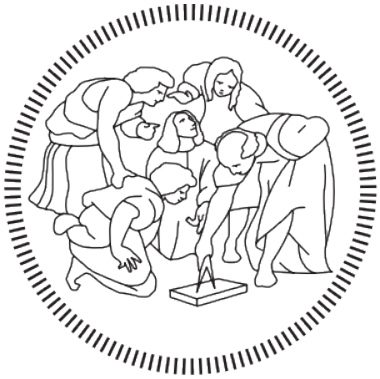

POLITECNICO DI MILANO

AIR Lab - Artificial Intelligence and Robotics Lab
Dipartimento di Elettronica, Informazione e Bioingegneria

MSc. in Computer Science and Engineering

Implementation of a 6 DoF EKF-SLAM for Leonardo Drone Contest

Author:
Diego Emanuel Avila
Matricola: 903988

Supervisor:
Prof. Matteo Matteucci

December 2020

Preface

Some preface

Diego Emanuel Avila
Milano
October 2020

Abstract

Abstract

Contents

List of Figures

List of Tables

List of Acronyms

Introduction 1

Theoretical Background 2

2.1 Robot Operating System

The *Robot Operating System* (ROS) is an open source middleware that provides inter-process communication through a message passing mechanism. It is, also, a collection of libraries, tools and conventions with the aim of simplify the development of software for robots. In this sense, one of the main components of the framework are nodes which encapsulate processes and/or algorithms.

2.1.1 Nodes

Nodes are processes that perform some computation and communicate between them using a publisher/subscriber infrastructure based on topics. A node subscribes to a particular topic, and it will receive messages of that type from a publisher node. This way, publisher is hid to the subscriber, reducing the coupling between them. The advantage of this mechanism is that nodes can be developed separately once the message structure is defined, forcing developers to implement clear interfaces for communication by using a message *Interface Description Language* (IDL). Moreover, this enables a modular and distributed development of the robotic system, while providing some fault tolerance and reduced code complexity.

2.1.2 Topics

As mentioned before, communication between nodes is done via a publisher/subscriber architecture based on topics, which are named buses over which messages are exchanged. A node that generates data, publishes it over a topic, and this information is consumed by those nodes subscribed to that topic. There can be several publishers (as well as subscribers) for a topic. The transport protocol used for exchanging messages is defined at runtime and can be of two types: TCPROS or UDPROS. The description of these protocols is beyond the scope of this document and the reader is invited to read the ROS documentation for detailed information.

2.1.3 Services

As well as topics, services are a way to communicate nodes. The difference is that topics are an asynchronous way of communication, while services are synchronous. This is a key difference, because nodes decide when to trigger a service and so, services are used to retrieve specific information or to ask another node to do something out of caller's node scope.

2.1.4 Tools

Regarding the tools provided by ROS, one that is be crucial for debugging and/or experimentation is the bag recording and playback. Since the message passing infrastructure is anonymous (meaning that nodes communicate between each others without knowing which node sent or received a message) it is possible to record the messages during a period of time,

without taking into consideration which node sent a message and which node received it. This recorder bag is useful for debugging because it can be played back, hence reproducing a previous experiment. Also, it can be used for development of new nodes that depend on the messages contained in the bag.

Another useful tool provided is *ROS Visualization* (RViz), a 3D visualization tool. With this tool it is possible to see the robot, orientation, reference frames, covariance matrices, etc. In addition, it is possible to draw lines, arrows, text and others, onto the environment in order to see useful information that cannot be extracted from messages.

2.2 Kalman Filter

Kalman Filter (KF) is a technique for filtering and prediction in *linear Gaussian systems* and applicable only to continuous states. It assures that the posteriors are Gaussian if the Markov assumption is hold, in addition to three conditions. The Markov assumption states that the past and future data are independent if one knows the current state; the other three conditions are the following:

1. The state transition probability $p(x_t|u_t, x_{t-1})$ must be a linear function in its arguments. This is guaranteed by $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$. Here, x_t and x_{t-1} are the state vectors, of size n, and u_t represents the control vector, of size m, at time t; A_t and B_t are matrices of size $n \times n$ and $n \times m$ respectively. In this way, the state transition function becomes linear in its arguments, hence KF assumes a linear system dynamics.

   ϵ_t represents the uncertainty introduced by the state transition, and it is a Gaussian random variable with zero mean and R_t covariance.

2. The measurement probability $p(z_t|x_t)$ must be linear in its arguments. This is guaranteed by $z_t = C_t x_t + \delta_t$, where z_t represents the measurement vector of size k, C_t is a matrix of size $k \times n$ and δ_t is a Gaussian noise with zero mean and Q_t covariance.

3. The initial belief should be normally distributed.

Given these three conditions, we are guaranteed that the posterior probability is Gaussian.

Algorithm 1: Kalman Filter algorithm

Input: μ_{t-1}, Σ_{t-1}, u_t, z_t

$\hat{\mu}_t = A_t \mu_{t-1} + B_t u_t$

$\hat{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

$K_t = \hat{\Sigma}_t C_t^T \left(C_t \hat{\Sigma}_t C_t^T + Q_t \right)^{-1}$

$\mu_t = \hat{\mu}_t + K_t (z_t - C_t \hat{\mu}_t)$

$\Sigma_t = (I - K_t C_t) \hat{\Sigma}_t$

return μ_t, Σ_t

The KF algorithm's input is the belief at time $t-1$, represented by its mean (μ_{t-1}) and its covariance (Σ_{t-1}), in addition to the control vector (u_t) and the observations (z_t). As a result, it returns the current belief characterized by its mean μ_t and covariance Σ_t.

The first step in the algorithm (lines 1 and 2), represent the prediction step. It calculates the current belief before incorporating the observations, but after adding the control vector. The estimated belief is characterized by its mean, $\hat{\mu}_t$, and its covariance $\hat{\Sigma}_t$.

The second step (from line 3 to 5) starts by calculating K_t, the Kalman gain, which specifies the degree in which the observation is incorporated to the new state estimate. Following, at line 4, the new mean is estimated by means of the Kalman gain and the *innovation*, which is the difference between the observation z_t and the expected measurement $C_t\hat{\mu}_t$. Finally, the new covariance is calculated.

2.2.1 Extended Kalman Filter

2.3 Simultaneous Localization and Mapping

2.3.1 EKF-SLAM

Bibliography

Robot operating system. URL `https://www.ros.org/`.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. The MIT Press, 2006. ISBN 978-0-262-20162-9.