ENGN4627 - SLAM Localisation using an Extended Kalman Filter.

u5801396 - Ryan Pike - Undergraduate Group 11

November 4, 2018

1 Findings

A key concept in control theory is the ability to estimate states in the face of noise. A key use for state estimation is the localisation and mapping of the real world environment. In this sense we use on board proprioceptive sensors of a robot to predict its position in the environment. This prediction is compared to observations from the exteroceptive sensors in such a way to update its position in the environment and update the environment itself.

> "One may separate the problem of physical realization into two stages: computations of the "best approximation" $\hat{x}(t_1)$ of the state from knowledge of $y(t)$ for $t \leqslant t_1$ and computation of $u(t_1)$ given $\hat{x}(t_1)$"
>
> – R. E. Kalman, "Contributions to the Theory of Optimal Control," 1960 [3]

Using a turtlebot in the ROS environment the task of localisation and mapping was as follows;

1. Place the turtlebot at the specified origin.
2. Drive the turtlebot (either remotely or autonomously) in the specified room.
3. Aim to identify and correctly store landmarks around the room. (These were specified via coloured cylinders)
4. Store the results of the cylinders and compare to the ground truth.

1.1 Method

The standard skeleton proposed in class was implemented effectively with several improvements to both robustness and stability. The pseudo-algorithm for this skeleton is as follows [5];

Algorithm 1 Extended Kalman Filter (With Mahalanobis value α)

procedure INPUT: $(\mu_{t-1}, \Sigma_{t-1}, \mu_t, z_t)$
Prediction:
  $\bar{\mu}_{x_t} \leftarrow f(\mu_{x_{t-1}}, \mu_{u_t})$.
  $\bar{\Sigma}_t \leftarrow F\Sigma_{t-1}F^\top + W\Sigma_{u_t}W^\top$
Update:
  $S = C_t\bar{\Sigma}_t C_t^\top + \Sigma_{z_{tj}}$
  if $\det S \neq 0, y^\top S^{-1} y > \alpha$ **then**
    $K_t \leftarrow \bar{\Sigma}_t C_t^\top S^{-1}$
    $\mu_t \leftarrow \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_{x_t}, \mu_{1_t}))$.
    $\Sigma \leftarrow (I - K_t C_t)\bar{\Sigma}_t$
return $\bar{\mu}_t, \bar{\Sigma}_t$

The state mean is of dimension $3 + 2n$ where n is the number of landmarks seen, the state covariance is simply that dimension squared. Line 7 also assumes time synchrosisation of the measurement readings. The complementary features to this algorithm are described as follows;

1.1.1 Matching

The matching procedure was ensuring the set of predicted features and observed features were within a sufficient distance of one another. This was performed using the *mahalanobis* distance which computes the euclidean distance of innovation y_t to the innovation covariance using standard deviations as its basis of distance [6]. This was computed as follows;

$$y^\top \Sigma_{IN} y < \alpha; \qquad \Sigma_{IN} = C_t \bar{\Sigma}_t C_t^\top + \Sigma_{z_{tj}} \tag{1}$$

For a particular probability with 3 dimensions we use the lower end of the chi-squared value for a certainty of 0.975 [5]. This made $\alpha = 7.378$ The second set of matching was performed through time synchronisation which is a message filter provided in the ROS environment. The control procedure was as follows. The messages are recevied with an associated time-stamp. If the sequence of messages have time-stamps within a certain distance of one another (0.07) then we initiate the respective callback function. This was performed using the message filter *ApproximateTimeSynchronizer* [7].

1.1.2 Filtering

The dimensions of the room, aswell as the linear and angular velocity are bounded. With this in mind it is possible to ensure no values that are unrealistic or impossible to be put through the algorithm. The following restrictions were placed on each iteration such that if such a value was seen the algorithm would ignore it or saturate it. The control was placed on both the odometry readings and also the landmark readings. The landmark readings were eliminated entirely in the following ranges (In sensor frame);

$$\text{Kinect Sensor Margins: } \begin{cases} Z > 2.6 \\ Z < 0.6 \\ X > 2.5 \\ X < -2.5 \end{cases} \tag{2}$$

For the odometry readings, ignoring the value could interrupt the continuity of the algorithm and as such we applied a saturation method;

$$\text{Odometry Saturation: } \begin{cases} \dot{x} > 0.22 & \implies \dot{x} = 0.22 \\ \dot{x} < -0.22 & \implies \dot{x} = -0.22 \\ \dot{\theta} > 1.1 & \implies \dot{\theta} = 1.1 \\ \dot{\theta} < -1.1 & \implies \dot{\theta} = -1.1 \end{cases} \tag{3}$$

If the $\dot{\theta}_t$ was within these values see (**1.1.4**)

1.1.3 Covariance Motivation

There are two covariances that can be seen from line 4, and 6 which are $\Sigma_{u_t}, \Sigma_{z_{tj}}$ Respectively. These can be initialised at each time step (provided a measurement is sequenced) and they relate to the uncertainty in the motion of the robot and also the uncertainty in the measurement. The former covariance was initialised as follows;

$$\Sigma_{u_t} = \|u_t\|_2 \cdot \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} \tag{4}$$

This is essentially proportional to the identity indicating no insight into the relationships between the motion variables $theta, \dot{x}, \dot{y}$ However the intuition was placed on the idea that if the robot was moving at a greater speed over the same time-step then the uncertainty should grow accordingly. As for the measurement covariance $\Sigma_{z_{tj}}$. The exact value of 0.2 was found through trial and error. The following paper *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Application* found that the error in the Kinect sensor increases quadratically from a few mm at 0.5 metres to 4mm at 3 metres [4]. From these findings the following quadratic was approximated and associated covariance matrix initialised to be;

$$f(z) = 0.00471429z^2 - 0.0106429z + 0.00714286 \tag{5}$$

$$\Sigma_{z_{tj}} = f(z) \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6}$$

1.1.4 IMU Usage

It is a general heuristic that integrating over long periods of time from a wheel encoder is more stable than the same integration over the gyro data. This is due to the drift factor attributed to both measurement devices. In this sense it was found that fusing both readings resulted in superior performance. The following control mechanism was implemented to decide which sensor to use. Assuming time synchronisation the encoder

Algorithm 2 Choice of Theta

1: **procedure** INPUT: $(\dot{\theta}_t^{ODOM}, \dot{\theta}_t^{IMU})$
2: **if** abs $\left(\dot{\theta}_t^{ODOM} - \dot{\theta}_t^{IMU} > (0.03 \times \frac{\pi}{180})/dt\right)$ **then**
3: $\dot{\theta}_t = \dot{\theta}_t^{IMU}$
4: **if** $\dot{x} \neq 0$ **then**
5: $\dot{\theta}_t = \dot{\theta}_t^{IMU}$
6: $\dot{\theta}_t = \dot{\theta}_t^{ODOM}$
7: **return** $\dot{\theta}_t$

tended to give poor reading when the linear velocity was non-zero and whenever the difference in values from both sensors was greater than the threshold (line 2). This choice was motivated by the paper *Gyrodometry: A New Method for Combining Data from Gyros and Odometry in Mobile Robots*

1.1.5 Visualisation

In order to get a sense of what the robot was seeing the visualisation environment RVIZ was used. Using the *MarkerArray* object it is possible to plot the markers in real-time, aswell as the orientation and pose of the robot. This was critical in overcoming some of the programming mistakes.

1.2 Results

Upon demo day we received an error to the ground truth of 0.9. This was an unexpected result as the trial runs performed leading up to the demo showed far greater accuracy ≈ 0.09. The visualisation environment showed a seemingly good result however upon performing the second loop in the two minute time frame a single measurment seemed to skew the entire environment. This could have been an ill-conditioned matrix or a measurement we simply had not filtered. It could also be the other end of the spectrum where the *mahalanobis* distance was not allowing the appropriate correction as the observation was too far from the predicted feature.

1.3 Discussion

With all of these demonstrations there are varying degrees of systematic and random error that is hard to account for. On the day the robot was not properly calibrated for both the IR sensor and the cylinder

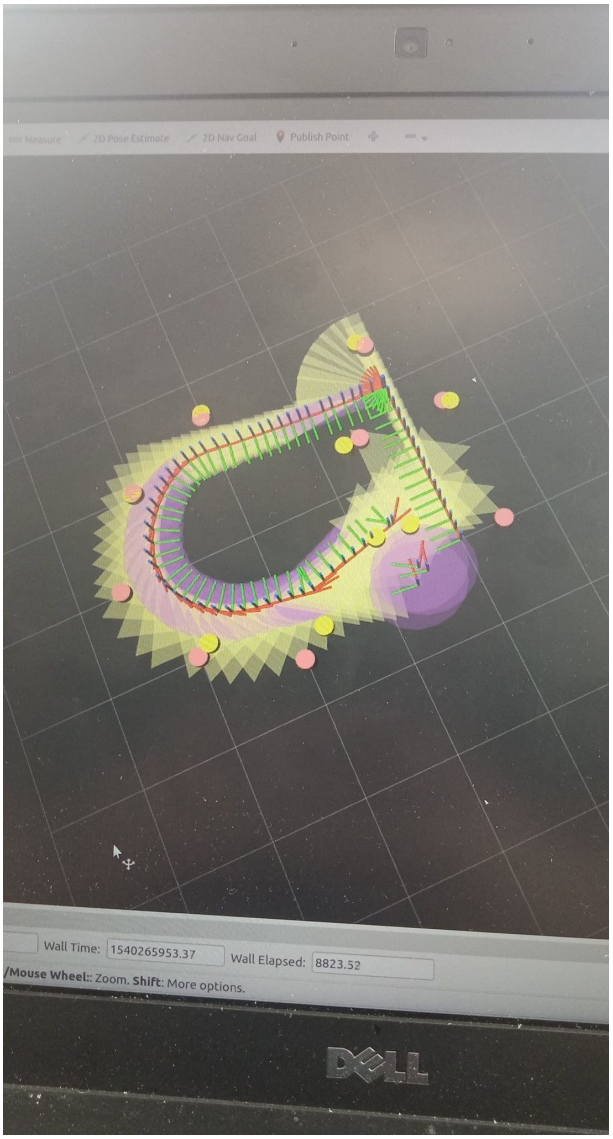
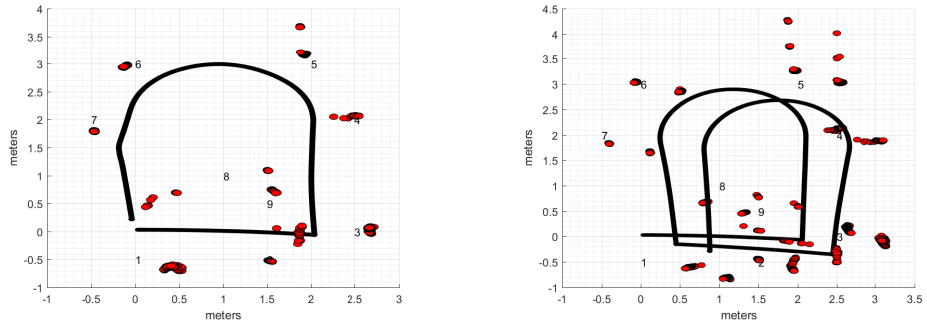


Figure 1: Visualisation environment showing the use of the *markerArray* object for IMU bag 2. *The dominant covariances for position and orientation are plotted in purple and yellow respectively*

detection package. Considering most trial runs performed previously were done at night time the effect of shadows was no substantiated for. This may have led to the discrepancy in the final result.

2 Task 1: Data Acquisition and Plotting

The data acquisition is as discussed in the lab template. As for plotting there was a method that can be run in real time (RVIZ) and then a 2D plot that was simulated afterwards. The following two figures are the open loop prediction for both IMU1 and IMU2 rosbags; As we can see from both plots the open loop



(a) IMU 1 Rosbag

(b) IMU 2 Rosbag

Figure 2: Using open loop prediction

operation is not accurate. It is simply plotting landmarks as it sees them and as such we see significant drift in both pose trajectories. There are cases where the cylinder is closely matched to the ground truth (Shown in blue) however there is no decision mechanism on if we should trust this measurement.

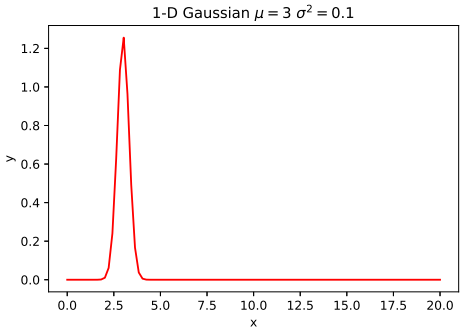
3 Task 2: State Estimation using Filtering

3.1 Gaussian Distribution

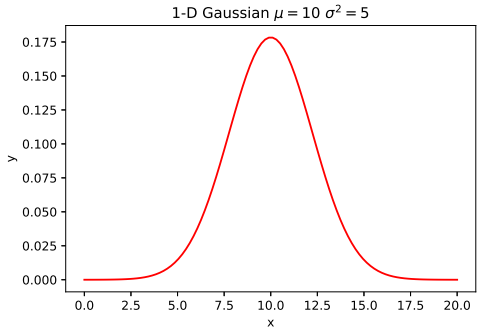
For a 1-Dimensional Gaussian we are simply moving the peak of the probability density function which takes on the value μ. The variance determines how fat or thin the PDF is. In the 2-D cases again the mean vector μ translates the peak of the surface whilst the covariance determines the kurtosis, skewness and how flat or sharp the peak is. These properties can be demonstrated quite succinctly in Figure 3. We can see in (d) how a larger covariance in the x-direction will cause the surface to spread out over this region whilst remaining thin in the other direction. We see that as the variance is decreased as in (c) the surface is very flat except for the sharp peak to ensure the integral remains at unity. If we continually increase the covariance we require a larger domain to see the entire surface and the kurtosis decreases.

3.2 Kalman Filter

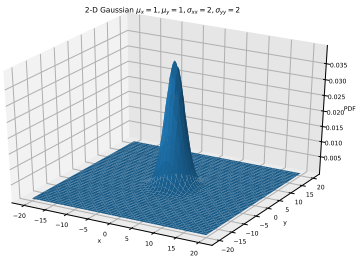
For the Kalman filter we are tasked with understanding how the robots initial position μ, initial uncertainty Σ_μ, motion error σ_u and measurement error σ_z affect performance. It can be shown Figure 4 that the filter is robust to initial uncertainty in its position. This can be seen from (b) and (d) where the KF converged to a very similar distribution. When increasing the measurement and motion accuracy the model tends to display poor behaviour surrounding its certainty in its position. Since initial uncertainty is robust we will omit the discourse in the 2-D case and instead focus on measurement and motion error. In Figure 5 we see that increasing the motion error increases the uncertainty after each prediction and it takes more measurements to try and gain certainty in its position. In a similar fashion when increasing the uncertainty in the measurment the robot is less certain of its observations however from (b) and (d) it shows more robustness to perturbation



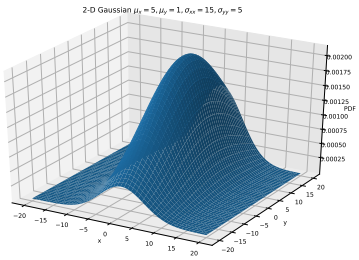
(a) Small variance



(b) Moderate variance (shifted mean)



(c) Small variance in the 2-D Case



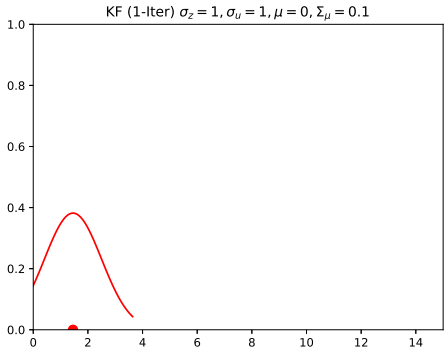
(d) Dominant x-covariance

Figure 3: Variations in the 1-D and 2-D Gaussian

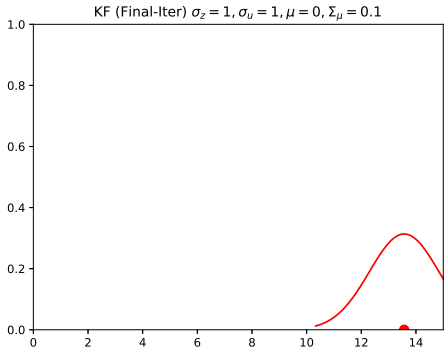
in motion error as opposed to measurement error. The primary difference between changing both is the step at which the error is propagated.

3.3 Particle Filter

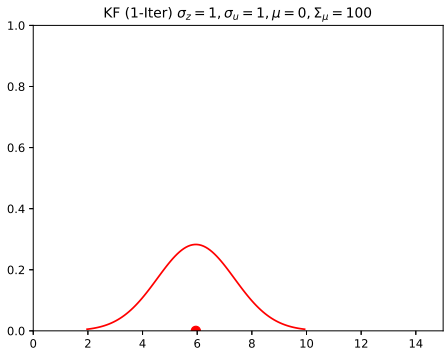
The final demonstration is a particle filter which instead of parametrizing distributions it takes in a set of weighted samples which can then compute the posterior from the given prior. This does require the discretisation of the space leading to the primary result that a higher particle density indicates a greater degree of certainty. The *particle Number* tends to give a bias towards the posterior estimate as there are only finitely many. It was found that increasing the number of particles gave a heightened degree of certainty of its position, however increased computational complexity. in (a) - (b) we see the samples selected are agglomerating to a stable position. When the *forward noise* is increased the sparsity or distance from the samples tends to increase, however there is still a level of convergence to the desired state. Similarly for *turn noise* we get the same effect. When *sensor noise* is increased we get very high uncertainty in its localisation and it seems the filter is not robust to this variable which can be seen in (e)-(f).



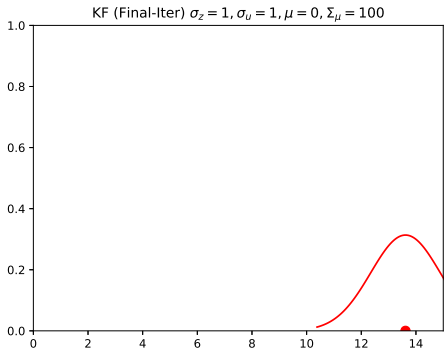
(a) Small initial uncertainty



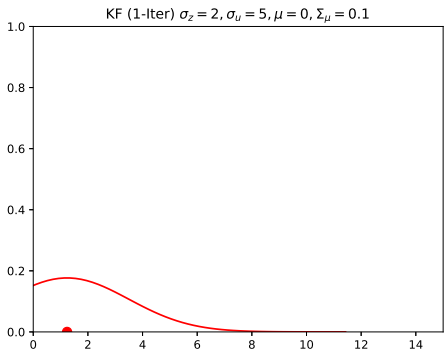
(b) Final iteration



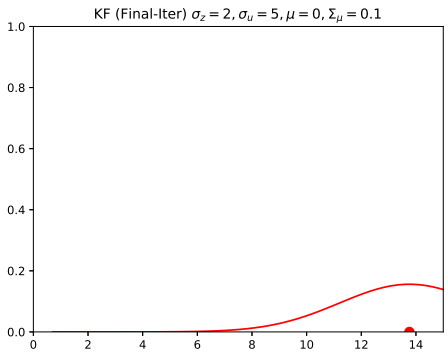
(c) Relatively large initial uncertainty



(d) Final Iteration (Converged to above)

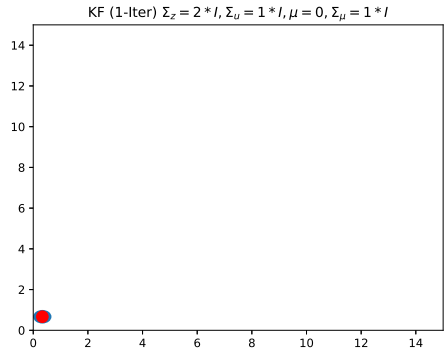


(e) Increased motion and measurement covariance

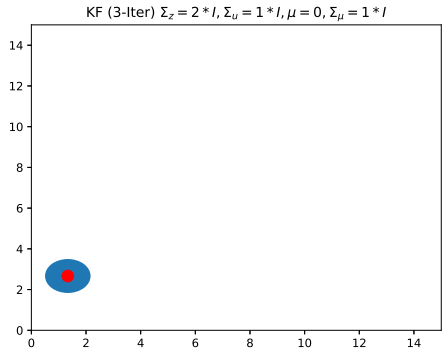


(f) Final iteration

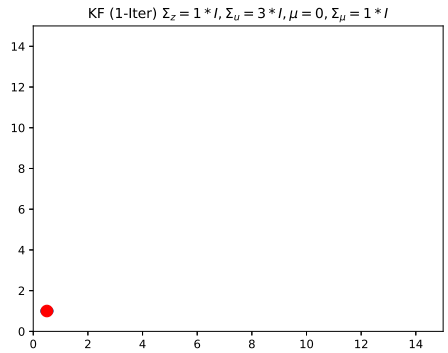
Figure 4: Variations in the 1-D and 2-D Gaussian



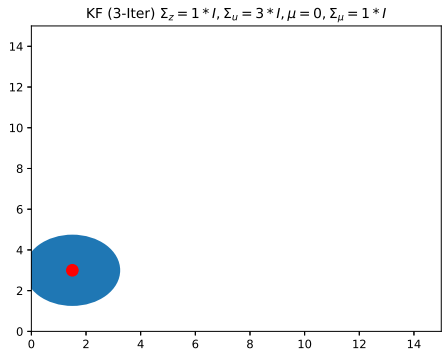
(a) Decreased motion uncertainty



(b) 3rd Iteration

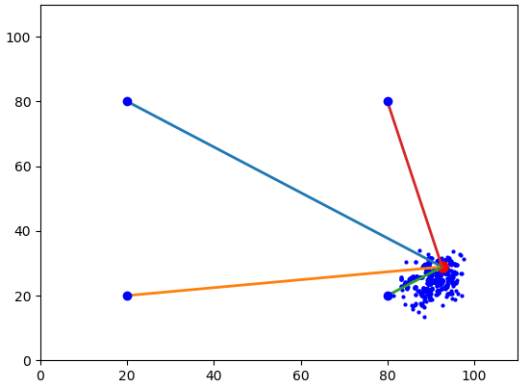


(c) Increased motion uncertainty

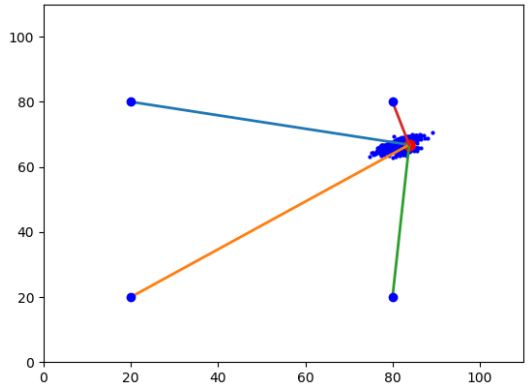


(d) 3rd iteration

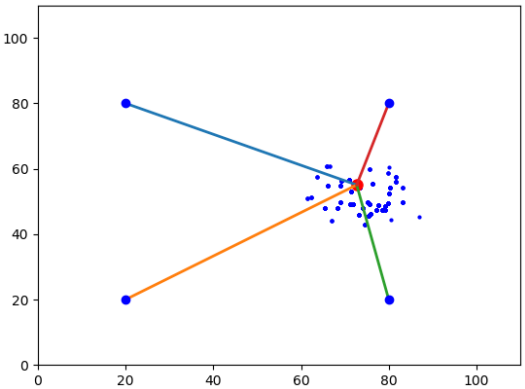
Figure 5: Error propagation in the Kalman Filter (Same starting mean and state covariance)



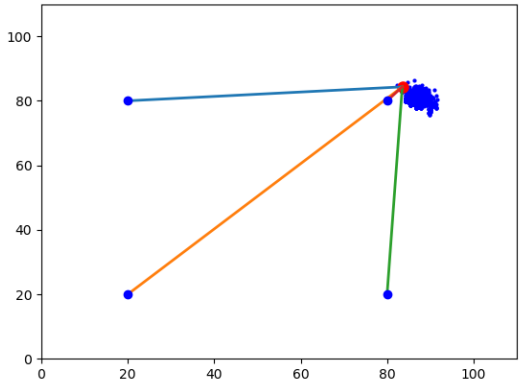
(a) 15000 particles iteration 1 1



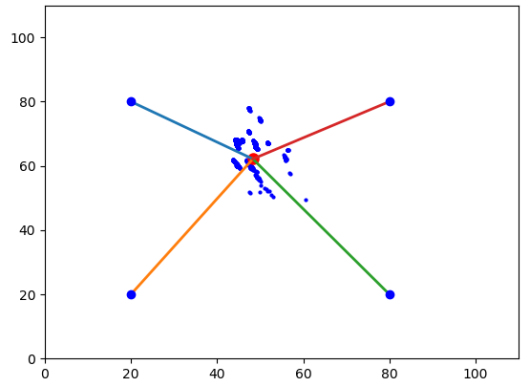
(b) 15000 particles iteration 10



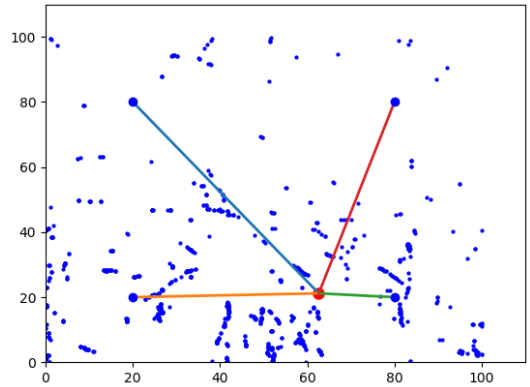
(c) Increased forward noise iteration 1



(d) Increased forward noise iteration 10



(e) Small sensor noise iteration 10



(f) Large sensor noise iteration 10

Figure 6: Particle Filter for varying levels of samples, forward noise and sensor noise

4 Lab Part I & II: Report Problems

4.1 Question 1: Theory

4.1.1 Bayesian formula

The Bayesian formula is a relationship between two conditional probabilities of the form $P(x \mid y)$ and $P(y \mid x)$. It is formally stated as [8];

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)} = \frac{P(y \mid x)P(x)}{\sum_x' P(y \mid x')} \qquad \text{(Discrete)} \tag{7}$$

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)} = \frac{P(y \mid x)P(x)}{\int_{x'} P(y \mid x')} \qquad \text{(Continuous)} \tag{8}$$

It gives us a way to compare the probability of an event occuring given prior knowledge of similar conditions.

4.1.2 Total probability theorem

Using the axioms of probability theory and features from conditional probability the total probability theorem can be stated as [8];

$$p(x) = \sum_y P(x \mid y)P(y) \qquad \text{(Discrete)} \tag{9}$$

$$p(x) = \int_y P(x \mid y)P(y)dy \qquad \text{(Continuous)} \tag{10}$$

4.1.3 Markov assumption

The markov assumption in the context of this demo would be that the current mean state $\bar{\mu}_t$ at time t is independent from both the previous state $\bar{\mu}_{t-1}$ and also the future state $\bar{\mu}_{t+1}$ This is a very strong assumption which is typically violated given and sub-optimal conditions.

4.1.4 SLAM

Simultaneous localisation and mapping is a dual problem of attempting to simultaneously build an environment and also track your position within said environment. The aim is to build both the environment (map) and the position (localisation) to varying degrees of accuracy depending on the context. This has massive uses in both military and space aswell as service robotics. Methods have improved greatly over time thanks to the powers of modern computers leading from in-direct feature extraction to large-scale direct SLAM all in real time.

4.1.5 Multivariate distribution

Assuming the continuous case, if we have a vector of random variables $\boldsymbol{X} := [X_1, X_2, \cdots, X_n]$ Then this vector will have a joint probability function and also a cumulative density function. The properties of such are as follows for the joint density function $f(\boldsymbol{X})$ [2];

$$f(x_1, \ldots, x_n) \geqslant 0 \tag{11}$$

$$\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \ldots, x_n)dx_1, \ldots, dx_n = 1 \tag{12}$$

$$P\left[(X_1, \ldots, X_n) \in A\right] = \int_A \cdots \int_A f(x_1, \ldots, x_n)dx_1, \ldots, dx_n \tag{13}$$

4.2 Question 2: Gaussian Motion Model

4.2.1 Distribution of priors

These are highly trivial computations which can be performed as follows. For the one-dimensional case (using values from Lab Part I);

$$\mu_{prior} = \mu_{posterior} + \mu_{move} = 2 + 1 = 3 \tag{14}$$

$$\sigma_{prior} = \sqrt{\sigma^2_{posterior} + \sigma^2_{move}} = 2\sqrt{5} \tag{15}$$

For the two dimensional case the only difference is dropping the squared term and simply adding the covariances. Therefore;

$$\mu_{prior} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \tag{16}$$

$$\Sigma_{prior} = \begin{bmatrix} 4 & 1 \\ 1 & 5 \end{bmatrix} \tag{17}$$

4.2.2 PDF of a weighted linear combination of Gaussian random variables

Let $Z := \omega_1 X + \omega_2 Y$, with CDF and PDF $F_Z(z), f_Z(z)$ respectively. We can show the following;

$$F_Z(z) = P(z \leqslant z) \tag{18}$$

$$= P(\omega_1 X + \omega_2 Y \leqslant z) = \int_{-\infty}^{\infty} P(\omega_1 x + \omega_2 Y \leqslant z) f_X(x) dx \tag{19}$$

$$\tag{20}$$

where $f_X(x)$ is the PDF of X. Now;

$$P(\omega_1 x + \omega_2 Y \leqslant z) \equiv P\left(Y \leqslant \frac{z - \omega_1 x}{\omega_2}\right) \tag{21}$$

$$\implies F_Z(z) = \int_{-\infty}^{\infty} F_Y\left(\frac{z - \omega_1 x}{\omega_2}\right) f_X(x) dx \tag{22}$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\frac{z-\omega_1 x}{\omega_2}} f_Y(t) dt f_X(x) dx \tag{23}$$

Using the fact that $F'_Z(z) = f_Z(z)$ we have that;

$$f_Z(z) = \frac{\partial}{\partial z} \int_{-\infty}^{\infty} \int_{-\infty}^{\frac{z-\omega_1 x}{\omega_2}} f_Y(t) dt f_X(x) dx \tag{24}$$

$$= \frac{1}{\omega_2} \int_{-\infty}^{\infty} f_Y\left(\frac{z - \omega_1 x}{\omega_2}\right) f_X(x) dx \tag{25}$$

$$\tag{26}$$

Noting the simplicity of this derivative comes from the realisation that the only place z occurs is in the upper place of the integral. We know the following properties of X and Y;

$$X \sim \mathcal{N}(u_1,\, s_1) \tag{27}$$

$$Y \sim \mathcal{N}(u_2,\, s_2) \tag{28}$$

Since these are both normally distributed we can compute the above integral in a symbolic package such as `Mathematica` and we arrive at the following result;

$$f_Z(z) = \frac{1}{\sqrt{2\pi}\sqrt{\omega_1^2 s_1 + \omega_2^2 s_2}} \exp\left\{ -\frac{[z - (\omega_1 u_1 + \omega_2 u_2)]^2}{2(\omega_1^2 s_1 + \omega_2^2 s_2)} \right\} \tag{29}$$

From this we see the following property of Z;

$$Z \sim \mathcal{N}\left(\omega_1 u_1 + \omega_2 u_2,\ \omega_1^2 s_1 + \omega_2^2 s_2\right) \tag{30}$$

4.3 Question 3: The Athenian Taxi Quandary

This is a question primarily concerning conditional probability. The prior distribution that is given to us is that 9/10 cars in Athens are green. Therefor let the true colour of the car be denoted T, and the Perceived colour of the car to be P. Therefore we have

$$P(T = green) = 0.9 \tag{31}$$
$$P(T = blue) = 0.1 \tag{32}$$

It is presupposed that discrimination between blue and green is 75% reliable. This statement is telling us the following relations;

$$P(T = green \mid P = green) = 0.75 \tag{33}$$
$$P(T = blue \mid P = blue) = 0.75 \tag{34}$$
$$P(T = green \mid P = blue) = 0.25 \tag{35}$$
$$P(T = blue \mid P = green) = 0.25 \tag{36}$$

We can now just use the sum rule of probability in conjunction with Bayes rule to see that;

$$P(T = green \mid P = blue) = \frac{P(P = blue \mid T = green)P(T = green)}{P(P = blue)} \tag{37}$$
$$= 0.75 \tag{38}$$
$$P(T = blue \mid P = blue) = \frac{P(P = blue \mid T = blue)P(T = blue)}{P(P = blue)} \tag{39}$$
$$= 0.25 \tag{40}$$

Here we have used the following relation;

$$P(P = blue) = P(P = blue \mid T = blue)P(T = blue) + P(P = blue \mid T = green)P(T = green) \tag{41}$$

Therefore it was three times as likely that the car was indeed green than blue.

4.4 Question 4: Markov Localisation

Suppose a robot is living in a 1D world. The world is divided in 5 cells and it is cyclic. The current belief of the robot is:

1/9	1/3	1/3	1/9	1/9

We are also given the following conditionals;

$$P(x_t \mid x_t) = 0.1 \tag{42}$$
$$P(x_{t+1} \mid x_t) = 0.8 \tag{43}$$
$$P(x_{t+2} \mid x_t) = 0.1 \tag{44}$$

The robots motion command is u which is set to 1. We can discretize the Markov localisation algorithm as follows [8];

Algorithm 3 Markov Localisation (Without Measurement)

1: **procedure** INPUT: BEL(x_{t-1}, u_t)
2: **for all** x_t **do**
3: $bel(x_t) = \sum_{x_t} P(x_t \mid u_t, x_{t-1}) bel(x_{t-1})$

This gives the following result;

Cell 1	$1/9 \times 0.1 + 1/9 \times 0.8 + 1/9 \times 0.1$	1/9
Cell 2	$1/3 \times 0.1 + 1/9 \times 0.8 + 1/9 \times 0.1$	2/15
Cell 3	$1/3 \times 0.1 + 1/3 \times 0.8 + 1/9 \times 0.1$	14/45
Cell 4	$1/9 \times 0.1 + 1/3 \times 0.8 + 1/9 \times 0.1$	14/45
Cell 5	$1/9 \times 0.1 + 1/9 \times 0.8 + 1/3 \times 0.1$	2/15

The prior is the very right column which to three decimal places is;

$bel(x_t) =$

0.111	0.133	0.311	0.311	0.133

Ultimately the computation is simply summing over all possible future states given the current belief. For the first cell for instance, the robot can either stay in the same position, move from cell 5 or overshoot from cell 4. There are no possible other ways that robot can be in that state. We express this through the formula of total probability.

4.5 Question 5: Kalman Filter Localisation

4.5.1 Structure of a KF

There are two primary steps to a kalman filter excluding the rudimentary but important step of initialisation, we have predict and update. In the predict state, the prior is estimated from the previous state at time t except for the control input which remains at u_{t-1}. In the update step the posterior belief is computed through methods described above.

4.5.2 Prior Derivation for a state

Using the following formula which is a case of the total probability theorem we have [5];

$$\overline{bel}(x_t) = \int_{x_t} P(x_t \mid u_t; x_{t-1}) bel(x_{t-1}) dx_{t-1} \tag{45}$$

We also know that $bel(x_{t-1}) \sim \mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$, $P(x_t \mid u_t; x_{t-1}) \sim \mathcal{N}(x_t; A_t x_{t-1} + B_t u_t, \Lambda)$ Which allows us to subsitute the corresponding probability density functions. This gives the following result;

$$\overline{bel}(x_t) = \eta \int \exp\left\{ \frac{-1}{2}(x_t - A_t x_{t-1} - B_t u_t)^\top \Lambda_t^{-1}((x_t - A_t x_{t-1} - B_t u_t) \right\} \cdot \tag{46}$$

$$\exp\left\{ \frac{-1}{2}(x_{t-1} - \mu_{t-1})^\top \Sigma_{t-1}^{-1}((x_{t-1} - \mu_{t-1}) \right\} dx_{t-1} \tag{47}$$

Here the normalising constant $\eta = (2\pi \det \Sigma_{t-1})^{-1/2}(2\pi \det \Lambda_t)^{-1/2}$

4.5.3 Markov vs. Kalman

The primary difference is in the initialisation step. The Markov algorithm allows any initial distribution for the robot state mean, whilst the Kalman filter has to be normally distributed. This means Markov localisation can begin from any unknown position as the states are disparate. This downfall requires a discrete representation of the space and memory and computational power can limit the precision of this representation. The KF filter on the other hand can be used in *continuous* representations of the physical world. The primary downside to the Kalman filter is if the algorithm or sensors fail to capture all the possible robot positions then we can get unbounded uncertainty. This then makes the robot completely lost unable to recover its position. For the *kidnapped robot problem* it would be best to use the markov localisation algorithm despite computational resources as its initial uncertainty is not specified which is a strict requirement of the Kalman filter.

References

[1] J. Borenstein, L. Feng *Gyrodometry: A New Method for Combining Data from Gyros and Odometry in Mobile Robots* The University of Michigan, 1996

[2] E.T. Jaynes *Probability Theory: The Logic of Science* Cambridge University Press, 2003

[3] R.E. Kalman. *Contributions to the theory of optimal control.* Boletin de la Sociedad Matẽmatica Mexicana, *5:102-119, 1960*

[4] Kourosh Khoshelham, Sander Oude Elberink *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications* MDPI Sensors *12: 1437-1454, 2012*

[5] ANU, College of Engineering and Computer Science *Lecture slides for ENGN4627 (Robotics)* Oct. 2018

[6] Roland Siegwart, Illah R. Nourbakhsh, Davide Scaramuzza *Introduction to Autonomous Mobile Robots 2nd, Edition* The MIT Press, Cambridge, Massachesetts 2011

[7] ROS wiki *http://wiki.ros.org/message_filters*

[8] Sebstian Thrun, Wolfram Burgard, Dieter Fox *Probabilistic Robotics.* Princeton Intelligent Robotics and Autonomous Agents series Edition, 2006.