ME8135 — Assignment 5 Report

Student: Arash Basirat Tabrizi
Submitted to: Dr. Sajad Saeedi
Due: June 23, 2023

Introduction:

In the previous assignments we explored a variety of estimation techniques and implemented them using Python programs. In this report we aim to summarize the basic fundamentals of the estimation techniques and discuss their pros and cons.

Terminologies:

A key concept in probabilistic robotics is that of a *belief*. A belief reflects a robot's internal knowledge about the state of the environment. We will denote belief over a state variable x_t by $bel(x_t)$, which is an abbreviation for the posterior probability distribution over the state x_t at time t. In the context of probabilistic filtering, the probability distribution $\overline{bel}(x_t)$ predicts the state at time t based on the previous state posterior, before incorporating the measurement at time t. Calculating $bel(x_t)$ from $\overline{bel}(x_t)$ is called *correction* or the *measurement update*. We will use these terminologies frequently in this report.

1. Recursive State Estimation Techniques

1.1. The Bayes Filter Algorithm

This principal algorithm calculates the belief distribution *bel* from measurement, z_t, and control data, u_t. The Bayes filter is **recursive**, that is, the belief $bel(x_t)$ is calculated from the belief $bel(x_{t-1})$:

Algorithm Bayes_filter$(bel(x_{t-1}), u_t, z_t)$**:**
  for all x_t *do*
    $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
    $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
  endfor
  return $bel(x_t)$

Figure 1: The general algorithm for Bayes filtering (page 27 of *Probabilistic Robotics* textbook).

In probabilistic robotics, Bayes filters are implemented in several different ways. Since the Bayes filter is not a practical algorithm, probabilistic algorithms use tractable approximations. As we will discuss in later sections of this report, there exist quite a variety of techniques and algorithms that are all derived from the Bayes filter.

1.2. Gaussian Filters

Historically, Gaussian filters constitute the earliest tractable implementations of the Bayes filter for continuous spaces. They are also by far the most popular family of techniques to date. This section introduces two basic Gaussian filter algorithms that we explored in *Assignment 2*.

1.2.1 Kalman Filter

Probably the best studied technique for implementing Bayes filters is the *Kalman filter* (KF). It is a technique for filtering and prediction in ***linear Gaussian systems***. The KF represents beliefs by the **moments parameterization** and implements the belief computation for continuous states (continuous-time motion model). KFs are classical recursive filters that are commonly used when the motion and observation models are linear, and the measurement and process noises are zero-mean Gaussian. The observation that any linear transformation of a Gaussian random variable results in another Gaussian random variable plays an important role in the derivation of the KF algorithm. The efficiency of the KF is then due to the fact that the parameters of the resulting Gaussian can be computed in closed form.

1.2.2 Extended Kalman Filter

In practice, motion and observation models are rarely linear. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by linear motion models. The *extended Kalman filter* (EKF) relaxes the linearity assumption by assuming that the motion and observation models can in fact be governed by ***nonlinear*** functions. The EKF calculates a Gaussian approximation to the true belief. The goal of the EKF is thus shifted from computing the exact posterior to efficiently estimating its mean and covariance. However, since these statistics cannot be computed in closed form, the EKF has to resort to an additional approximation called ***linearization***. The EKF can often be effective for mildly nonlinear, Gaussian or non-Gaussian systems.

1.3. Non-Gaussian Filters

In order to able to handle non-Gaussian noise and nonlinear observation and motion models, a popular alternative to Gaussian filters are ***nonparametric filters***.

1.3.1 Particle Filter

The *Particle filter* (PF) is a nonparametric implementation of the Bayes filter. The key idea of the PF is to represent the posterior belief $bel(x_t)$ by a set of random state samples drawn from this posterior. In this approach we draw a large number of samples from the input density, transform each one of these samples through the nonlinearity exactly, and then build the output density from the transformed samples (***Monte Carlo Method***). Here the samples of a posterior distribution are called *particles*. Just like all other Bayes filter algorithms discussed thus far, the PF algorithm constructs the belief $bel(x_t)$ recursively from the belief $bel(x_{t-1})$ one time step earlier. The real novelty of the PF lies in its use of the ***importance sampling*** algorithm which we discussed in detail in *Assignment 3*.

1.4. Other Parametric Techniques for Estimation

There are several techniques that we can use to fit a parametric model to data. For example, in the problem of *pose estimation*, we wish to fit a homographic transformation H given a set of pairs of corresponding points in presence of outliers.

1.4.1 RANSAC

Random sample consensus (RANSAC) is an iterative method to fit a parameterized model to a set of observed data containing outliers. In *Assignment 4* we explored how the basic RANSAC algorithm can be refined to estimate homography. In the context of pose estimation, the refined RANSAC algorithm can help find the optimal pose.

2. Comparison of Estimation Techniques

2.1. Kalman Filter

1. **Computational efficiency**: computationally efficient and is particularly suitable for systems with linear dynamics and Gaussian noise. It has a relatively low computational cost compared to more complex filters.

2. **Accuracy of the approximation**: it assumes that the motion and observation models follow Gaussian distributions. If these assumptions hold, the KF can provide accurate state estimates and covariance matrices.

3. **Ease of implementation:** has a relatively straightforward implementation process when the motion and observation models are linear.

2.2. Extended Kalman Filter

1. **Computational efficiency**: introduces nonlinearity into the state estimation process by linearizing the motion and observation models. Linearization process increases the computational complexity compared to the KF but is still relatively efficient.

2. **Accuracy of the approximation**: approximation accuracy depends on the linearity of the motion and observation models. If the nonlinearities are significant, the linearization approximation can produce very faulty results.

3. **Ease of implementation:** requires calculating the motion and observation models and their associated Jacobian matrices. This process can be more challenging than the basic KF, as it involves differentiating the nonlinear functions.

2.3. Particle Filter

1. **Computational efficiency**: tends to be computationally more expensive compared to the KF and EKF. The complexity increases with the number of particles used for estimation, and the filtering process involves resampling and weighting steps that can be computationally intensive.

2. **Accuracy of the approximation**: more flexible than the KF and EKF as they can handle nonlinear systems and non-Gaussian noise. They provide better approximations for nonlinear and non-Gaussian state estimation problems. However, the accuracy depends on the number of particles used and the quality of the proposal distribution.

3. **Ease of implementation:** more challenging than the traditional KF or EKF. It requires defining the proposal distribution, the resampling strategy, and handling the weight updates for particles. Additionally, selecting an appropriate number of particles to balance computational cost and estimation accuracy can be a non-trivial task.

2.4. RANSAC

1. **Pros**: simple and easily implementable. Successful in different contexts. Robust to outliers and can provide accurate pose estimation even in the presence of noise.

2. **Cons**: requires many parameter tuning. The accuracy of the pose estimation depends on the percentage of outliers and the quality of the inlier-outlier separation criterion. The number of iterations required by RANSAC can be high to achieve good accuracy, particularly in scenarios with a large number of outliers. Additionally, there always exists a trade-off of accuracy-vs-time.