1 \beta_2}}{2\left(y_1 - y_2\right)} \tag{18}$$

Finally, the observation model can be derived quite simply, being that the measurements are estimating the states directly, giving 19 and its Jacobian 20.

$$\boldsymbol{h}_k = \begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\theta}_k \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{19}$$

$$\boldsymbol{H}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

D. EKF

For the actual EKF, it was split into two parts, the first is a standard EKF using only the primary measurements as its inputs, and the second part is the fusing of the delayed secondary measurements with the primary using Alexander's Method as described above.

For the primary measurement, first the priori state and error covariance estimate are calculated.

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{f}_{k-1}\left(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}\right) \tag{21}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1}^+\boldsymbol{F}_{k-1}^\top + \boldsymbol{Q}_{k-1} \tag{22}$$

Then the Kalman gain is update.

$$\boldsymbol{K}_{1,\,k} = \boldsymbol{P}_k^- \boldsymbol{H}_{1,\,k}^\top \left(\boldsymbol{H}_{1,\,k}\boldsymbol{P}1,\, k^- \boldsymbol{H}_{1,\,k}^\top + \boldsymbol{R}_{1,\,k}\right)^{-1} \tag{23}$$

Finally, the posterior state and error covariance estimates are calculated.

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_{1,\,k}\left(\boldsymbol{y}_{1,\,k} - \boldsymbol{h}_{1,\,k}\right) \tag{24}$$

$$\boldsymbol{P}_{1,\,k}^+ = \left(\boldsymbol{I} - \boldsymbol{K}_{1,\,k}\boldsymbol{H}_{1,\,k}\right)\boldsymbol{P}_{1,\,k}^- \tag{25}$$

When the secondary measurement is received, the Kalman gain, and posterior error covariance are updated.

$$\boldsymbol{K}_{2,\,k} = \boldsymbol{P}_{1,\,k}^- \boldsymbol{H}_{2,\,k}^\top \left(\boldsymbol{H}_{2,\,k}\boldsymbol{P}_{1,\,k}^- \boldsymbol{H}_{2,\,k}^\top + \boldsymbol{R}_{2,\,k}\right)^{-1} \tag{26}$$

$$\boldsymbol{P}_{2,\,k}^+ = \left(\boldsymbol{I} - \boldsymbol{K}_{2,\,k}\boldsymbol{H}_{2,\,k}\right)\boldsymbol{P}_{1,\,k}^- \tag{27}$$

Finally the posterior state estimate is recalculated. As it's known that the secondary measurement will be delayed by $s + N$ samples, where s is the sample with which the measurement refers to, then the recalculation of the fused posterior state estimate adds one more term, $\boldsymbol{W}$. to account for the delay.

$$\begin{aligned} \hat{\boldsymbol{x}}_k^+ =& \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_{1,\,k}\left(\boldsymbol{y}_{1,\,k} - \boldsymbol{h}_{1,\,k}\right) \\ &+ \boldsymbol{W}\boldsymbol{K}_{2,\,s}\left(\boldsymbol{y}_{2,\,k} - \boldsymbol{H}_{2,\,s}\hat{\boldsymbol{x}}_s^-\right) \end{aligned} \tag{28}$$

where

$$\boldsymbol{W} = \prod_{i=1}^{i=N}\left(\boldsymbol{I} - \boldsymbol{K}_{s+i}\boldsymbol{H}_{1,\,s+1}\right)\boldsymbol{F}_{s+i-1} \tag{29}$$

IV. RESULTS AND DISCUSSION

Before proper testing could commence, covariances for the simulation had to be calculated from error measurements. This was done by measuring the error from the odometry topic, which in the simulation is the ground truth, to the various sensors over 400 samples while the bot traversed an arbitrary path. It should be noted that for the sensors that involve dead reckoning (Model, Joint, and IMU), the covariance in table 1 is calculated on the assumption that every measurement was based off an absolute reference. As dead reckoning techniques have the issue of accumulating error, the value shown would be incremental for every sample beyond the first.

Simulation performance was quite good for yaw, although a small error in yaw and a larger error in position can be seen when at faster velocities. The exact cause of this issue in simulation is still unknown. The physical LiDAR has a delay between each individual range measurement which would cause each measurements to be taken at different positions. This is not compensated for in the calculations, however, the simulation scan topic reports both scan time and time increment as 0.0 seconds. Meaning either the simulation does not publish these values as intended, or the range measurements are all taken simultaneously and the issue lies elsewhere.

Physical performance was acceptable considering the lack of controls. The field was not a perfect rectangle, the beacons that were used, though cylindrical, had voids and cutouts, and proper covariances were not calculated beforehand. This was mostly due to time constraints and restricted access to the

TABLE 1: Table of measured simulation covariances.

Sensor	Measurement	Covariance
Model	x	$5.5 * 10^{-6}$
Model	y	$2.7 * 10^{-6}$
Model	θ	$7.1 * 10^{-5}$
Joint	x & y	$1.6 * 10^{-5}$
Joint	θ	$7.2 * 10^{-5}$
IMU	θ	$5.7 * 10^{-5}$
LiDAR	x	$5.0 * 10^{-3}$
LiDAR	y	$5.0 * 10^{-3}$
LiDAR	θ	$5.0 * 10^{-3}$

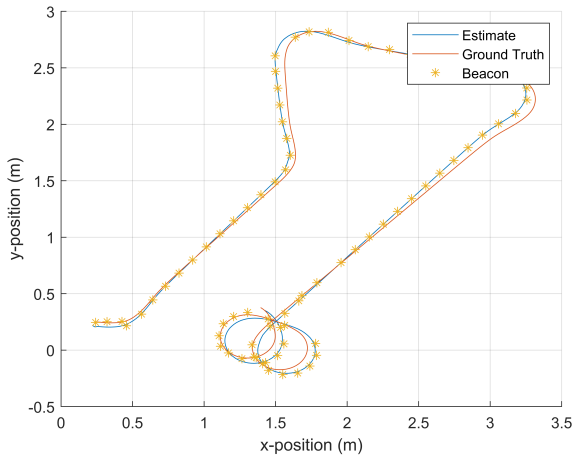


FIGURE 4: Position estimates versus ground truth (Simulation).

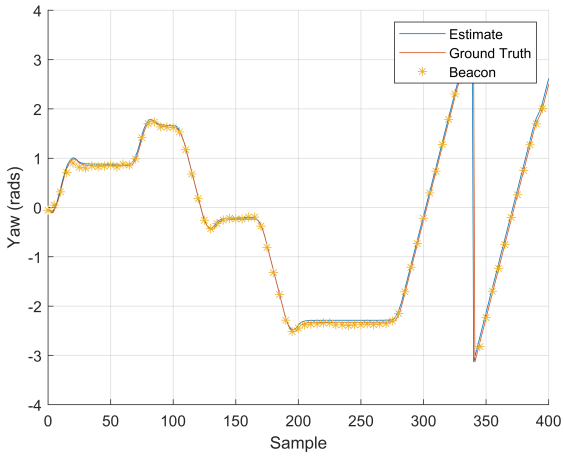


FIGURE 5: Yaw estimates versus ground truth (Simulation.

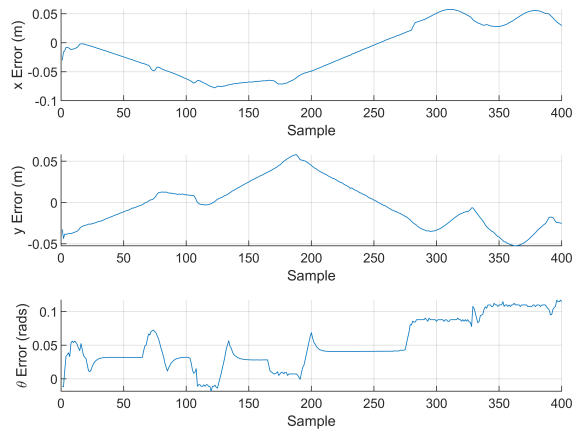


FIGURE 6: Error plot of figure 4 and 5.

bots/lab beforehand. The code however, was usable between simulation and physical with only minor changes.

An issue that was unable to be resolved in the physical implementation however, was approximately once every 200-300 samples, an old sample would seemingly be republished, causing a jump in the position estimate of up to 30 cm. Although the system recovers in a handful of samples (once the next secondary measurement is received), the exact cause of this error was also not able to be determined.

V. CONCLUSION

In simulation, the method used above showed very promising results, with its biggest downfalls showing the possibility of being rectified. The systemic error in the state estimates was not due to the filtering process itself, and such should be fixable with a more accurate model of how the LiDAR works both in simulation and physically. The other issue was that of the update rate of the filter, due to the experiment being run entirely within MATLAB with local parallel processes, as well as the simulation running in a virtual machine running on the same machine, the fastest update rate achievable was 5Hz for the primary measurements, and 1Hz for the secondary. This was not a limitation of the dependent topics, as they updated at a rate of at least 25Hz based off initial tests. If the code was written in something like Python, and the physical TurtleBot used instead of the simulation, the processing could be more easily paralleled and distributed to offload the computation.

The methods as they stand are suitable as are as part of a more complex path planning and motion control algorithms as long as the systemic errors mentioned are taken into account for tolerances.

REFERENCES

[1] M. M. Atia, S. Liu, H. Nematallah, T. B. Karamat, and A. Noureldin, "Integrated indoor navigation system for ground vehicles with automatic 3-d alignment and position initialization," IEEE Transactions on Vehicular Technology, vol. 64, no. 4, pp. 1279–1292, 2015.

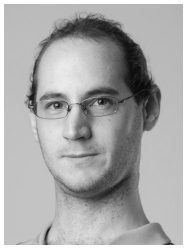
[2] M. Wu, H. Ma, M. Fu, and C. Yang, "Particle filter based simultaneous localization and mapping using landmarks with rplidar," in Intelligent Robotics and Applications, H. Liu, N. Kubota, X. Zhu, R. Dillmann, and D. Zhou, Eds. Cham: Springer International Publishing, 2015, pp. 592–603.

[3] S. Konatowski, P. Kaniewski, and J. Matuszewski, "Comparison of estimation accuracy of ekf, ukf and pf filters," Annual of Navigation, vol. 23, 12 2016.

[4] A. Gopalakrishnan, N. Kaisare, and S. Narasimhan, "Incorporating delayed and infrequent measurements in extended kalman filter based nonlinear state estimation," Journal of Process Control, vol. 21, pp. 119–129, 01 2011.

[5] H. L. Alexander, "State estimation for distributed systems with sensing delay," in Data Structures and Target Classification, V. Libby, Ed., vol. 1470, International Society for Optics and Photonics. SPIE, 1991, pp. 103 – 111. [Online]. Available: https://doi.org/10.1117/12.44843

[6] S. Safari, F. Shabani, and D. Simon, "Multirate multisensor data fusion for linear systems using kalman filters and a neural network," Aerospace Science and Technology, vol. 39, pp. 465 – 471, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963814001126



MICHAEL J. DUKE is was born in Adelaide, Australia in 1991. He is currently studying to receive the BEng(Hons) degree in mechatronic engineering for University of South Australia, Adelaide, Australia.

From 2009 to 2019, he held a full-time position as a Precision Agriculture Technician for Rocky River Ag Services in Crystal Brook, Australia. Since 2019, he has been working as a Mechatronic Design Engineer Undergraduate for Applidyne Australia Pty Ltd. in Adelaide, Australia. His current research interests include control systems, model based design and digital twins, and electric vehicle drive train and battery technology.