Applied Estimation(EL2320) Lab 2 Instructions

1 Introduction

This lab consists of two parts:

1. A series of preparatory questions that we will go over in class
2. Two seperate matlab exercises
   - A preparatory case study with the particle filter where you learn more about the behavior of the particle filter. Similar to the warm up part of the lab 1, very little extra code is needed.
   - The main lab II problem in which you need to complete an implementation of a Monte Carlo Localization(MCL) algorithm.

It is suggested to go through the lecture notes again before you start this lab. If you are using the recommended course book, it is a good time to read chapter 4 to do a recap about the particle filters and chapter 8 to further familiarize yourself with the MCL problem. In order to pass this lab you need to

1. Provide written answers to the preparatory questions, Part I.
2. Follow the instructions given for the warm up particle filter problem
3. Write the code to complete the (incomplete) functions in the MCL problem
4. Provide written answers to the questions in the sections 2 and 3
5. Upload your [working!] code along with sample plots (png, jpg, or pdf) at illustrative times during the runs on each data set. The plots should be analysed in the report with a short text on each that explains why it looks the way it does compared to the other plots. Note that you do NOT need to upload the datasets, test cases or warmup code.

The answers to most questions are 1 (sometimes up to 3) sentence per part. So a question that asks 'What is A and why is B?' can be answered in two sentences most of the time. Being concise and correct (or wrong) will get you better feedback than writting a few paragraphs that are hard to follow.

There will be help session for the lab. You need to do the readings and answer the questions in sections 2 and 3 (as many as you can) before you start to write the code and attend the help sessions.

PART I - Preparitory Questions

These questions are intended to encourage you to think about what we have covered so far on the Particle filter and localization.

You should write out your answers to the questions neatly and legibly and bring them to class on the assigned day. Be sure to include your name. It is advised that you go ahead and use a pdf document that you then print out for class as you will need to upload these answers later. This will be graded and give bonus points as in lab 1. You will then be able to correct the answers before uploading the lab report to the course web assignment. In other words getting all the answers correct for class is not necessary, but participation is of great benefit to you.

As you see the number of questions is less than on the first lab as you are all now familiar with the basic problem of estimation and localization from the first lab. However there are other issues with this lab that have to do with the need for efficient Matlab code. These will be also disscussed during the preparative session.

Particle Filters:

1. What are the particles of the particle filter?
2. What are importance weights, target distribution, and proposal distribution and what is the relation between them?
3. What is the cause of particle deprivation and what is the danger?
4. Why do we resample instead of simply maintaining a weight for each particle always.
5. Give some examples of the situations which the average of the particle set is not a good representation of the particle set.
6. How can we make inferences about states that lie between particles.
7. How can sample variance cause problems and what are two remedies?
8. For robot localization for a given quality of posterior approximation, how are the pose uncertainty (spread of the true posteriori) and number of particles we chose to use related.

PART II - Matlab Excercises

2 Warm up problem with the Particle Filter

In this section we will go through a rather simple solution to a 2D estimation problem using the Sampling-Importance Re-sampling(SIR) algorithm(Alg 1).

Algorithm 1 SIR General Algorithm

Inputs: Particle set $S_{t-1} = \{< x^i_{t-1}, w^i_{t-1} > | i = 1, ..., N\}$, Observation z_t, Control signal u_t
$\bar{S}_t = S_t = \emptyset$
for m = 1 to M **do**
  sample $x^m_t \sim p(x_t | u_t, x^m_{t-1})$ {Prediction}
end for
for m = 1 to M **do**
  $w^m_t = p(z_t | x^m_t)$ {Weighting}
end for
$\bar{S}_t = \bigcup_{m=1}^{M} < x^m_t, w^m_t >$
for m=1 to M **do**
  draw $i \propto w^i_t$ {Re-sampling}
  $S_t = S_t \cup < x^i_t, \frac{1}{M} >$
end for
return S_t

Now, consider the problem of 2D target tracking: we want to repeatedly estimate where the object is located at each time-step given some [inaccurate] information about how the target has moved from the previous time-step, some measurements and the initial position of the target (the position of the target in the first time step) . The system is described by (1).

$$\begin{aligned} x_t &= \begin{bmatrix} x_{t,x} \\ x_{t,y} \end{bmatrix} \\ x_t &= x_{t-1} + u_t + \varepsilon_t \\ z_t &= x_t + \delta_t \end{aligned} \tag{1}$$

Where ε_t and δ_t are the process and the measurement noises and they are not necessarily Gaussian nor white. The measurements are given using a vision algorithm and therefore, $z_{t,x}$ and $z_{t,y}$ are the pixel coordinates of the [mass center of the] target in some input images.
Download everything in the folder el2320_lab2 and go to the "warm-up" folder. Load the "visiondata.mat" file. You can find the data for three types of experiments:

1. Stationary target: variables starting with "fixed".
2. Target moving on the diagonal axis of the image frame: variables starting with "mov"
3. Target rotating around a point: variables starting with "circ"

For each type of experiments there are three sets of measurements differing in the measurement noise:

1. Measurement noise is white gaussian with a small standard deviation: variables ending with 1
2. Measurement noise is white gaussian with a big standard deviation: variables ending with 2
3. Measurement noise is white gaussian with a small standard deviation but with approximately 50% outliers: variables ending with 3

Measurement noises are similar for different experiment types with the same ending number. You can find the ground truth information about each experiment type in the variables ending with "true". Take a look at the data using the function visualize_vision_data.m: for example try:

visualize_vision_data(fixed_meas_1,fixed_true);

We do not know about the control sequences(u_t), but we can choose a model and compensate with model imperfection with compensation noises. Compensation noise is an artificial noise that we have to consider in the model due to model imperfection. We can consider for example a fixed target, a target moving on a line, target moving with a fixed angular velocity or any other motion model that you can think of! We will go through some examples of the motion model and will analyze their drawbacks and advantages. The choice of the motion model affects the modeled system and it might even change the dimension of the model's state-space. In the following sections, $\bar{x}_t$ represents the modeled state and $\bar{u}_t$ represents the motion model and the following holds for all models

$$\begin{aligned} \bar{x}_t &= \bar{x}_{t-1} + \bar{u}_t + R \\ \bar{z}_t &= \bar{C}\bar{x}_t + Q \end{aligned} \tag{2}$$

where R is the modeled process noise and Q is the modeled observation noise. Although there is no constraint on the process noise being Gaussian in particle filter based approaches, one usually models Gaussian noises mainly because working with Gaussians is easy. Therefore, in the following, we assume that the noises are white Gaussians: $Q \sim \mathcal{N}(0, \Sigma_{Q(2\times 2)})$.

2.1 Prediction

In general, assuming that the process noise is independent of the state, we can break down the prediction step to two steps(applying motion + diffusion)(3).

$$\underbrace{\bar{x}_t^m}_{\texttt{predicted state}} = \underbrace{x_{t-1}^m + \bar{u}_t}_{\texttt{applying motion}} \underbrace{+\mathcal{N}(0, \Sigma_R)}_{\texttt{diffusion}} \tag{3}$$

Where Σ_R is the modeled process noise covariance matrix(assuming the noise is white Gaussian).

2.1.1 2D State Space

$$\begin{aligned} \bar{x}_t &= \begin{bmatrix} \bar{x}_{t,x} \\ \bar{x}_{t,y} \end{bmatrix} \\ C &= I \end{aligned} \tag{4}$$

1. **Fixed target**

$$\bar{u}_t = 0 \tag{5}$$

2. **Target moving on a line**

$$\bar{u}_t = \begin{bmatrix} dx_0 \\ dy_0 \end{bmatrix} = v_0 dt \begin{bmatrix} \cos\theta_0 \\ \sin\theta_0 \end{bmatrix} \tag{6}$$

2.1.2 3D State Space

$$\begin{aligned} \bar{x}_t &= \begin{bmatrix} \bar{x}_{t,x} \\ \bar{x}_{t,y} \\ \bar{x}_{t,\theta} \end{bmatrix} \\ \bar{C} &= (I|0)_{(2\times 3)} \end{aligned} \tag{7}$$

1. **Target moving on a line**

$$\bar{u}_t = \begin{bmatrix} dx_t \\ dy_t \\ 0 \end{bmatrix} = v_0 dt \begin{bmatrix} \cos x_{t-1,\theta} \\ \sin x_{t-1,\theta} \\ 0 \end{bmatrix} \tag{8}$$

   - **Question 1:** What are the advantages/drawbacks of using (6) compared to (8)? Motivate.

2. **Target moving on a circle**

$$\bar{u}_t = \begin{bmatrix} dx_t \\ dy_t \\ d\theta_0 \end{bmatrix} = dt \begin{bmatrix} v_0 \cos x_{t-1,\theta} \\ v_0 \sin x_{t-1,\theta} \\ \omega_0 \end{bmatrix} \tag{9}$$

   - **Question 2:** What types of circular motions can we model using (9)? What are the limitations(what do we need to know/fix in advance)?

2.2 Sensor Model

Having modeled a Gaussian noise represented by Σ_Q, the likelihood function for an observation is given by (10).

$$p(z|x, \Sigma_Q, \bar{C}) = \frac{1}{2\pi|\Sigma_Q|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}[z - \bar{C}x]^T \Sigma_Q^{-1} [z - \bar{C}x]\right) \tag{10}$$

Obviously, how you model the measurement noise affects how the likelihood function will look like. A simple interpretation is that if you model an accurate sensor(smaller $|\Sigma_Q|$s), the likelihood function will be sharper(and vice versa).

- **Question 3:** What is the purpose of keeping the constant part in the denominator of (10)?

2.3 Re-Sampling

2.3.1 Vanilla(Multinomial) Re-Sampling

The vanilla re-sampling method is the most simple method to carry out the re-sampling step. The method draws particles independently using N random variables and the Cumulative Distribution Function(CDF) of the weights of the particle set. Alg 2 shows the Multinomial re-sampling method.

Algorithm 2 Multinomial Re-Sampling

$S_t = \emptyset$
for m = 1 to M **do**
  $CDF(m) = \sum_{i=1}^{m} w_t^i$
end for
for m=1 to M **do**
  $r_m = rand\{0 \leq r_m \leq 1\}$
  $i = arg\min_j CDF(j) \geq r_m$
  $S_t = S_t \cup < x_t^i, \frac{1}{M} >$
end for
return S_t

2.3.2 Systematic Re-Sampling

Systematic re-sampling(Stochastic Universal re-sampling) is an alternative method for re-sampling which offers better speed in addition to better variance. Alg 3 shows the Systematic re-sampling method.

Algorithm 3 Systematic Re-Sampling

$S_t = \emptyset$
for m = 1 to M **do**
  $CDF(m) = \sum_{i=1}^{m} w_t^i$
end for
$r_0 = rand\{0 \leq r_0 \leq \frac{1}{M}\}$
for m=1 to M **do**
  $i = \min j : CDF(j) \geq r_0 + \frac{m-1}{M}$
  $S_t = S_t \cup < x_t^i, \frac{1}{M} >$
end for
return S_t

- **Question 4:** How many random numbers do you need to generate for the Multinomial re-sampling method? How many do you need for the Systematic re-sampling method?
- **Question 5:** With what probability does a particle with weight $w = \frac{1}{M} + \epsilon$ survive the re-sampling step in each type of re-sampling (vanilla

and systematic)? What is this probability for a particle with $0 \leq w < \frac{1}{M}$? What does this tell you? (Hint: it is easier to reason about the probability of not surviving, that is M failed binary selections for vanilla, and then subtract that amount from 1.0 to find the probability of surviving.

2.4 Experiments

Load the file: "visiondata.mat" if you do not have it in loaded. The entrance function to the code is "pf_track" which takes a measurement set, a ground truth set and a verbose level as input arguments. You can run the code using for example

pf_track(fixed_meas_1,fixed_true,2);

- **Question 6:** Which variables model the measurement noise/process noise models?

Read through the code and make sure you understand what is going on in the code. Model the process and measurement noises for the fixed_meas_1 measurement set. Run the particle filter algorithm and observe how the algorithm is working. At the end of the simulation, the mass center of the particle set at each time step is considered to be the estimate of the filter and the estimates are compared to the ground truth.

Try to adjust the number of particles for a 2D sate space to have a precise estimate for the fixed_meas_1 measurement set(precise: mean absolute error should be less than 1 pixels). Campare now to the 3D state space without changing the parameters.

- **Question 7:** What happens when you do not perform the diffusion step? (You can set the process noise to 0)
- **Question 8:** What happens when you do not re-sample? (set RESAMPLE MODE=0)
- **Question 9:** What happens when you increase/decrease the standard deviations(diagonal elements of the covariance matrix) of the observation noise model? (try values between 0.0001 and 10000)
- **Question 10:** What happens when you increase/decrease the standard deviations(diagonal elements of the covariance matrix) of the process noise model? (try values between 0.0001 and 10000)

Now, try the moving targets. For the next three questions try to think about the answer and write what you think will happen.

- **Question 11:** How does the choice of the motion model affect a reasonable choice of process noise model?

- **Question 12:** How does the choice of the motion model affect the precision/accuracy of the results? How does it change the number of particles you need?
- **Question 13:** What do you think you can do to detect the outliers in third type of measurements? Hint: what happens to the likelihoods of the observation when it is far away from what the filter has predicted?

Play with the parameters and observe how they affect the filter and how they change the behavior of the filter in different scenarios. Does this confirm what you wrote for the last three questions?

- **Question 14:** Using 1000 particles, what is the best precision you get for the second type of measurements of the object moving on the circle when modeling a fixed, a linear or a circular motion(using the best parameter setting)? How sensitive is the filter to the correct choice of the parameters for each type of motion?

Congratulations! you have finished the warm-up problem (and most of questions)!

3 Main problem: Monte Carlo Localization

In this section, we will go through the MCL problem. You will encounter both the tracking problem and the global localization problem. We will be using the SIR algorithm(Alg 1) similar to the previous section, with slightly different changes for the prediction and update steps. We will utilize a 3D state-space(x,y,θ) in the following.

3.1 Prediction

For the prediction step, we will utilize a (motion+diffusion) approach similar to (3). The motion model is similar to the motion model in the EKF Localization problem. We will use odometry as the motion model: consult (4) in the lab 1 instructions.

3.2 Measurement Model

The sensor model(the measurement model) is also similar to the measurement model in the EKF Localization problem: Assuming a range-bearing measurement setting and a Gaussian measurement noise, the measurement model for the sensor is given by (11)

$$\begin{aligned} h(x_t, W, j) &= \begin{bmatrix} \sqrt{(W_{j,x} - x_{t,x})^2 + (W_{j,y} - x_{t,y})^2} \\ atan2(W_{j,y} - x_{t,y}, W_{j,x} - x_{t,x}) - x_{t,\theta} \end{bmatrix} \\ z_t^i &= h(x_t, W, j(i)) + \eta \\ \eta &\sim N(0, Q) \end{aligned} \tag{11}$$

where W is the given map of the environment.

3.3 Data Association

We will utilize the maximum likelihood data association(Alg 4).

Algorithm 4 Maximum Likelihood Data Association Algorithm for the tth time step

for all observations i in z_t **do**
  for m=1 to M **do**
    for all landmarks k in W **do**
      $\hat{z}_t^{k,m} = h(x_t^m, W, k)$ {`Predict Measurement`}
      $\nu_t^{i,k,m} = z_t^i - \hat{z}_t^{k,m}$ {`Calculate Innovation`}
      $\psi_t^{i,k,m} = \frac{1}{2\pi|Q|^{\frac{1}{2}}} exp[-\frac{1}{2}(\nu_t^{i,k,m})^T(Q)^{-1}\nu_t^{i,k,m}]$ {`Calculate Likelihood`}
    end for
    $\hat{c}_t^{i,m} = \arg\max_k \psi_t^{i,k,m}$
    $\Psi_t^{i,m} = \psi_t^{i,\hat{c}_t^{i,m},m}$
  end for
end for

3.4 Weighting

Having computed the associations, the weighting is straightforward:

$$p(z_t|x_t^m, \Psi_t^{i_{1:n},m}) = \prod_{j=1}^{n} \Psi_t^{j,m} \tag{12}$$

where n is the number of observations in t^{th} time step. The particle weights are proportional to $p(z_t|x_t^m, \Psi_t^{i_{1:n},m})$.

3.5 Outlier Detection

Having computed the associations, we can define simple outlier detection methods using measures like the average likelihood of particles. A simple outlier detection measure is given in 13:

$$\hat{o}_t^i = \frac{1}{M}\sum_{m=1}^{M} \Psi_t^{j,m} \leq \lambda_\Psi \tag{13}$$

where λ_Ψ is a threshold on the average likelihood of the particles taking a measurement.

- **Question 15:** What parameters affect the mentioned outlier detection approach? What will be the result of the mentioned method if you model a very weak measurement noise $|Q| \to 0$?
- **Question 16:** What happens to the weight of the particles if you do not detect outliers?

3.6 Re-Sampling

The re-sampling methods in this lab are the same as in the warm-up part: Multinomial Re-sampling and Systematic Re-sampling.

3.7 Instructions

Take a look into the "MCL" folder. The **runlocalization_MCL** function is the entrance point to the code. The structure of the code is very similar to the code you worked with in the previous lab. Instead of ekf_localization.m you will be working with mcl.m.
Unlike the previous lab, you should be able to perform both global localization and tracking using the code in this lab. You can define a tracking problem by defining the initial position(the **start_pose** input variable). If you want the code to perform global localization, you need to pass an empty array(i.e. start_pose = []) to the entrance function.
Read the code and get some idea about what is going on in it. In this part, we will represent the particle set with a matrix instead of a structure: the weight of the particles are concatenated to the particle set. Therefore, the particle set will be of dimension $4 \times M$.
Similar to the EKF localization, you can start with the USE_KNOWN_ASSOCIATIONS flag enabled for debugging purposes. However, this time you need to complete a function to be able to use the known associations. In particulare the outputs (outliers and psi) are used to compute the weights. You need to complete the following functions

1. **init.m**: Depends on the problem at hand
2. **calculate_odometry.m**: Use (4) in the lab 1 instructions
3. **predict.m**: Use (3)
4. **observation_model.m**: Similar to lab 1
5. **associate_known.m**: Use Alg 4
6. **associate.m**: Use Alg 4
7. **weight.m**: Use (12)
8. **multinomial_resample.m**: Use Alg 2
9. **systematic_resample.m**: Use Alg 3

The number of lines of code you write would be approximately 70-80 lines while you can reuse code from what is provided and what you have done in the EKF Localization.

3.8 Data Sets

The three data sets provided for the previous lab are also included here in addition to new data sets. Fig 1 depicts the ground-truth and odometry information related to the new data sets in this lab. Start with the data sets you are already familiar with. Start with a tracking problem. Try different number of particles. Play with the process and measurement noise covariance matrices and pay attention how the performance of the filter is affected by them. If you have done everything right and your code is functioning well, you will realize than you need compensation noises in order to make the filter work correctly(you need to model stronger noises than the actual noises in the underlying system). You can use the so_pb_10_outlier data set to test if your outlier detection method works correctly.
Now, try the global localization problem. Note that you need to use more particles in a global localization problem compared to a tracking problem specially in cases that multiple hypotheses match the measurements and the particle set also agrees with the hypotheses. For example, if you perform a global localization on the so_pb_40_no data set, as the environment is perfectly symmetric, you have 160 valid hypotheses for the entire simulation!
Run your code on the following data sets:

1. **map_sym2.txt + so_sym2_nk**: This data set corresponds to a perfectly symmetric environment with 4 landmarks. Notice that you need to increase the part_bound variable(in init.m, e.g. part_bound = 20) to be able to keep all hypotheses(why?). Try tracking and global localization in this data set. How many valid hypotheses you have for these 4 landmarks? Start with 1000 particles. Does you filter keep all the hypotheses reliably? How about 10000 particles? What do you think is the reason? Try multinomial sampling. How well does it preserve multiple hypotheses? How well are you hypotheses preserved when you model stronger/weaker measurement noises?

2. **map_sym3.txt + so_sym3_nk**: This data set corresponds to a nearly symmetric environment with 5 landmarks. At first, there are multiple valid hypotheses but, approximately after the time step 180 the robot observes the top right landmark which breaks the symmetry and only one hypothesis remains valid. Does your filter converge to the correct hypothesis? Include in your report an image just before and just after the convergence, (You can do this easiest by inserting a line like:

   'if $(count > 170)\&(count < 250)$ pause; end'

   then save the plot and hit enter until you get one hypothesis, then save again.)

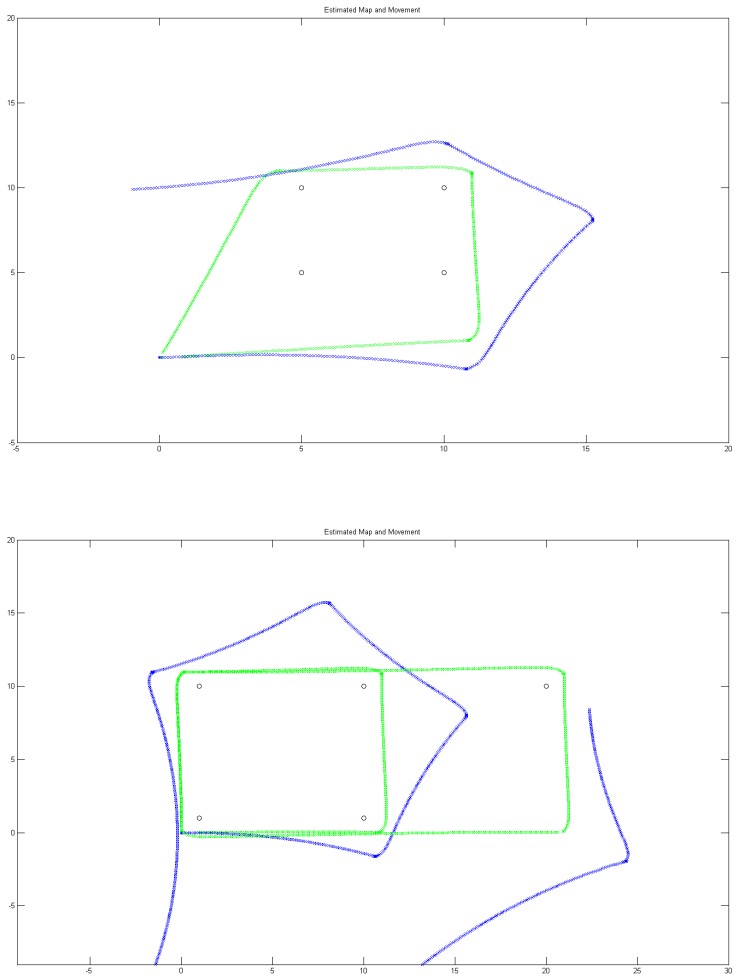

Figure 1: The data sets, top: so_sym2_nk, bottom: so_sym3_nk